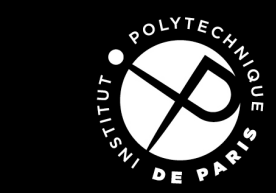
Learning and Algorithms for Online Matching

Flore Sentenac



INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAG005

Thèse de doctorat

Learning and Algorithms for Online Matching

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École nationale de la statistique et de l'administration économique

École doctorale n°574 École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat: Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 11 juillet 2023, par

Flore Sentenac

Composition du Jury :

Alfred Galichon Professeur, New York University	Président du jury
Olivier Tercieux Paris School of Economics (PSE) and CNRS	Rapporteur
Laurent Massoulié Professeur, Inria Paris, Université DIENS PSL	Rapporteur
Shie Mannor Professeur, Technion	Examinateur
Jaouad Mourtada Professeur adjoint, CREST, ENSAE	Examinateur
Vianney Perchet Professeur, CREST, ENSAE	Directeur de Thèse

Remerciements

Mes remerciements vont d'abord à mon directeur de thèse Vianney, pour son encadrement et son soutien tout au long de la thèse. Je t'ai contacté en juillet 2019, sans bourse ni projet de thèse très précis, et pourtant déjà là tu m'as soutenue, présenté d'autres chercheurs et des projets de recherche potentiels. J'ai commencé à travailler avec toi quelques mois après, et les quatre années qui ont suivi ont été si riches que j'aurais du mal à en faire un résumé complet. Tu m'as offert un environnement de recherche incroyablement stimulant, plein de rencontres, poussé à la curiosité et soutenu dans tous mes projets, le tout avec enthousiasme et décontraction. Je n'aurais certainement pas fait le quart de la moitié de tout ce que j'ai fait sans ça. Merci pour tout, je mesure la chance que j'ai eue, et pour un peu, j'aurais du mal à partir.

Un grand merci à Laurent Massoulié et à Olivier Tercieux pour le temps qu'ils ont consacré à la lecture de mon manuscrit et pour leurs rapports détaillés et constructifs.

Je tiens également à exprimer ma gratitude à Alfred Galichon, Shie Mannor et Jaouad Mourtada pour avoir accepté d'être membres de mon jury. C'est un honneur et un plaisir de présenter mon travail en votre présence.

Je voudrais aussi remercier particulièrement Claire. Mon passage à Londres a bien sûr été vraiment enrichissant grâce à toi, mais pas seulement. Tous les conseils que tu m'as donnés, toutes les discussions qu'on a eues m'ont beaucoup apporté, autant personnellement qu'intellectuellement, et j'espère que nous aurons (beaucoup) d'autres occasions de nous recroiser.

Julien, mon "binôme" de thèse, tu sais déjà que ces années n'auraient pas du tout été les mêmes sans toi. Tu m'as aidé à résoudre autant de problèmes de maths que de crises

existentielles. Merci pour toutes ces répétitions de présentation, ces relectures de mail et de candidatures. J'aimerais te dire que je vais arrêter d'abuser de ta bienveillance, mais je ne veux pas mentir par écrit.

Au cours de ma thèse, j'ai eu la chance de collaborer avec de nombreux autres chercheurs. Nathan, qui a été présent tout au long du parcours et m'a beaucoup appris. Etienne, Hugo, Corentin, Mathieu et Nadav qui sont des amis autant que des collaborateurs (in english for you Nadav: thank you for illuminating my slides with funny pictures).

Merci aussi à Milan, Clément, Matthieu et Laurent de m'avoir partagé votre expérience. Malgré la coupure du covid, j'ai eu la chance de profiter de l'environnement du CREST et de son équipe de permanents. Merci à Arnak, Nicolas, Anna, Victor-Emmanuel, Jaouad , Cristina, Guillaume et Azadeh pour toutes ces discussions.

J'ai aussi eu la chance de partager ce moment avec toute une équipe de co-thésards et post-docs géniaux, les anciens, Pierre, Firas et Evrard, ceux qui sont arrivés en cours de route, Maria, Côme, Mike, Sasila, Hafedh et Ziyad, Felipe et Dorian, et le seul qui ait été là tout du long, Lorenzo. Merci pour tous les group meeting et les reading group (surtout pour votre soutien aux moments de faible compréhension). Vos gâteaux vont me manquer.

Il y a aussi eu tous ceux du CREST avec qui j'ai partagé les déjeuners Magnan, les siestes poufs, les cafés sur la terrasse et la "table des enfants" du CIRM de décembre Arya, Evgenii, Nicolas, Lu, Suzanne, Etienne, Clara, Hugo, Younès, Amir, Meyer, Théo et Nina. Mention particulière à toi Nayel, pour m'avoir tant de fois épargné le RER de 7h du matin. Mon sommeil t'en remercie encore.

Je remercie aussi tout le personnel administratif et en particulier Leyla, Djamila et Fanda qui m'ont, entre autres, aidé à gérer mes inscriptions et permis d'assister aux différentes conférences et summer schools.

Je voudrais aussi remercier sans les nommer mes amis pour tous les moments passés avec vous pendant ces quatre années. Des confinements aux verres en terrasse, vous mettez des paillettes dans ma vie.

Je remercie aussi ma famille. Mon père Jean, pour m'avoir transmis le goût des maths, et ma mère Bénédicte, pour m'avoir rappelé qu'il n'y avait pas que les maths. Mes grands-parents, Eric et Béatrice, qui m'ont beaucoup accueilli en télétravail, et m'ont poussé à la curiosité depuis la petite enfance. Mes deux sœurs, Camille et Philippine, mon parrain Bernard, ma marraine Charlotte, mon oncle Harold, ma tante Claire, et le petit Victor, qui je l'espère sera sage pendant la soutenance.

Enfin, merci à Ayman de partager ma vie. T'avoir à mes côtés est un privilège. Et il va bien falloir que je l'admette, tu avais raison quand tu me disais il y a presque dix ans que j'aimerais la recherche.

Contents

Chapter 1

Introduction (version française)

Contents

Dans les scénarios séquentiels, une option parmi plusieurs est irrévocablement choisie à chaque itération et les performances globales de l'algorithme sont mesurées à la fin de l'exécution. Cela modélise de nombreux problèmes de la vie réelle et est particulièrement pertinent pour les applications numériques.

Le cadre du Bandit Multi-Bras est au cœur de la section Section 1.1. Il a été introduit dans le contexte des essais cliniques Robbins (1952); Thompson (1933), et a récemment attiré à nouveau l'attention en raison de son application aux systèmes de recommandation séquentiels. Le modèle de matching séquentiel, détaillé dans la section Section 1.2.2, est particulièrement pertinent pour la publicité numérique.

1.1 Apprentissage séquentiel

Dans cette section, nous introduisons les bandits multi-bras stochastiques ainsi que les principaux résultats dans ce modèle. La section se termine par une brève introduction à une sélection d'algorithmes de bandits. Cela fournira une base pour une partie du travail présenté dans cette thèse. Pour une introduction plus approfondie aux bandits multi-bras, nous renvoyons le lecteur à la monographie Lattimore and Szepesvári (2020).

1.1.1 Bandits Multi-Bras Stochastiques et Regret

Au début de l'exécution, un agent dispose de K options disponibles, appelées "bras". Chacune de ces options $i \in K$ a une récompense moyenne associée $\mu_i \in [0, 1]$. À chaque itération t jusqu'à l'horizon T, il choisit une option $a(t) \in [K]$. Nous disons que l'agent "tire le bras $a(t)$". Il reçoit ensuite et observe la récompense bruitée $r(t) = \mu_{a(t)} + \varepsilon_t$, où $(\varepsilon_t)_{t\in[T]}$ est une séquence i.i.d. tirée d'une distribution 1-sous-gaussienne à moyenne nulle.

Lorsque les valeurs des moyennes des différents bras, $(\mu_i)_{i\in[K]}$, sont connues, la stratégie optimale consiste à choisir le bras avec la moyenne la plus élevée μ^* à chaque itération. Cependant, ces moyennes sont inconnues de l'agent au début de l'exécution. L'objectif de l'agent est alors de concevoir une stratégie qui minimise la différence entre la récompense espérée obtenue en suivant cette stratégie et la récompense espérée pour la stratégie optimale. Cette quantité est appelée regret et est formellement définie pour une stratégie π comme suit :

$$R_\pi(T) := \mathrm{E}\left[\sum_{t\in T} \mu^* - \mu_{a(t)}\right].$$

L'espérance est prise par rapport à la stratégie de l'agent.

D'une part, à une certaine itération, l'agent peut **exploiter** sa connaissance actuelle pour maximiser la récompense instantanée de manière greedy. D'autre part, à chaque itération, l'agent n'observe qu'une version bruitée de la moyenne du bras tiré. Ainsi, apprendre les moyennes des bras nécessite de les tirer, ce qui implique que les stratégies raisonnables incorporent un niveau d'**exploration**.

Dans cette exposition, nous nous concentrerons sur les bornes configuration-dépendante. Ce sont des bornes supérieures et inférieures sur les performances des algorithmes pour des valeurs de paramètres fixes. Nous renvoyons les lecteurs intéressés par les bornes minimax à la monographie Lattimore and Szepesvári (2020).

1.1.2 Borne inférieure

Dans cette section, nous exposons une borne inférieure sur le regret configuration-dépendant atteignable par des algorithmes "raisonnables", définis dans la littérature comme la classe suivante d'algorithmes asymptotiquement cohérents.

Definition 1.1.1 (Algorithme asymptotiquement cohérent)**.** *Un algorithme π est asymptotiquement cohérent si, pour chaque instance de bandit, pour tout $\alpha > 0$, $R_\pi(T) = o(T^\alpha)$.*

Remarquez que pour tout algorithme asymptotiquement cohérent, la récompense accumulée est asymptotiquement de même ordre que celle de la stratégie optimale qui tire le meilleur bras à chaque itération.

La borne inférieure suivante est dérivée à partir d'arguments d'information théorique que nous ne détaillerons pas ici.

Theorem 1.1.2 (Lai and Robbins (1985))**.** *Considérons une instance de bandit où le bruit $\varepsilon_t \sim \mathcal{N}(0,1)$. Alors tout algorithme asymptotiquement cohérent a la borne inférieure suivante sur son regret asymptotique :*

$$\liminf_{T\to+\infty} \frac{R(T)}{\log(T)} \geq \sum_{\mu_k<\mu^*} \frac{2}{\mu^* - \mu_k}.$$

Bien que cette borne inférieure soit asymptotique, elle indique que la borne supérieure sur le regret de tout algorithme raisonnable ne peut pas être inférieure à $O\left(\sum_{\mu_k<\mu^*} \frac{\log(T)}{\mu^*-\mu_k}\right)$.

1.1.3 Algorithmes classiques de bandits

Un large éventail d'algorithmes a été proposé pour traiter le cadre des bandits multi-bras stochastiques. Ici, nous nous concentrerons sur deux stratégies seulement : une première dans laquelle l'exploration est réalisée en une seule fois au début, et une deuxième dans laquelle l'exploration et l'exploitation sont entrelacées. Nous fournissons des pseudo-codes succincts et des bornes supérieures sur le regret sans démonstration. Certains algorithmes de cette thèse s'appuient sur les idées des deux stratégies présentées ici.

Dans les présentations des deux algorithmes, nous notons :

- $N_k(t)$ le nombre de fois où le bras k a été tiré jusqu'au temps t,
- $\hat{\mu}_k(t) = \frac{1}{N_k(t)} \sum_{s=1}^{t-1} r(t) \mathbb{1}\{a(t) = k\}$, la moyenne empirique pour le bras k à l'itération t,
- $L_k(t) = \hat{\mu}_k(t) - \sqrt{\frac{2\log(T)}{N_k(t)}}$, $U_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log(T)}{N_k(t)}}$, bornes inférieure et supérieure à l'itération t sur la vraie moyenne du bras k avec une forte probabilité.

Algorithme Explore-Then-Commit La stratégie la plus simple d'Explore-Then-Commit consiste à explorer uniformément tous les bras, puis à se concentrer sur le bras avec la plus haute moyenne empirique. L'algorithme présenté dans Algorithm 1 est une version légèrement affinée de ce principe, également appelée Successive-Elimination. L'algorithme maintient un ensemble de bras actifs $\mathcal{A}$, qui sont candidats pour être le bras avec la plus haute moyenne. Les bras de cet ensemble sont échantillonnés uniformément. Lorsqu'un bras est jugé sous-optimal avec une forte probabilité, il est définitivement éliminé de cet ensemble et donc jamais tiré à nouveau.

Algorithm 1: Explore-Then-Commit

Entrée : ensemble de bras $[K]$ et horizon T

$\mathcal{A} = [K]$;
while $|\mathcal{A}| > 1$ **do**
  Tirer une fois chaque bras dans $\mathcal{A}$;
  $\forall k \in [K]$ tel que $U_k(t) \leq \max_{i\in\mathcal{A}} L_i(t)$, enlever k de $\mathcal{A}$;
end
Tirer le bras restant dans $\mathcal{A}$ jusqu'à la fin;

Theorem 1.1.3 (Perchet and Rigollet (2013))**.** *Le regret de Algorithm 1 avec l'horizon connu T est borné supérieurement par :*

$$R(T) = O\left(\sum_{\mu_k<\mu^*} \frac{\log(T)}{\mu^* - \mu_k}\right).$$

Comme le montre la Section 1.1.2, la borne sur le regret obtenue pour cet algorithme simple n'est qu'une constante supérieure à la borne inférieure asymptotique sur le regret configuration-dépendant. Cependant, il est connu que le regret de toute stratégie Explore-Then-Commit est sous-optimal d'une constante multiplicative Garivier and Kaufmann (2016). Néanmoins, la simplicité de l'algorithme est avantageuse et peut s'avérer utile dans des contextes plus complexes.

Algorithme Upper-Confidence-Bound L'algorithme Upper-Confidence-Bound (UCB), Auer et al. (2002a), est un exemple d'algorithme de bandit entièrement adaptatif. À chaque itération, l'indice de chaque bras $U_k(t)$ est une borne supérieure de la vraie moyenne du bras avec une forte probabilité, et le bras avec l'indice le plus élevé est tiré.

Algorithm 2: Upper-Confidence-Bound

Entrée : ensemble de bras $[K]$ et horizon T

for $t = 1, \dots, T$ **do**
  Tirer $\arg\max_{k \in [K]} U_k(t)$;
end

Theorem 1.1.4 (Auer et al. (2002a))**.** *Le regret de Algorithm 2 avec l'horizon connu T est borné supérieurement par :*

$$R(T) \leq \sum_{\mu_k < \mu^*} \frac{16 \log(T)}{\mu^* - \mu_k} + 3(\mu^* - \mu_k).$$

Encore une fois, l'algorithme UCB atteint asymptotiquement la borne inférieure Section 1.1.2 à constante multiplicatrice près. Mieux encore, pour certaines distributions, l'algorithme UCB avec une version légèrement modifiée des bornes supérieures est asymptotiquement optimal Cappé et al. (2013).

Ainsi, Algorithm 2 fonctionne strictement mieux que Algorithm 1, mais il est parfois plus difficile à généraliser dans des contextes plus complexes.

1.2 Algorithmes séquentiel

Par définition, un algorithme séquentiel est un algorithme qui traite séquentiellement ses entrées sans avoir accès à la séquence complète au début de l'exécution.

Un exemple classique est le problème de la location de skis. Une personne arrive en vacances et peut décider chaque jour de louer des skis pour une journée au prix de p ou de les acheter au prix de B. Si la personne sait qu'elle va passer D jours en vacances, la décision optimale est évidente : acheter des skis le premier jour si $B < pD$, sinon les louer chaque jour. Lorsque D est inconnu, la situation est plus complexe, mais il existe toujours un moyen de s'assurer de ne pas payer plus de deux fois le prix optimal : louer des skis pendant les premiers $\lfloor \frac{B}{p} \rfloor$ jours, puis les acheter le jour suivant.

Généralement, comme dans l'exemple précédent, les algorithmes séquentiels sont évalués par comparaison avec les algorithmes optimaux qui ont connaissance de la séquence complète des entrées.

La prochaine section est consacrée à la définition du ratio de compétitivité (**C.R.**), une mesure classique de la performance des algorithmes séquentiels. Ensuite, nous donnerons plus

de détails sur le cadre de matching séquentiel, qui est l'un des principaux points d'intérêt de cette thèse.

Dans cette section et tout au long de la thèse, un algorithme séquentiel désigne un algorithme pour lequel la séquence est révélée de manière séquentielle (comme celui qui n'a pas accès à D dans l'exemple de la location de skis), tandis qu'un algorithme hors ligne désigne un algorithme qui a accès à la séquence complète des entrées (comme celui qui connaît D dans le même exemple).

1.2.1 Ratio de compétitivité

On peut considérer à la fois des algorithmes de minimisation des coûts et des algorithmes de maximisation de la récompense. Comme nous traitons des algorithmes de maximisation de la récompense dans le cadre du problème de matching séquentiel, cette section formule tout en termes d'algorithmes de maximisation de la récompense avec des récompenses positives.

Soit ALG(G) le score d'un algorithme pour une entrée G et OPT l'algorithme hors ligne optimal.

Definition 1.2.1 (Ratio de compétitivité (**C.R.**))**.** *Un algorithme de maximisation de la récompense ALG atteint un rapport de compétitivité α pour une classe d'entrées $\mathcal{G}$ s'il existe une constante c telle que pour tout $G \in \mathcal{G}$,*

$$\mathrm{E}[ALG(G)] \geq \alpha \mathrm{E}[OPT(G)] + c,$$

où les espérances sont prises par rapport aux éventuelles variations aléatoires des algorithmes et de l'entrée G.

Remarque : Pour un algorithme de minimisation des coûts, le
min serait remplacé par un max et l'inégalité serait inversée.

Notez que le ratio de compétitivité (**C.R.**) est borné entre 0 et 1, plus la valeur est élevée, mieux c'est. Deux éléments influencent sa valeur : l'algorithme lui-même et la classe d'entrées $\mathcal{G}$ sur laquelle l'algorithme est évalué. Si des hypothèses plus restrictives sont faites sur $\mathcal{G}$, il est alors plus facile d'obtenir des algorithmes avec un ratio de compétitivité élevé.

Dans l'exemple de la location de skis, une entrée G est un ensemble de trois paramètres p, B, D. Si ces paramètres peuvent prendre n'importe quelle valeur positive, ce qui correspond à la classe d'entrées la plus large possible, le plus petit ratio de compétitivité (**C.R.**) atteignable est 2. Cependant, si nous supposons que ces paramètres sont tirés d'une distribution connue, il est possible d'obtenir un ratio de compétitivité (**C.R.**) plus petit.

Remarque : Selon la définition 1.1.1, tout algorithme de bandit asymptotiquement consistant a un ratio de compétitivité asymptotique de 1.

1.2.2 Matching séquentiel dans les graphes bipartis

Matching hors ligne. Nous commençons par détailler la version hors ligne du problème de matching avant de passer à sa contrepartie séquentielle. Cette thèse porte sur les matching dans les graphes bipartis.

Definition 1.2.2 (Graphe biparti)**.** *Un graphe biparti $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ est un graphe avec deux ensembles de sommets U et V et un ensemble d'arêtes E tel que aucun sommet du même ensemble n'est adjacent.*

Tout au long de ce manuscrit, un matching est définie comme suit.

Definition 1.2.3 (Matching)**.** *Une matching M sur le graphe G est un sous-ensemble de $\mathcal{E}$ tel que chaque sommet de G soit l'extrémité d'au plus une arête dans M. Si un sommet $u \in \mathcal{U} \cup \mathcal{V}$ est l'extrémité d'une arête dans M, nous disons que u est* ***matché****, sinon il est* ***libre*** *ou* ***non matché****. Un matching de cardinalité maximale est un* ***matching maximale****.*

Le problème de trouver un matching maximal dans un graphe biparti peut être résolu à l'aide de l'algorithme de Hopcroft-Karp, avec un temps d'exécution de $O\left(|\mathcal{U} \cup \mathcal{V}|^{1/2}|\mathcal{E}|\right)$. Dans le cas d'une matching pondéré, où chaque arête $(u, v) \in \mathcal{E}$ est associée à un poids $w_{u,v}$, et l'objectif est de trouver un matching maximisant la somme des arêtes sélectionnées, l'algorithme hongrois peut être utilisé, avec un temps d'exécution de $O\left(|\mathcal{U} \cup \mathcal{V}|^3\right)$.

Matching séquentiel. La version en ligne du problème peut être formulée comme suit :

1. Au début de l'exécution, tous les sommets du côté "hors ligne" $\mathcal{U}$ sont disponibles.
2. À chaque itération, un sommet $v \in \mathcal{V}$ est révélé, ainsi que ses arêtes.
3. v peut être associé à l'un de ses voisins libres, et le match choisi (s'il y en a un) est irrévocable.

Ce cadre théorique est particulièrement adapté à la publicité en ligne : $\mathcal{U}$ est l'ensemble des campagnes/publicités qu'un annonceur peut diffuser et les utilisateurs $v_1, v_2, \ldots, v_T$ arrivent séquentiellement (Manshadi et al., 2012; Mehta, 2012). Certains d'entre eux sont éligibles pour un sous-ensemble important de campagnes, tandis que d'autres ne le sont pas (généralement en fonction de leurs attributs/caractéristiques, tels que la localisation géographique, l'historique de navigation ou toute autre information pertinente). L'objectif d'un annonceur (dans ce modèle simplifié) est de maximiser le nombre de publicités affichées. En pratique, les campagnes/publicités ne sont pas affichées une seule fois, mais ont un budget maximal d'impressions (par exemple, une publicité spécifique peut être affichée uniquement 10 000 fois par jour). Une astuce possible consiste à dupliquer les sommets de $\mathcal{U}$ autant de fois que le budget le permet.

La version séquentielle du matching est évidemment plus difficile que la version hors ligne, notamment car on ne peut généralement pas obtenir un matching maximal. Nous détaillons dans la section suivante, en fonction des hypothèses sur la classe de graphes considérée, les valeurs du **C.R.** réalisables pour les algorithmes de matching en ligne.

1.2.3 Cadres classiques de matching séquentiel et algorithmes

Dans cette section, nous détaillerons les hypothèses classiques faites sur la classe de graphes considérée $\mathcal{G}$, ainsi que les algorithmes proposés dans la littérature dans chaque contexte.

Cadre adversarial Dans le cadre adversarial, aucune restriction n'est imposée sur la classe d'entrée $\mathcal{G}$. Le graphe peut être n'importe quel graphe et les sommets arrivent dans n'importe quel ordre. C'est pourquoi on parle d'adversarial, car en se référant à la définition du **C.R.** dans Theorem 1.2.1, on constate que dans ce paramètre, le **C.R.** d'un algorithme est évalué sur le graphe sur lequel il a les performances les plus faibles.

Il est peut-être surprenant que, même dans ce cadre difficile, une garantie de 1/2 sur le **C.R.** soit facilement obtenue avec l'algorithme suivant :

Pour être cohérent avec le reste de la thèse, nous définissons GREEDY comme l'algorithme qui choisit le match du sommet entrant uniformément au hasard, mais le résultat suivant est valable pour tous les algorithmes choisissant n'importe quel match dès qu'il en existe un disponible.

Algorithm 3: Algorithme GREEDY

```
1 for t = 1, ..., |V| do
2 |   Associer v_t à un voisin libre choisi uniformément au hasard;
3 end
```

Theorem 1.2.4. *Dans le cadre adversarial,*

$$\textbf{\textit{C.R.}}(\textit{GREEDY}) \geq \frac{1}{2}.$$

Proof. Considérons l'événement où GREEDY échoue à matcher à un sommet v_t qui est apparié dans le matching maximal du graphe final. Cela ne peut se produire que si l'appariement de v_t dans le matching optimal était déjà apparié à un autre sommet. Ainsi, pour tout événement de "manqué" (GREEDY n'arrive pas à apparier un sommet qui est matché dans le matching maximal), il y a au moins un événement de "match" (GREEDY apparie un sommet). Cela implique qu'il y a au plus deux fois le nombre de sommets appariés dans le matching maximale par rapport au matching construit par GREEDY, d'où la borne inférieure de 1/2 sur le **C.R.**. □

Cette borne inférieure est optimale.

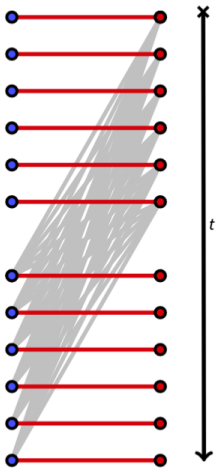


Figure 1.1: Une instance difficile pour GREEDY

Proposition 1.2.5. *Dans le cadre adversarial,*

$$\textbf{\textit{C.R.}}(\textit{GREEDY}) = \frac{1}{2}.$$

Proof. Considérons la famille de graphes $(\mathcal{G}_n)_{n\in\mathbb{N}}$ suivante, illustrée dans Line 3. Il y a $2n$ sommets de chaque côté de $\mathcal{G}_n$, qui sont divisés en deux moitiés, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ et $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, avec $\mathcal{U}_1 = \{u_1, \ldots, u_n\}, \mathcal{U}_2 = \{u_{n+1}, \ldots, u_{2n}\}$, de même pour $\mathcal{V}$. Nous avons l'ensemble d'arêtes suivant :

$$\mathcal{E} = \{(u_i, v_i), \forall i \in [2n], (u_{n+i}, v_j)\forall (i,j) \in [n]^2\}.$$

Dans chaque $\mathcal{G}_n$, il existe un matching de cardinalité $2n$, $\mathcal{M} = \{(u_i, v_i), \forall i \in [2n]\}$. Analysons l'exécution de GREEDY. Tout d'abord, nous observons que tout sommet $u_i \in \mathcal{V}_1$ choisira son matching dans $\mathcal{V}_2$ avec une probabilité supérieure à $\frac{n-i+1}{n-i+2}$, car il n'a qu'un seul voisin dans $\mathcal{U}_1$, et à l'itération i, au plus $i-1$ sommets ont déjà été matchés dans $\mathcal{U}_2$. Cela implique qu'en moyenne, $n - o(n)$ sommets de $\mathcal{V}_1$ sont matchés avec un sommet $u_j \in \mathcal{U}_2$. Pour tous ces $u_j \in \mathcal{U}_2$, leur match dans la correspondance maximale reste non apparié, et nous avons donc :

$$\lim_{n\to+\infty} \frac{\mathrm{E}[\textsc{greedy}(\mathcal{G}_n)]}{2n} = \frac{1}{2}.$$

□

Pour dépasser ce **C.R.** de 1/2, un algorithme différent est nécessaire. L'algorithme RANKING Karp et al. (1990) utilise ce que l'on appelle informellement une "randomisation corrélée" et a un **C.R.** plus élevé dans le cadre adversarial.

Il fonctionne comme suit : au début de l'exécution, une permutation aléatoire π est tirée et le sommet $u_i \in \mathcal{U}$ se voit attribuer le rang $\pi(i)$. Lorsqu'un sommet $v_t \in \mathcal{V}$ arrive, il est apparié

à son voisin libre de rang le plus bas. On parle de "randomisation corrélée" car le choix du voisin pour v_t, bien qu'étant toujours aléatoire, est corrélé d'une itération à l'autre.

Algorithm 4: Algorithme RANKING

```
1 Tirer une permutation aléatoire π;
2 for i = 1, ..., |U| do
3 |   Attribuer à u_i le rang π(i);
4 end
5 for t = 1, ..., |V| do
6 |   Apparier v_t avec son voisin libre de rang le plus bas;
7 end
```

On peut comprendre pourquoi RANKING a un **C.R.** plus élevé que GREEDY en examinant l'exemple difficile pour GREEDY. Dans cet exemple, chaque sommet de $\mathcal{U}_2$ a un degré élevé, et à chaque fois qu'un de ses voisins arrive, il a une opportunité d'être choisi, ce qui signifie qu'ils ont tendance à être appariés tôt dans le processus, et plus généralement GREEDY semble favoriser l'appariement des sommets de degré élevé en premier. L'intuition dicterait l'opposé : les sommets de degré élevé devraient être appariés aussi tard que possible comme option de secours pour les sommets n'ayant pas d'autre option. Avec la randomisation corrélée, RANKING corrige en partie ce biais : au début de l'exécution, le voisin libre de rang le plus bas du sommet entrant $u_i \in \mathcal{U}_1$ se trouve toujours dans $\mathcal{U}_2$. Cependant, à mesure que les sommets de rang bas de $\mathcal{U}_2$ sont appariés, la probabilité que le voisin libre de rang le plus bas du sommet entrant $u_i \in \mathcal{U}_1$ soit son appariement dans l'appariement maximal augmente. Ainsi, en moyenne, RANKING apparie plus de sommets que GREEDY sur cet exemple.

Theorem 1.2.6. *Dans le cadre adversarial,*

$$\boldsymbol{C.R.}(\textit{RANKING}) \geq 1 - \frac{1}{e} \approx 0.63.$$

Il existe plusieurs preuves de ce résultat. Pour une preuve directe basée sur la correspondance entre les événements de "manqué" et les événements de "match", voir Birnbaum and Mathieu (2008). Pour une preuve basée sur l'analyse primal-dual, voir Devanur et al. (2013).

Dans l'article qui a introduit l'algorithme RANKING, Karp et al. (1990), l'auteur a démontré que la borne inférieure sur le **C.R.** est optimale avec l'exemple des graphes triangulaires supérieurs $(T_n)_{n\in\mathrm{N}}$, illustrés dans la Line 7, qui ont n sommets de chaque côté et l'ensemble des arêtes suivant :

$$\mathcal{E} = \{(u_i, v_j), (i,j) \in [n]^2 \text{ avec } j \leq i\}.$$

Il existe un matching de taille n dans ces graphes, et les auteurs ont prouvé que RANKING apparie $O\left((1-\frac{1}{e})n\right)$ sommets sur ces exemples. À l'aide de ces mêmes exemples, ils ont également prouvé le résultat plus fort suivant.

Proposition 1.2.7. *Pour tout algorithme **ALG**, dans le cadre adversarial :*

$$\boldsymbol{C.R.}(\boldsymbol{ALG}) \leq 1 - \frac{1}{e}.$$

Proof. Considérons un algorithme ALG quelconque (aléatoire ou non). Soit π_n une permutation quelconque sur $[n]$, $\pi_n \in S_n$, et Π_n la distribution uniforme sur S_n. Notons (T_n, π_n) le graphe où les lignes sont permutées par π et les colonnes arrivent dans l'ordre $1, \ldots, n$. Notons $\mathcal{D}$ la classe de tous les algorithmes déterministes. En utilisant le lemme de Yao :

$$\min_{\pi_n \in S_n} \mathrm{E}\left[\mathrm{ALG}\left((T_n, \pi_n)\right)\right] \leq \max_{A \in \mathcal{D}} \mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right].$$

La deuxième étape de la preuve consiste à relier les termes $\mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right]$ pour tout algorithme déterministe à la performance de GREEDY (appelé RANDOM dans Karp et al. (1990)) sur T_n :

$$\mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right] = \mathrm{E}\left[\textsc{greedy}\left(T_n\right)\right].$$

Cette égalité est obtenue par induction sur les deux propriétés suivantes :

1. Pour l'algorithme A sur (T_n, π_n), avec $\pi_n \sim \Pi_n$, ainsi que pour GREEDY sur T_n, s'il y a k lignes éligibles au moment t, elles ont toutes la même probabilité d'être n'importe quel ensemble de k lignes parmi les $n - t + 1$ premières lignes de T_n.

2. Pour chaque k, la probabilité qu'il y ait k lignes éligibles au moment k est la même pour GREEDY exécuté sur T_n que pour A exécuté sur (T_n, π_n).

Il reste à calculer $\mathrm{E}\left[\textsc{greedy}\left(T_n\right)\right]$. Définissons $x(t) = n - t + 1$ comme étant le nombre de colonnes restantes à l'itération t et $y(t)$ comme étant le nombre de voisins libres de v_t lors de son arrivée à l'itération t. Tant que $y(t) > 1$, $y(t)$ diminue de 2 si u_t est libre à l'itération t et n'est pas sélectionné. Avec la première propriété de la récurrence ci-dessus, cela se produit avec une probabilité de $\left(\frac{y(t)-1}{y(t)}\frac{y(t)}{x(t)}\right)$. Sinon, cela diminue de 1. Ainsi, nous obtenons :

$$\mathrm{E}[y(t+1) - y(t)] = -1 - \frac{y(t) - 1}{x(t)}.$$

Comme $x(t+1) - x(t) = -1$, nous pouvons réécrire :

$$\frac{\mathrm{E}[y(t+1) - y(t)]}{\mathrm{E}[x(t+1) - x(t)]} = 1 + \frac{y(t) - 1}{x(t)}.$$

Ainsi, par le théorème de Kuts, avec une probabilité tendant vers 1 lorsque $n \to +\infty$, $y(t) = g(t) + o(n)$ avec $g(t)$ la solution de l'équation différentielle suivante :

$$\frac{dg}{dx} = 1 + \frac{g(t) - 1}{x(t)}$$

avec la condition initiale $g(1) = n$. Nous obtenons $g(t) = 1 + x(t)\left(\frac{n-1}{n} - \ln\left(\frac{x(t)}{n}\right)\right)$ en résolvant l'EDO. Ainsi, $g(t) = 1$ pour $x(t) = \frac{n}{e} + o(n)$. Cela implique que la taille de l'appariement construit par GREEDY sur T_n est $\left(1 - \frac{1}{e}\right)n + o(n)$. □

Ordre Aléatoire Dans ce cas, le graphe peut toujours être n'importe quel graphe, mais les sommets arrivent dans un ordre aléatoire. Sans surprise, le **C.R.** de RANKING et de GREEDY est plus élevé dans ce contexte que dans le cadre adversarial.

Tout d'abord, GREEDY a un **C.R.** de $1 - \frac{1}{e}$. Nous avons déjà obtenu une borne supérieure lors de l'étude de GREEDY sur les graphes triangulaires supérieurs T_n. Quant à la borne inférieure, elle découle de la borne inférieure sur le **C.R.** de RANKING dans le cadre adversarial. En effet, GREEDY dans le cadre de l'ordre aléatoire émule RANKING et son classement aléatoire avec les rôles des deux côtés inversés.

Le **C.R.** exact de RANKING dans ce cadre est toujours une question ouverte. Une borne inférieure de 0.696 a été obtenue par une méthode assistée par ordinateur appelée "factor revealing LP" Karande et al. (2011); Mahdian and Yan (2011). Une borne supérieure de 0.75 est connue pour le graphe montré dans la Line 7, Karande et al. (2011).

Le meilleur **C.R.** réalisable dans ce cadre est également un problème ouvert. La borne inférieure la plus élevée connue est celle du **C.R.** de RANKING. La borne supérieure la plus basse est celle qui s'applique dans le cadre "Known i.i.d.", un modèle plus restreint détaillé dans le paragraphe suivant.

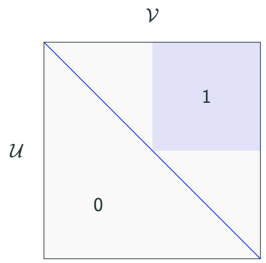


Figure 1.2: Un exemple difficile pour RANKING dans le cadre de l'ordre aléatoire

Known i.i.d. Les paramètres adversarial et ordre aléatoire peuvent sembler pessimistes car, en pratique, certaines informations sur le graphe peuvent être disponibles. Le cadre appelé "Known i.i.d.", introduit par Feldman et al. (2009), modélise cette connaissance par un ensemble de K types connus de sommets entrants. Les sommets d'un même type ont un ensemble prédéfini d'arêtes connues de l'algorithme, par exemple, tous les sommets de type 1 sont connectés à u_1, u_6 et u_7, tous les sommets de type 2 à u_2, u_3 et u_7, etc. Lorsque le type i a une arête avec un sommet $u \in \mathcal{U}$, nous notons cela $(u, i) \in \mathcal{E}$. À chaque itération, un type est tiré selon la distribution $\mathcal{D}$ sur $[K]$. Le type i est tiré avec une probabilité p_i, $\sum_{i\in[K]} p_i = 1$. La valeur de chaque p_i est également disponible pour l'algorithme. Nous surchargeons la notation et notons également $G \sim \mathcal{D}$ lorsque le graphe G est généré avec des types tirés selon la distribution $\mathcal{D}$.

Pour simplifier, nous supposons dans cette section que $p_i n \in \mathrm{N}$ pour chaque $i \in [K]$, ce qui signifie informellement que chaque type arrive un nombre entier de fois en espérance. Nous expliquons à la fin de la section comment supprimer cette hypothèse. Sans perte de généralité, nous pouvons même supposer que $p_i n = 1$, car nous pouvons dupliquer les types $p_i n$ fois sans modifier le processus de génération du graphe.

Les informations sur les types aident grandement à concevoir des algorithmes avec un taux de compétitivité plus élevé. À titre d'illustration, nous présentons l'algorithme SUGGESTED-MATCHING (Line 5), introduit dans Feldman et al. (2009), un algorithme naïf atteignant un **C.R.** de $1 - \frac{1}{e}$.

Grâce à l'hypothèse des taux d'arrivée unitaires, nous pouvons calculer un graphe espéré $\hat{G}$, dans lequel $\mathcal{V}$ est l'ensemble de tous les types. Ensuite, nous calculons un appariement maximum $\hat{\mathcal{M}}$ dans ce graphe espéré $\hat{G}$. Ensuite, dès la première arrivée d'un type i, le sommet est apparié à son appariement dans $\hat{\mathcal{M}}$ s'il en existe un. S'il arrive une deuxième fois ou plus, il reste non apparié.

Algorithm 5: SUGGESTED-MATCHING

```
1  Construire le graphe espéré Ĝ;
2  Calculer le matching maximum M̂ dans Ĝ;
3  for t = 1, .., |V| do
4  |  Tirer le type d'arrivée i_t ~ D;
5  end
6  if i_t arrive pour la première fois then
7  |  Apparier v_t avec son appariement dans M̂ s'il en existe un;
8  end
9  else
10 |  Le laisser non apparié
11 end
```

Remarque Trouver un matching maximum dans $\hat{G}$ revient à résoudre la programmation linéaire en nombres entiers suivante :

$$\begin{aligned} \text{maximiser} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{sous contraintes} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall i \in [K], \\ & x_{ui} \in \{0,1\}, \forall (u,i) \in \mathcal{E}. \end{aligned} \tag{Matching-ILP}$$

Pour les graphes bipartis, les contraintes d'intégralité peuvent être transformées en contraintes de positivité sans modifier la valeur du programme linéaire, Lovász and Plummer (2009a). Cela implique que $|\hat{\mathcal{M}}|$ est la valeur du programme linéaire suivant :

$$\begin{aligned} \text{maximiser} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{sous contraintes} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall i \in [K], \\ & x_{ui} \geq 0, \forall (u,i) \in \mathcal{E}. \end{aligned} \tag{Matching-LP}$$

Theorem 1.2.8. *Dans le cadre des arrivées* ***connues i.i.d.*** *avec des taux d'arrivée entiers :*

$$\boldsymbol{C.R.}(\textsc{suggested-matching}) \geq 1 - \frac{1}{e}.$$

Proof. Considérons un type i qui est apparié dans $\hat{\mathcal{M}}$. Aucun sommet de type i n'est apparié par SUGGESTED-MATCHING sauf si ce type n'arrive jamais. Cela se produit avec une probabilité $\left(1 - \frac{1}{n}\right)^n$. Ainsi, nous avons :

$$\begin{aligned} \mathrm{E}_{G\sim\mathcal{D}}[\textsc{suggested-matching}(G)] = & \sum_{i \text{ apparié dans } \hat{\mathcal{M}}} \mathrm{E}[\mathbb{1}\{i\text{arrive au moins une fois}\}] \\ = & |\hat{\mathcal{M}}| \left(1 - \frac{1}{n}\right)^n \\ \geq & |\hat{\mathcal{M}}| \left(1 - \frac{1}{e}\right). \end{aligned}$$

Il reste à prouver que $|\hat{\mathcal{M}}| \geq \mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)]$. Cela découle de la caractérisation des matchings maximums sur les graphes bipartis en tant que solutions d'une programmation linéaire en nombres entiers (ILP). Considérons une réalisation donnée $G \sim \mathcal{D}$, et notons r_i^G le nombre de fois où le type i est présent dans cette réalisation. En effet, trouver un matching

maximum dans le graphe G revient à trouver une solution du programme ILP suivant :

$$\begin{aligned} \text{maximiser} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{sous contraintes} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq r_i^G, \forall i \in [K], \\ & x_{ui} \in \mathbb{N}, \forall (u,i) \in \mathcal{E}. \end{aligned}$$

Désignons par $(x_{ui}^G)_{(u,i)\in\mathcal{U}\times[K]}$ la solution du programme ci-dessus et $\bar{x}_{ui} = \mathrm{E}_{G\sim\mathcal{D}}\left[x_{ui}^G\right]$. Nous avons :

$$\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)] = \sum_{(u,i)\in\mathcal{U}\times[K]} \bar{x}_{ui}.$$

Étant donné que $\mathrm{E}[r_i] = 1$, $(\bar{x}_{ui})_{(u,i)\in\mathcal{U}\times[K]}$ satisfait les contraintes du PL

Matching-LP, ce qui implique que $\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)]$ est dominé par sa valeur $|\hat{\mathcal{M}}|$. Ainsi, $\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)] \leq |\hat{\mathcal{M}}|$. □

L'algorithme Line 5 est sous-optimal car tout type qui arrive pour la deuxième fois ou plus est laissé sans appariement. Un long travail a été consacré à l'amélioration de cet algorithme avec plusieurs choix pour chaque type. Dans une première séquence, des algorithmes avec deux choix ont été proposés, augmentant le CR des algorithmes à 0.706 (sans l'hypothèse des taux d'arrivée entiers), Feldman et al. (2009),Manshadi et al. (2012),Jaillet and Lu (2014). La construction de ces deux choix reposait sur la solution d'un LP soigneusement conçu. Récemment seulement, un algorithme avec plus de deux choix a été proposé. Il repose également sur la solution d'un LP, avec des techniques supplémentaires de résolution des conflits en ligne (Online Contention Resolution Scheme, OCRS), Huang et al. (2022). Nous rapportons ici le CR le plus élevé connu à ce jour pour ce cadre.

Theorem 1.2.9 (Huang et al. (2022))**.** *Dans le cadre des arrivées* ***connues i.i.d.****, il existe un algorithme avec un CR de 0.711.*

Le meilleur CR réalisable dans ce cadre est encore un problème ouvert, mais la borne supérieure suivante sur le CR de tout algorithme est connue.

Proposition 1.2.10 (Manshadi et al. (2012))**.** *Il existe une instance du problème de matching stochastique en ligne pour laquelle aucun algorithme ne peut atteindre un rapport de compétitivité meilleur que 0.823.*

1.2.4 Algorithmes "learning-augmented"

Une récente ligne de recherche a proposé d'améliorer les algorithmes en ligne grâce à des prédictions d'entrée Mitzenmacher and Vassilvitskii (2020). Comme cela a été fait dans l'article introduisant le concept, illustrons ce principe sur le problème de location de skis, où une personne passe un nombre inconnu de jours D en vacances et doit décider chaque jour s'il loue des skis pour un prix p ou les achète pour B. Nous avons vu dans l'introduction de la section comment obtenir un CR de 2 lorsque D est inconnu. Imaginons maintenant que nous ayons accès à une prédiction $\hat{D}$ et définissons $\eta = |D - \hat{D}|$, la différence entre le nombre réel et prédit de jours de ski. L'application de l'algorithme hors ligne optimal avec $\hat{D}$ comme le nombre réel de jours de ski, c'est-à-dire l'achat de skis si $b\hat{D} > D$, peut conduire à des performances

arbitrairement médiocres. En effet, dans le cas où $D = 1$ et $\hat{D} = 2B$, le coût de l'algorithme en ligne est de B, tandis que le coût de l'algorithme hors ligne optimal est de b.

Maintenant, soit $\lambda \in [0, 1]$ un paramètre représentant la méfiance dans la prédiction (une valeur élevée pour λ correspond à une faible confiance dans la prédiction). Si $\hat{D} > B$, le skieur achète le jour $\lceil \lambda B/b \rceil$, et sinon, il achète le jour $\lceil B/b\lambda \rceil$.

Remarquons d'abord que si $\lambda = 1$, c'est-à-dire si la prédiction n'est pas du tout fiable, nous retrouvons l'algorithme pire des cas original. Une analyse cas par cas permet d'obtenir que le rapport de compétitivité de cet algorithme est borné par :

$$1 + \min\left(\frac{1}{\lambda}, \lambda + \frac{\eta}{(1-\lambda)OPT}\right).$$

Lorsque l'erreur de prédiction tend vers 0, le ratio de compétitivité n'est pas supérieur à $1 + \lambda < 2$. Même lorsque l'erreur est grande, le ratio n'est jamais pire que $1 + 1/\lambda$.

Plus généralement, les algorithmes améliorés par l'apprentissage visent à être :

1. **Consistants** : leurs performances devraient être proches de celles de l'algorithme hors ligne optimal lorsque la prédiction est bonne.

2. **Robustes** : même lorsque la prédiction est mauvaise, le rapport de compétitivité doit rester borné.

1.3 Plan et Contributions

1. **Apprentissage séquentiel**

   - **Matchings successifs**, Chapter 3 : Dans les jeux en ligne ou les plateformes de travail en ligne, certaines activités, telles que le travail d'équipe ou le jeu à 2 contre 2, impliquent d'apparier successivement des utilisateurs. Certaines applications de jeux en ligne apparient les joueurs entre eux (par exemple, Go, des quiz compétitifs et des dessins), mais les joueurs ne participent que s'ils le souhaitent tous les deux. Dans un modèle simple, le joueur i décide de participer avec une probabilité u_i, et les joueurs appariés i et j participent à un jeu avec une probabilité $u_i u_j$. Le nombre attendu de jeux joués à partir d'un appariement proposé m est alors $\sum_{(i,j)\in m} u_i u_j$. Les revenus de telles applications proviennent généralement des publicités affichées pendant les jeux, donc plus il y a de jeux joués, mieux c'est. Dans ce cadre, nous avons affiné certaines techniques de bandits existantes pour tirer parti de la structure spécifique du modèle, et nous avons proposé un nouvel algorithme résolvant le problème des matchings successifs.
   - **Systèmes de files d'attente**, Chapter 4 : Les systèmes de files d'attente séquentiels sont composés de files d'attente qui reçoivent des paquets à des débits différents. Ils envoient périodiquement des paquets aux serveurs, chacun d'eux traitant au plus un paquet à la fois. Les paquets non traités sont renvoyés aux files d'attente. Dans ce modèle, chaque file d'attente est un agent en concurrence pour le service des serveurs. Un aspect important est l'effet de report des paquets d'une ronde à l'autre. Il est traditionnel dans la littérature sur les jeux répétés de supposer que les rondes successives du jeu sont indépendantes. Cette hypothèse n'est pas réaliste dans de nombreuses applications pratiques, par exemple dans le cas du routage des paquets dans les réseaux informatiques. Les systèmes de files d'attente séquentiels ont été introduits dans Gaitonde and Tardos (2020a), en tant que cadre où cette

hypothèse est levée, dans le but de combler le fossé entre la théorie des jeux répétés et certaines applications. Dans notre projet, inspiré par les techniques des bandits, nous avons conçu un algorithme d'apprentissage décentralisé atteignant des performances similaires à celles de l'algorithme centralisé.

2. **Algorithmes séquentiels**

   - **Contraintes sur le modèle de matching séquentiel**, Chapter 5 : Plutôt que d'évaluer un algorithme dans le pire des cas, notre approche donne une loi générale pour le score d'un algorithme de matching en ligne dans des classes spécifiques de graphes aléatoires. Nous avons étudié l'algorithme glouton - le plus simple des algorithmes de matching séquentiel - dans une classe appelée modèle de configuration. Ce travail nous a permis d'illustrer théoriquement que l'algorithme glouton fonctionne beaucoup mieux que sa performance adversariale sur une grande classe de graphes. Nous avons également mis en évidence comment les caractéristiques des graphes influencent les performances de l'algorithme.
     Ce choix du modèle de configuration a été motivé par l'observation suivante : supposons que les campagnes peuvent être soit "intensives" (avec de nombreux utilisateurs éligibles), soit "sélectives/légères" (peu d'utilisateurs éligibles), avec une proportion empirique de, disons, 20%/80%. Que l'annonceur gère 100 campagnes en même temps ou 10 000, il aura toujours à peu près cette proportion de campagnes intensives par rapport aux campagnes légères. De même, certains utilisateurs sont plus précieux que d'autres et sont donc éligibles à plus de campagnes que d'autres ; la proportion de chaque type étant indépendante de la taille totale de la population.
   - **Introduction de features**, Chapter 6 : Un aspect clé manquant dans le modèle précédent est l'asymétrie dans l'association des utilisateurs aux campagnes. Dans la plupart des applications, les utilisateurs et les campagnes ont des caractéristiques inhérentes, et les premiers sont éligibles aux seconds s'ils sont "suffisamment similaires". Dans un travail ultérieur toujours en cours, nous modélisons ces interactions sous la forme d'un graphe aléatoire biparti géométrique : les caractéristiques des $2N$ sommets (N utilisateurs et N campagnes) sont tirées indépendamment dans un espace métrique et une arête est présente entre un nœud de campagne et un nœud d'utilisateur si la distance entre leurs caractéristiques est inférieure à c/N, où $c > 0$ est le paramètre du modèle.
   - **Allocation avec apprentissage**, Chapter 7 : Une approche populaire pour aller au-delà de l'analyse dans le pire des cas des algorithmes en ligne consiste à supposer l'existence de prédictions qui peuvent être exploitées pour améliorer les performances. Ces prédictions sont généralement fournies par des sources externes qui ne peuvent pas être entièrement fiables. Au lieu de cela, nous soutenons que des prédictions fiables peuvent être construites par des algorithmes pendant leur exécution. Nous étudions cette idée dans le contexte de scheduling statique avec des tailles de tâches exponentielles. En nous appuyant sur des idées issues de la littérature sur les bandits, nous avons conçu un algorithme d'apprentissage efficace dont les performances asymptotiques (en fonction du nombre de tâches) correspondent à celles du meilleur algorithme ayant accès à une prédiction des tailles des tâches.

1.4 Liste des Publications

Les chapitres de cette thèse sont basés soit sur des publications dans des conférences de machine learning, soit sur des travaux actuellement soumis, comme indiqué ci-dessous.

- Chapter 3: "Pure exploration and regret minimization in matching bandits", avec Jialin Yi, Clément Calauzenes, Vianney Perchet et Milan Vojnovic. ICML (2021).
- Chapter 4: "Decentralized learning in online queuing systems", avec Etienne Boursier et Vianney Perchet, NeurIPS (2021).
- Chapter 5: "Online matching in sparse random graphs: Non-asymptotic performances of greedy algorithm", avec Nathan Noiry et Vianney Perchet. NeurIPS (2021).
- Chapter 6: "Online Matching in Geometric Random Graphs", avec Vianney Perchet, Nathan Noiry, Laurent Ménard et Matthieu Lerasle. Working paper.
- Chapter 7: "On Preemption and Learning in Stochastic Scheduling", avec Nadav Merlis, Hugo Richard, Mathieu Molina, Corentin Odic et Vianney Perchet. ICML (2023).

L'auteur a également participé aux travaux publiés suivants, qui ne sont pas discutés dans cette thèse.

- "Robust estimation of discrete distributions under local differential privacy", avec Julien Chhor. ALT (2023).

Chapter 2

Introduction to Online Learning and Algorithms

Contents

In online scenarios, an option out of several is irrevocably picked at every iteration and the overall performance of the algorithm is measured at the end of the run. This models many real-life problems and is particularly relevant for digital applications.

The Multi-Armed Bandit framework is the focus of Section 2.1. It was introduced in the context of clinical trials Robbins (1952); Thompson (1933), and has received renewed attention recently due to its application to online recommendation systems. The Online Matching model, detailed in Section 2.2.2, is particularly relevant for digital advertising.

2.1 Online Learning

In this section, we introduce stochastic multi-armed bandits as well as the main results in that model. The section ends with a quick introduction to selected bandit algorithms. This will provide a ground for some of the work presented in this thesis. For a more thorough introduction to Multi-Armed Bandits, we refer the reader to the survey Lattimore and Szepesvári (2020).

2.1.1 Stochastic Multi-Armed Bandits and Regret

At the beginning of the run, an agent has K available options, called "arms". Each $i \in K$ of those options has an associated mean reward $\mu_i \in [0, 1]$. At every iteration t up to the horizon T, he picks an option $a(t) \in [K]$. We say the agent "pulls arm $a(t)$". He then receives and observes the noisy reward $r(t) = \mu_{a(t)} + \varepsilon_t$, where $(\varepsilon_t)_{t\in[T]}$ is an i.i.d. sequence drawn from a mean-zero 1-sub-gaussian distribution.

When the values of the means of the different arms, $(\mu_i)_{i\in[K]}$, are known, the optimal strategy is to pick the arm with the highest mean μ^* at every iteration. However, those means are unknown to the agent at the beginning of the run. The goal of the agent is then to design a strategy minimizing the difference between the expected reward received by following that strategy at the expected reward for the optimal strategy. This quantity is called the regret and is formally defined for a strategy π as follows:

$$R_\pi(T) := \mathrm{E}\left[\sum_{t\in T} \mu^* - \mu_{a(t)}\right].$$

The expectation is taken with respect to the agent's strategy.

On the one hand, at a given iteration, the agent may **exploit** his current knowledge to maximize the instantaneous reward in a greedy manner. On the other hand, at every iteration, the agent observes only a noisy version of the mean of the pulled arm. Thus, learning the arms mean requires pulling them, which entails that reasonable strategies incorporate a level of **exploration**.

In this exposition, we will focus on instance-dependent bounds. Those are upper and lower bounds on the performance of algorithms for fixed parameter values. We refer the reader interested in minimax bounds to the survey Lattimore and Szepesvári (2020).

2.1.2 Lower Bound

In this section, we expose a lower bound on the instance dependent regret achievable by "reasonable" algorithms, which are defined in the literature as the following class of asymptotically consistent algorithms.

Definition 2.1.1 (Asymptotically consistent algorithm)**.** *An algorithm π is asymptotically consistent if for every bandit instance, any $\alpha > 0$, $R_\pi(T) = o(T^\alpha)$.*

Note that for any asymptotically consistent algorithm, the accumulated reward is asymptotically in T of the same order as that of the optimal strategy which pulls the best arm at every iteration.

The following lower bound is derived through information-theoretic arguments that we will not detail here.

Theorem 2.1.2 (Lai and Robbins (1985))**.** *Consider a bandit instance where the noise $\varepsilon_t \sim \mathcal{N}(0, 1)$. Then any asymptotically consistent algorithm has the following lower bound on its*

asymptotic regret:

$$\liminf_{T\to+\infty} \frac{R(T)}{\log(T)} \geq \sum_{\mu_k<\mu^*} \frac{2}{\mu^* - \mu_k}.$$

Although this lower bound is asymptotic, it indicates that the upper bound on the regret of any reasonable algorithm cannot be smaller than $O\left(\sum_{\mu_k<\mu^*} \frac{\log(T)}{\mu^*-\mu_k}\right)$.

2.1.3 Classical Bandit Algorithms

A wide array of algorithms have been proposed to deal with the stochastic MAB setting. Here, we will focus on two strategies only, a first one in which exploration is done all at once at the beginning, and a second one in which exploration and exploitation are weaved in together. We provide brief pseudo-codes and upper bounds on the regret without proof. Some algorithms in this thesis build upon the ideas of the two strategies presented here.

In both algorithms presentations, we denote

- $N_k(t)$ the number of pulls of arm k up to time t,
- $\hat{\mu}_k(t) = \frac{1}{N_k(t)} \sum_{s=1}^{t-1} r(t)\mathbb{1}\{a(t) = k\}$, the empirical mean for arm k at iteration t,
- $L_k(t) = \hat{\mu}_k(t) - \sqrt{\frac{2\log(T)}{N_k(t)}}$, $U_k(t) = \hat{\mu}_k(t) + \sqrt{\frac{2\log(T)}{N_k(t)}}$, lower and upper bounds at iteration t on the true mean of arm k w.h.p..

Explore-Then-Commit Algorithm The simplest **Explore-Then-Commit** strategy consists in exploring uniformly all the arms, then committing to the arm with the highest empirical mean. The algorithm presented in Algorithm 6 is a slightly refined version of that principle, also called Successive-Elimination. The algorithm maintains a set of active arm $\mathcal{A}$, that are candidates to be the arm with the highest mean. Arms from this set are sampled uniformly. When an arm is deemed sub-optimal w.h.p., it is definitively eliminated from this set and thus never pulled again.

Algorithm 6: Explore-Then-Commit

input : set of arms $[K]$ and horizon T

$\mathcal{A} = [K]$;
while $|\mathcal{A}| > 1$ **do**
  Pull once every arm in $\mathcal{A}$;
  $\forall k \in [K]$ s.t. $U_k(t) \leq \max_{i\in\mathcal{A}} L_i(t)$, remove k from $\mathcal{A}$;
end
Pull remaining arm in $\mathcal{A}$ until the end;

Theorem 2.1.3 (Perchet and Rigollet (2013))**.** *The regret of Algorithm 6 with known horizon T is upper bounded as:*

$$R(T) = O\left(\sum_{\mu_k<\mu^*} \frac{\log(T)}{\mu^* - \mu_k}\right).$$

As shown in Section 2.1.2, the bound on the regret obtained for this simple algorithm is only a constant larger than the asymptotic lower bound on the instance-dependent regret. However, it is known that the regret of any Explore-Then-Commit strategy is a multiplicative constant sub-optimal, Garivier and Kaufmann (2016). Still, the simplicity of the algorithm is advantageous and can prove useful in more complex settings.

Upper-Confidence-Bound Algorithm The Upper-Confidence-Bound (UCB) Algorithm, Auer et al. (2002a), in an example of fully adaptive bandit algorithm. At every iteration, the index of each arm $U_k(t)$ is an upper bound on the true mean of the arm w.h.p., and the arm with the highest index is pulled.

Algorithm 7: Upper-Confidence-Bound

input : set of arms $[K]$ and horizon T
for $t = 1, \dots, T$ **do**
  Pull $\arg\max_{k \in [K]} U_k(t)$;
end

Theorem 2.1.4 (Auer et al. (2002a))**.** *The regret of Algorithm 7 with known horizon T is upper bounded as:*

$$R(T) \leq \sum_{\mu_k < \mu^*} \frac{16 \log(T)}{\mu^* - \mu_k} + 3(\mu^* - \mu_k).$$

Again, the UCB algorithm asymptotically reaches the lower bound Section 2.1.2 up to a multiplicative constant. Even better, for some distributions, the UCB algorithm with a slightly modified version of the upper bounds is asymptotically optimal, Cappé et al. (2013).

Thus, Algorithm 7 performs stricly better than Algorithm 6, but it is sometimes harder to generalize in more complex settings.

2.2 Online Algorithms

By definition, an online algorithm is any algorithm that sequentially processes its input without having access to the whole sequence at the beginning of the run.

A classical example is the ski rental problem. A person arrives on holiday and can decide every day to rent skis for one day for a price p or buy them for a price B. If the person knows they are going to spend D days on holiday, the optimal decision is obvious: buy skis on the first day if $B < pD$, rent them every day otherwise. When D is unknown, the situation is trickier, but there is still a way to ensure one does not pay more than twice the optimal price: rent skis for the first $\lfloor \frac{B}{p} \rfloor$ days, buy them on the following one.

Generally, as in the previous example, online algorithms are evaluated in comparison with the optimal algorithms aware of the full input sequence.

The next section is dedicated to the definition of the Competitive ratio (**C.R.**), a classical measure of performance for online algorithms. Then, we give more detail on the Online Matching setting, which is one of the main focus points of this thesis.

In this section and the remainder of the thesis, an online algorithm is an algorithm for which the sequence is revealed sequentially (the one that does not have access to D in the ski rental example), and an offline algorithm is an algorithm that has access to the full input sequence (the one that knows D in the same example).

2.2.1 Competitive Ratio

We could consider both cost minimizing algorithms or reward maximizing ones. As in the Online Matching problem we deal with reward maximizing ones, this section formulates everything in

terms of reward maximizing algorithms, with positive rewards.

We denote as ALG(G) the score of an algorithm over input G and OPT the optimal offline algorithm.

Definition 2.2.1 (Competitive Ratio (**C.R.**)). *A reward maximizing algorithm ALG reaches competitive ratio α over input class $\mathcal{G}$ if there exists a constant c s.t. for any $G \in \mathcal{G}$*

$$\mathrm{E}[ALG(G)] \geq \alpha \mathrm{E}[OPT(G)] + c,$$

where the expectations are taken with respect to the possible randomness in the algorithms and the input G.

Remark: For a cost minimizing algorithm, the min would be replaced by a max and the inequality would be reversed.

Note that the **C.R.** is bounded between 0 and 1, the higher the better. Two elements impact its value: the algorithm and the class of input $\mathcal{G}$ over which the algorithm is evaluated. If more restrictive assumptions are made on $\mathcal{G}$, then it is easier to get algorithms with a high competitive ratio.

In the ski rental example, an input G is a set of three parameters p, B, D. If those parameters can take any positive values, which corresponds to the largest input class possible, 2 is the smallest **C.R.** achievable. However, if we assume that those parameters are drawn from a known distribution, it may be possible to get a smaller **C.R.**.

Remark: By Theorem 2.1.1, any asymptotically consistent bandit algorithm has an asymptotic **C.R.** of 1.

2.2.2 Online Matching in Bipartite Graphs

Offline Matching. We start by detailing the offline version of the matching problem before moving on to its online counterpart. This thesis focuses on matchings in bipartite graphs.

Definition 2.2.2 (Bipartite Graph). *A bipartite graph $G = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$ is a graph with two sets of vertices U and V and set of edges E such that no vertices within the same set are adjacent.*

Throughout this manuscript, a matching is defined as follows.

Definition 2.2.3 (Matching). *A matching M on graph G is a subset of $\mathcal{E}$ s.t. each vertex of G is the endpoint of at most one edge in M. If a vertex $u \in \mathcal{U} \cup \mathcal{V}$ is the endpoint of an edge in M, we say that u is **matched**, otherwise it is **free** or **unmatched**. A matching of maximal cardinality is a **maximum matching**.*

The problem of finding a maximum matching in a bipartite graph can be solved using the Hopcroft-Karp algorithm, with running time $O\left(|\mathcal{U} \cup \mathcal{V}|^{1/2}|\mathcal{E}|\right)$. In the case of a weighted matching, where each edge $(u, v) \in \mathcal{E}$ is associated with a weight $w_{u,v}$, and the goal is to find a matching maximizing the sum of the selected edges, the Hungarian algorithm can be used, with running time $O\left(|\mathcal{U} \cup \mathcal{V}|^3\right)$.

Online Matching. The online version of the problem can be stated as follows:

1. at the beginning of the run, all vertices of the "offline" side $\mathcal{U}$ are available,
2. at every iteration, a vertex $v \in \mathcal{V}$ is revealed, along with its edges,

3. v can be matched to one of its free neighbors, and the chosen match (if there is one) is irrevocable.

This theoretical setting is particularly well suited for online advertising: $\mathcal{U}$ is the set of campaigns/ads that an advertiser can run and users $v_1, v_2, \ldots, v_T$ arrive sequentially (Manshadi et al., 2012; Mehta, 2012). Some of them are eligible for a large subset of campaigns, while others are not (usually based on their attributes/features, such as geographic localization, browsing history, or any other relevant information). The objective of an advertiser (in this over-simplified model) is to maximize the number of displayed ads. In practice, campaigns/ads are not displayed only once but have a maximal budget of impressions (say, a specific ad can be displayed only 10.000 times each day). A possible trick consists of duplicating the vertices of $\mathcal{U}$ as many times as the budget.

The online matching version is obviously harder than the offline one, notably one can not obtain a maximum matching in most cases. We detail in the next section, depending on the assumptions on the class of graphs considered, the values of the attainable **C.R.** for online matching algorithms.

2.2.3 Classical Online Matching Settings and Algorithms

In this section, we will detail the classical hypothesis made on the considered class of graphs $\mathcal{G}$, as well as the algorithms proposed in the literature in each context.

Adversarial setting In the adversarial setting, no restriction is put on the class of input $\mathcal{G}$. The graph may be any graph and the vertices arrive in any order. Hence the name adversarial, as looking back at Theorem 2.2.1, we see that in that setting, the **C.R.** of an algorithm is evaluated on the graph on which it has the poorest performance.

Perhaps surprisingly, even in that hard setting, a guarantee of 1/2 on the **C.R.** is easily obtained with

Algorithm 8: GREEDY Algorithm

```
for t = 1, ..., |V| do
    Match v_t to a free neighbor chosen uniformly at random;
end
```

For consistency with the rest of the thesis, we define GREEDY as the algorithm which picks the match of the incoming vertex uniformly at random, but the following result holds for all algorithms picking any match as soon as one is available.

Theorem 2.2.4. *In the adversarial setting,*

$$\textbf{\textit{C.R.}}(\textsc{greedy}) \geq \frac{1}{2}.$$

Proof. Consider the event that GREEDY fails to match a vertex v_t which is matched in the maximum matching of the final graph. This can only happen if the match of v_t in the optimal matching was previously matched to another vertex. Thus, for any "miss" event (GREEDY fails to match a vertex that is matched in the maximum matching), there is at list one "match" event (GREEDY matches a vertex). This implies that there are at most twice the number of matched vertices in the maximum matching as in the matching constructed by GREEDY, hence the lower bound of 1/2 on the **C.R.**. □

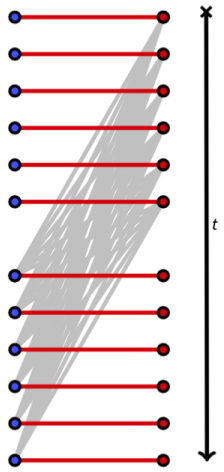


Figure 2.1: A difficult instance for GREEDY

This lower bound is tight.

Proposition 2.2.5. *In the adversarial setting,*

$$\textbf{\textit{C.R.}}(\textsc{greedy}) = \frac{1}{2}.$$

Proof. Consider the following family of graphs $(\mathcal{G}_n)_{n\in\mathrm{N}}$, illustrated in Line 3. There $2n$ vertices on either side of $\mathcal{G}_n$, which are split in half, $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$ and $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, with $\mathcal{U}_1 = \{u_1, \dots, u_n\}$, $\mathcal{U}_2 = \{u_{n+1}, \dots, u_{2n}\}$, similarly for $\mathcal{V}$. We have the following set of edges:

$$\mathcal{E} = \{(u_i, v_i), \forall i \in [2n], (u_{n+i}, v_j)\forall (i,j) \in [n]^2\}.$$

In each $\mathcal{G}_n$, there exists a matching of cardinality $2n$, $\mathcal{M} = \{(u_i, v_i), \forall i \in [2n]\}$. Let us analyse the run of GREEDY. First we observe that any vertex $u_i \in \mathcal{V}_1$ will pick its match in $\mathcal{V}_2$ with probability greater than $\frac{n-i+1}{n-i+2}$, as it has only one neighbor in $\mathcal{U}_1$, and at iteration i, at most $i-1$ vertices have already been matched in $\mathcal{U}_2$. This implies that in expectation, $n - o(n)$ vertices of $\mathcal{V}_1$ are matched with vertex $u_j \in \mathcal{U}_2$. For all of those $u_j \in \mathcal{U}_2$, there match in the maximum matching remains unmatched, and we thus have:

$$\lim_{n\to+\infty} \frac{\mathrm{E}[\textsc{greedy}(\mathcal{G}_n)]}{2n} = \frac{1}{2}.$$

□

To get beyond this **C.R.** of 1/2, a different algorithm is needed. The RANKING algorithm Karp et al. (1990), uses what is informally referred to as "correlated randomness" and has a higher **C.R.** in the adversarial setting.

It proceeds as follows: at the beginning of the run, a random permutation π is drawn, and vertex $u_i \in \mathcal{U}$ is assigned rank $\pi(i)$. When a vertex $v_t \in \mathcal{V}$ arrives, it is matched to its lowest ranked free neighbor. We say there is "correlated randomness", since the choice of neighbor for v_t, albeit still random, is correlated from iteration to iteration.

Algorithm 9: RANKING Algorithm

```
1 Draw a random permutation π;
2 for i = 1, .., |U| do
3 |   Assign to u_i rank π(i);
4 end
5 for t = 1, .., |V| do
6 |   Match v_t to its lowest ranked free neighbor;
7 end
```

We can get some intuition on why RANKING has a higher **C.R.** than GREEDY by looking back at the difficult instance for GREEDY. In that instance, each vertex of $\mathcal{U}_2$ has a high degree, and every time one of its neighbors arrives, it has an opportunity to be picked, which means they tend to get matched early on in the process, and more generally GREEDY seems biased towards matching high degree vertices early. Intuition would dictate the opposite: high degrees vertices should be matched as late as possible as a fallback option for vertices with no other option. With correlated randomness, RANKING partly corrects this bias: at the beginning of the

run, the lowest ranked free neighbor of the incoming vertex $u_i \in \mathcal{U}_1$ is still in $\mathcal{U}_2$. However, as the low ranked vertices of $\mathcal{U}_2$ get matched, the probability that the lowest rank free neighbor of the incoming vertex $u_i \in \mathcal{U}_1$ is its match in the maximum matching increases. Thus, in expectation RANKING matches more vertices than GREEDY on that instance.

Theorem 2.2.6. *In the adversarial setting,*

$$\boldsymbol{C.R.}(\textit{RANKING}) \geq 1 - \frac{1}{e} \approx 0.63.$$

There exist multiple proofs of this result. For a direct proof based on the mapping of "miss" events to "match" events, see Birnbaum and Mathieu (2008). For a proof based on Primal-Dual Analysis see Devanur et al. (2013).

In the article that introduced the RANKING algorithm, Karp et al. (1990), the author proved that the lower bound on the **C.R.** is tight with the example of the upper triangular graphs $(T_n)_{n\in\mathbb{N}}$,illustrated in Line 7, that has n vertices on both sides and set of edges:

$$\mathcal{E} = \{(u_i, v, j), (i, j) \in [n]^2 \text{ s.t.} j \leq i\}.$$

There is a matching of size n in these graphs, and the authors prove that RANKING matches $O\left((1 - \frac{1}{e})n\right)$ vertices on those instances. Through those same instances, they also proved the following stronger result.

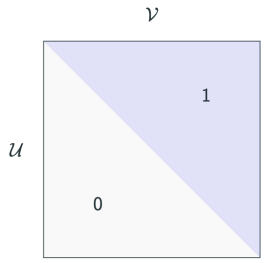


Figure 2.2: A difficult instance for RANKING

Proposition 2.2.7. *For any algorithm **ALG**, in the adversarial setting:*

$$\boldsymbol{C.R.}(\boldsymbol{ALG}) \leq 1 - \frac{1}{e}.$$

Proof. Consider any (random or not) algorithm ALG. Let π_n be any permutation over $[n]$, $\pi_n \in S_n$, and Π_n the uniform distribution over S_n. We note (T_n, π_n) the graph where the rows are permuted by π and the columns arrive in order $1, \ldots, n$. Note $\mathcal{D}$ the class of all deterministic algorithms. By Yao's Lemma:

$$\min_{\pi_n \in S_n} \mathrm{E}\left[\mathrm{ALG}\left((T_n, \pi_n)\right)\right] \leq \max_{A \in \mathcal{D}} \mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right].$$

The second step of the proof is to relate the $\mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right]$ for any deterministic algorithm with the performance of GREEDY (called RANDOM in Karp et al. (1990)) on T_n:

$$\mathrm{E}_{\pi_n \sim \Pi_n}\left[\mathrm{A}\left((T_n, \pi_n)\right)\right] = \mathrm{E}\left[\textsc{greedy}\left(T_n\right)\right].$$

This equality is obtained by induction on the two following properties:

1. For algorithm A on (T_n, π_n), with $\pi_n \sim \Pi_n$, as well as for GREEDY on T_n, if there are k eligible rows at time t, then they are equally likely to be any set of k rows from among the first $n - t + 1$ rows of T_n.

2. For each k, the probability that there are k eligible rows at time k is the same for GREEDY run on T_n as it is for A run on (T_n, π_n).

It remains to compute $\mathrm{E}\left[\textsc{greedy}\left(T_n\right)\right]$. Define $x(t) = n - t + 1$ the number of columns remaining at iteration t and $y(t)$ the number of free neighbors of v_t upon its arrival at iteration

t. As long as $y(t) > 1$, it decreases by 2 if u_t is free at iteration t and does not get picked. With the first property in the above recursion, this happens with probability $\left(\frac{y(t)-1}{y(t)}\frac{y(t)}{x(t)}\right)$. Otherwise, it decreases by 1. Thus, we obtain:

$$\mathrm{E}[y(t+1) - y(t)] = -1 - \frac{y(t)-1}{x(t)}.$$

As $x(t+1) - x(t) = -1$, we can rewrite:

$$\frac{\mathrm{E}[y(t+1) - y(t)]}{\mathrm{E}[x(t+1) - x(t)]} = 1 + \frac{y(t)-1}{x(t)}.$$

Thus, by Kuts theorem, with probability tending to 1 as $n \to +\infty$, $y(t) = g(t) + o(n)$ with $g(t)$ the solution of the following differential equation:

$$\frac{dg}{dx} = 1 + \frac{g(t)-1}{x(t)}$$

with initial condition $g(1) = n$. Solving the ODE, we get $g(t) = 1 + x(t)\left(\frac{n-1}{n} - \ln\left(\frac{x(t)}{n}\right)\right)$. Thus, $g(t) = 1$ for $x(t) = \frac{n}{e} + o(n)$. This implies that the size of the matching constructed by GREEDY on T_n is $\left(1 - \frac{1}{e}\right)n + o(n)$.

□

Random Order In this setting, the graph can still be any graph, but the vertices arrive in random order. Unsurprisingly, the **C.R.** of both RANKING and GREEDY are higher in that setting than in the adversarial one.

First, GREEDY has a **C.R.** of $1 - \frac{1}{e}$. We already obtained in upper bound when studying GREEDY on the upper triangular graphs T_n. As for the lower bound, it is a consequence of the lower bound on the **C.R.** of RANKING in the adversarial setting. Indeed, GREEDY in the random order setting emulates RANKING and its random ranking with the roles of the two sides switched.

The exact **C.R.** of RANKING in that setting is still an open question. A lower bound of 0.696 was obtained through a computer assisted method called "factor revealing LP", Karande et al. (2011); Mahdian and Yan (2011). An upper bound of 0.75 is known for the graph shown in Line 7, Karande et al. (2011).

Th best **C.R.** achievable in that setting is also an open problem. The highest known lower bound is the one on the **C.R.** of RANKING. The lowest upper bound is one that holds in the Known i.i.d. setting, a more restricted model detailed in the next paragraph.

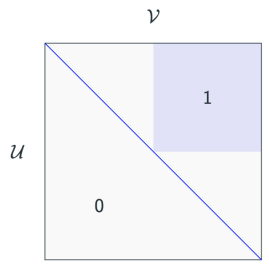


Figure 2.3: A difficult instance for RANKING in the Random Order Setting

Known i.i.d. The adversarial and random order settings may seem pessimistic as in practice, some information about the graph can be available. The so-called **Known i.i.d.** setting, introduced by Feldman et al. (2009), models that knowledge by a set of K known types of incoming vertices. Vertices of one type have a predefined set of edges known to the algorithm, for instance, all vertices of type 1 are connected to u_1, u_6 and u_7, all vertices of type 2 to u_2, u_3 and u_7, etc... When type i has an edge with a vertex $u \in \mathcal{U}$, we note this $(u, i) \in \mathcal{E}$. At every

iteration, a type is drawn from distribution $\mathcal{D}$ on $[K]$. Type i is drawn with probability p_i, $\sum_{i\in[K]} p_i = 1$. The value of each p_i is also available to the algorithm. We overload the notation and also note $G \sim \mathcal{D}$ when the graph G is generated with types drawn from distribution $\mathcal{D}$.

For simplicity, we assume in that section $p_i n \in \mathrm{N}$ for each $i \in [K]$, which informally means that each type arrives an integral number of times in expectation. We detail at the end of the section how to remove this hypothesis. W.l.o.g., we can even assume $p_i n = 1$, as we can duplicate types $p_i n$ times without modifying the graph generating process.

The information of the types greatly helps in designing algorithms with a higher competitive ratio. As a mean of illustration, we expose the SUGGESTED-MATCHING algorithm (Line 10), introduced in Feldman et al. (2009), a naive algorithm reaching a **C.R.** of $1 - \frac{1}{e}$.

Thanks to the unitary arrival rates hypothesis, we can compute an expected graph $\hat{G}$, in which $\mathcal{V}$ is the set of all types. Next, compute a maximum matching $\hat{\mathcal{M}}$ in that expected graph $\hat{G}$. Then upon the first arrival of a type i, the vertex gets matched to its match in $\hat{\mathcal{M}}$ if there is one. If it arrives a second time or more, it remains unmatched.

Algorithm 10: SUGGESTED-MATCHING

```
1  Construct expected graph Ĝ;
2  Compute maximum matching M̂ in Ĝ;
3  for t = 1, .., |V| do
4  |  Draw incoming type i_t ~ D;
5  end
6  if i_t arrives for the first time then
7  |  Match v_t to its match in M̂ if there is one;
8  end
9  else
10 |  Leave it unmatched
11 end
```

Remark Finding a maximum matching in $\hat{G}$ is equivalent to solving the following ILP:

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{s.t.} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall i \in [K], \\ & x_{ui} \in \{0,1\}, \forall (u,i) \in \mathcal{E}. \end{aligned} \tag{Matching-ILP}$$

For bipartite graphs, the integrality constraints may be turned into positivity constraints without modifying the value of the LP, Lovász and Plummer (2009a). This implies that $|\hat{\mathcal{M}}|$ is

the value of the following LP:

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{s.t.} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall i \in [K], \\ & x_{ui} \geq 0, \forall (u,i) \in \mathcal{E}. \end{aligned} \tag{Matching-LP}$$

Theorem 2.2.8. *In the **Known i.i.d.** setting with integral arrival rates:*

$$\boldsymbol{C.R.}(\textsc{suggested-matching}) \geq 1 - \frac{1}{e}.$$

Proof. Consider a type i that is matched in $\hat{\mathcal{M}}$. No vertex of type i is matched by SUGGESTED-MATCHING only if this type never arrives. This happens with probability $\left(1 - \frac{1}{n}\right)^n$. Thus we have:

$$\begin{aligned} \mathrm{E}_{G\sim\mathcal{D}}[\textsc{suggested-matching}(G)] = & \sum_{i \text{ matched in } \hat{\mathcal{M}}} \mathrm{E}[\mathbb{1}\{i\text{arrives at least once}\}] \\ = & |\hat{\mathcal{M}}| \left(1 - \frac{1}{n}\right)^n \\ \geq & |\hat{\mathcal{M}}| \left(1 - \frac{1}{e}\right). \end{aligned}$$

It remains to prove that $|\hat{\mathcal{M}}| \geq \mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)]$. This stems from the characterisation of maximum matchings on bipartite graphs as solutions of an ILP. Consider a given realisation $G \sim \mathcal{D}$, and note r_i^G the number of times type i is present in that realisation. Indeed, finding a maximum matching in the graph G is equivalent to finding a solution of the following ILP:

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,i)\in\mathcal{U}\times[K]} x_{ui}, \\ \text{s.t.} \quad & \sum_{i:(u,i)\in\mathcal{E}} x_{ui} \leq 1, \forall u \in \mathcal{U}, \\ & \sum_{u:(u,i)\in\mathcal{E}} x_{ui} \leq r_i^G, \forall i \in [K], \\ & x_{ui} \in \mathbb{N}, \forall (u,i) \in \mathcal{E}. \end{aligned}$$

Denote as $(x_{ui}^G)_{(u,i)\in\mathcal{U}\times[K]}$ the solution of the above program and $\bar{x}_{ui} = \mathrm{E}_{G\sim\mathcal{D}}\left[x_{ui}^G\right]$. We have:

$$\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)] = \sum_{(u,i)\in\mathcal{U}\times[K]} \bar{x}_{ui}.$$

Since $\mathrm{E}[r_i] = 1$, $(\bar{x}_{ui})_{(u,i)\in\mathcal{U}\times[K]}$ satisfies the constraints of the LP Matching-LP, whihch implies that $\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)]$ is dominated by its value $|\hat{\mathcal{M}}|$. Thus $\mathrm{E}_{G\sim\mathcal{D}}[\textsc{opt}(G)] \leq |\hat{\mathcal{M}}|$. □

Line 10 is sub-optimal as any type arriving for the second time or more is left unmatched. A long line of work has been dedicated to improving this algorithm with multiple choices for each type. In a first sequence, algorithms with two choices where proposed, raising the

CR of algorithms to 0.706 (without the integral arrival rates hypothesis), Feldman et al. (2009),Manshadi et al. (2012),Jaillet and Lu (2014). The construction of those two choices relied on the solution of a carefully designed LP. Only recently, an algorithm with more than two choices was proposed. It also relies on the solution of a LP, with additional Online Contention resolution Scheme (OCS) techniques, Huang et al. (2022). We report here the highest CR known for that setting to date.

Theorem 2.2.9. *In the **Known i.i.d.** setting, there exists an algorithm with a CR of 0.711.*

The best achievable CR in that setting is still an open problem, but the following upper bound on the CR of any algorithm is known.

Proposition 2.2.10 (Manshadi et al. (2012))**.** *There is an instance of the online stochastic matching problem for which no algorithm can achieve a competitive ratio better than 0.823.*

2.2.4 Learning Augmented Algorithms

A recent line of work has proposed to improve online algorithms through predicted input Mitzenmacher and Vassilvitskii (2020). As was done in the article introducing the concept, let us illustrate this principle on the ski rental problem, where a person spends an unknown number of days D on holiday and has to decide each day whether to rent skis for a price p or buy them for B. We saw in the introduction of the section how to achieve a CR of 2 when D was unknown. Let us now imagine that we have access to a prediction $\hat{D}$, and define $\eta = |D - \hat{D}|$, the difference between the true and predicted number of skied days. Applying the optimal offline algorithm with $\hat{D}$ as the true number of skied days, i.e. buying skis if $b\hat{D} > D$, can lead to arbitrarily poor performances. Indeed, in the case $D = 1$ and $\hat{D} = 2B$, the cost of the online algorithm is B, whereas the cost of the optimal offline one is b.

Now, let $\lambda \in [0, 1]$ be a parameter representing the mistrust in the prediction (a high value for lambda λ corresponds with a low trust in the prediction). If $\hat{D} > B$, the skier buys on day $\lceil \lambda B/b \rceil$, and otherwise, he buys on day $\lceil B/b\lambda \rceil$.

Note first that if $\lambda = 1$, i.e. the prediction is not trusted at all and we recover the original worse case algorithm. Through a case by case analysis one obtains that the competitive ratio of that algorithm is bounded by:

$$1 + \min\left(\frac{1}{\lambda}, \lambda + \frac{\eta}{(1-\lambda)OPT}\right).$$

As the prediction error drops to 0, the competitive ratio is no larger than $1 + \lambda < 2$. Even when the error is large the ratio is never worse than $1 + 1/\lambda$.

More generally, learning augmented algorithms aim to be:

1. **Consistent**: their performance should be close to that of the offline optimum algorithm when the prediction is good.

2. **Robust**: even when the prediction is bad, the competitive ratio should remain bounded.

2.3 Outline and Contributions

1. **Online Learning**

   - **Successive Matchings**, Chapter 3: In online gaming or online labor platforms, some activities, such as teamwork or 2v2 gameplay, involve successively matching users

together. Some online gaming apps match players together (e.g. Go, competitive quizzes, and drawings) but players will participate only if they both want to. In a simple model, player i decides to participate with probability u_i, and matched players i and j participate in a game with probability $u_i u_j$. The expected number of played games from a proposed matching m is then $\sum_{(i,j)\in m} u_i u_j$. The revenue of such apps typically comes from ads displayed during games, so the more games played the better. In that setting, we refined some existing bandit techniques to leverage the specific structure of the model, and proposed a novel algorithm solving the successive matching problem.

- **Queuing systems**, Chapter 4: Online queuing systems are composed of queues that receive packets at different rates. Repeatedly, they send packets to servers, each of them treating only at most one packet at a time. The untreated packets are sent back to the queues. In this model, each queue is an agent, competing for the service of servers. An important aspect is the carryover effect of the packets from round to round. It is traditional in the repeated game literature to assume that the successive rounds of the game are independent. This assumption is not realistic in many practical applications, for instance in the case of packet routing in computer networks. Online queuing systems were introduced in Gaitonde and Tardos (2020a), as a setting where that assumption is removed, in an effort to bridge the gap between the theory of repeated games and some applications. In our project, inspired by bandit techniques, we designed a decentralized learning algorithm reaching performances similar to those of the centralized one.

2. **Online Algorithm**

- **Constraining the online matching model**, Chapter 5: Rather than evaluating an algorithm in the worst case, our approach gives a general law for the score of an online matching algorithm in specific classes of random graphs. We studied the greedy algorithm - the simplest sequential matching algorithm - in a class called the configuration model. This work allowed us to illustrate theoretically that the greedy algorithm performs much better than its adversarial performance on a large class of graphs. We have also highlighted how the characteristics of the graphs affect the performance of the algorithm.

  This choice of the configuration model was motivated by the following observation: assume that campaigns can either be "intensive" (with many eligible users) or "selective/light" (few eligible users), with an empirical proportion of, say, 20%/80%. Then whether an advertiser handles 100 campaigns at the same time or 10.000, it will always have roughly this proportion of intensive vs. light campaigns. Similarly, some users are more valuable than others, and are thus eligible for more campaigns than others; the proportion of each type being independent of the total population size.

- **Introducing features**,Chapter 6: A key aspect missing in the previous model is the asymmetry in the association of users to campaigns. In most applications, both users and campaigns have inherent features, and the former is eligible to the latter if they are "similar enough". In a subsequent still ongoing work, we model these interactions as a bipartite geometric random graph: the features of the $2N$ vertices (N users and N campaigns) are drawn independently in a metric space and an edge is present between a campaign and a user node if the distance between their features is smaller than c/N, where $c > 0$ is the parameter of the model.

- **Allocating while learning**,Chapter 7: A popular approach to go beyond the worst-case analysis of online algorithms is to assume the existence of predictions that can be leveraged to improve performances. Those predictions are usually given by some external sources that cannot be fully trusted. Instead, we argue that reliable predictions can be built by algorithms, while they run. We investigate this idea in the context of static scheduling with exponential job sizes. Leveraging ideas from the bandit literature, we designed an efficient learning algorithm with asymptotic (in the number of jobs) performances matching that of the best algorithm with access to a prediction of the sizes of the jobs.

2.4 List of Publications

The chapters of this thesis are based either on publications in proceedings of machine learning conferences or works currently submitted, as listed below.

- Chapter 3: "Pure exploration and regret minimization in matching bandits", with Jialin Yi, Clément Calauzenes, Vianney Perchet and Milan Vojnovic. ICML (2021).
- Chapter 4: "Decentralized learning in online queuing systems", with Etienne Boursier and Vianney Perchet, NeurIPS (2021).
- Chapter 5: "Online matching in sparse random graphs: Non-asymptotic performances of greedy algorithm", with Nathan Noiry and Vianney Perchet. NeurIPS (2021).
- Chapter 6: "Online Matching in Geometric Random Graphs", with Vianney Perchet, Nathan Noiry, Laurent Ménard and Matthieu Lerasle. Working paper.
- Chapter 7: "On Preemption and Learning in Stochastic Scheduling", with Nadav Merlis, Hugo Richard, Mathieu Molina, Corentin Odic and Vianney Perchet. ICML (2023).

The author also participated in the following published work, which is not discussed in this thesis.

- "Robust estimation of discrete distributions under local differential privacy", with Julien Chhor. ALT (2023).

Part I

Online Learning

Chapter 3

Pure Exploration and Regret Minimization in Matching Bandits

Finding an optimal matching in a weighted graph is a standard combinatorial problem. We consider its semi-bandit version where either a pair or a full matching is sampled sequentially. We prove that it is possible to leverage a rank-1 assumption on the adjacency matrix to reduce the sample complexity and the regret of off-the-shelf algorithms up to reaching a linear dependency in the number of vertices (up to poly log terms).

Contents

3.1 Introduction

Finding *matchings* in graphs, i.e., subsets of edges without common vertices, is a long standing problem Lovász and Plummer (2009a) with many different applications in economics Roth et al. (2004), operations research Wheaton (1990), and machine learning Mehta (2012). We consider here its sequential variant where at each time step t, an agent chooses a matching m_t of some graph, defined by its unknown weighted adjacency matrix W (with bounded elements in $(0, 1)$), and observes noisy evaluations of the chosen entries $\{X_{i,j,t} : (i, j) \in m_t\}$, with $\mathbb{E}[X_{i,j,t}] = W_{i,j}$. This problem obviously falls in the realm of combinatorial bandits Cesa-Bianchi and Lugosi (2012), but we aim at leveraging a specific structural property: in many relevant examples, W is a rank 1 matrix.

Two different types of graphs are relevant for matchings, *bipartite* and *monopartite*, and we are going to consider both of them (even though the latter is more intriguing, the former, maybe more intuitive, will serve as a warm-up and to convey insights). In the bipartite case, the set of vertices is separated in two distinct subsets $\mathcal{U}$ and $\mathcal{V}$ (of respective sizes N and M) and edges only exist across subsets, not within. The rank-1 adjacency matrix W is then a $N \times M$ matrix, that can be written as $W = uv^\top$ for some $u \in (0, 1)^N$ and $v \in (0, 1)^M$. The canonical application of this setting is online advertising, where the probability that a user clicks on an ad depends on both the position at which the ad is displayed and its relevance to the user Katariya et al. (2017b). Other motivations come from two-sided markets, where matching occurs between offers and demands, e.g. in online labor markets, u_i may represent the utility for a user seeking a solution to a project and v_j may represent the expertise of a project solver.

On the other hand, monopartite graphs have $2N$ vertices and their rank-1 adjacency matrix W can be written as $W = uu^\top$ for some $u \in (0, 1)^{2N}$. This setting models collaborative activities that arise in teamwork, online gaming, and online labor platforms Johari et al. (2018). For instance, some online gaming apps match players together (e.g. Go, competitive quizzes, and drawings) but players will participate only if they both want to. In a simple model, player i decides to participate with probability u_i, and matched players i and j participate in a game with probability $u_i u_j$. The expected number of played games from a proposed matching m is then $\sum_{(i,j)\in m} u_i u_j$. The revenue of such apps typically comes from ads displayed during games, so the more games played the better. In these examples, the app will match (say, everyday) as many pairs of players as possible and not just one (as in the bipartite example).

We will consider the aforementioned two variants of these sequential choices of matchings: either the matching has to be "minimal", i.e., it has to be a single pair of vertices, or it has to be "maximal", i.e., a choice of N distinct pairs. We will refer the former to as *pair selection* and the latter as *matching selection* problem. As standard in multi-armed bandits, we shall investigate both the regret minimization over an arbitrary given time horizon and the pure exploration in a PAC learning setting.

3.1.1 Related work

The matching problems defined above are special classes of combinatorial bandit problems with semi-bandit feedback Cesa-Bianchi and Lugosi (2012) with many recent improvements for regret minimization Combes et al. (2015); Cuvelier et al. (2021); Degenne and Perchet (2016); Perrault et al. (2020); Wang and Chen (2021) as well as pure exploration Chen et al. (2014); Garivier and Kaufmann (2016). The combinatorial structure is quite clear, as the cardinality of the set of matchings is equal to $(2N)!/(2^N N!) \sim \sqrt{2}\Big(2N/e\Big)^N$.

Off-the-shelf combinatorial bandits algorithms would incur a regret scaling as $\tilde{O}(N^2 \log(T)/\Delta_{\min})$,

where $\Delta_{\min}$ denotes the expected reward gap between an optimal matching and the best sub-optimal matching. This has been recently improved, but only in the aforementioned bipartite case, where the rank-1 structure has been leveraged in the line of work of *stochastic rank-1 bandits* Katariya et al. (2017a,b); Trinh et al. (2020), yet either with sub-optimal parameter dependencies or with asymptotic performances. This quadratic dependency would also appear in pure exploration, as standard algorithms would require in the bipartite case $O(NM \log(1/\delta))$ iterations to find the best pair with probability at least $1 - \delta$ Garivier and Kaufmann (2016).

The classical matching problem has strong connections with ranking/sorting; it is obviously the same with their sequential variants Rejwan and Mansour (2020); Zoghi et al. (2017) even though they do not directly handle the bandit feedback.

3.1.2 Organization of the chapter and our contributions

The remaining of the chapter is divided in four main parts. First, we formally introduce the general model in Section 3.2. Then we investigate the pair selection problem (both for regret minimization and pure exploration) in Section 3.3, and afterwards the matching selection problem (again, for regret and pure exploration) in Section 3.4. Finally, we present numerical results in Section 3.5. They validate the tightness of our results and demonstrate competitiveness and performance gains obtained by our proposed algorithms over some state-of-the-art baseline algorithms.

Our contributions can be summarized as follows:

i) For the pair selection problem in the bipartite case, we introduce a new algorithm, called Pair-Elim, in Section 3.3.1 with an optimal (up to a multiplicative constant) regret bound: perhaps interestingly, the algorithm eliminates sub-optimal rows and columns on different timescales. This result is extended to the monopartite case in the same Section 3.3.1.

   For pure exploration, we simply adapt Pair-Elim; it still leverages the rank-1 structure to find the optimal pair with a linear (instead of quadratic) sampling complexity in $O((N + M) \log(1/\delta))$.

ii) The monopartite case, still with pair sampling, is investigated in Section 3.3.2; we transform the above algorithm into Pair-Elim-Mono, that can handle both regret minimization and pure exploration. It is also extended for best matching identification, with again optimal bound (up to a multiplicative constant).

iii) Section 3.4.1 is dedicated to pure exploration with matching sampling; a new algorithm is developed with optimal sample complexity for non-degenerate ranges of parameters (i.e., it equals the new lower bounds proved up to multiplicative constants).

iv) Finally, regret minimization in the matching selection problem is investigated in Section 7; we introduce a new Adaptive-Matching algorithm with a linear (instead of quadratic) dependency in N since its regret scales as $\tilde{O}(N \log(T)/\Delta_{\min})$.

   Roughly speaking, this algorithm elies on a divide and conquer type of approach.

3.2 Objectives and problem statement

Noise model. We assume the noisy observation X_t of W is generated as follow: for any $(i, j, t), X_{i,j,t} = W_{i,j} + \varepsilon_{i,j,t}$ where $\varepsilon_{i,j,t}$ are independent, zero-mean, sub-Gaussian random variables.

Optimal matching. The objective is to find a matching m, either minimal or maximal depending on the setting, that maximizes the expected reward $\mathbb{E}[\sum_{(i,j)\in m} X_{i,j,t}] = \sum_{(i,j)\in m} W_{i,j}$. It turns out that in both the bipartite case and the monopartite one, under the rank-1 assumption, the optimal matching is the one that pairs better items together. More formally and without loss of generality, for the bipartite case ($W = uv^\top$), we assume that $u_1 \geq \cdots \geq u_N$ and $v_1 \geq \cdots \geq v_M$. The optimal matching is the one that associates (u_1, v_1), then (u_2, v_2), and so on[1]. Similarly, for the monopartite case ($W = uu^\top$), we assume that $u_1 \geq u_1 \geq \cdots \geq u_{2N}$ and the optimal matching associates any odd index with its successor, i.e. (u_1, u_2) then (u_3, u_4) and so on. In both cases finding the optimal matching boils down to finding the order of the entries of u and v.

Pure exploration. A first objective the agent can aim for is to identify the best matching with *high probability* and *as fast as possible*. Formally, given a confidence level $0 < \delta < 1/2$, the agent seeks to minimize the worst-case number of samples τ_δ needed for the algorithm to finish and return the optimal matching with probability at least $1 - \delta$.

Regret minimization. Another objective for the agent is to find the best matching while *playing sub-optimally as few times as possible* in the process. Formally, her goal is to minimize the regret, i.e., the difference between the cumulative reward of the oracle (that knows the best pair or the best matching) and her cumulative reward. Denoting by $\mathcal{M}$ the set of matching considered – e.g. minimal matchings for *pair selection* or maximal matchings for *matching selection* – the regret after T steps is defined as:

$$R(T) = T \max_{m\in\mathcal{M}} \sum_{(i,j)\in m} W_{i,j} - \sum_{t=1}^{T} \sum_{(i,j)\in m_t} W_{i,j}\,. \tag{3.1}$$

Universal vs. parameter dependent constants. In order to avoid cumbersomeness, we shall use the notations $c_{\mathfrak{u}}$ to denote some universal constant and $c_{\mathfrak{p}}$ to denote constants (w.r.t. T) but that can depend on other problem parameters. They might change from one statement to another, but they are always defined explicitly in the proofs.

3.3 Pair selection problem

In this section we consider the pair selection problem. Even though playing one pair (i, j) of items at each time step may seem very similar to dueling bandits Yue et al. (2012) in the monopartite case, the reward information structure is very different. In dueling bandits the information is *competitive*, one observes which i or j is best (in expectation). Here, the information is *collaborative*, the higher the parameters of both i and j, the higher the observation (in expectation). Thus, in our case, playing the pair (i, j) does not provide information on the relative order of i and j. Instead, to get information about the relative order of i and j, it is necessary to use a third item j' as a point of comparison and play both (i, j') and (j, j'). We will refer to this as comparing i with j *against* j'. A crucial idea, that is key to several of the algorithms presented in the chapter, is that the fastest way to compare two items i and j is to compare them against the item j' with the highest possible parameter value. It turns out this last remark also holds in the bipartite case.

[1]This is a direct consequence of the *rearrangement inequality*.

3.3.1 Bipartite case

As the bipartite case has already been studied and might be simpler to grasp, we start with it and then extend the results to the monopartite case. The fastest way to compare row items is to compare them against the best column, and reciprocally for columns. Similarly to Rank1Elim Katariya et al. (2017b), the algorithm maintains a list of *active* rows (resp. columns) that are, with high probability, *non-provably dominated* as defined by confidence sets computed from the samples. As Rank1Elim, Pair-Elim performs an Explore Then Commit (ETC) strategy, playing all active rows against randomly chosen active columns to collect samples and update the confidence sets. Then it uses a similar ETC strategy on columns. The main difference with Rank1Elim resides in Pair-Elim eliminating row and columns at different timescales. Pair-Elim implements an ETC policy with horizon T for rows. Simultaneously, it runs an ETC policy for columns, but over shorter time windows w (referred to as "blocks") between steps T_{w-1} and $T_w - 1$ where $T_w := T_{w-1} + 2^{2^w}$. Within a block, columns are only *temporarily* eliminated. They are reinstated as active at the beginning of the next block, when a new instance of the ETC is run in the new horizon. The detailed pseudo-code is given in Appendix 3.A.1.

Algorithm 11: Pair-Elim

```
Set target precision to δ;
for t = 0... do
    Identify active rows and columns;
    Sample all active columns against a random active row;
    Sample all active rows against a random active column;
    if t > Σ_{s=0}^{w} 2^{2^s} then
        Reset samples for columns;
        w = w + 1;
    end
    Update confidence sets on rows and columns;
    if Optimal pair detected with confidence ≥ 1 − δ then
        Output optimal pair;
    end
end
```

The key intuition is that the algorithm is more aggressive in the elimination of columns (i.e. with lower confidence) as they are only temporarily eliminated. Thus, sub-optimal rows will be eliminated after $\log(T)$ samples while the number of samples of sub-optimal columns only increases logarithmically in the current number of samples. This implies that pairs (i, j) that are "doubly-suboptimal", i.e. both i and j are sub-optimal, are eliminated after $\log(\log(T))$ samples. On the other hand, pairs that are only suboptimal, but not doubly, are eliminated after $\log(T)$ samples.

It is noteworthy that this cannot be achieved with a standard Explore Then Commit (ETC) independent on rows and columns, as the number of samples of sub-optimal decisions would then scale linearly (not logarithmically) with the number of samples – before elimination. This is the reason why the algorithm Rank1Elim is sub-optimal.

The complexity of bandit problems is characterized by the *gaps*, i.e., the difference in expectation between basic item performances. In this case, we need to differentiate gaps on rows and columns, namely, $\Delta_i^U = \max_{i' \in [N]} u_{i'} - u_i$ and $\Delta_j^V = \max_{j' \in [M]} v_{j'} - v_j$ that appear both in the regret and in the sample complexity.

Theorem 3.3.1. *For any time horizon $T > 0$, the expected regret of* Pair-Elim *with $\delta = (1/T)$ is upper bounded as,*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} A(u, v) \log(T) + c_{\mathfrak{p}} \log(\log(T))$$

where

$$A(u, v) = \sum_{i \in [N]: \Delta_i^U > 0} \frac{1}{v_1 \Delta_i^U} + \sum_{j \in [M]: \Delta_j^V > 0} \frac{1}{u_1 \Delta_j^V}.$$

The proof of Theorem 3.3.1 is available in Appendix 3.A.1.

Theorem 3.3.2. *For any $\delta \in (0, 1)$,* Pair-Elim *outputs the best pair with probability at least $1 - \delta$ at stage τ_δ s.t.*

$$\mathbb{E}[\tau_\delta] \leq c_{\mathfrak{u}} A(u, v) \log(1/\delta) + c_{\mathfrak{p}} \log\log(1/\delta)$$

where

$$A(u, v) = \sum_{i \in [N]: \Delta_i^U > 0} \frac{1}{(v_1 \Delta_i^U)^2} + \sum_{j \in [M]: \Delta_j^V > 0} \frac{1}{(u_1 \Delta_j^V)^2}$$

which is tight up to a multiplicative constant.

Proofs of the upper bound of Theorem 3.3.2 is in Appendix 3.A.1. The lower was proven in previous work Katariya et al. (2017b).

The improvement compared to Rank1Elim is visible, as the leading term of the regret scales as the inverse of the parameters mean for Rank1Elim, while it only scales as the inverse of the best parameter, for Pair-Elim.

3.3.2 Monopartite case

We introduce a new algorithm, Pair-Elim-Mono, that generalizes the main ideas of Pair-Elim. Instead of working on monopartite graph (of size $2N$), it first duplicates items and create a bipartite graph with $\mathcal{U} = \mathcal{V} = [2N]$. Then, as in the previous section, an elimination policy is run over rows with horizon T and over columns by blocks.

The major difference between the mono and bipartite case is that, in the former, two items of $\mathcal{U}$ and $\mathcal{V}$ are optimal (instead of only one, because of the initial duplication). As a consequence, active pairs are tracked instead of active rows and columns. Pairs containing item i are all eliminated after they have been deemed smaller than two other distinct items. If i is deemed smaller than another item j, all pairs containing item i are eliminated except (i, j). If entry (i, j) is eliminated as a consequence of row i's sub-optimality, entry (j, i) is also eliminated.

Note that the fastest way to compare item 1 with item $i > 2$ is to compare them against item 2 and the fastest way to compare item 2 with item $i > 2$ is to compare them against item 1. Similarly, it is harder to identify the second best item than the best item, as simple computations yield

$$u_1 \Delta_{2,i} \leq u_2 \Delta_{1,i}.$$

where $\Delta_{i,j} = u_i - u_j$.

Apart from the algorithm itself, another difference with the bipartite case is the way to measure the gaps, but the guarantees are very similar to the bipartite case.

Theorem 3.3.3. *The expected regret of* Pair-Elim-Mono *satisfies, for any time horizon $T > 0$,*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} A(u) \log(T) + c_{\mathfrak{p}} \log\log(T) \tag{3.2}$$

where

$$A(u) = \sum_{i \in \{3, \ldots, 2N\}: \Delta_{2,i} > 0} \frac{1}{u_1 \Delta_{2,i}}.$$

Algorithm 12: PAIR-ELIM-MONO

```
Set target precision to δ;
Initiate rows 𝒰 = [2N] and columns 𝒱 = [2N];
for t = 0, 1, ... do
    Identify active pairs;
    Sample all active pairs;
    if t > Σ_{s=0}^{w} 2^{2^s} then
        Reset samples for columns;
        w = w + 1;
    end
    Update confidence sets on rows and columns;
    if Optimal pair detected with confidence ≥ 1 − δ then
        Output optimal pair;
    end
end
```

The proof is provided in Appendix 3.A.2.

Theorem 3.3.4. *For any $\delta \in (0,1)$, the sample complexity of the* PAIR-ELIM-MONO *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} A(u) \log(1/\delta) + c_{\mathfrak{p}} \log\log(1/\delta)$$

with probability at least $1-\delta$, where

$$A(u) = \sum_{i \in [2N]:\Delta_i > 0} \frac{1}{(u_1 \Delta_{2,i})^2}$$

which is tight up to a multiplicative constant.

The proof is provided in Appendix 3.A.2.

Towards maximal matchings. The *matching selection* setting can be seen as a constrained versions of *pair selection* where the agent has the constraint that N consecutively sampled pairs should form a maximal matching instead of being chosen freely. Hence, before diving in this setting, we can wonder what would be the sample complexity to identify the best maximal matching, but by freely choosing the pairs, which is arguably a simpler problem than *matching selection*. This can be done in two steps: **1)** identify the two best items using PAIR-ELIM-MONO, **2)** sample all unranked items against them until the full best matching is identified. We refer to this two-step algorithm as PAIR-SELECT. There again, the sample complexity of the algorithm is optimal up to a multiplicative constant, proofs are deferred to Appendix 3.A.3.

Theorem 3.3.5. *For any $\delta \in (0,1)$, the sample complexity of the* PAIR-SELECT *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} A(u) \log(1/\delta) + c_{\mathfrak{p}} \log\log(1/\delta)$$

with probability at least $1-\delta$, where

$$A(u) = \sum_{i \in [2N]:\Delta_i > 0} \frac{1}{(u_1 \Delta_i)^2}\,,$$

$$\text{with } \Delta_{2,i} = u_{2i} - u_{2i+1} \quad \text{and} \quad \Delta_{2i-1} = u_{2i-2} - u_{2i-1}$$

which is tight up to a multiplicative constant.

Surprisingly, identifying the full order of the items rather than simply the top two ones does not require N times more samples. Rather the degradation appears in the gap, that are not anymore the gap to the second-best item, but rather the gap between consecutive matched pairs.

In the following section, we investigate if similar guarantees hold in the *matching selection* setting or if the constraints on the choice of consecutive pairs have a stronger impact.

3.4 Matching selection problem

In this section, we present our results for the matching selection problem, where recall at each iteration step a maximal matching of items needs to be selected. We first consider the objective of pure exploration, and then consider the regret minimization objective. In this section, pair i refers to the i^{th} pair of consecutive items (u_{2i-1}, u_{2i}), the pair with the i^{th} highest expected reward in the optimal matching.

Contrarily to the pair selection setting, where the same algorithm has tight guarantees for both regret minimization and pure exploration, the matching selection setting requires different algorithms. This can be understood from a simple example where the smallest gap between consecutive items $i, i+1$ appears at the bottom of the ranking (between low quality items). From a pure exploration point of view, the fastest way to rank them is to compare them against the top ranked item. However, this is sub-optimal from a regret point of view as misranking i and $i+1$ incurs a low regret, while playing $(1, i)$ or $(1, i+1)$ incurs a high regret. The pure exploration objective and the regret minimization objective are treated separately in the following as they require the use of different algorithms.

3.4.1 Pure exploration for matching

The algorithm for matching selection with an objective of pure exploration, referred to as MATCHING-ID, is based on this idea of comparing items with the top ones to rank them quickly (remember finding an optimal matching amounts to finding an order over the items). It makes use of two non-exclusive sets of items: $\mathcal{S} \subseteq [2N]$ is the set of un-ranked items (their exact rank is unknown) and $\mathcal{B}$ the set of items that are still potentially amongst the $|\mathcal{S}|$ best items. The algorithm proceeds through iterations, in each iteration sampling matchings from $\mathcal{B} \cup \mathcal{S}$ such that all distinct pairs of items are sampled once. Items from $\mathcal{S}$ are ranked by using the samples collected from matches with items in $\mathcal{B}$. This procedure continues until the rank of all items is known.

Algorithm 13: MATCHING-ID

$\mathcal{S} = [2N]$;
while $\mathcal{S} \neq \emptyset$ **do**
  Compute the set $\mathcal{S}$ of unranked items;
  Compute the set $\mathcal{B}$ of candidate $|\mathcal{S}|$ best items;
  Sample each item in $\mathcal{B} \cup \mathcal{S}$ once against each other item in that set;
  Update confidence intervals for the items in $\mathcal{S}$ using observed outcomes of matches with items in $\mathcal{B}$;
end

We introduce the following notations

$$\mu_{[2N]\setminus\{2k,2k+1\}} = \frac{1}{2N} \sum_{i\in[2N]\setminus\{2k,2k+1\}} u_i.$$

Let s and h denote the indices of the smallest and the second smallest gap, i.e. $s = \arg\min_{k\in[2,N-1]} \Delta_{2k,2k+1}$ and $h = \arg\min_{k\in[N-1]\setminus\{s\}} \Delta_{2k,2k+1}\mu_{[2N]\setminus\{2k,2k+1\}}$. We also define

$$\gamma_{\min} = \min_{k\in[N-1]} \{\mu_{[2N]\setminus\{2k,2k+1\}}\Delta_{2k,2k+1}\}.$$

To simplify the exposition of the result, we assume that the smallest gap is not between the two best pairs (general version of the result is given in Appendix 3.A.7).

Theorem 3.4.1 (upper bound)**.** *For any $\delta > 0$, the sample complexity of the* Matching-id *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{\gamma_{\min}^2} \log(1/\delta) + c_{\mathfrak{p}}$$

with probability at least $1-\delta$. Moreover, by denoting,

$$\alpha := \min\left\{\frac{1}{2}\frac{(u_1+u_2)\Delta_{2s,2s+1}}{\mu_{[2N]\setminus\{2h,2h+1\}}\Delta_{2h,2h+1}}, 1\right\},$$

the following holds with probability at least $1-\delta$

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{(1-\alpha)^2(u_1^2+u_2^2)\Delta_{2s,2s+1}^2} \log(1/\delta) + c_{\mathfrak{p}}.$$

The proof of these upper-bounds is deferred to Appendix 3.A.7. These upper bounds are tight, up to multiplicative factors, for some interesting regimes of parameters, as stated below.

Theorem 3.4.2. *Assume that stochastic rewards of item pairs have Gaussian distribution with unit variance. Then, for any δ-PAC algorithm, we have*

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}} \sum_{i\in[2N]:\Delta_i>0} \frac{1}{\sum_{j=1}^{2N} u_j^2} \frac{1}{\Delta_i^2} \log(1/\delta)$$

and

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}} \frac{1}{u_1^2+u_2^2} \frac{1}{\Delta_{2s,2s+1}^2} \log(1/\delta).$$

In particular, if all gaps are equal, the first lower bound of Theorem 3.4.2 matches the first upper bound of Theorem 3.4.1 up to a multiplicative constant. On the other hand, in the opposite regime where one gap is substantively smaller than all others, then it is the second bounds that are equivalent.

These results with matching matching should be put into perspective with their pendant, Theorem 3.3.5, for pair selection. In this case, the sample complexity of the latter is N times bigger than the one of the former. This might seem surprising at first sight as pair selection is an "easier" problem (without constraints). The reason is that with matching selection, the algorithm gets to observe N pairs at each iteration (and not just one). As a consequence, the overall number of pairs evaluation $X_{i,j,s}$ are actually of the same order.

This is quite surprising as selection matchings is much more constrained than selecting batches of N arbitrary pairs (possibly with repetitions and/or single items sampled more than just once in a batch). The main consequence is that the matching identification problem is *as*

difficult with minimal than maximal matchings selection (and therefore in any intermediate case).

Regret minimization

We first describe and analyze a simplified matching divide and conquer algorithm, which is correct for problem instances satisfying a condition on model parameters introduced shortly. This simplified algorithm allows us to convey the key idea that underlies the design of a more complicated algorithm. Having described this simplified algorithm and shown the regret upper bound, we will remove the aforementioned condition and show a regret upper bound that holds for a matching divide and conquer algorithm. In the following, we assume there is a unique optimal matching m^*.

Simple Adaptive Matching

We consider problem instances under the following assumption on model parameters:

Assumption 1. *The model parameters $u_1, \dots, u_{2N}$ are assumed to satisfy $u_{2i-1} = u_{2i}$, for all $i \in [N]$.*

Under Assumption 1, in the optimal matching every pair of matched items have equal parameter values. This assumption avoids some complications that arise due to uneven clusters in the divide and conquer procedure.

The Simple Adaptive Matching (SIMPLE-ADAPTIVE-MATCHING) successively partitions items into an ordered sequence of clusters (ranked clusters), such that all items in a cluster have a higher rank than all items in any lower-ranked cluster with a high probability. Items in a cluster $\mathcal{S}$ of size $|\mathcal{S}| = 2K$ are matched according to a round-robin tournament, which runs over $2K - 1$ iterations.

A cluster is split into two sub-clusters as soon as the upper bound for the reward of each item in one of the sub-clusters is smaller than the lower bound for the reward of each item in the other sub-cluster. Assumption 1 guarantees that, with high probability, at each iteration step, all clusters contain an *even* number of items, so it is always possible to match all items within each cluster.

Functions `sample_matching` and `conf_bound` of Algorithm 14 are detailed in Appendix 3.A.4. At the high level, `sample_matching` samples matchings that ensure that each $|\mathcal{S}| - 1$ iterations, any given item in $\mathcal{S}$ has been matched once with any other item in $\mathcal{S}$. `conf_bound` builds confidence intervals for the total reward of each item per match with items in the same cluster or in lower ranked cluster.

The regret of SIMPLE-ADAPTIVE-MATCHING can be bounded by using the following additional notation

$$\Delta_{\min} = \min_{m \in \mathcal{M}: m \neq m^*} \left\{ \sum_{(i,j) \in m^*} u_i u_j - \sum_{(i,j) \in m} u_i u_j \right\}$$

and the proof is again deferred to Appendix 3.A.4.

Theorem 3.4.3. *The expected regret of* SIMPLE-ADAPTIVE-MATCHING *satisfies, for any horizon $T > 0$,*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} \frac{N \log(N)}{\Delta_{\min}} \log(T) + c_{\mathfrak{p}}.$$

Algorithm 14: SIMPLE-ADAPTIVE-MATCHING

input : set of items $[2N]$ and horizon T

$t = 0, C = X = \tilde{C} = \tilde{X} = [0]^{2N \times 2N}, \mathfrak{S} = \{[2N]\}$;
for $t = 1 \dots T$ **do**
  $m_t \leftarrow$ `sample_matching`$(\mathfrak{S}, t)$;
  for $(i, j) \in m_t$ **do**
    $\tilde{X}(i, j) \leftarrow \tilde{X}(i, j) + X_{i,j,t}$;
    $\tilde{C}(i, j) \leftarrow \tilde{C}(i, j) + 1$;
  end
  for $\mathcal{S} \in \mathfrak{S}$ **do**
    if $\exists i \in \mathcal{S}$ *s.t.* $\sum_{j \in S} \tilde{C}(i, j) = |\mathcal{S}| - 1$ **then**
      $X([s], :), C([\mathcal{S}], :) += \tilde{X}([\mathcal{S}], :), \tilde{C}([\mathcal{S}], :)$;
      $\tilde{X}([\mathcal{S}], :), \tilde{C}([\mathcal{S}], :) = 0$;
    end
  end
  $Q_+, Q_- \leftarrow$ `conf_bound`$(X, C, T, Q_+, Q_-, \mathfrak{S})$;
  for $\mathcal{S} \in \mathfrak{S}$ **do**
    Order items in $\mathcal{S}$ according to Q_+;
    for $i \in \{2, \dots, |\mathcal{S}|\}$ **do**
      if $Q_+[i] < Q_-[i-1]$ **then**
        Split $\mathcal{S}$ between $i - 1$ and i;
      end
    end
  end
end

ADAPTIVE-MATCHING **algorithm**

In general, when Assumption 1 does not hold, some of the clusters may have odd sizes. In these odd-size clusters, uniform sampling within a cluster is infeasible, and we need to match items residing in different clusters. In order to deal with these complications, we use a new ADAPTIVE-MATCHING algorithm.

ADAPTIVE-MATCHING is defined as an extension of SIMPLE-ADAPTIVE-MATCHING. It uses the same policy for splitting clusters. The SAMPLE-MATCHING procedure extends that of the previous algorithm to unevenly split clusters. This procedure ensures that any two mutually un-ranked items are matched similarly to other items, which guarantees that the expected rewards for those items are scaled similarly. At the high level, it defines a list of matchings that respect the desired item sampling proportion, then samples the matchings in this list in a round-robin.

A detailed description of SAMPLE-MATCHING, and the proof of the following result, are given in Appendix 3.A.5.

Theorem 3.4.4. *The expected regret of* ADAPTIVE-MATCHING *is bounded, for any horizon* $T > 0$*, as*

$$\mathbb{E}[R(T)] \leq c_{\mathfrak{u}} \frac{N \log(N)}{\Delta_{\min}} \log(T) + c_{\mathfrak{p}}.$$

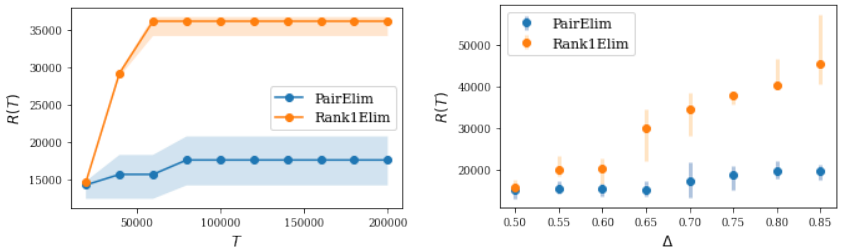


Figure 3.1: Regret comparison for PAIR-ELIM vs RANK1ELIM: (top) regret versus T for fixed $\Delta = 0.75$ and (bottom) regret versus the gap parameter Δ for fixed $T = 2,000,000$. We used 20 independent runs. The shaded areas show the range between the 5% and 95% percentile.

Comparison with an exploration policy

The ADAPTIVE-MATCHING algorithm matches high parameter value items together as soon as they are identified in order to exploit this for accruing reward. On the other hand, our algorithms, for the pair sampling problem and the pure exploration matching identification problem, used the detected high parameter value items to *explore* the un-ranked ones. Without a comparison, it is unclear which of the two strategies will lead to a smaller regret in the end. For this reason, we consider an *exploration-first* algorithm that matches identified high parameter value items with other items to speed up the learning of the rank of these other items, and compare it with ADAPTIVE-MATCHING.

We do this under assumption that both algorithms are given as input the two best items, as well as a set of $2(N-1)$ un-ranked items. We chose as a comparison metric the upper bound on the total regret incurred by the two algorithms until the un-ranked items can be partitioned into two or more ranked clusters of items. We denote with U_I the upper bound on the regret R_I for the exploration-first strategy and with U_D the upper bound on the regret R_D for the ADAPTIVE-MATCHING algorithm.

The following Lemma 3.4.5 states that unless the second best item is sufficiently worse than the best item, the regret of the ADAPTIVE-MATCHING algorithm is at most of the same order as that of the exploration-first algorithm, and can be arbitrarily smaller depending on the problem parameters. If the ratio between the parameter values of the first and the second best item goes to zero, then the exploration-first algorithm becomes infinitely better than the ADAPTIVE-MATCHING algorithm.

Lemma 3.4.5. *If $u_2/u_1 > 1/2$, then $U_D/U_I \leq c_u N$, and it can be arbitrarily close to 0 depending on the problem parameters. If $u_3/u_2 > 1/2$, then this ratio is smaller than a constant independent from the parameters of the problem. On the other hand, $\lim_{u_2/u_1 \to 0} R_D/R_I = +\infty$.*

In summary, the lemma tells that ADAPTIVE-MATCHING is essentially as good as the exploration-first strategy for any problem instance such that the parameter value of the second best item is at least a constant factor of the best item.

3.5 Experiments

In this section we present numerical results, which demonstrate tightness of our theoretical bounds and compare our proposed algorithms with some state-of-the-art baseline algorithms. We first consider the pair selection problem and then the matching selection problem. In summary, our numerical results validate our theoretical results and demonstrate that significant performance gains can be achieved against some previously proposed algorithms.

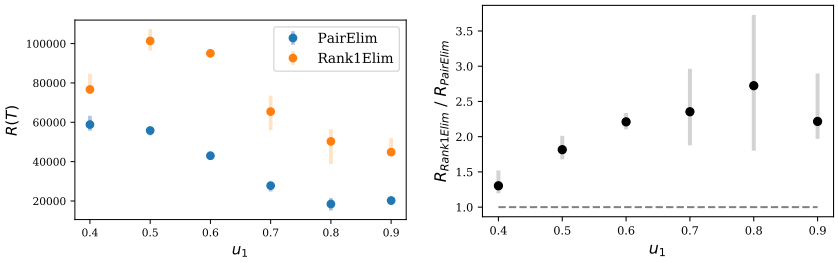


Figure 3.2: Regret comparison for PAIR-ELIM vs RANK1ELIM for different values of maximum item parameter value: (top cumulative regrets, and (bottom) ratio of the cumulative regrets. The parameter setting is $N = 8$, $\mu_U = 0.2$, and we used 20 independent runs.

All the code used for obtaining the results in this section is available from this public Gitlab repository: `https://gitlab.com/roka/matching-bandit`

3.5.1 Pair selection

We consider RANK1ELIM as a baseline for comparison. As noted in the introduction, RANK1ELIM has a regret upper bound that is sub-optimal with respect to the problem parameters, which is in contrast to our algorithm, PAIR-ELIM that has optimal regret bound up to a multiplicative constant. We demonstrate that significant performance gains that can be achieved by using PAIR-ELIM versus RANK1ELIM for some problem instances.

In all experiments, the variables considered are Bernoulli variables. We consider the bipartite case with $N = M$. Each problem instance is defined by a tuple (N, u_1, Δ), where $0 \leq \Delta \leq u_1 \leq 1$, and assuming that $u_1 = v_1$. The row parameter values $u_2, \ldots, u_N$ are defined as sorted values of independent random variables according to uniform distribution on $[0, 2(u_1 - \Delta)]$, where Δ is the expected gap between the value of the best item and the value of any other item. Note that we have $\mu_U = u_1 - (1 - 1/N)\Delta$. For fixed value of parameter u_1 and increasing expected gap Δ, we have problem instances with fixed maximum row parameter value and decreasing mean row parameter value μ_U. We similarly define the column parameter values, and all the observations above made for row parameter values hold for column parameter values. According to our regret analysis, PAIR-ELIM algorithm will outperform RANK1ELIM when Δ is large for fixed u_1. To confirm this claim, we ran the two algorithms on a set of problem instance with $N = 8$ and $u_1 = 0.9$. The results are shown in Figure 3.1.

A notable improvement of PAIR-ELIM compared to RANK1ELIM is that the cumulative regret bound is inversely proportional to the *maximums* of row and column item parameter values, instead of the *means* of row and column item parameter values. For our problem instances, the former values correspond to u_1 and v_1 (with $u_1 = v_1$), while the latter values correspond to μ_U and μ_V (with $\mu_U = \mu_V$). We thus expect that PAIR-ELIM would outperform RANK1ELIM for problem instances for which there is a significant gap between the maximum and mean values of the row and column item parameter values. We demonstrate this for problem instances defined as follows.

We consider the bipartite case with $N = M$. Each problem instance is defined by a tuple (N, u_1, Δ), where $0 \leq \Delta \leq u_1 \leq 1$, and assuming that $u_1 = v_1$. Other row item parameter values $u_2, \ldots, u_N$ are sorted values of independent random variables according to uniform distribution over $[0, 2(u_1 - \Delta)]$, where Δ is the expected gap between the value of the best item and the value of any other item. Note that we have $\mu_U = u_1 - (1 - 1/N)\Delta$. For fixed value of parameter u_1 and increasing expected gap Δ, we have problem instances with fixed maximum row item parameter value and decreasing mean row item parameter value μ_U. We similarly define the

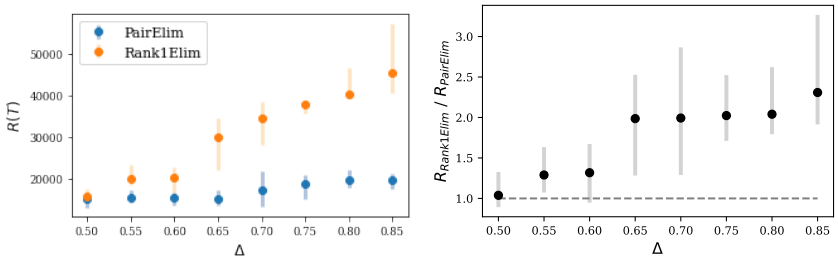


Figure 3.3: Regret comparison for PAIR-ELIM vs RANK1ELIM for different values of the gap parameter Δ: (top) cumulative regrets, and (bottom) ratio of the cumulative regrets. The parameter setting is $N = 8$, $u_1 = 0.9$, and we used 20 independent runs.

column item parameter values, and all the observations above made for row item parameter values hold for column item parameter values.

We first consider problem instances such that the mean values of row and column item parameters are fixed to value $\mu_U = 0.2$, and we vary the value of parameter u_1 in $[0.4, 0.9]$. The results are shown in Figure 3.2. We observe that PAIR-ELIM provides a significant performance gain over RANK1ELIM, which for some problem instances is for as much as nearly a 1/3 reduction of the cumulative regret. We also observe that there is a general trend of PAIR-ELIM outperforming RANK1ELIM more for larger gaps between the mean and maximum values of item parameter values.

We next consider problem instances that have the maximum row and column item parameter values fixed to value $u_1 = 0.9$ and varying gap parameter Δ taking values in $[0.5, 0.85]$. Note that with the value of the gap parameter Δ increasing, the means of row and column item parameter values decrease. As a consequence, the cumulative regret of RANK1ELIM should increase. On the other hand, as the maximum values of row and column item parameter values are kept fixed, the cumulative regret of PAIR-ELIM should remain roughly the same. These claims are confirmed in Figure 3.3 (top). In Figure 3.3 (bottom), we observe a trend of PAIR-ELIM outperforming RANK1ELIM more for larger values of Δ, and that this can be for a significant amount.

3.5.2 Matching selection

In this section we evaluate the performance of our algorithm for the matching selection problem. Our goal is twofold. We first demonstrate numerical results according to which our proposed algorithm has expected regret that scales proportionally to $N \log(N)/\Delta_{\min}$ for a fixed horizon T. We then compare its performance with that of ESCB, which, as discussed in the introduction, has the expected regret bound $O(N^2 \log^2(N)/\Delta_{\min})$ for fixed T. We demonstrate that our algorithm can achieve significant performance gains over ESCB. In our evaluations. We use our SIMPLE-ADAPTIVE-MATCHING algorithm, for problem instances such that there is a unique optimum matching with matched items having equal parameter values.

We first show that the regret of our proposed algorithm scales in the order of $N \log(N)/\Delta_{\min}$. We consider a set of problem instances, each is defined by a tuple $(N, \tilde{\Delta})$, where $0 < (N-1)\tilde{\Delta} \leq 1$ and $\tilde{\Delta}$ is the gap between parameter values of adjacent matched pairs in the optimum matching, so that $\tilde{\Delta} = \sqrt{\Delta_{\min}}$. The parameter values are defined by $u_{2i-1} = u_{2i} = (N-i)\tilde{\Delta}$ for $i \in [N]$. We run our algorithm on problem instances with $\tilde{\Delta} = 0.1$. The results shown in Figure 3.4 suggest that the cumulative regret of SIMPLE-ADAPTIVE-MATCHING scales as $(1/\Delta_{\min})N \log(N)$ as established in Theorem 3.4.3.

We next compare the performance of our algorithm and ESCB. We consider problem

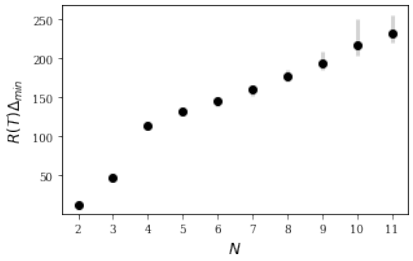


Figure 3.4: Normalized regret of SIMPLE-ADAPTIVE-MATCHING versus N for $u_{2i-1} = u_{2i} = (N-i)\tilde{\Delta}$, for $i \in [N]$, with $\tilde{\Delta} = 0.1$ and $T = 200,000$.

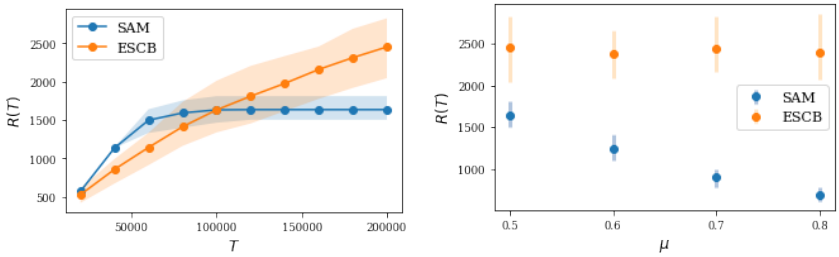


Figure 3.5: Regret comparison for SIMPLE-ADAPTIVE-MATCHING (SAM) vs ESCB: (top) regret versus T for fixed $\mu = 0.5$ and (bottom) regret vs μ for fixed $T = 200,000$.

instances defined by a tuple $(N, \mu, \tilde{\Delta})$ where μ is the mean of parameter values and $\tilde{\Delta}$ is the gap between parameter values. The values of item parameters are set as $u_{2i-1} = u_{2i} = \mu + (N+1-2i)\tilde{\Delta}/2 \in [0,1]$ for $i \in [N]$. Such problem instances allow us to vary μ while keeping other parameters fixed; we fix $N = 4$ and $\tilde{\Delta} = 0.1$. In Figure 3.5 (top) we show the regret versus the time horizon T for fixed $\mu = 1/2$, which shows that our algorithm outperforms ESCB for large enough values of T. We expect our algorithm to perform better than ESCB as we increase μ. The results in Figure 3.5 (bottom) confirm this claim. We have performed these experiments for a small value of N because of the computation complexity of ESCB. ESCB requires solving an NP-hard problem in each iteration, and has overall computation complexity $O(|\mathcal{M}|T)$ where $\mathcal{M}$ is the set of all arms. For the matching selection problem, $|\mathcal{M}|$ scales as $\sqrt{2}(2N/e)^N$.

3.A Appendix

3.A.1 Pair-Elim algorithm

Algorithm description and pseudo-code

In this section, we describe Pair-Elim algorithm with the pseudo-code provided in Algorithm 15.

The key feature of Pair-Elim is the usage of two different time scales for elimination of rows and columns. The rows are eliminated by using an ETC policy with horizon T, and the columns are eliminated by using a similar ETC policy over time windows w (referred to as "blocks") between steps T_{w-1} and $T_w - 1$.

The ETC policy relies on maintaining confidence intervals, Q_+^U, Q_-^U and Q_+^V, Q_-^V for rows and columns, respectively. The confidence intervals for the rows are computed from all samples gathered since the beginning of the run, while the confidence intervals for the columns are limited to the samples gathered during the current time window. The target precision of the confidence intervals also varies: for the rows, it is set to optimize the regret over the horizon T, for the columns, it is less precise and only optimizes the regret over the current time window.

A domination mapping h^U is determined from the confidence intervals Q_+^U and Q_-^U. It maps any dominated row to to one of those that dominates it. In the column exploration step, a random row is picked and all non-dominated rows are sampled against $h^U(i)$. h^V is defined and used similarly in the row explorations step.

Note that this domination mapping is not necessary to obtain the proven upper bound on the regret of the algorithm. Sampling against any potentially optimal row or column would provide the same theoretical guarantee. However, we observed experimentally that using the domination mapping gave consistently better results.

When the algorithm is run in the regret minimization mode, the precision of the row confidence intervals is set to the required precision so that they hold until the horizon is reached. The precision of the column confidence intervals is set to the required precision so that they hold until the end of the running time window.

When the algorithm is run in the regret minimization mode, the precision of the row confidence intervals is set to the required precision so that they hold with probability ate least $1-\delta$. Until the best row is detected, the precision of the column confidence intervals is set to the required precision so that they hold until the end of the running time window. Once the best row is detected, it is set so that they hold with probability ate least $1-\delta$.

The algorithm uses function `domination_map` that defines h^U by using Q_+^U and Q_-^U through the following equation:

$$h^U(i) = i\mathbb{1}_{\{\exists k\in[0,M] \text{ s.t. } \forall j\in[N], Q_+(k,i)>Q_-(k,j)\}} + \max_{j\in[N]} j\mathbb{1}_{\{\exists k\in[0,M] \text{ s.t. } Q_+(k,i)<Q_-(k,j)\}}, \ \forall i \in U.$$

The algorithm uses function `confidence_bound`, with input arguments X, C, h, Q_+, Q_-, and p, and outputs Q_+ and Q_- defined as follows. For each $(k,i) \in U \times V$, if $C(k,i) = k_l$, with $k_l := \lceil 4^{l+1}\log(\beta_l)\rceil$ for some integer $l > 0$, then:

$$Q_-(k,i) = \frac{X(k,i)}{C(k,i)} - \sqrt{\frac{\log(\beta_l)}{k_l}} \text{ and } Q_+(k,i) = \frac{X(k,i)}{C(k,i)} + \sqrt{\frac{\log(\beta_l)}{k_l}}.$$

For each $i \in U$, if $\sum_{k=1}^M C(k,i) = k_l$ for some k_l,then:

$$Q_-(0,i) = \frac{\sum_{k=1}^M X(k,i)}{\sum_{k=1}^M C(k,i)} - \sqrt{\frac{\log(\beta_l)}{k_l}} \text{ and } Q_+(0,i) = \frac{\sum_{k=1}^M X(k,i)}{\sum_{k=1}^M C(k,i)} + \sqrt{\frac{\log(\beta_l)}{k_l}}.$$

Algorithm 15: PAIR-ELIM

input : set of rows U, set of columns V, precision δ (or $\frac{1}{T}$) and pure_explore

$t = 0, C = X = C_w = X_w = [0]^{N \times M}$, column_explore =True;
for t... **do**
  for *window* $w = 0, 1, 2, \ldots$ **do**
    // columns;
    $Q_+^V, Q_-^V \leftarrow$ `confidence_bound`$(X_w, C_w, 2^{2^w}, Q_+^V, Q_-^V,$ column_explore$)$;
    $h^V \leftarrow$ `domination_map`(Q_+^V, Q_-^V);
    $J \leftarrow \bigcup_{j \in V} \left\{ h^V(j) \right\}$;
    // rows;
    $Q_+^U, Q_-^U \leftarrow$ `confidence_bound`$(X, C, T, Q_+^U, Q_-^U,$ pure_explore$)$;
    $h^U \leftarrow$ `domination_map`(Q_+^U, Q_-^U);
    $I \leftarrow \bigcup_{i \in U} \left\{ h^U(i) \right\}$;
    $j \leftarrow h^V\left(\text{Unif}(V)\right)$;
    for $i \in I$ **do**
      $X(i,j) \leftarrow X(i,j) + x_{i,j,t}$;
      $C(i,j) \leftarrow C(i,j) + 1$;
      $s \leftarrow s + 1$, $t \leftarrow t + 1$;
    end
    $i \leftarrow h^U\left(\text{Unif}(U)\right)$;
    for $j \in J$ **do**
      $X_w(i,j) \leftarrow X_w(i,j) + x_{i,j,t}$;
      $C_w(i,j) \leftarrow C_w(i,j) + 1$;
      $s \leftarrow s + 1$, $t \leftarrow t + 1$;
    end
    if $s > 2^{2^w}$ *and row_explore =True* **then**
      // Change time window;
      $s = 0, C_w = X_w = [0]^{N \times M}$;
    end
    if *column_explore =False,* $|I|$*=1 and pure_explore* **then**
      // Start looking for the best column with high probability;
      $w = \log_2 \log_2(T)$;
      column_explore =True;
    end
    if *column_explore =True,* $|J|$*=1 and pure_explore* **then**
      // Return best entry;
      return (I, J);
    end
  end
end

In all other cases, values in Q_+ and Q_- remain unchanged. The value used for parameter β_l depends on h and p. In the case where p is false, $\beta_l = h$. When p is true $\beta_l = \pi\sqrt{(N+1)(M+1)h/3} \cdot l$.

Proof of the upper bound

In this section, we prove Theorems 3.3.1 and 3.3.2. The proof starts by a lemma that bounds the probability that any of the built confidence interval fails. It is given in a slightly more general form than needed, as it will be used in other proofs.

Let $x_1, ..., x_k$ be a sequence of k independent samples from some distributions, and $\hat{x}$ denote the empirical mean

$$\hat{x} = \frac{1}{k}\sum_{s=1}^{k} x_s.$$

Let us denote $x^* = \mathbb{E}[\hat{x}]$. The lower and upper bounds for x^* are defined for $k = k_l$, with $k_l = \lceil 4\Delta_l^{-2}\log(\beta_l)\rceil$ where $\Delta_l = 1/2^l$ and β_l is a parameter, as follows

$$L_l(\hat{x}) = \hat{x} - \sqrt{\frac{\log(\beta_l)}{k_l}} \text{ and } U_l(\hat{x}) = \hat{x} + \sqrt{\frac{\log(\beta_l)}{k_l}}.$$

Let $\mathcal{E}_l(\hat{x})$ be the event that x^* lies within the interval $[L_l(\hat{x}), U_l(\hat{x})]$, i.e.

$$\mathcal{E}_l(\hat{x}) = \{x^* \in [L_l(\hat{x}), U_l(\hat{x})]\}.$$

Lemma 3.A.1. *The event $\mathcal{E}_l(\hat{x})$ holds with probability at least $1 - 2/\beta_l^2$.*

The proof of the lemma follows by direct application of Hoeffding's inequality and is omitted. □

We also state a general purpose lemma that bounds the number of samples before two sets of samples can be ranked.

Let $x_1, \ldots, x_k$ and $y_1, \ldots, y_k$ be two sequences of sampled values, and $\hat{x}$ and $\hat{y}$ be their respective empirical means. Let $\Delta(\hat{x}, \hat{y}) = x^* - y^* > 0$.

Lemma 3.A.2. *If $\Delta_l < \Delta(\hat{x}, \hat{y})/2$, then*

$$U_l(\hat{y}) < L_l(\hat{x}).$$

Proof of Theorem 3.3.1 (Regret):

The proof proceeds as follows. Let $\mathcal{A}$ be the "good event" that all row confidence intervals hold. Lemma 3.A.1 together with a union bound on the $N(M+1)$ tracked parameters and all confidence interval's updates up to horizon T gives:

$$\mathbb{P}[\overline{\mathcal{A}}] \leq 2\frac{N(M+1)}{T}.$$

A first bound on the regret follows

$$\begin{aligned} R(T) &\leq \mathbb{E}[R(T) \mid \mathcal{A}] + T\mathbb{P}[\overline{\mathcal{A}}] \\ &\leq \mathbb{E}[R(T) \mid \mathcal{A}] + 2N(M+1). \end{aligned} \tag{3.3}$$

Let $\mathcal{B}_w$ be the "good event w" that all column related confidence intervals hold during time window w. Repeating the same arguments, we have

$$2^{2^w}\mathbb{P}[\overline{\mathcal{B}}_w] \leq 2(N+1)M.$$

Note that w ranges from 0 to $w_{\max}$, where $w_{\max}$ is the largest integer k such that $2^{2^{k-1}} \leq T$. Hence, we have

$$w_{\max} \leq \log_2(\log_2(T)) + 1.$$

Which gives the following bound:

$$\sum_{w=0}^{w_{\max}} 2^{2^w} \mathbb{P}[\overline{\mathcal{B}}_w] \leq 2(N+1)M(\log_2(\log_2(T)) + 2). \tag{3.4}$$

Under event $\mathcal{B}_w$, optimal column 1 is never eliminated in time window w. Let $n_{1,j,w}$ denote the number of times column j is sampled against the optimal row 1 during time window w. Note that this number is equal for all non-eliminated columns. This, together with Lemma 3.A.2, gives the following bound

$$n_{1,j,w} \leq \left\lceil \frac{64 \log(2^{2^w})}{(u_1 \Delta_j^V)^2} \right\rceil.$$

Let $R_{1,j,w}$ be the total regret incurred by the sampling of pair $(1, j)$, at a sampling step where 1 is the picked active row with which columns are sampled against, during time window w. We show

$$\sum_{w=0}^{w_{\max}} R_{1,j,w} \leq \frac{512}{u_1 \Delta_j^V} \log(T) + 2\log_2(\log_2(T)) + 4. \tag{3.5}$$

This follows from the following inequalities

$$\begin{aligned}
\sum_{w=0}^{w_{\max}} R_{1,j,w} &\leq 2u_1 \Delta_j^V \sum_{w=0}^{w_{\max}} n_{1,j,w} \\
&\leq \frac{128}{u_1 \Delta_j^V} \left(\sum_{w=0}^{w_{\max}} 2^w \right) \log(2) + 2w_{\max} + 2 \\
&\leq \frac{512}{u_1 \Delta_j^V} \log(T) + 2\log_2(\log_2(T)) + 4.
\end{aligned}$$

The factor 2 in the first inequality accounts for the possibility that active column j is the column that active rows are sampled against at a given sampling step where 1 is the picked row.

Similarly, under event $\mathcal{A}$, the number of times row i is sampled against the optimal column 1, denoted as $n_{i,1}$, is bounded as

$$n_{i,1} \leq \left\lceil \frac{64 \log(T)}{(v_1 \Delta_i^U)^2} \right\rceil.$$

Thus, we have

$$\sum_{w=0}^{w_{\max}} R_{i,1,w} \leq 2v_1 \Delta_i^U n_{i,1} \leq \frac{128}{v_1 \Delta_i^U} \log(T) + 1. \tag{3.6}$$

Under event $\mathcal{A}$, when row i is eliminated, it is mapped by h^U to an item with higher parameter. Thus, the following property always holds

$$\sum_{i \in [N]} u_{h^U(i)} \geq \sum_{i \in [N]} u_i = N\mu_U.$$

We let $n_{j,w}$ denote the number of sampling steps in time window w before suboptimal

column j is eliminated. Thus, under event $\mathcal{B}_w$, according to Lemma 3.A.2, we have

$$n_{j,w} \leq \left\lceil \frac{64 \log(2^{2^w})}{(\mu_U \Delta_j^V)^2} \right\rceil .$$

If the stochastic rewards are Bernoulli random variables, arms including column j such that $v_j = 0$ return a deterministic reward of value 0. Thus, those columns are eliminated before any other. We define $v_{\min} = \min\{v_j : j \in [C], v_j > 0\}$. If the stochastic rewards are not Bernoulli random variables, then $v_{\min} = v_N$.

The following property always holds

$$\sum_{j \in V} v_{h^V(j)} \geq M \min\{v_{\min}, \mu_V\}.$$

We let $\tilde{v}_{\min} = \min\{v_{\min}, \mu_V\}$ and n_i be the number of sampling steps before row i is eliminated. The previous inequality together with Lemma 3.A.2 gives that under event $\mathcal{A}$,

$$n_i = \left\lceil \frac{64 \log(T)}{(\tilde{v}_{\min} \Delta_i^U)^2} \right\rceil .$$

This means that row i stops being sampled by the end of the first time window w_i such that $2^{2^{w_i}} > (N+M) n_i > 2^{2^{w_i - 1}}$. Thus, we have

$$2^{w_i + 1} \leq 4 \log_2(n_i) + 4 \log_2(N+M).$$

Let $R_{i,j,w}$ be the regret incurred by the sampling of pair (i,j) at a sampling step where row i and/or column j are the active rows/columns other entries are sampled against. In the special case where both row i and column j are picked, $R_{i,j,w}$ also account for the "useless" samples $(i,1)$ and $(1,j)$, useless in the sense that they are not used to update the confidence intervals related to items i and j. The following relations hold:

$$\begin{aligned}
\sum_{w=0}^{w_{\max}} R_{i,j,w} \leq & 4(u_1 v_1 - u_i v_j) \sum_{w=0}^{w_i} n_{j,w} \\
\leq & 256 \frac{\Delta_i^U + \Delta_j^V}{(\mu_U \Delta_j^V)^2} \log(2) \sum_{w=0}^{w_i} 2^w + w_i + 1 \\
\leq & 256 \frac{\Delta_i^U + \Delta_j^V}{(\mu_U \Delta_j^V)^2} \log(2) 2^{w_i + 1} + 4 w_i + 4 \\
\leq & 1024 \frac{\Delta_i^U + \Delta_j^V}{(\mu_U \Delta_j^V)^2} \log\left(\frac{64 \log(T)}{(\tilde{v}_{\min} \Delta_i^U)^2} + 1 \right) + 4 \log_2 \left(4 \log \left(\frac{64 \log(T)}{(\tilde{v}_{\min} \Delta_i^U)^2} + 1 \right) + 1 \right) \\
& + 1024 \frac{\Delta_i^U + \Delta_j^V}{(\mu_U \Delta_j^V)^2} \log(N+M) + 4 \log_2(4 \log_2(N+M)) + 4.
\end{aligned} \tag{3.7}$$

Summing up the obtained terms (3.3), (3.4), (3.6), (3.5) and (3.7), completes the proof of the theorem. □

Proof of Theorem 3.3.2 (Pure exploration):

The proof goes as follows. Let $\mathcal{A}$ be the "good event" that all high probability confidence intervals hold. Lemma 3.A.1 together with a union bound on the $N(M+1) + M$ tracked

parameters and all confidence interval's updates gives:

$$\mathbb{P}[\overline{\mathcal{A}}] \leq \delta.$$

Note that the algorithm runs in two phases: the first phase where the best row is identified, which is followed by the second phase in which the best column is identified, using sampling against the best row identified in the first phase.

At every sampling step, the expected values of the column active rows are sampled against is at least $v_{\min}$. Lemma 3.A.2 implies that rows 1 and i are guaranteed to be relatively ranked by the first sampling step where $\sum_{j=1}^{M} C(i,j) = k_l$ with:

$$\frac{1}{2^{l+1}} < \sqrt{\frac{\log(\beta_l)}{k_l}} < \frac{1}{4} v_{\min} \Delta_i^U.$$

This implies

$$\frac{1}{4} v_{\min} \Delta_i^U \leq \frac{1}{2^l}.$$

Thus, we have

$$k_l \leq \frac{32}{(v_{\min}\Delta_i^U)^2} \log\left(\frac{1}{\delta}\right) + \frac{32}{(v_{\min}\Delta_i^U)^2} \log\left(\frac{\pi^2 N(M+1)}{3}\right) + 64 \log\left(\log_2\left(\frac{1}{v_{\min}\Delta_i^U}\right) + 2\right).$$

Let τ^U be the number of sampling steps after which the best row is detected with probability at least $1-\delta$, i.e. τ^U is the length of the first phase. The previous inequality implies:

$$\tau^U \leq \frac{32}{(v_{\min}\Delta_2^U)^2} \log\left(\frac{1}{\delta}\right) + \frac{32}{(v_{\min}\Delta_2^U)^2} \log\left(\frac{\pi^2 N(M+1)}{3}\right) + 64 \log\left(\log_2\left(\frac{1}{v_{\min}\Delta_2^U}\right) + 2\right) \quad (3.8)$$

We start by bounding the number of "bad" samples, that is samples (i,j) with $j > 1$. Repeating the computations used to prove equation 3.4, the expected number of "bad" samples due to the failures of a confidence interval during any time window w is bounded as :

$$\sum_{w=0}^{w_{\max}} 2^{2^w} \mathbb{P}[\overline{\mathcal{B}}_w] \leq 2(N+1)M(\log_2(\log_2((N+M)\tau^U)) + 2). \quad (3.9)$$

At every sampling step, the expected values of the column active rows are sampled against is at least μ_U. Repeating the computations used to obtain equation 3.6, under the good event the confidence intervals hold during the successive time windows, the number of sampling steps where sub-optimal column j is active during the first phase, $\tau_j^V(1)$, is bounded as:

$$\tau_j^V(1) \leq 256 \frac{1}{(\mu_U \Delta_j^V)^2} \log\left((N+M)\tau^U\right) + \log_2\left(4\log((N+M)\tau^U)) + 1\right). \quad (3.10)$$

In a sampling step where column j is active, it is sampled at most $N+1$ times. Thus the number of "bad" samples involving column j is less than $(N+1)\tau_j^V(1)$.

The number of sampling steps where 1 is the picked column active rows are sampled against

before sub-optimal entry i is eliminated, τ_i^U, is bounded as:

$$\tau_i^U(1) \leq \frac{32}{(v_1\Delta_i^U)^2}\log\left(\frac{1}{\delta}\right) + \frac{32}{(v_1\Delta_i^U)^2}\log\left(\frac{\pi^2 N(M+1)}{3}\right) + 64\log\left(\log_2\left(\frac{1}{v_1\Delta_i^U}\right)+2\right).$$

The best row 1 is sampled at every sampling step until all other rows have been eliminated. Thus, we have

$$\tau_1^U(1) \leq \frac{32}{(v_1\Delta_2^U)^2}\log\left(\frac{1}{\delta}\right) + \frac{32}{(v_1\Delta_2^U)^2}\log\left(\frac{\pi^2 N(M+1)}{3}\right) + 64\log\left(\log_2\left(\frac{1}{v_1\Delta_2^U}\right)+2\right).$$

Also, in those sampling steps, column 1 is sampled against an active row. The number of "good" samples $(i,1)$ is thus bounded by

$$3\tau_1^U(1) + \sum_{i=2}^{N}\tau_i^U(1). \tag{3.11}$$

Similarly, under the good event that the confidence intervals hold, during the second phase of the algorithm, column $j>1$ is eliminated after $\tau_j^V(2)$ samples, with

$$\tau_j^V(2) \leq \frac{32}{(u_1\Delta_j^V)^2}\log\left(\frac{1}{\delta}\right) + \frac{32}{(u_1\Delta_i^V)^2}\log\left(\frac{\pi^2 M}{3}\right) + 64\log\left(\log_2\left(\frac{1}{u_1\Delta_j^V}\right)+2\right). \tag{3.12}$$

The best column 1 is sampled at every sampling step until all other columns have been eliminated. Thus, we have

$$\tau_1^V(2) \leq \frac{32}{(u_1\Delta_2^V)^2}\log\left(\frac{1}{\delta}\right) + \frac{32}{(u_1\Delta_2^V)^2}\log\left(\frac{\pi^2 M)}{3}\right) + 64\log\left(\log_2\left(\frac{1}{u_1\Delta_2^V}\right)+2\right). \tag{3.13}$$

Summing up equations (3.9), $(N+1)$(3.10), (3.11), (3.12) and (3.13) gives the result. □

Proof of the lower bound

Let $d(x,y) = x\log(x/y) + (1-x)\log((1-x)/(1-y))$ denote the binary relative entropy. Let F_μ denote the distribution of the stochastic reward of a row-column pair with the product of the parameters equal to μ. We admit the following assumption.

Assumption 2. *For every $p,q\in[0,1]$, and $\kappa\geq 1$ such that $\kappa p, \kappa q\in[0,1]$, it holds*

$$\mathrm{D_{KL}}(F_p||F_q) \leq \mathrm{D_{KL}}(F_{\kappa p}||F_{\kappa q}).$$

The assumption holds for normal distributions and Bernoulli distributions as explained next. Let F_{μ_1} and F_{μ_2} be two normal distribution with means μ_1 and μ_2, and variances σ_1^2 and σ_2^2. Then, we have $\mathrm{D_{KL}}(F_{\mu_1}||F_{\mu_2}) = (1/(2\sigma_2^2))(\mu_1-\mu_2)^2 + \sigma_1^2/(2\sigma_2^2) + \log(\sigma_1/\sigma_2) - 1$. From this, it is readily observed that Assumption 2 holds. Now, assume that F_{μ_1} and F_{μ_2} are two Bernoulli distributions with means μ_1 and μ_2, then by the data processing inequality in Theorem 2.2 and Proposition 6 in Polyanskiy and Wu (2014), it follows that Assumption 2 holds.

We next prove the following theorem:

Theorem 3.A.3 (lower bound)**.** *Assume that stochastic rewards of row-column pairs satisfy Assumption 2. Then, for any δ-PAC algorithm, the expected sampling complexity is lower bounded as*

$$\mathbb{E}[\tau_\delta] \geq A_F(u,v)d(\delta, 1-\delta)$$

where

$$A_F(u,v) \;:=\; \frac{1}{2}\left(\sum_{i=2}^{N}\frac{1}{\mathrm{D_{KL}}(F_{u_iv_1}||F_{(u_i+\Delta_i^U)v_1})}+\sum_{j=2}^{M}\frac{1}{\mathrm{D_{KL}}(F_{u_1v_j}||F_{u_1(v_j+\Delta_j^V)})}\right).$$

Proof: Let $N_{i,j}(t)$ denote the number of time steps in which pair (i,j) is sampled until step t, i.e.

$$N_{i,j}(t) = \sum_{s=1}^{t}\mathbb{1}_{\{(i_t,j_t)=(i,j)\}}.$$

We use the following lemma of Kaufmann et al. (2016) adapted to our setting.

Lemma 3.A.4. . *Consider two problem instances with parameters $A=(u,v)$ and $A'=(u',v')$. For any almost-surely finite stopping time σ with respect to the filtration $\mathcal{F}_t$, we have*

$$\sum_{i<j}\mathbb{E}_A\left[N_{i,j}(\sigma)\right]\mathrm{D_{KL}}(F_{u_iv_j}||F_{u'_iv'_j}) \geq \sup_{\mathcal{E}\in\mathcal{F}_\sigma} d\left(\mathbb{P}_A(\mathcal{E}),\mathbb{P}_{A'}(\mathcal{E})\right)$$

For any given $u=(u_1,\ldots,u_N)$ and $\alpha>0$, we define the following model alternatives

$$U_\alpha = \{u_{\alpha,i}=(u_1,..,u'_i,...,u_N) : u'_i = u_i+\Delta_i^U+\alpha), 1<i\leq N\}$$

and we define V_α analogously.

Let $\mathcal{E}$ denotes the event that the algorithm outputs pair $(1,1)$. Let $A_{\alpha,i}=(u_{\alpha,i},v)$. For any δ-PAC algorithm, $\mathbb{P}_A(\mathcal{E})>1-\delta$ and $\mathbb{P}_{A_{\alpha,i}}(\mathcal{E})<\delta$ (similarly for $(u,v_{\alpha,i})$). This, together with Lemma 3.A.4 gives that the solution to the following linear program is a lower bound on the sample complexity of any δ-PAC algorithm:

$$\begin{aligned}
&\mathbf{LP}_\alpha: \\
&\text{minimize } \sum_{i,j} x_{i,j} \\
&\text{subject to } \sum_{j=1}^{M} x_{i,j}\mathrm{D_{KL}}(F_{u_iv_j}||F_{u_{\alpha,i}v_j}) \geq d(\delta,1-\delta), \text{ for } i\in[N] \\
&\qquad\qquad \sum_{i=1}^{N} x_{i,j}\mathrm{D_{KL}}(F_{u_iv_j}||F_{u_iv_{\alpha,j}}) \geq d(\delta,1-\delta), \text{ for } j\in[M].
\end{aligned}$$

Let $x^*=(x^*_{i,j}:(i,j)\in[N]\times[M])$ be an optimal solution to $\mathbf{LP}_\alpha$, and let $\hat{x}=(\hat{x}_{i,j}:(i,j)\in[N]\times[M])$ be defined as follows

$$\hat{x}_{i,j} = \begin{cases} \sum_{j'=1}^{M} x^*_{1,j} + \sum_{i'=1}^{N} x^*_{i',1} & \text{for } i=1 \text{ and } j=1 \\ \sum_{i'=1}^{N} x^*_{j,i'} & \text{for } i=1 \text{ and } 1<j\leq M \\ \sum_{j'=1}^{M} x^*_{i,j'} & \text{for } 1<i\leq N \text{ and } j=1 \\ 0 & \text{otherwise.} \end{cases}$$

The vectors x^* and $\hat{x}$ satisfy the following equation

$$\sum_{i,j} \hat{x}_{i,j} = 2 \sum_{i,j} x^*_{i,j}.$$

By Assumption 2, it follows that $\hat{x}$ is an admissible solution to the following linear program:

$$\begin{aligned}
&\mathbf{LP}'_\alpha: \\
&\text{minimize } \frac{1}{2}\sum_{i=1}^{N} x_{i,1} + \frac{1}{2}\sum_{j=1}^{M} x_{1,j} \\
&\text{subject to } x_{i,1} \geq \frac{1}{\mathrm{D}_{\mathrm{KL}}(F_{u_i v_1} || F_{u_{\alpha,i} v_1})} d(\delta, 1-\delta), \text{ for } i \in [N] \\
&\qquad\qquad x_{1,j} \geq \frac{1}{\mathrm{D}_{\mathrm{KL}}(F_{u_1 v_j} || F_{u_1 v_{\alpha,j}})} d(\delta, 1-\delta), \text{ for } j \in [M].
\end{aligned}$$

Thus, the optimal solution to $\mathbf{LP}'_\alpha$ is a lower bound on the sample complexity. This lower bound holds for any $\alpha > 0$. Letting α go to zero establishes the proof of Theorem 3.A.3. □

We next show lower bounds on the sampling complexity for stochastic rewards according to normal distributions, and then after according to Bernoulli distributions.

Theorem 3.A.5. *Assume that stochastic rewards of item pairs have normal distributions with unit variance. Then, we have*

$$\mathbb{E}[\tau_\delta] \geq A(u,v) \left(\log\left(\frac{1}{\delta}\right) - 1 \right)$$

where

$$A(u,v) = \sum_{i \in [N]: \Delta_i^U > 0} \frac{1}{(v_1 \Delta_i^U)^2} + \sum_{j \in [M]: \Delta_j^V > 0} \frac{1}{(u_1 \Delta_j^V)^2}.$$

Proof: For stochastic rewards of item pairs according to normal distributions with unit variances, we have $\mathrm{D}_{\mathrm{KL}}(F_{\mu_1} \,||\, F_{\mu_2}) = (\mu_1 - \mu_2)^2/2$. This combined with the fact,

$$d(\delta, 1-\delta) = (1-2\delta) \log\left(\frac{1-\delta}{\delta}\right) \geq \log\left(\frac{1}{\delta}\right) - 1$$

yields the statement of the theorem. □

Theorem 3.A.6. *Assume that stochastic rewards of item pairs are Bernoulli random variables. Then, we have*

$$\mathbb{E}[\tau_\delta] \geq \frac{\min\{u_1 v_1, 1 - u_1 v_1\}}{4} A(u,v) \left(\log\left(\frac{1}{\delta}\right) - 1 \right).$$

Proof: If for some $\alpha \in (0, 1/2]$, $\alpha \leq q \leq 1 - \alpha$, then

$$d(p,q) \leq \frac{2}{\alpha}(p-q)^2.$$

Let us first consider the term $\mathrm{D}_{\mathrm{KL}}(F_{u_i v_1} || F_{(u_i + \Delta_i^U) v_1})$. Let $p = u_i v_1$ and $q = (u_i + \Delta_i^U) v_1$. Note that $q = u_1 v_1$. We can apply the above upper bound for the KL divergence by taking

$\alpha = \min\{u_1v_1, 1-u_1v_1\}$, and note that $(p-q)^2 = (v_1\Delta_i^U)^2$. It follows

$$\mathrm{D_{KL}}(F_{u_iv_1}||F_{(u_i+\Delta_i^U)v_1}) \leq \frac{2(v_1\Delta_i^U)^2}{\min\{u_1v_1, 1-u_1v_1\}}.$$

By the same arguments, we have

$$\mathrm{D_{KL}}(F_{u_1v_j}||F_{u_1(v_j+\Delta_j^V)}) \leq \frac{2(u_1\Delta_j^V)^2}{\min\{u_1v_1, 1-u_1v_1\}}.$$

□

3.A.2 PAIR-ELIM-MONO algorithm

Algorithm description and pseudo-code

The PAIR-ELIM-MONO algorithm is similar to the PAIR-ELIM algorithm. The main differences are in the definition of the confidence intervals and the set of active pairs. The pseudo-code of the algorithm is shown in Algorithm 16.

Algorithm 16: PAIR-ELIM-MONO

input : set of items U, precision δ (or $\frac{1}{T}$) and pure_explore

$t = 0, C = X = C_w = X_w = [0]^{2N\times 2N}, S = \{(i,j)|i \in [2N], j \in [2N], i \neq j\}$;
while $t \leq T$ **do**
  for *window* $w = 0, 1, 2, \ldots$ **do**
    $Q_+^V, Q_-^V \leftarrow$ `confidence_bound`$(X_w, C_w, 2^{2^w}, Q_+^V, Q_-^V, \text{False})$;
    $Q_+^U, Q_-^U \leftarrow$ `confidence_bound`$(X, C, T, Q_+^U, Q_-^U,$ pure_explore$)$;
    $S \leftarrow$ `active_entries`$(Q_+^U, Q_-^U, Q_+^V, Q_-^V, S)$;
    for $(i,j) \in S$ **do**
      $X(i,j) \leftarrow X(i,j) + x_{i,j,t}$;
      $C(i,j) \leftarrow C(i,j) + 1$;
      $X_w(j,i) \leftarrow X_w(j,i) + x_{i,j,t+1}$;
      $C_w(j,i) \leftarrow C_w(j,i) + 1$;
      $s \leftarrow s+2$, $t \leftarrow t+2$::;
    end
    if $s > 2^{2^w}$ **then**
      // Change time window;
      $s = 0, C_w = X_w = [0]^{N\times M}$;
    end
    if *pure_explore and* `optimal_pair`(Q_+^U, Q_-^U) **then**
      return optimal pair;
    end
  end
end

The function `confidence_bound`(X, C, h, Q_+, Q_-, p) updates Q_+ and Q_- as follows. For each $(i,j) \in U \times U$, if $C(i,j) = k_l$, with $k_l := \lceil 4^{l+1}\log(h)\rceil$ for some integer $l > 0$, then:

$$Q_-(i,j) = \frac{X(i,j)}{C(i,j)} - \sqrt{\frac{\log(h)}{k_l}} \text{ and } Q_+(i,j) = \frac{X(i,j)}{C(i,j)} + \sqrt{\frac{\log(h)}{k_l}}.$$

If $C'(i,j) + \sum_{k=1,k\neq i,j}^{2N} C(i,k) = k_l$ for some k_l, then:

$$Q_-(i,2N+j) = \frac{\sum_{k=1,k\neq i,j}^{2N} X(i,k)}{\sum_{k=1,k\neq i,j}^{2N} C(i,k)} - \sqrt{\frac{\log(h)}{k_l}}$$

and

$$Q_+(i,2N+j) = \frac{\sum_{k=1,k\neq i,j}^{2N} X(i,k)}{\sum_{k=1,k\neq i,j}^{2N} C(i,k)} + \sqrt{\frac{\log(h)}{k_l}}.$$

The value used for parameter β_l depends on h and p. In the case where p is false, $\beta_l = h$. When p is true $\beta_l = \pi\sqrt{4N(2N-1)h/3}\cdot l$

The function `active_entries` updates S as follows. For any item i:

- if there exist two entries j and j' s.t. for some k, $Q_+^U(i,k) < Q_-^U(j,k)$ and for some k', $Q_+^U(i,k') < Q_-^U(j',k')$, then all entries (i,l) and (l,i) are removed from S.
- if there exist two entries j and j' s.t. for some k, $Q_+^V(i,k) < Q_-^V(j,k)$ and for some k', $Q_+^V(i,k') < Q_-^V(j',k')$, then all entries (l,i) are removed from S.
- if there exist an entry j s.t. for some k, $Q_+^U(i,k) < Q_-^U(j,k)$ then all entries (i,l) and (l,i) are removed from S except for (i,j) and (j,i).
- if there exist an entry j s.t. for some k, $Q_+^V(i,k) < Q_-^V(j,k)$ then all entries (l,i) are removed from S except for (j,i).

Proof of the upper bounds

Many computational steps of the two following proofs are similar to those given in Appendix3.A.1 and are not repeated here.

Proof of Theorem 3.3.3 (Regret) The proof goes as follows. Pair-Elim-Mono algorithm tracks $2N(N-1)$ row parameters, and there are less than T confidence interval updates. Thus, Lemma 3.A.1 together with a union bound yield the following bound

$$\mathbb{P}[\overline{\mathcal{A}}] \leq 4\frac{N(N-1)}{T}. \tag{3.14}$$

Similarly, we have

$$\sum_{w=0}^{w_{\max}} 2^{2^w}\mathbb{P}[\overline{\mathcal{B}}_w] \leq 4N(N-1)(\log_2(\log_2(T)) + 2). \tag{3.15}$$

From Lemma 3.A.2, under event $\mathcal{A}$ and $\mathcal{B}_w$, the number of sampling steps where column j is sampled against row 1 or row 2, $n_{1,j,w}$ and $n_{2,j,w}$, during time window w are bounded as

$$n_{1,j,w} \leq \left\lceil \frac{64\log(2^{2^w})}{(u_1\Delta_{2,j})^2} \right\rceil \text{ and } n_{2,j,w} \leq \left\lceil \frac{64\log(2^{2^w})}{(u_2\Delta_{1,j})^2} \right\rceil.$$

Thus, the total regret for sampling of pair $(1,j)$, $R_{1,j}(T)$, is bounded as

$$R_{1,j}(T) \leq \frac{512}{u_1\Delta_{2,j}}\log(T) + 2\log_2(\log_2(T)) + 4 \tag{3.16}$$

and the fact $u_1\Delta_{2,i} < u_2\Delta_{1,i}$ gives

$$R_{2,j}(T) \leq R_{1,j}(T). \tag{3.17}$$

Similarly, we have

$$R_{i,1}(T) \leq \frac{128}{u_1 \Delta_{2,i}} \log(T) + 1 \tag{3.18}$$

and

$$R_{i,2}(T) \leq R_{i,1}(T). \tag{3.19}$$

Under assumption that event $\mathcal{A}$ occurs, rows 1 and 2 are active at every iteration. Thus, at every sampling step, column j is sampled against the two optimal rows and is definitely eliminated at most once it is found smaller than column 2, which gives

$$n_{j,w} \leq \left\lceil \frac{64 \log(2^{2^w})}{(u_1 \Delta_{2j})^2} \right\rceil.$$

Consider a doubly sub-optimal pair (i, j) with $i > j$. Doubly sub-optimal pair (i, j) definitely stops being sampled as soon as i has been deemed smaller than any other item $\neq j$. Before this happens, i can be compared with 1 or 2 against at least one of the active columns. The number of sampling steps before doubly sub-optimal pair (i, j) is definitely eliminated is thus upper bounded as

$$n_{ij} = \left\lceil \frac{128 \log(T)}{(\tilde{u}_{\min} \Delta_{2,i})^2} \right\rceil.$$

with $\tilde{u}_{\min} = \min\{u_{\min}, \mu_U\}$, $u_{\min} = \min\{u_j : j \in [C], u_j > 0\}$.

Consider a doubly sub-optimal pair (i, j) with $i < j$. It may happen that the order $i < j$ is discovered before any other. In that case, i is only sampled against j and can no longer be compared with either 1 or 2 at every sampling step, since j might get eliminated as a column. However, doubly sub-optimal pair (i, j) is still definitely eliminated as soon as j is deemed smaller than any other entry, and number of sampling steps before doubly sub-optimal entry (i, j) is definitely eliminated is thus upper bounded as

$$n_{ij} = \left\lceil \frac{128 \log(T)}{(\tilde{u}_{\min} \Delta_{2,j})^2} \right\rceil.$$

Thus, the regret for the sampling of a doubly sub-optimal entry (i, j) is bounded as

$$\begin{aligned} \sum_{w=0}^{w_{\max}} R_{i,j,w} \leq & 512 \frac{\Delta_{2,i} + \Delta_{2,j}}{(u_1 \Delta_{2,j})^2} \log\left(\frac{128}{(\tilde{u}_{\min} \min\{\Delta_{2,i}, \Delta_{2,j}\})^2} \log(T) + 1 \right) \\ & + \log_2\left(4 \log\left(\frac{128 \log(T)}{(\tilde{u}_{\min} \min\{\Delta_{2,i}, \Delta_{2,j}\})^2} + 1 \right) + 1 \right) \\ & + 512 \frac{\Delta_i^U + \Delta_j^V}{u_1 \Delta_{2,j}^2} \log\left(4N(2N-1)\right) + 4 \log_2(4 \log_2(4N(2N-1))) + 4. \end{aligned} \tag{3.20}$$

The results follows from equations (3.14), (3.15), (3.16), (3.17), (3.18), (3.19) and (3.20).□

Proof of Theorem 3.3.4 (Pure Exploration): The proof goes as follows. Let $\mathcal{A}$ be the "good event" that all high probability confidence intervals hold. Lemma 3.A.1 together with a union bound on the $4N(2N-1)$ tracked parameters and all confidence interval updates gives

$$\mathbb{P}[\overline{\mathcal{A}}] \leq \delta.$$

Let τ^U be the number of sampling steps after which the best pair is detected with probability at least $1-\delta$. The following holds:

$$\begin{aligned}\tau^U &\leq \frac{64}{(\tilde{u}_{\min}\Delta_{23})^2}\log\left(\frac{1}{\delta}\right)+\frac{64}{(\tilde{u}_{\min}\Delta_{23})^2}\log\left(\frac{\pi^2 N(M+1)}{3}\right)\\ &\quad +128\log_2\left(\log_2\left(\frac{1}{\tilde{u}_{\min}\Delta_{23}}\right)+2\right)\end{aligned}$$

We call bad samples the sampled pairs (i,j) with $j>2$. The expected number of "bad" samples due to the failures of a confidence interval during any time window w is bounded as

$$\sum_{w=0}^{w_{\max}} 2^{2^w}\mathbb{P}[\overline{\mathcal{B}}_w] \leq 2N(2N-1)(\log_2(\log_2(2N(2N-1)\tau^U))+2).$$

Under the good event the confidence intervals hold, the number of sampling steps where column j is sampled, τ_j, $j>2$, is bounded as

$$\tau_j \leq \frac{256}{(u_1\Delta_{2j})^2}\log\left(2N(2N-1)\tau^U\right)+\log_2\left(4\log(2N(2N-1)\tau^U))+1\right).$$

In each of those sampling steps, column j is sampled less than $2(2N-1)$ times.

Let τ_2 be the number of samples to detect 1 when sampling against 2. The following holds

$$\begin{aligned}\tau_2 &\leq 64\left(\sum_{i=3}^{N}\frac{1}{(u_2\Delta_{1,i})^2}\right)\log\left(\frac{1}{\delta}\right)\\ &\quad +128\left(\sum_{i=2}^{N}\frac{1}{(u_2\Delta_{1,i})^2}\left(\frac{1}{2}\log\left(\frac{\pi^2 2N(2N-1)}{3}\right)+\log_2\left(\log_2\left(\frac{1}{u_2\Delta_{1,i}}\right)+2\right)\right)\right).\end{aligned}$$

Let τ_1 be the necessary number of samples to detect 2 when sampling against 1. The following holds

$$\begin{aligned}\tau_1 &\leq 64\left(\sum_{i=3}^{N}\frac{1}{(u_1\Delta_{2,i})^2}\right)\log\left(\frac{1}{\delta}\right)\\ &\quad +128\left(\sum_{i=3}^{N}\frac{1}{(u_1\Delta_{2,i})^2}\left(\frac{1}{2}\log\left(\frac{\pi^2 2N(2N-1)}{3}\right)+\log_2\left(\log_2\left(\frac{1}{u_1\Delta_{2,i}}\right)+2\right)\right)\right).\end{aligned}$$

The inequality $u_1\Delta_{2,i}\leq u_2\Delta_{1,i}$ gives $\tau_2\leq\tau_1$.

Under event $\mathcal{A}$, the total number of samples is less than $4\tau_1+2(2N-1)\tau_j+\sum_{w=0}^{w_{\max}}2^{2^w}\mathbb{P}[\overline{\mathcal{B}}_w]$, which gives the result. □

Proof of the lower bound As in Appendix 3.A.1, we admit Assumption 2. We prove the following theorem.

Theorem 3.A.7. *For any δ-PAC algorithm:*

$$\mathbb{E}[\tau_\delta] \geq \frac{1}{2}\left(\sum_{i=2}^{n} \frac{1}{d\left(\theta_i\theta_1, (\theta_i + \Delta_{1,i} + \alpha)\theta_1\right)}\right) d(\delta, 1-\delta) \tag{3.21}$$

and

$$\mathbb{E}[\tau_\delta] \geq \frac{1}{2}\left(\sum_{i=3}^{n} \frac{1}{\tilde{d}\left(\theta_i\theta_2, \theta_1\theta_2\right)}\right) d(\delta, 1-\delta) \tag{3.22}$$

where $\tilde{d}\left(u_iu_2, u_1u_2\right) = \min\{d\left(u_iu_2, u_1u_2\right), d\left(u_1u_2, u_iu_2\right)\}$.

We prove (3.21) by considering the classes of alternative models:

$$U_\alpha = \{\boldsymbol{u}_i^\alpha = (u_1, .., u_i', ..., u_n) \text{ with } u_i^\alpha = u_i \pm (\Delta_i + \alpha) \mid i \in [2, N]\}.$$

Let $\mathcal{E}_{\mathcal{M}_u}$ denote the event that the algorithm outputs pair $(1, 2)$. For any δ-PAC algorithm, $\mathbb{P}_u(\mathcal{E}_{\mathcal{M}_u}) \geq 1-\delta$ and $\mathbb{P}_{\boldsymbol{u}_i^\alpha}(\mathcal{E}_{\mathcal{M}_u}) \leq \delta$. This together with Lemma 3.A.4 gives that the solution to the following linear program is a lower bound on the sample complexity of any $\delta - PAC$ algorithm:

$$\begin{aligned}
&\mathbf{LP}_\alpha: \\
&\text{minimize } \sum_{i<j} x_{i,j} \\
&\text{subject to } \sum_{j=1}^{i-1} x_{j,i} d\left(u_iu_j, u_i'u_j\right) + \sum_{j=i+1}^{2N} x_{i,j} d\left(u_iu_j, u_i^\alpha u_j\right) \geq d(\delta, 1-\delta),\ i \in \{2, \ldots, 2N\}
\end{aligned}$$

Let $x^* = (x_{i,j}^* : (i,j) \in [2N] \times [2N])$ be an optimal solution to $\mathbf{LP}_\alpha$, and let $\hat{x} = (\hat{x}_{i,j} : (i,j) \in [2N] \times [2N])$ be defined as follows

$$\hat{x}_{i,j} = \begin{cases} \hat{x}_{1,i} = \sum_{j=1}^{i-1} x_{j,i}^* + \sum_{j=i+1}^{N} x_{i,j}^* & \text{for } i \in \{2, \ldots, 2N\} \\ \hat{x}_{l,i} = 0 & \text{for } l > 1,\ i \in \{2, \ldots, 2N\} \\ 0 & \text{otherwise.} \end{cases}$$

x^* and $\hat{x}$ satisfy the following equation

$$\sum_{i,j} \hat{x}_{i,j} = 2 \sum_{i,j} x_{i,j}^*.$$

Thus, the solution of the following linear program is a lower bound on the sample complexity:

$$\begin{aligned}
&\mathbf{LP}_\alpha': \\
&\text{minimize } \frac{1}{2}\sum_{i=2}^{n} x_{i,1} \\
&\text{subject to } x_{i,1} \geq \frac{1}{d\left(u_iu_1, u_i^\alpha u_1\right)} d(\delta, 1-\delta),\ i \in \{2, \ldots, 2N\}
\end{aligned}$$

Equation (3.22) is obtained by considering the class of alternative models:

$$u_{\text{switch}} = \{\boldsymbol{u}_i = (u_i, .., u_1, ..., u_n) \mid i \in \{3, \dots, 2N\}\}.$$

Considering the event $\mathcal{E}_{\mathcal{M}_u}$ and Lemma 3.A.4 , we get that the solution of the following linear program is a lower bound on the sample complexity of any $\delta - PAC$ algorithm:

$\mathbf{LP}_{\text{switch}}$:

$$\text{minimize } \sum_{i<j} x_{i,j}$$

$$\text{subject to } \sum_{j=2, j\neq i}^{n} x_{1,j} d\left(u_1 u_j, u_i u_j\right) + \sum_{j=2, j\neq i}^{n} x_{j,i} d\left(u_i u_j, u_1 u_j\right) \geq d(\delta, 1-\delta),\ i \in \{3, \dots, 2N\}$$

Using the same technique, from an optimal solution x^* we can build an alternative solution that satisfies the constraints.

$$\forall i \in \{1\} \cup [3, N]: \ \hat{x}_{2,i} = \sum_{j=2}^{i-1} x_{j,i} + \sum_{j=i+1}^{N} x_{j,i}$$
$$\hat{x}_{i,j} = 0, j \in \{3, \dots, 2N\}$$

As before, we also have

$$\sum_{i,j} \hat{x}_{i,j} = 2 \sum_{i,j} x^*_{i,j}.$$

Thus, the optimal solution to the following linear program is a lower bound on the sampling complexity:

$$\text{minimize } \frac{1}{2}\left(x_{1,2} + \sum_i x_{2,i}\right)$$
$$\text{subject to } x_{i,2} d\left(u_i u_2, u_1 u_2\right) + x_{1,2} d\left(u_1 u_2, u_i u_2\right) \geq d(\delta, 1-\delta),\ i \in \{3, \dots, 2N\}$$

which gives Equation (3.22).

As in Appendix 3.A.1, specific lower bounds can be obtained for Bernoulli and unit-variance Gaussian random variables.

3.A.3 Pair-Select algorithm

Algorithm description and pseudo-code

Algorithm 17: Pair-Elim-Mono

input : set of items U, precision δ

1 Detect items $1, 2$ with Pair-Elim-Mono;

2 Sample unranked items against items $1, 2$;

Proof of the upper bound

The proof goes as follows. Recall the following definitions:

$$\Delta_{2,i} = u_{2i} - u_{2i+1} \quad \text{and} \quad \Delta_{2i-1} = u_{2i-2} - u_{2i-1}.$$

with the convention $u_{2N+1} = u_{2N-2}$.

During the second phase of the algorithm, at least half of the gathered samples involving item i are pairs $(1, i)$. Repeating the computations used to obtain Equation (3.8), the number of samples gathered during the second phase of the algorithm, τ_m, is bounded as:

$$\begin{aligned}\tau_m \quad \leq \quad & 64 \left(\sum_{i=3}^{N} \frac{1}{(u_1 \Delta_i)^2} \right) \log\left(\frac{1}{\delta}\right) \\ & +128 \left(\sum_{i=3}^{N} \frac{1}{(u_1 \Delta_i)^2} \left(\frac{1}{2} \log\left(\frac{\pi^2 2N(2N-1)}{3} \right) + \log_2 \left(\log_2 \left(\frac{1}{u_1 \Delta_i} \right) + 2 \right) \right) \right).\end{aligned}$$

The upper bound on the number of samples to detect 1 and 2, is the same as the upper bound on the sample complexity of Pair-Elim-Mono. Noticing that $u_1 \Delta_i \leq u_1 \Delta_{2,i}$ and summing the bounds on the number of samples during each of the two phase completes the proof. □

Proof of the lower bound

Note that the lower bound of Theorem 3.A.7 still holds.

To ease the notations, we will note

$$u_{2i} \pm \Delta_{2,i} = u_{2i+1} \quad \text{and} \quad u_{2i-1} \pm \Delta_{2i-1} = u_{2i-2}.$$

Theorem 3.A.8. *For any δ-PAC algorithm, we have*

$$\mathbb{E}[\tau_\delta] \geq \frac{1}{2} \left(\sum_{i=2}^{n} \frac{1}{d\left(u_i u_1, (u_i \pm \Delta_i) u_1\right)} \right) d(\delta, 1-\delta) \tag{3.23}$$

We prove the theorem by considering the following classes of alternative models:

$$u_\alpha = \{ \boldsymbol{u}_i^\alpha = (u_1, .., u_i', ..., u_{2N}) \text{ with } u_i^\alpha = u_i \pm (\Delta_i + \alpha) \mid i \in \{2, \ldots, 2N\} \}.$$

Let $\mathcal{E}_{\mathcal{M}_u}$ denote the event that the algorithm returns the optimal matching. For any δ-PAC algorithm, $\mathbb{P}_u(\mathcal{E}_{\mathcal{M}_u}) \geq 1 - \delta$ and $\mathbb{P}_{\boldsymbol{u}_i^\alpha}(\mathcal{E}_{\mathcal{M}_u}) \leq \delta$. Using this with 3.A.4 we obtain that the solution to the following linear program is a lower bound on the sample complexity of any δ-PAC algorithm:

$$\begin{aligned}
&\mathbf{LP}_\alpha: \\
&\text{minimize } \sum_{i<j} x_{i,j} \\
&\text{subject to } \sum_{j=1}^{i-1} x_{j,i} d\left(u_i u_j, u_i' u_j\right) + \sum_{j=i+1}^{2N} x_{i,j} d\left(u_i u_j, u_i^\alpha u_j\right) \geq d(\delta, 1-\delta),\ i \in \{2, \ldots, 2N\}
\end{aligned}$$

Let $x^* = (x^*_{i,j})$ be an optimal solution to $\mathbf{LP}_\alpha$.
We can build an alternative solution satisfying the constraints of the linear program:

$$\begin{aligned}
\forall i \in \{2, \ldots, 2N\}:\ \hat{x}_{1,i} &= \sum_{j=1}^{i-1} x^*_{j,i} + \sum_{j=i+1}^{2N} x^*_{i,j} \\
\hat{x}_{l,i} &= 0, \forall l > 1.
\end{aligned}$$

We have

$$\sum_{i,j} \hat{x}_{i,j} = 2 \sum_{i,j} x^*_{i,j}.$$

Thus, the solution of the following linear program is a lower bound on the sample complexity:

$$\begin{aligned}
&\mathbf{LP}'_\alpha \\
&\text{minimize } \frac{1}{2} \sum_{i=2}^{2N} x_{i,1} \\
&\text{subject to } x_{i,1} \geq \frac{1}{d\left(u_i u_1, u_i^\alpha u_1\right)} d(\delta, 1-\delta),\ i \in \{2, \ldots, 2N\}
\end{aligned}$$

This lower bound holds for any $\alpha > 0$, so we can let α go to zero and get Equation (3.A.6).

As in Appendix 3.A.1, specific lower bounds can be obtained for Bernoulli and unit-variance Gaussian random variables.

3.A.4 SIMPLE-ADAPTIVE-MATCHING algorithm

Algorithm description and pseudo-code

The items are split into even clusters with high probability. The `sample_matching` procedure samples the items in the clusters following a round-robin tournament. For a cluster S, the round-robin tournaments guarantees that each item has been matched once with any other item in the cluster each $|S| - 1$ iterations.

The goal of the `sample_matching` procedure is to ensure that at every iteration, any two items i, i' in the same cluster S have been matched the same number of times with any other item. This means that, for any item $j \in [2N] \setminus \{i, i'\}$, $C(i,j) = C(i',j)$, which guarantees:

$$\frac{\mathbb{E}[X(i,j) - X(i',j)]}{C(i,j)} = u_j \Delta_{i,i'}. \tag{3.24}$$

Algorithm 18: SIMPLE-ADAPTIVE-MATCHING

input : set of items $[2N]$ and horizon T

$t = 0, C = X = \tilde{C} = \tilde{X} = [0]^{2N \times 2N}, \mathfrak{S} = \{[2N]\}$;
for $t = 1 \dots T$ **do**
  $m_t \leftarrow$ `sample_matching`(S, t);
  for $(i, j) \in m_t$ **do**
    $\tilde{X}(i, j) \leftarrow \tilde{X}(i, j) + x_{i,j,t}$;
    $\tilde{C}(i, j) \leftarrow \tilde{C}(i, j) + 1$;
  end
  for $S \in \mathfrak{S}$ **do**
    if $\exists i \in S$ *s.t.* $\sum_{j \in S} \tilde{C}(i, j) = |S| - 1$ **then**
      // the notation $X([S], :)$ is a vectorized numpy index notation;
      $X([S], :), C([S], :) += \tilde{X}([S], :), \tilde{C}([S], :)$;
      $\tilde{X}([S], :), \tilde{C}([S], :) = 0$;
    end
  end
  $Q_+, Q_- \leftarrow$ `confidence_bound`(X, C, T, Q_+, Q_-, S);
  for $S \in \mathfrak{S}$ **do**
    Order items in s according to Q_+;
    for $i \in \{2, \dots, |S|\}$ **do**
      if $Q_+[i] < Q_-[i-1]$ **then**
        Split S between i and $i - 1$;
      end
    end
  end
end

This implies that it is possible to compare to items within the same cluster by comparing the total reward received for both of those items.

We define an ordered sequence of clusters as a sequence $S_k, \dots, S_{k'}$ such that, for any $l \in \{k, \dots, k' - 1\}$

$$\forall (i, j) \in S_l \times S_{l+1} : \ i < j.$$

At every iteration, $\mathfrak{S}$ is an ordered sequence of clusters. We note $\mathfrak{S}[k]$ the k^{th} cluster of the sequence.

The function `confidence_bound` computes confidence bounds for the total reward per item when matched with an item within the same cluster or in a lower ranked cluster. Consider any item $i \in [2N]$ and k such that $i \in \mathfrak{S}[k]$. If $\sum_{j=1}^{2N} C(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}} = k_l$ for some l, then

$$Q_-(i) = \frac{\sum_{j=1}^{2N} X(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l} - \sqrt{\frac{\log(T)}{k_l}}$$

and

$$Q_+(i) = \frac{\sum_{j=1}^{2N} X(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l} + \sqrt{\frac{\log(T)}{k_l}}.$$

Equation 3.24 implies that for any two items i, i' ins the same cluster $\mathfrak{S}[k]$:

$$\mathbb{E}[\frac{\sum_{j=1}^{2N}(X(i,j)-X(i,j))\mathbb{1}_{\{j\in S[k']|k'\leq k\}}}{k_l}] = \Delta_{i,i'}\frac{\sum_{j=1}^{2N}u_jC(i,j)\mathbb{1}_{\{j\in S[k']|k'\leq k\}}}{k_l}$$
$$\geq \Delta_{i,i'}\frac{1}{|\mathfrak{S}[k]|-1}\sum_{j\in\mathfrak{S}[k]\setminus\{i,i'\}}u_j \tag{3.25}$$

Proof of the upper bound

The proof goes as follows. The SIMPLE-ADAPTIVE-MATCHING algorithm tracks the expected rewards per item. Lemma 3.A.1 together with a union bound on the $2N$ parameters and the confidence interval updates implies

$$\mathbb{E}[R(T)] \leq \mathbb{E}[R(T)\mid\mathcal{A}] + T\mathbb{P}(\bar{\mathcal{A}}) \leq \sum_{i=1}^{N}\sum_{j=i+1}^{N}R_{i,j}(T) + 4N.$$

We call inter-pair match between pair i and pair j the match of an item of pair i with an item of pair j. The per iteration average regret of a tournament on a cluster S can be decomposed into the sum of regrets of inter-pair matches.

Lemma 3.A.9. *Assume that items in $S = \{2k-1,\dots,2l\}$, for $1\leq k<l\leq K$, such that $|S| = 2K$, are matched according to a round-robin tournament, so that every pair (i,j) with $i\neq j$ is matched exactly once over $2K-1$ iterations. Then, the expected average regret, denoted as R_S, is given as*

$$R_S = \frac{1}{2K-1}\sum_{i=1}^{K}\sum_{j=i+1}^{K}r_{i,j}$$

where

$$r_{i,j} = (u_{2i}-u_{2j-1})(u_{2i-1}-u_{2j}) + (u_{2i-1}-u_{2j-1})(u_{2i}-u_{2j}).$$

This proof of Lemma 3.A.9 is given in Appendix 24.

Lemma 3.A.10. *The total number of sampled matchings T_S before the set of items $[2K]$ is separated is upper bounded as*

$$T_S \leq \frac{64}{(\tilde{\mu}_{-\{k,k+1\}}\Delta_{k,k+1})^2}\log(T) + 2K - 1$$

for all $k\in[2K-1]$,where

$$\tilde{\mu}_{-\{k,k+1\}} = \frac{1}{2K-1}\left(\sum_{i=1}^{2K}u_i - u_k - u_{k+1}\right). \tag{3.26}$$

The last lemma follows from Lemma 3.A.2 and equation 3.25.

Let $R_{i,j}(T)$ denote the total regret incurred due to interactions between pairs i and j, written

$$R_{i,j}(T) = \sum_{t=1}^{T}\sum_{\{k,l\}\in\{2i-1,2i\}\times\{2j-1,2j\}}\mathbb{1}_{\{(k,l)\in m_t\}}\left(\frac{1}{2}(u_{2i-1}u_{2i}+u_{2j-1}u_{2j}) - u_lu_k\right).$$

Let $\mathcal{E}_{i,j}(S)$ denote the event that all inter-pair matches between pairs i and j are sampled

before S is separated. Note that

$$\mathbb{E}[R_{i,j}(T) \mid \mathcal{E}_{i,j}(S)] \leq \frac{T_S}{|S|-1} r_{i,j}$$

which with some computations yield the following lemma.

Lemma 3.A.11. *Assume that all inter-pair matches between pairs i and j are sampled before S is separated, for a set S such that $|S| = 2K$ and $K \geq 3$. Then under event $\mathcal{A}$, the total regret due to inter-pair matches between pairs i and j is upper bounded as*

$$\mathbb{E}[R_{i,j}(T) \mid \mathcal{E}_{i,j}(S)] \leq \frac{640}{K-2} \frac{1}{\Delta_{\min}} \log(T) + 2.$$

If $K = 2$, the bound is

$$\mathbb{E}[R_{i,j}(T) \mid \mathcal{E}_{i,j}(S)] \leq 384 \frac{1}{\Delta_{\min}} \log(T) + 2.$$

The proof of Lemma 3.A.11 is given in Appendix 24.

The total regret incurred for inter-pair matches between pairs i and j is bounded as

$$\mathbb{E}[R_{i,j}(T)] \leq \max_{S \subseteq [2N]: (i,j) \in S} \mathbb{E}[R_{i,j}(T) \mid \mathcal{E}_{i,j}(S)].$$

Adjacent pairs can interact in a cluster S with $|S| = 2K$ and either $2 < K \leq N$ or $K = 2$. The total regret for interactions between pair i and pair $i+1$ is thus upper bounded by Lemma 3.A.11 as

$$\begin{aligned} R_{i,i+1}(T) &\leq \max_{S \subset [2N]: (i,i+1) \in S} \mathbb{E}[R_{i,i+1}(T) \mid \mathcal{E}_{i,i+1}(S)] \\ &\leq \max \left\{ 384, \max_{2 < K \leq N} \frac{640}{K-2} \right\} \frac{\log(T)}{\Delta_{\min}} + 2 \\ &\leq \frac{640}{\Delta_{\min}} \log(T) + 2. \end{aligned}$$

All non-adjacent items can only i and j interact in a cluster s.t. $K \geq j - i + 1 \geq 3$. Thus, Lemma 3.A.11 gives

$$R_{i,j}(T) \leq \frac{640}{(j-i-1)\Delta_{\min}} \log(T) + 2.$$

Putting the pieces together, we have

$$\begin{aligned} R(T) &\leq \left(N + \sum_{i=1}^{N} \sum_{j=i+2}^{N} \frac{1}{j-i-1} \right) \frac{640 \log(T)}{\Delta_{\min}} + N^2 \\ &\leq \left(N + \sum_{i=1}^{N} \sum_{k=1}^{N-i-1} \frac{1}{k} \right) \frac{640 \log(T)}{\Delta_{\min}} + N^2 \\ &\leq \left(2N + \sum_{i=1}^{N} \log(N-i-1) \right) \frac{640 \log(T)}{\Delta_{\min}} + N^2 \\ &\leq 640 \frac{N(\log(N)+2)}{\Delta_{\min}} \log(T) + N^2. \end{aligned}$$

Proof of Lemma 3.A.9 The items in set $S = \{2k-1, \dots, 2l\}$ are matched according to a round-robin tournament. This means that each pair (i, j) such that $i \neq j$ is matched exactly once over $2K-1$ iterations. Therefore, the total expected reward over $2K-1$ iterations is:

$$\sum_{i=2k-1}^{2l} \sum_{j=i+1}^{2l} u_i u_j.$$

And the optimal way to match items in set S gives the expected reward

$$\sum_{i=k}^{l} u_{2i-1} u_{2i}.$$

It follows that the expected regret per iteration over $2K-1$ iterations is

$$\begin{aligned}
R_S &= \sum_{i=k}^{l} u_{2i-1}u_{2i} - \frac{1}{2(2K-1)} \sum_{i=2k-1}^{2l} \sum_{j=2k-1, j\neq i}^{2l} u_i u_j \\
&= \frac{1}{2(2K-1)} \sum_{i=l}^{p} \sum_{j=k, j\neq i}^{l} [u_{2i-1}(u_{2i} - u_{2j-1}) + u_{2i-1}(u_{2i} - u_{2j}) \\
&\qquad + u_{2i}(u_{2i-1} - u_{2j-1}) + u_{2i}(u_{2i-1} - u_{2j})] \\
&= \frac{1}{2K+1} \sum_{i=k}^{l} \sum_{j=i+1}^{K-1} r_{i,j}.
\end{aligned}$$

Proof of Lemma 3.A.11 We first compute the bound for a cluster S of size $2K$ where $K > 2$, and then consider the case $K = 2$. Without loss of generality, we enumerate the items as $1, \dots, 2K$.

Case: $K \geq 3$ Consider any two pairs i and j cluster S. The total regret for interactions between the two pairs until time T_S is bounded as follows:

$$\begin{aligned}
\mathbb{E}[R_{i,j}(T) \mid \mathcal{E}_{i,j}(S)] &= \sum_{t=1}^{T_S} \sum_{\{k,l\} \in \{2i-1,2i\} \times \{2j-1,2j\}} \mathbb{1}_{\{\{u_k, u_l\} \in m_t\}} \left(\frac{1}{2}(u_{2i-1}u_{2i} + u_{2j-1}u_{2j}) - u_l u_k \right) \\
&\leq \frac{T_S}{2K-1} r_{i,j} \\
&\leq \frac{T_S}{2K-1} r_{1,K} \\
&\leq \frac{T_S}{2K-1} \left[(u_1 - u_{2K})(u_2 - u_{2K-1}) + (u_1 - u_{2K-1})(u_2 - u_{2K}) \right] \\
&\leq \frac{T_S}{2K-1} 2\Delta^2_{2,2K-1}
\end{aligned}$$

where the last inequality holds by Assumption 1.

Let $m = \arg\max\{\Delta_{k,k+1} : 1 < k \leq 2K-1\}$ and $\gamma = \min_{i \in [K-2]} \Delta_{2i,2i+1}$. Note that

$$\Delta_{2,2K-1} \leq (K-1)\Delta_{m,m+1} \text{ and } \Delta_{\min} \leq \gamma^2$$

and

$$u_k \geq u_{2N} + \sum_{i=\lceil k/2 \rceil}^{N-1} \Delta_{2i,2i+1}.$$

Recall the definition of $\mu_{-m,m+1}$ in Equation (3.26). We have

$$\begin{aligned}(2K-1)\tilde{\mu}_{-\{m,m+1\}} &= \sum_{k=1}^{2K} u_k - (u_m + u_{m+1}) \\ &\geq \sum_{k=3}^{2K} u_k \geq (2K-2)u_{2K} + \sum_{k=3}^{2K} \sum_{i=\lceil \frac{k}{2} \rceil}^{N-1} \gamma \\ &\geq (2K-2)u_{2K-1} + (K-1)(K-2)\gamma \\ &\geq (K-1)(K-2)\gamma\end{aligned}$$

Combining with Lemma 3.A.10, we have

$$T_S \leq \frac{64(2K-1)^2}{(K-1)^2(K-2)^2\gamma^2\Delta_{m,m+1}^2} \log(T) + 2K - 1.$$

Therefore, we have

$$\begin{aligned}R_{i,j}(T) &\leq 128 \frac{(K-1)^2\Delta_{m,m+1}^2(2K-1)^2}{(2K-1)(K-1)^2(K-2)^2\gamma^2\Delta_{m,m+1}^2} \log(T) + 2 \\ &\leq 128 \frac{(2K-1)}{(K-2)^2\gamma^2} \log(T) + 2 \\ &\leq \frac{640}{(K-2)\Delta_{\min}} \log(T) + 2\end{aligned}$$

which completes the proof.

Case: $K = 2$ Under Assumption 1, the regret can be bounded as follows

$$\begin{aligned}R_{0,1}(T) &\leq 64 \frac{2 \cdot 3\log(T)(u_2 - u_3)^2}{(u_1 + u_4)^2(u_1 - u_2)^2} + 2 \\ &\leq \frac{384\log(T)}{(u_1 + u_4)^2} + 2 \leq \frac{384\log(T)}{(u_2 - u_3)^2} + 2 \\ &\leq \frac{384\log(T)}{\Delta_{\min}} + 2.\end{aligned}$$

3.A.5 Adaptive-Matching algorithm

Pseudo-code

The main difficulty when removing Assumption 1, is that there is no reason to believe the created clusters will contain an even number of items. For instance, set of items $\{1, \ldots, 6\}$ may be split into $\{1, 2, 3\}, \{4, 5, 6\}$. In that case, items can no longer be matched within the cluster they belong to alone. To tackle this difficulty, we introduce the Sample-Matching procedure. As in the simplified setting, this procedure guarantees that two items within a cluster are matched the same number of time with any other item.

The rest of the algorithm is similar to the simple-Adaptive-Matchingalgorithm.

Algorithm 19: ADAPTIVE-MATCHING

input : set of items $[2N]$ and horizon T

$t = 0, C = X = \tilde{C} = \tilde{X} = [0]^{2N \times 2N}, \mathfrak{S} = \{[2N]\}$;
for $t = 1 \dots T$ **do**
  $m_t \leftarrow$ SAMPLE-MATCHING$(\mathfrak{S}, t)$;
  for $(i, j) \in m_t$ **do**
    $\tilde{X}(i, j) \leftarrow \tilde{X}(i, j) + x_{i,j,t}$;
    $\tilde{C}(i, j) \leftarrow \tilde{C}(i, j) + 1$;
    $\tilde{X}(j, i) \leftarrow \tilde{X}(j, i) + x_{i,j,t}$;
    $\tilde{C}(j, i) \leftarrow \tilde{C}(j, i) + 1$;
  end
  for $S \in \mathfrak{S}$ **do**
    // $r(S)$and $L(S)$ defined in the description of SAMPLE-MATCHING;
    if $\exists i \in S$ *s.t.* $\sum_{j \in S} \tilde{C}(i, j) = r(S)|L(S)|$ **then**
      $X([S], :), C([S], :) += \tilde{X}([S], :), \tilde{C}([S], :)$;
      $\tilde{X}([S], :), \tilde{C}([S], :) = 0$;
    end
  end
  $Q_+, Q_- \leftarrow$ `confidence_bound`$(X, C, T, Q_+, Q_-, \mathfrak{S})$;
  for $S \in \mathfrak{S}$ **do**
    Order items in S according to Q_+;
    for $i \in [2, |s|]$ **do**
      if $Q_+[s(i)] < Q_-[s(i-1)]$ **then**
        Split S between $S(i)$ and $S(i-1)$;
        if *S is part of a chain of clusters H* **then**
          $\tilde{X}([H], :), \tilde{C}([H], :) = 0$;
        end
      end
    end
  end
end

As in SIMPLE-ADAPTIVE-MATCHING, the function `confidence_bound` computes confidence bounds for the total reward per item when matched with an item within the same cluster or in a lower ranked cluster. Consider any item $i \in [2N]$ and k such that $i \in \mathfrak{S}[k]$. If $\sum_{j=1}^{2N} C(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}} = k_l$ for some l, then

$$Q_-(i) = \frac{\sum_{j=1}^{2N} X(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l} - \sqrt{\frac{\log(T)}{k_l}}$$

and

$$Q_+(i) = \frac{\sum_{j=1}^{2N} X(i, j) \mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l} + \sqrt{\frac{\log(T)}{k_l}}.$$

The main difference between the SIMPLE-ADAPTIVE-MATCHING and the ADAPTIVE-MATCHING algorithms is the SAMPLE-MATCHING procedure, which is detailed in subsection 29.

Cluster properties

Before presenting the SAMPLE-MATCHING procedure and the analysis of the algorithm, we introduce some needed terminology.

Recall that an ordered sequence of clusters is a sequence $S_k, \dots, S_{k'}$ such that, for any $l \in \{k, \dots, k'-1\}$

$$\forall (i,j) \in S_l \times S_{l+1} : \ i < j.$$

In the optimal matching, the items are matched in decreasing order. Hence, only the lowest and highest ranked items of a cluster S_l can be optimally matched outside the cluster.

- **Isolated cluster**: A cluster is said to be an *isolated cluster* if all items in this cluster have their optimal matching partners within this cluster.
- **Chain of clusters**: A *chain of clusters* is an ordered sequence of clusters such that every cluster in the chain contains an item whose optimal match is in the following cluster (except for the last one). Also, each item and its optimal matching partner either belong to the same cluster or to adjacent clusters in the chain.
- **Trivial chain of clusters**: A chain is referred to be a *trivial chain* if it is of the form $\{2k-1\}, \{2k, \dots, 2l-1\}, \{2l\}$, for any $1 \leq k < l \leq N$.

Note that if cluster S_i and cluster S_{i+1} are adjacent clusters in a chain, the highest ranked item of cluster S_i is optimally matched with the lowest ranked item of cluster S_{i+1}.

Note that within any cluster S, the items are not sorted in monotonic order. Therefore, knowing the highest -or lowest- ranked item of S does not provide any information about which item it should be matched with, as it could be any of the other un-ranked items. Also, the sampling policy and confidence interval's updates remain unchanged after splitting S between the top -or bottom- item and the rest of the items. Thus, for simplicity, even if the highest or the lowest element of the cluster (alone) is discovered, in the analysis, we consider that cluster S is not yet split.

SAMPLE-MATCHING procedure

At a high level, SAMPLE-MATCHING is a procedure that constructs a list of matchings $\tilde{\mathcal{M}}$, then iteratively samples each matching in this ensemble. For each pair (i,j), we denote with $p_{i,j}$ the proportion of draws of pair (i,j) in the list $\tilde{\mathcal{M}}$:

$$p_{i,j} = \frac{1}{|\tilde{\mathcal{M}}|} \sum_{m \in \mathcal{M}} \mathbb{1}_{\{(i,j) \in \mathcal{M}\}}$$

Sampling guarantees The SAMPLE-MATCHING procedure guarantees the following properties:

- Items residing in an isolated cluster S are uniformly matched among themselves:

$$p_{i,j} = \frac{1}{|S|-1}, \ \forall (i,j) \in S^2, \ i \neq j.$$

- For every chain of clusters $S_k, \dots, S_{k'}$, items residing within each cluster S_i of the chain, are uniformly matched among themselves

$$p_{i,j} = \frac{|S_l|-2}{|S_l|(|S_l|-1)}, \ \forall (i,j) \in S_l^2, \ i \neq j, l \in \{k+1, \dots, k'-1\},$$

$$p_{i,j} = \frac{1}{|S_l|},\ \forall (i,j) \in S_l^2,\ i \neq j, l = k \text{ or } l = k'.$$

- All potential pairs $(i,j) \in S_l \times S_{l+1}$, for $l \in \{k, \ldots, k'-1\}$, are sampled uniformly;

$$p_{i,j} = \frac{1}{|S_l||S_{l+1}|},\ \forall (i,j) \in S_l \times S_{l+1}, l \in [k, k'-1].$$

Let $\nu(S)$ be the sampling frequency of cluster S, which has values as follows:

$$\nu(S) = \begin{cases} |S| - 1 & \text{if } S \text{ is an isolated cluster} \\ |S| & \text{if } S \text{ is the first or the last cluster of a chain} \\ |S| + 1 & \text{if } S \text{ is an intermediate cluster of a chain.} \end{cases} \tag{3.27}$$

Let i and i' be to items in cluster $\mathfrak{S}[k]$. If $\mathfrak{S}[k]$ is an isolated or the first cluster of a chain, then:

$$\mathbb{E}[\frac{\sum_{j=1}^{2N}(X(i,j) - X(i,j))\mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l}] \geq \Delta_{i,i'} \frac{1}{\nu(\mathfrak{S}[k])} \sum_{j \in \mathfrak{S}[k] \setminus \{i,i'\}} u_j \tag{3.28}$$

If $\mathfrak{S}[k]$ is an within or the last cluster of a chain, then:

$$\mathbb{E}[\frac{\sum_{j=1}^{2N}(X(i,j) - X(i,j))\mathbb{1}_{\{j \in \mathfrak{S}[k'] | k' \leq k\}}}{k_l}] \geq \Delta_{i,i'} \frac{1}{\nu(\mathfrak{S}[k])} \left[\sum_{j \in \mathfrak{S}[k] \setminus \{i,i'\}} u_j + \frac{1}{|\mathfrak{S}[k-1]|} \sum_{j \in \mathfrak{S}[k-1]} u_j \right] \tag{3.29}$$

Note that in the case of isolated clusters, equations (3.25) and (3.28) are equivalent.

General procedure The Sample-Matching procedure constructs ensemble $\tilde{\mathcal{M}}$ as follows. Each cluster $S = [k,l]$ is associated with a list of *matching schemes* $L(S) = (l_1, \ldots, l_r)$. Each *matching schemes* l_i indicates which items should be paired together. For example, $l_i = \{(a,b), \ldots, (y,z)\}$ indicates that items a and b, and items y and z should be matched together. If S is part of a chain, *matching schemes* l_i may contain at most one pair (a, next) to indicate that item i should be matched with an item of the next cluster. It may also contain at most one pair $(a, \text{previous})$) to indicate a match with the previous cluster. Constructing the list of *matching schemes* therefore depends on the type of the cluster: whether the cluster is an isolated cluster, the first, an intermediate, or the last cluster of the chain.

Each cluster S is associated with sampling rate r_S, that depends on the position of the cluster in the chain. For each cluster S, the *matching schemes* in L_S are chosen according to a deterministic round-robin schedule, with each matching being chosen r_S times before moving on to the next one. At iteration t, the Sample-Matching procedure builds a full matching by aggregating the *matching schemes* chosen for individual clusters.

Construction of a list of *matching schemes* We discuss separately how lists of *matching schemes* are constructed for each type of the cluster.

- **Isolated cluster:** The list of *matching schemes* $L(S)$ is the scheduling list of an all-meet-all tournament over items in S.

  Note that in this case we have $|L(S)| = |S| - 1$.

- **First or last cluster in a chain:** A virtual item "next" or "previous" is added to cluster S, and, then, $L(S)$ is built using the isolated clusters' method.

  Note that in this case we have $|L(S)| = |S|$.

- **Intermediate cluster in a chain:** An auxiliary list of matchings $\tilde{L}(S) = (l_1, \dots, l_{|S|-1})$ is built using the isolated clusters' method. The list $L(S)$ is iteratively constructed from $\tilde{L}(S)$ using Algorithm 20.

  Note that $|L(S)| = (|S| - 1)|S|$. For each item a in S, pairs (a, prev) and (a, next) each appear in $|S| - 1$ *matching schemes* . Each pair $(a, b) \in S^2,\ a \neq b$ appears in $|S| - 2$ *matching schemes*. Also note that the *matching schemes* in $L(S)$ are ordered so that all pairs (a, prev) and (a, next), $a \in S$, appear once in each sub-list $(l_{n|S|+1}, \dots, l_{(n+1)|S|})$, $n \in \{0, \dots, |S| - 2\}$.

Algorithm 20: InterList

```
L(S) = ∅;
L̃(S) = (l_1, ..., l_{|S|−1});
for l_i ∈ L̃(S) do
    for (a, b) ∈ l_i do
        l_i ← l_i \ (a, b);
        L(S) ← L(S) ∪ (l_i, (a, next), (b, prev));
        L(S) ← L(S) ∪ (l_i, (b, next), (a, prev));
    end
end
```

Sampling rates and synchronization: If (a, next) and (prev, b) appear in the matchings of adjacent clusters S_1 and S_2, pair (a, b) is sampled. A synchronization method is used in Sample-Matching to ensure all pairs $(a, b) \in S_1 \times S_2$ are sampled uniformly.

Note that their is no need for such a synchronization for an isolated cluster S: the sampling rate r_S is set to 1, and Sample-Matching iterates over $L(S)$. Note that Sample-Matching and `sample_matching` are equivalent on an isolated cluster.

The following paragraph explains, without loss of generality, the synchronisation method of Sample-Matching for specific chain $S_1, \dots, S_K$.

Each cluster S_i in the chain is associated with a sampling rate r_{S_i} that depends on its position in the chain and the size of the adjacent clusters.

- If i is odd, $r_{S_i} = 1$.
- If i is even, $r_{S_i} = \text{LCM}(|S_{i-1}|, |S_{i+1}|)$, with the convention $|S_{K+1}| = 1$, and where $\text{LCM}(a, b)$ denotes the least common multiplier of a and b.

Sample-Matching selects the same *matching schemes* for cluster S, r_S times, before switching to the next one. Suppose Sample-Matching starts on the chain $S_1, \dots, S_K$ at iteration t_{init}. The following properties hold:

- If i is odd, all items in cluster S_i are sampled against the same items between iterations t_{init} and $t_{\text{init}} + n|L(S)|$, for every integer n.
- If i is even, all items in cluster S_i are sampled against the same items between iterations t_{init} and $t_{\text{init}} + nr(S)|L(S)|$, for every integer n.

Consider the event that a cluster S_i is separated at iteration t_{split}. SAMPLE-MATCHING determines the list of *matching schemes* and the sampling rates associated with the newly created clusters, and from iteration $t_{split}+1$, the matchings are sampled following those new list of *matching schemes* and the sampling rates.

It may happen that at iteration t_{split}, SAMPLE-MATCHING is stopped in the middle of the round on $L(S)$ for some cluster S. Thus, some couple of items in the cluster have not been uniformly sampled against the same other items. To correct this, all samples gathered after iteration $(t_{split} - [(t_{split} - t_{\text{init}}) \pmod{r(S)|L(S)|}])$ are dropped for the items in S.

Lemma 3.A.12. *The sum of the regret for all dropped samples is at most* $32N^6$.

Proof: Note that for any cluster S, it always holds that $r(S)|L(S)| < (2N)^4$. Thus, the samples of at most $(2N)^4$ iterations are not yet stored in X and C at any given split. Those iterations all cost at most N in terms of regret, and there are at most $2N$ splits. □

Proof of the upper bound

The proof goes as follows. The ADAPTIVE-MATCHING algorithm tracks estimators of the expected value of the total reward received for each of the $2N$ parameters. Lemma 3.A.1, together with a union bound and the fact that the regret per iteration is always smaller than N imply

$$\mathbb{E}[R(T)] \leq \mathbb{E}[R(T) \mid \mathcal{A}] + T\mathbb{P}(\bar{\mathcal{A}}) \leq \mathbb{E}[R(T) \mid \mathcal{A}] + 4N^2.$$

We denote with $R_{S_k,\dots,S_{k'}}$ the average regret of the SAMPLE-MATCHING procedure on the chain of clusters $S_k, \dots, S_{k'}$. The following lemma states that the average regret of SAMPLE-MATCHING on any chain of clusters can be decomposed into the sum of average regrets for isolated clusters and trivial chains of clusters.

Lemma 3.A.13 (Chain regret)**.** *The expected average regret of the* SAMPLE-MATCHING *procedure on a chain of clusters* $S_k, \dots, S_{k'}$ *is equal to*

$$R_{S_k,\dots,S_{k'}} = R_{S_k \cup \{v(S_{k+1})\}} + \sum_{i=k+1}^{k'-1} R_{\{l_{i-1}\},S_i,\{v(S_{i+1})\}} + R_{\{l_{k'-1}\} \cup S_{k'}}$$

where $v(S_i)$ *is a virtual item with parameter value* $\bar{u}_{S_i} = \frac{1}{|S_i|}\sum_{k \in S_i} u_k$*, and* l_i *is the highest ranked item in* S_i*.*

To transform the problem on chains of cluster onto a problem on isolated clusters and trivial chains of clusters, we introduce some additional notations. We denote with $S_t(i)$ the cluster to which item i belongs to at iteration t, and we let $\mathcal{E}_i(t)$ be the event $\{S_t(2i) = S_t(2i-1)\}$. Let $h_t(j)$ be a virtual item such that

$$h_t(j) = \mathbb{1}_{\mathcal{E}_j(t)} j + \mathbb{1}_{\overline{\mathcal{E}}_j(t)} v(S_t(j)).$$

The *virtual match event* $\mathcal{V}_t(l,j)$, is defined as follows:

$$\mathcal{V}_t(l,j) = (\{l \in S(j)\} \cap \{(l,j) \in m_t\}) \cup (\{l \in S(n^*(j))\} \cap \overline{\mathcal{E}}_j(t) \cap \{m_t(l) \in S(j)\}).$$

This means that an item l is "virtually" matched with item j when either item j belongs to the same cluster as l and l is matched with j at iteration t, or l belongs to the same cluster as j's optimal neighbor $n^*(j)$, j belongs to another cluster S and l is matched with an item in S at iteration t.

We define $\tilde{R}_{i,j}(T)$ as follows:

$$\begin{aligned}\tilde{R}_{i,j}(T) &= \sum_{t=1}^{T} \sum_{k\in\{2i-1,2i\}} \mathbb{1}_{\mathcal{V}_t(2j-1,k)} \left(\frac{1}{2}(u_{2i-1}u_{2i} + u_{2j-1}u_{2j}) - u_l u_{2j-1}\right) \\ &\quad + \sum_{t=1}^{T} \sum_{k\in\{2i-1,2i\}} \mathbb{1}_{\mathcal{V}_t(k,2j)} \left(\frac{1}{2}(u_{2i-1}u_{2i} + u_{2j-1}u_{2j}) - u_l u_{h_t(2j)}\right).\end{aligned}$$

Note that under Assumption 1, $\tilde{R}_{i,j}(T)$ and $R_{i,j}(T)$ are equivalent, since pairs of items are never split into different clusters.

According to Lemma 3.A.13, the following equation holds:

$$\sum_{i=1}^{N}\sum_{j=i+1}^{N} R_{i,j}(T) = \sum_{i=1}^{N}\sum_{j=i+1}^{N} \tilde{R}_{i,j}(T).$$

It remains to bound $\tilde{R}_{i,j}(T)$.

Lemma 3.A.14 (Trivial chain regret)**.** *The average regret of the* Sample-Matching *procedure on a trivial chain of clusters is upper bounded as*

$$\begin{aligned}R_{\{2k-1\},[2k,2l-1],\{2l\}} \leq & \frac{2}{2K+1}(u_{2k-1} - u_{2l})(u_{2k} - u_{2l-1}) \\ & + \frac{1}{2K+1}\sum_{j=k+1}^{l-1}\sum_{j'=j+1}^{l-1} r_{j,j'} + \frac{1}{2K+1}\sum_{j=k+1}^{p-1} r_{j,l} + r_{k,j}\end{aligned}$$

where $K = l - k$.

Note that this is exactly the regret of uniform matching over items $\{2k-1,\dots,2l\}$, except that all inter-pair matches between pairs k and l are the most favorable ones, namely $(2k-1, 2l-1)$ and $(2k, 2l)$. Recall that an inter-pair match between a pair i and a pair j is the match of an item of pair i with an item of pair j.

Recall the definition of $\nu(S)$ in equation (3.27). Note that $\tilde{R}_{i,j}(T)$ has the following properties:

- If $S_t(2i) \neq S_t(2j-1)$:
$$\tilde{R}_{i,j}(t+1) - \tilde{R}_{i,j}(t) = 0.$$
- If pairs i and j belong to the same cluster S between iterations t and $t+\nu(S)$, then
$$\tilde{R}_{i,j}(t+\nu(S)) - \tilde{R}_{i,j}(t) \leq r_{i,j}$$
which follows from Lemmas 3.A.14 and 3.A.9.
- If at least three items of pairs i and j belong to the cluster S, the last property holds with $r_{i,j}$ replaced with
$$r_{i,j,t} = (u_{2i} - u_{2j-1})(u_{2i-1} - u_{h_t(2j)}) + (u_{2i-1} - u_{2j-1})(u_{2i} - u_{h_t(2j)})$$
- If only items $2i$ and $2j-1$ belong to the same cluster S between iterations t and $t+\nu(S)$, we have
$$\tilde{R}_{i,j}(t+\nu(S)) - \tilde{R}_{i,j}(t) \leq 2(u_{2i} - u_{2j-1})(u_{2i-1} - u_{h_t(2j)})$$

which follows from Lemma 3.A.14. Notice that in that case, S is necessarily an intermediate cluster in a chain.

The following equality holds:

$$\begin{aligned} r_{i,j} &= (u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}) + (u_{2i-1} - u_{2j-1})(u_{2i} - u_{2j}) \\ &= 2(u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}) + \Delta_{2i-1,2i}\Delta_{2j-1,2j}. \end{aligned}$$

Let us define

$$\bar{u}_j(t) = \frac{1}{t}\sum_{s=1}^{t} u_{h_t(j)}$$

and $\mathcal{E}_{i,j}(S)$ is redefined to be the event that cluster S is the last cluster to contain both items $2i$ and $2j-1$. And $\tilde{\mathcal{E}}_{i,j}(S')$ is the event that cluster S' is the last cluster to contain at least three items of pairs i and j. For a cluster S, T_S is the minimal number s.t. if for some $i \in S$, $\sum_{j\in[2N]} C(i,j) > T_S$, then S is separated. The following relation holds

$$\begin{aligned} \mathbb{E}[\tilde{R}_{i,j}(T) \mid \mathcal{E}_{i,j}(S) \cap \tilde{\mathcal{E}}_{i,j}(S')] \leq & \frac{T_S}{\nu(S)} 2(u_{2i} - u_{2j-1})(u_{2i-1} - \bar{u}_{2j}(T_S)) \\ & + \frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}(u_{2j-1} - \bar{u}_{2j}(T_{S'})) \\ & + \underbrace{h(i,j) + \frac{r(S')|L(S')|}{\nu(S')} + \frac{r(S)|L(S)|}{\nu(S)}}_{:=b(i,j)} \end{aligned}$$

where $h(i,j)$ accounts for the regret due to dropped samples. According to lemma 3.A.12:

$$\sum_{(i,j)} h(i,j) \leq 32N^6.$$

Note that for any cluster S, it always holds that $|L(S)|/\nu(S) < N$. Thus:

$$\frac{r(S')|L(S')|}{\nu(S')} + \frac{r(S)|L(S)|}{\nu(S)} \leq 2N^3$$

Lemma 3.A.15. *Consider an isolated cluster $S = \{2l-1, \ldots, 2l'\}$, under event $\mathcal{A}$, the minimal number T_S s.t. if for some $i \in S$, $\sum_{j\in[2N]} C(i,j) > T_S$, then S is separated, is upper bounded as*

$$T_S \leq 64\frac{1}{(\tilde{\mu}_{-\{k,k+1\}}\Delta_{k,k+1})^2}\log(T), \quad \textit{for all } k \in \{2l, \ldots, 2l'-2\}$$

and

$$T_S \leq 64\max\left\{\frac{1}{(\tilde{\mu}_{-\{2l-1,2l\}}\Delta_{2l-1,2l})^2}, \frac{1}{(\tilde{\mu}_{-\{2l'-1,2l'\}}\Delta_{2l'-1,2l'})^2}\right\}\log(T).$$

where

$$\tilde{\mu}_{-\{k,k+1\}} = \frac{1}{\nu(S)}\left(\sum_{i=2l-1}^{2l'} u_i - u_k - u_{k+1}\right).$$

If cluster $S = \{l, \ldots, l'\}$ belongs to a chain and is ranked between clusters S_{-1} and S_{+1} (those clusters may be empty if S is the first or last element of the chain), under event $\mathcal{A}$, T_S is upper

bounded as

$$T_S \leq 64 \frac{1}{(\tilde{\mu}_{-\{k,k+1\}} \Delta_{k,k+1})^2} \log(T), \quad \textit{for all } k \in \{l, \dots, l'-1\}$$

where

$$\tilde{\mu}_{-\{k,k+1\}} = \frac{1}{\nu(S)} \left(\bar{u}_{S_{-1}} + \left(\sum_{i=l}^{l'} u_i \right) - u_k - u_{k+1} \right).$$

This lemma is a consequence of Lemma 3.A.2 and equations (3.28),(3.29).

The following lemma bounds the regret for interactions between pair i and j.

Lemma 3.A.16. *Under event $\mathcal{A}$, the total regret for interaction between pairs i and j such that $j - i \geq 2$ is upper bounded as follows*

$$\tilde{R}_{i,j}(T) - b(i,j) \leq \frac{c_{\mathrm{u}}}{(j-i-1)} \frac{1}{\Delta_{\min}} \log(T).$$

If $j - i = 1$, the following bound holds

$$\tilde{R}_{i,j}(T) - b(i,j) \leq \frac{c_{\mathrm{u}}}{\Delta_{\min}} \log(T).$$

The proof of Lemma 3.A.16 is given in Appendix 9.

It remains to sum the bounds over all possible pairs:

$$\begin{aligned}
R(T) &\leq c_{\mathrm{u}} \left(N + \sum_{i=1}^{N} \sum_{j=i+2}^{N} \frac{1}{j-i-1} \right) \frac{\log(T)}{\Delta_{\min}} + 32N^6 + N^5 + 4N^2 \\
&\leq c_{\mathrm{u}} \left(N + \sum_{i=1}^{N} \sum_{k=1}^{N-i-1} \frac{1}{k} \right) \frac{\log(T)}{\Delta_{\min}} + 34N^6 \\
&\leq c_{\mathrm{u}} \left(2N + \sum_{i=1}^{N} \log(N-i-1) \right) \frac{\log(T)}{\Delta_{\min}} + 34N^6 \\
&\leq c_{\mathrm{u}} \frac{N(\log(N)+2)}{\Delta_{\min}} \log(T) + 34N^6.
\end{aligned}$$

□

Proof of Lemma 3.A.14 Let $p_{i,j}$ be the proportion of draws of pair (i,j) by Sample-Matching on the chain of clusters $\{2k-1\}, S, \{2l\}$. Let $S = \{2k\} \cup A \cup \{2l-1\}$ so that A is the set of items in S not in pair k or l. As noted in the analysis of Sample-Matching $p_{i,j}$ is defined as

$$p_{i,j} = \begin{cases} \frac{1}{|S|} & \text{if } i \in S, j = 2k-1 \text{ or } j = 2l \\ 0 & \text{if } i = 2k-1 \text{ and } j = 2l \\ \frac{1}{|S|} \frac{|S|-2}{|S|-1} & \text{if } (i,j) \in S^2, i \neq j \end{cases}.$$

By definition, we have

$$R_{\{2l-1\},S,\{2p\}} = \frac{1}{2} \sum_{i,j \in \{u_{2l-1}\} \cup S \cup \{u_{2p}\}} p_{i,j} u_i (u_{m^*(i)} - u_j).$$

The terms on the right hand side can be grouped to obtain the desired expression.

The first group of terms corresponds to a scaled-down uniform matching on set A,

$$R_1 = \frac{|S|-2}{|S|(|S|-1)} \sum_{j=l+1}^{p-1} \sum_{j'=j+1}^{p-1} r_{j,j'}.$$

The second group of terms corresponds to matches $(2l, 2p)$ and $(2l-1, 2p-1)$, with value

$$R_2 = \frac{1}{|S|}(u_{2l-1} - u_{2p})(u_{2l} - u_{2p-1}).$$

The third group of terms corresponds to matches $(2l-1, j)$, $(2l, m^*(j))$, and $(2p-1, j)$, $(2p, m^*(j))\}$, with respective values

$$R_3 = \frac{|S|-2}{|S|(|S|-1)} \sum_{j=l+1}^{p-1} r_{l,j} \text{ and } R_3' = \frac{|S|-2}{|S|(|S|-1)} \sum_{j=l+1}^{p-1} r_{j,p}.$$

The last group of terms corresponds to matches $(2l-1, j)$, $(2p, m^*(j))$, and $(2p-1, 2l)$, with value

$$R_4 \leq \frac{|S|-2}{|S|(|S|-1)}(u_{2l-1} - u_{2p})(u_{2l} - u_{2p-1})$$

where the inequality comes from the fact that matches $(2l-1, 2p-1)$, $(j, m^*(j))$, and $(2l, 2p)$ have a smaller expected reward than matches $(2l-1, j)$, $(2p, m^*(j))$, and $(2p-1, 2l)$.

Summing $R_1 + R_2 + R_3 + R_3' + R_4$ gives the result of the lemma.

Proof of Lemma 3.A.13 Let $S_j, \ldots, S_{j'}$ be a chain of clusters, and l_i denote the highest-ranked item in cluster i. Let $p_{k,k'}$ denote the proportion of draws of pair (k, k') by Sample-Matching on $S_j, \ldots, S_{j'}$. We use the convention $S_{j-1} = S_{j'+1} = \emptyset$. Let $R_{S_j,\ldots,S_{j'}}$ denote the expected average regret of Sample-Matching on $S_j, \ldots, S_{j'}$.

We have the following relations

$$\begin{aligned}
2R_{S_j,\ldots,S_{j'}} =& \sum_{i=j}^{j'} \sum_{k \in S_i} u_k u_{m^*(k)} - \sum_{i=j}^{j'} \sum_{k,k' \in S_i} p_{k,k'} u_k u_{k'} \\
&- 2 \sum_{i=j}^{j'} \sum_{k \in S_i} \sum_{k' \in S_{i-1}} p_{k,k'} \left(u_k u_{k'} - u_k u_{l_{i-1}} + u_k u_{l_{i-1}} \right) \\
&+ 2 \sum_{i=j}^{j'} \sum_{k \in S_i} \sum_{k' \in S_{i+1}} p_{k,k'} \left(u_k u_{k'} - u_{k'} u_{l_i} + u_{k'} u_{l_i} \right) \\
=& \sum_{i=j}^{j'} \left(2u_{l_i} \sum_{k \in S_i, k' \in S_{i+1}} p_{k,k'} u_{k'} + \sum_{k \in S_i \cup \{l_{i-1}\} \setminus \{l_i\}} u_k u_{m^*(k)} \right) \\
&- \sum_{i=j}^{j'} \left(\sum_{k,k' \in S_i} p_{k,k'} u_k u_{k'} + 2 \sum_{k \in S_i, k' \in S_{i+1}} p_{k,k'} u_k u_{k'} + 2 \sum_{k' \in S_{i-1}, k \in S_i} p_{k,k'} u_k u_{l_{i-1}} \right).
\end{aligned}$$

Let us define

$$\bar{u}_S = \frac{1}{|S|}\sum_{k\in S} u_k$$

and let us use the following convention $u_{l_{j-1}} = 0$ and $\bar{u}_{S_{p+1}} = 0$.

Note that the following equations hold:

$$\begin{aligned}
\sum_{k\in S_i, k'\in S_{i+1}} p_{k,k'}u_{k'} &= \bar{u}_{S_{i+1}} \\
\sum_{k\in S_i, k'\in S_{i+1}} p_{k,k'}u_k u_{k'} &= \sum_{k\in S_i}\frac{1}{|S_i|}u_k\bar{u}_{S_{i+1}} \\
\sum_{k'\in S_{i-1}, k\in S_i} p_{k,k'}u_k u_{l_{i-1}} &= \sum_{k\in S_i}\frac{1}{|S_i|}u_k u_{l_{i-1}}.
\end{aligned}$$

The equation above for $R_{S_j,\dots,S_{j'}}$ can be simplified to

$$\begin{aligned}
2R_{S_j,\dots,S_{j'}} =& \sum_{i=j}^{j'}\left(2u_{l_i}\bar{u}_{S_{i+1}} + \sum_{k\in S_i\cup\{l_{i-1}\}\setminus\{l_i\}} u_k u_{m^*(k)}\right) \\
&- \sum_{i=j}^{j'}\left(\sum_{k,k'\in S_i} p_{k,k'}u_k u'_k + 2\sum_{k\in S_i}\frac{1}{|S_i|}u_k\bar{u}_{S_{i+1}} + 2\sum_{k\in S_i}\frac{1}{|S_i|}u_k u_{l_{i-1}}\right)
\end{aligned}$$

which can be written as

$$R_{S_j,\dots,S_{j'}} = R_{S_j\cup\{v_{j+1}\}} + \sum_{i=j+1}^{j'-1} R_{\{l_{i-1}\},S_i,\{v_{i+1}\}} + R_{\{l_{j'-1}\}\cup S_{j'}}$$

where v_k is a virtual item with parameter value $\bar{u}_{S_k}$.

Proof of Lemma 3.A.16 For any two clusters S and S', the following holds:

$$\begin{aligned}
\mathbb{E}[\tilde{R}_{i,j}(T) \mid \mathcal{E}_{i,j}(S)\cap\tilde{\mathcal{E}}_{i,j}(S')] - b(i,j) \leq& \frac{T_S}{\nu(S)}2(u_{2i}-u_{2j-1})(u_{2i-1}-\bar{u}_{2j}(T_S)) \\
&+ \frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}(u_{2j-1}-\bar{u}_{2j}(T_{S'})).
\end{aligned}$$

To simplify the proof, we first assume that $S' = \{2i-1,\dots,2j\}$ and $S = \{2i,\dots,2j-1\}$ or $S = \{2i-1,\dots,2j\}$, and then argue that the obtained result holds for arbitrary S and S'. Note that this implies that $S \subset S'$, hence S is either isolated or in the trivial chain $\{2i-1\}, S, \{2j\}$.

The section starts with the presentations and proofs of the two following lemmas. The proof of Lemma 3.A.16 is provided afterwards.

Lemma 3.A.17 (Intermediate result)**.** *Assume that $S' = \{2i-1,\dots,2j\}$ or $S' = \{2i,\dots,2j-1\}$ and $S = \{2i,\dots,2j-1\}$, with $j-i \geq 2$. Then, we have*

$$\mathbb{E}[\tilde{R}_{i,j}(T) \mid \mathcal{E}_{i,j}(S)\cap\tilde{\mathcal{E}}_{i,j}(S')] - b(i,j) \leq \frac{c_{\mathrm{u}}}{(j-i-1)\Delta_{\min}}\log(T).$$

Lemma 3.A.18 (Neighboring pairs)**.** *Suppose $S' = \{2i-1,\dots,2j\}$ with $j-i=1$. Then, we have*

$$\mathbb{E}[\tilde{R}_{i,j}(T) \mid \mathcal{E}_{i,j}(S)\cap\tilde{\mathcal{E}}_{i,j}(S')] - b(i,j) \leq \frac{c_{\mathrm{u}}}{\Delta_{\min}}\log(T).$$

Proof of Lemma 3.A.17 Let $m = \arg\max_{k\in\{2i,\dots,2j-1\}} \Delta_{k,k+1}$. The following inequality holds

$$(u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}) \leq (2(j-i)-1)^2\Delta_{m,m+1}^2 + \Delta_{m,m+1}(\Delta_{2i-1,2i} + \Delta_{2-1,2j}). \quad (3.30)$$

Let $\gamma = \min_{k\in[i,j-1]} \Delta_{2i-1,2i+2}$. We have the following inequalities

$$\Delta_{\min} \leq \gamma^2 \text{ and } \Delta_{\min} \leq \gamma\Delta_{m,m+1}.$$

For all $k \in [2i-1, 2j-1]$,

$$u_k \geq u_{2j} + \frac{1}{2}\Delta_{2j-1,2j} + \frac{1}{2}\sum_{l=\lceil\frac{k+1}{2}\rceil}^{j-1} \gamma.$$

Hence, it follows

$$\begin{aligned}(2(j-i)+1)\mu_{-m,m+1} \geq& \sum_{k=2i-1}^{2j-1} u_k - u_m - u_{m+1} \quad (3.31)\\ \geq& u_{2i-1} + \sum_{k=2i+2}^{2j-1} u_k \\ \geq& \frac{j-i}{2}\Delta_{2j-1,2j} + \Delta_{2i-1,2i} + \frac{1}{2}\left(\sum_{k=2i+1}^{2j-1}\left(j - \left\lceil\frac{k+1}{2}\right\rceil\right)\right)\gamma \\ \geq& \frac{j-i}{2}\Delta_{2j-1,2j} + \Delta_{2i-1,2i} + \frac{(j-i-1)^2}{2}\gamma.\end{aligned}$$

Combining with Equation (3.30), we have

$$\begin{aligned}A_{i,j}(S) :=& \frac{T_S}{2(j-i)+1}2(u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}(T_S)) \\ \leq& 64\frac{2(j-i)+1}{\Delta_{m,m+1}^2}\frac{(2(j-i)-1)^2\Delta_{m,m+1}^2 + \Delta_{m,m+1}(\Delta_{2i-1,2i} + \Delta_{2j-1,2j})}{(\frac{j-i}{2}\Delta_{2j-1,2j} + \Delta_{2i-1,2i} + \frac{(j-i-1)^2}{2}\gamma)^2}\log(T).\end{aligned}$$

The following holds

$$\begin{aligned}a_1 :=& \frac{2(j-i)+1}{\Delta_{m,m+1}^2}\frac{(2(j-i)-1)^2\Delta_{m,m+1}^2}{(\frac{j-i}{2}\Delta_{2j-1,2j} + \Delta_{2i-1,2i} + \frac{(j-i-1)^2}{2}\gamma)^2} \\ \leq& \frac{4(2(j-i)+1)(2(j-i)-1)^2}{(j-i-1)^4}\frac{1}{\gamma^2} \\ \leq& \frac{180}{j-i-1}\frac{1}{\Delta_{\min}}\end{aligned}$$

and

$$\begin{aligned} a_2 &:= \frac{2(j-i)+1}{\Delta_{m,m+1}^2} \frac{\Delta_{m,m+1}(\Delta_{2i-1,2i}+\Delta_{2j-1,2j})}{(\frac{j-i}{2}\Delta_{2j-1,2j}+\Delta_{2i-1,2i}+\frac{(j-i-1)^2}{2}\gamma)^2} \\ &\leq \frac{2(j-i)+1}{\Delta_{m,m+1}\gamma} \frac{\Delta_{m,m+1}(\Delta_{2i-1,2i}+\Delta_{2j-1,2j})}{(\Delta_{2j-1,2j}+\Delta_{2i-1,2i})(j-i-1)^2\Delta_{m,m+1}} \\ &\leq \frac{5}{j-i-1}\frac{1}{\Delta_{\min}}. \end{aligned}$$

Putting the bounds on a_1 and a_2 together gives

$$A_{i,j}(S) \leq \frac{c_{\mathrm{u}}}{j-i-1}\frac{1}{\Delta_{\min}}.$$

We note

$$B_{i,j}(S') := \frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}\Delta_{2j-1,2j}.$$

We separately consider two different cases as follows.

- **Case $\Delta_{2i-1,2i} \leq \Delta_{m,m+1}$ or $\Delta_{2j-1,2j} \leq \Delta_{m,m+1}$.** By definition of S and S', S' is a cluster containing S. Hence, we have

  $$\frac{T_{S'}}{\nu(S')} \leq \frac{T_S}{\nu(S)}.$$

  Then, we have

  $$\frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}\Delta_{2j-1,2j} \leq \frac{T_S}{\nu(S)}\Delta_{m,m+1}(\Delta_{2i-1,2i}+\Delta_{2j-1,2j}) = 64a_2$$

  which implies, according to the bound on a_2,

  $$B_{i,j}(S') \leq \frac{320}{j-i-1}\frac{1}{\Delta_{\min}}\log(T).$$

- **Case $\Delta_{2i-1,2i} > \Delta_{m,m+1}$ and $\Delta_{2j-1,2j} > \Delta_{m,m+1}$.** According to Lemma 3.A.15, we have

  $$B_{i,j}(S') \leq \max\left\{\frac{64}{\Delta_{2i-1,2i}^2\mu_{-2i-1,2i}^2}, \frac{64}{\Delta_{2j-1,2i}^2\mu_{-2j-1,2j}^2}\right\}\frac{\Delta_{2i-1,2i}\Delta_{2j-1,2j}}{2(j-i)+1}\log(T).$$

  Repeating the computations in Equation (3.31) gives

  $$(2(j-i)+1)\mu_{-2j-1,2j} \geq \Delta_{2i-1,2i} + \frac{(j-i-1)^2}{2}\gamma.$$

  Hence, we have

  $$\begin{aligned} \frac{\Delta_{2i-1,2i}\Delta_{2j-1,2j}}{(2(j-i)+1)\Delta_{2j-1,2j}^2\mu_{-2j-1,2j}^2} &\leq \frac{2(j-i)+1}{(j-i-1)^2\gamma\Delta_{2j-1,2j}} \\ &\leq \frac{5}{(j-i-1)\Delta_{\min}}. \end{aligned}$$

Similarly, we obtain

$$(2(j-i)+1)\mu_{-2i-1,2i} \geq \Delta_{2j-1,2j} + \frac{(j-i-1)^2}{2}\gamma.$$

This implies that the following holds

$$B_{ij}(S') \leq \frac{320}{(j-i-1)\Delta_{\min}} \log(T).$$

Combining the bounds on $A_{i,j}(S)$ and $B_{i,j}(S')$ gives the following bound

$$\mathbb{E}[\tilde{R}_{i,j}(T) \mid \mathcal{E}_{i,j}(S) \cap \tilde{\mathcal{E}}_{i,j}(S')] \leq \frac{c_{\mathrm{u}}}{j-i-1}\frac{1}{\Delta_{\min}} \log(T) \tag{3.32}$$

when $S' = \{2i-1, \ldots, 2j\}$ and $S = \{2i, \ldots, 2j-1\}$, $j - i \geq 2$. □

Proof of Lemma 3.A.18 The following inequality holds

$$\Delta_{\min} \leq u_{2i-1}\Delta_{2i,2j-1}.$$

Hence, we have

$$\begin{aligned}\frac{T_S}{\nu(S)} 2(u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}) &\leq 64 \cdot 6 \frac{(u_{2i} - u_{2j-1})}{\Delta_{2i,2j-1} u_{2i}^2} \log(T) \\ &\leq \frac{384}{\Delta_{2i,2j-1} u_{2i}} \log(T) \\ &\leq \frac{384}{\Delta_{\min}} \log(T).\end{aligned}$$

If $\Delta_{2i-1,2i} \leq \Delta_{2i,2j-1}$ or $\Delta_{2j-1,2j} \leq \Delta_{2i,2j-1}$, then

$$\frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}\Delta_{2j-1,2j} \leq \frac{T_S}{\nu(S)}(u_{2i} - u_{2j-1})(u_{2i-1} - u_{2j}).$$

If $\Delta_{2i-1,2i} > \Delta_{2i,2j-1}$ and $\Delta_{2j-1,2j} > \Delta_{2i,2j-1}$, then

$$\begin{aligned}\frac{T_{S'}}{\nu(S')}\Delta_{2i-1,2i}\Delta_{2j-1,2j} &\leq 64 \max\left\{ \frac{3\Delta_{2i-1,2i}\Delta_{2j-1,2j}}{\Delta_{2i-1,2i}^2 (u_{2j-1} + u_{2j})^2}, \frac{3\Delta_{2i-1,2i}\Delta_{2j-1,2j}}{\Delta_{2j-1,2j}^2 (u_{2i-1} + u_{2i})^2} \right\} \log(T) \\ &\leq 192 \max\left\{ \frac{1}{\Delta_{2i-1,2i} u_{2j-1}}, \frac{1}{\Delta_{2j-1,2j} u_{2i-1}} \right\} \log(T) \\ &\leq \frac{192}{\Delta_{\min}} \log(T).\end{aligned}$$

□

Proof of Lemma 3.A.16 Up to now, S' was an isolated cluster. Consider now it is the first element of a chain $S' = \{2i-1, \ldots, 2j-1\}, S_2, \ldots$. All previous computations hold with $\bar{u}_{2j}(T'_S)$ instead of u_{2j}, hence the result remains true.

If S' is the last element of a chain, u_{2i-1} needs to be replaced by $\bar{u}_{2i-1}(T'_S)$ in all expressions for $\mu_{-k,k+1}$. Since $u_{S_{-1}} \geq u_{2i-1}$, it holds that $\bar{u}_{2i-1}(T'_S) \geq u_{2i-1}$, so all previous results remain true.

Therefore, for any cluster S', such that $S' = \{2i, \dots, 2j\}$,$S' = \{2i-1, \dots, 2j-1\}$ or $S' = \{2i-1, \dots, 2j\}$, and $S = S'$ or $S = \{2i, \dots, 2j-1\}$:

$$\mathbb{E}[\tilde{R}_{i,j}(T)|\mathcal{E}_{i,j}(S), \tilde{\mathcal{E}}_{i,j}(S')] - b(i,j) \leq c_{\mathrm{u}} \min(1, \frac{1}{j-i-1}) \frac{1}{\Delta_{\min}} \log(T)$$

For any $k \leq i < j < l$ or $k < i < j \leq l$,

$$r_{i,j}(t) \leq 2(u_{2k} - u_{2l-1})(u_{2k-1} - \bar{u}_{2l}(t)).$$

Thus, for any clusters such that $S = [a, b]$ with $a \in (2l-1, 2l)$ and $b \in (2k-1, 2k)$, and any subset S', we have

$$\begin{aligned}
\mathbb{E}[\tilde{R}_{i,j}(T)|\mathcal{E}_{i,j}(S), \tilde{\mathcal{E}}_{i,j}(S')] - h(i,j) &\leq \frac{T_S}{\nu(S)} 2(u_{2k} - u_{2l-1})(u_{2k-1} - \bar{u}_{2l}(t)) \\
&\leq c_{\mathrm{u}} \min(1, \frac{1}{l-k-1}) \frac{1}{\Delta_{\min}} \log(T) \\
&\leq c_{\mathrm{u}} \min(1, \frac{1}{j-i-1}) \frac{1}{\Delta_{\min}} \log(T)
\end{aligned}$$

where the second inequality comes from the computations of the proof of Lemma 3.A.17. This proves that the result still holds if the clusters S and S' are larger (in the sense of inclusion) than those studied in Lemma 3.A.17, and completes the proof. □

3.A.6 Comparison of ADAPTIVE-MATCHING with an exploration policy

Proof of Lemma 3.4.5

Consider a matching bandit problem on the ordered sequence of cluster

$$\mathfrak{S} = \{1, 2\}, \{3, ..., 2N\}$$

where the two best items are identified.

We consider an *information first strategy* that matches the items in $\mathfrak{S}$ following a round-robin tournament. The ADAPTIVE-MATCHING algorithm matches 1 with 2 and items in set $\{3, ..., 2N\}$ following a round-robin tournament. We denote with R_I and R_D the regrets incurred by these two strategies, respectively, before they can rank any two items m and $m+1$, $m \in \{3, \dots, 2N\}$. According to Lemmas 3.A.2 and 3.A.9, the following bounds hold

$$R_D \leq U_D := \frac{2N-3}{\Delta^2_{m,m+1}} \frac{\sum_{i=2}^{N} \sum_{j=i+1}^{N} r_{i,j}}{(\sum_{i=3}^{2N} u_i - u_m - u_{m+1})^2} \tag{3.33}$$

and

$$R_I \leq U_I := \frac{2N-2}{\Delta^2_{m,m+1}} \frac{\sum_{j=2}^{N} r_{1,j} + \sum_{i=2}^{N} \sum_{j=i+1}^{N} r_{i,j}}{(\sum_{i=1}^{2N} u_i - u_m - u_{m+1})^2}. \tag{3.34}$$

Let us define $\rho_1, \dots, \rho_{2N}$ such that

$$u_i = \prod_{k=1}^{i} \rho_k, \text{ for } i \in [2N].$$

$$
\begin{aligned}
r_{i,j} &= 2(u_{2i-1}-u_{2j})(u_{2i}-u_{2j-1})+(u_{2i-1}-u_{2i})(u_{2j-1}-u_{2j}) \\
&= 2\left(\prod_{k=1}^{2i-1}\rho_k\right)^2\rho_{2i}\left(1-\prod_{k=2i+1}^{2j}\rho_k\right)\left(1-\prod_{k=2i}^{2j}\rho_k\right)+\prod_{k=1}^{2i-1}\rho_k\prod_{k=1}^{2j-1}\rho_k(1-\rho_{2i})(1-\rho_{2j}) \\
&= \left(\prod_{k=1}^{2i-1}\rho_k\right)^2\rho_{2i}\underbrace{\left(2\left(1-\prod_{k=2i+1}^{2j-1}\rho_k\right)\left(1-\prod_{k=2i}^{2j-1}\rho_k\right)+\prod_{k=2i+1}^{2j-1}\rho_k(1-\rho_{2i})(1-\rho_{2j})\right)}_{a_{i,j}} \\
&= \left(\prod_{k=1}^{2i-1}\rho_k\right)^2\rho_{2i}a_{i,j}
\end{aligned}
$$

The expressions for U_D and U_I simplify to

$$
\begin{aligned}
U_D &= \frac{2N-3}{\Delta_{m,m+1}^2}\frac{\sum_{i=2}^N\sum_{j=i+1}^N\left(\prod_{k=1}^{2i-1}\rho_k\right)^2\rho_{2i}a_{i,j}}{(\sum_{i=3,i\neq m,m+1}^{2N}\prod_{k=1}^i\rho_k)^2} \\
&= \frac{2N-3}{\Delta_{m,m+1}^2}\rho_4\frac{\sum_{j=3}^N a_{2,j}+\sum_{i=3}^N\sum_{j=i+1}^N\left(\prod_{k=5}^{2i-1}\rho_k\right)^2\rho_4\rho_{2i}a_{i,j}}{(1+\sum_{i=4,i\neq m,m+1}^{2N}\prod_{k=4}^i\rho_k)^2} \\
&= \frac{2N-3}{\Delta_{m,m+1}^2}\frac{A_2}{(1+\sum_{i=4,i\neq m,m+1}^{2N}\prod_{k=4}^i\rho_k)^2}
\end{aligned}
$$

where

$$
A_2 = \rho_4\left(\sum_{j=3}^N a_{2,j}+\sum_{i=3}^N\sum_{j=i+1}^N\left(\prod_{k=5}^{2i-1}\rho_k\right)^2\rho_4\rho_{2i}a_{i,j}\right)
$$

and

$$
\begin{aligned}
U_I &= \frac{2N-1}{\Delta_{m,m+1}^2}\frac{\sum_{i=1}^N\sum_{j=i+1}^N\left(\prod_{k=1}^{2i-1}\rho_k\right)^2\rho_{2i}a_{i,j}}{(\sum_{i=1,i\neq m,m+1}^{2N}\prod_{k=1}^i\rho_k)^2} \\
&= \frac{2N-1}{\Delta_{m,m+1}^2}\rho_2\frac{\sum_{j=2}^N a_{1,2}+\sum_{i=2}^N\sum_{j=i+1}^N\left(\prod_{k=3}^{2i-1}\rho_k\right)^2\rho_2\rho_{2i}a_{i,j}}{(1+\sum_{i=2,,i\neq m,m+1}^{2N}\prod_{k=2}^i\rho_k)^2} \\
&= \frac{2N-1}{\Delta_{m,m+1}^2}\rho_2\frac{\sum_{j=2}^N a_{1,j}+\rho_3^2\rho_2A_2}{(1+\sum_{i=2,i\neq m,m+1}^{2N}\prod_{k=2}^i\rho_k)^2}.
\end{aligned}
$$

From the last two expressions, we have

$$
\lim_{\rho_2\to 0}\frac{U_D}{U_I} = +\infty.
$$

The following lower bound holds

$$a_{1,j} = \left(2\left(1-\prod_{k=3}^{2j-1}\rho_k\right)\left(1-\prod_{k=2}^{2j-1}\rho_k\right)+\prod_{k=3}^{2j-1}\rho_k(1-\rho_2)(1-\rho_{2j})\right) \geq 2(1-\rho_3)^2.$$

Thus, U_I is lower bounded as

$$U_I \geq \rho_2 \frac{2N-1}{\Delta^2_{m,m+1}} \frac{2N(1-\rho_3)^2+\rho_3^2\rho_2 A_2}{(1+\sum_{i=2,i\neq m,m+1}^{2N}\prod_{k=2}^{i}\rho_k)^2}.$$

Note that A_2 is at most of order N^2, thus if $\rho_2 > 1/2$, the ratio U_D/U_I is upper bounded by $c_{\mathrm{u}}N$. If, in addition, $\rho_3 > 1/2$, then the ratio U_D/U_I is upper bounded by a constant c_{u}.

3.A.7 MATCHING-ID algorithm

Algorithm description and pseudo-code

The algorithm is defined by the pseudo-code in Algorithm 21.

Algorithm 21: MATCHING-ID

input : set of items $[2N]$, required precision δ

$U = [2N], X = C = [0]^{[2N]\times[2N]}$;
while $U \neq \emptyset$ **do**
  $\mathcal{B} = \{$candidate $|\mathcal{S}|$ best items$\}$;
  $\mathcal{A} = \mathcal{B} \cup \mathcal{S}$;
  if $|\mathcal{A}|$ *odd* **then**
    $k = \arg\max[2N] \setminus \mathcal{A}$;
    $A = A \cup k$;
  end
  $L(A) = \{$round-robin tournament on $A\}$;
  $d = \{$arbitrary matching on $[2N] \setminus A\}$;
  for $m \in L(A)$ **do**
    Sample $m \cup d$;
    for $i \in U$ **do**
      if $(i,j) \in m$ *and* $j \in T$ **then**
        $X(i,j) += x_{i,j,t}$;
        $C(i,j) += 1$;
      end
    end
  end
  $Q^U_+, Q^U_- \leftarrow$ `confidence_bound`(X, C, U);
  $U \leftarrow$ `unranked`(Q^U_+, Q^U_-, U);
end

The function `confidence_bound` in Algorithm 21 computes confidence bounds for the expected reward per item when matched with an item within T. If $\sum_{j=1}^{2N} C(i,j) = k_l$ with

$k_l = \lceil 4^{l+1} \log(\beta_l) \rceil$, $\beta_l = \pi\sqrt{(2N)/(3\delta)} \cdot l$, for some l, then

$$Q_-(i) = \frac{\sum_{j=1}^{2N} X(i,j)}{k_l} - \sqrt{\frac{\log(\beta_l)}{k_l}}$$

and

$$Q_+(i) = \frac{\sum_{j=1}^{2N} X(i,j)}{k_l} + \sqrt{\frac{\log(\beta_l)}{k_l}}.$$

The function `unranked` in Algorithm 21 removes an item i from $\mathcal{S}$ if its associated confidence interval intersects with no other intervals, or if it intersects only with that of another item that would be its optimal match independently of their relative order.

Proof of the upper bound

We prove the following theorem.

Theorem 3.A.19 (upper bound)**.** *For any $\delta > 0$, the sample complexity of the* Matching-id *algorithm satisfies*

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{\gamma_{\min}^2} \log(1/\delta) + c_{\mathfrak{p}}$$

with probability at least $1-\delta$. Moreover, if $s \geq 2$, then by denoting,

$$\alpha := \min\left\{ \left(\frac{1}{4} \frac{(u_1+u_2)\Delta_{2s,2s+1}}{\mu_{[2N]\setminus\{2h,2h+1\}}\Delta_{2h,2h+1}} \right)^2 1 \right\},$$

the following holds with probability at least $1-\delta$,

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{(1-\alpha)^2(u_1^2+u_2^2)\Delta_{2s,2s+1}^2} \log(1/\delta) + c_{\mathfrak{p}}.$$

else, if $s = 1$, then by denoting,

$$\alpha := \min\left\{ \left(\frac{u_1/4\Delta_{2s,2s+1}}{\mu_{[2N]\setminus\{2h,2h+1\}}\Delta_{2h,2h+1}} \right)^2, 1 \right\}$$

the following holds with probability at least $1-\delta$,

$$\tau_\delta \leq c_{\mathfrak{u}} \frac{1}{(1-\alpha)^2(u_1^2)\Delta_{2s,2s+1}^2} \log(1/\delta) + c_{\mathfrak{p}}.$$

The set constructed by the algorithm, $\mathcal{B}$ and $\mathcal{S}$ have the following properties.

Proposition 3.A.20. *The mean of the elements in $|\mathcal{B}|$ increases at the set of un-ranked items $\mathcal{S}$ gets smaller.*

The proposition is implied by the definition of $\mathcal{B}$.

Proposition 3.A.21. *It always holds that $|\mathcal{B}| \leq 2|\mathcal{S}|$.*

Proof: If item i with $i > |\mathcal{S}|$ is ranked, then for any other item j that is not i's neighbor, it is known whether $i < j$ or $j < i$. If j is i's neighbor, either it is known whether $i < j$ or $j < i$, or for all other item k, it is known whether $k < \min\{i,j\}$ or $k > \max\{i,j\}$. Thus $\mathcal{B} \subset [i]$. □

We introduce the following notation:

$$\mu_t(\mathcal{B} \setminus \{k,l\}) = \frac{1}{t}\sum_{s=1}^{t}\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}\setminus\{k,l\}} u_i.$$

Two relatively un-ranked i,j items are items in U s.t. $Q_+(i) > Q_-(j)$ and $Q_+(j) > Q_-(i)$. Any two relatively un-ranked items i,j either both belong to $\mathcal{B}$, in which case,

$$\frac{\sum_{l\in\mathcal{B}\setminus i} u_i u_l}{|\mathcal{B}\setminus j|} - \frac{\sum_{l\in\mathcal{B}\setminus j} u_j u_l}{|\mathcal{B}\setminus j|} = (u_i - u_j)\frac{\sum_{l\in T\setminus\{i,j\}} u_l}{|\mathcal{B}|-1},$$

or, none of them belongs to $\mathcal{B}$, in which case

$$\frac{\sum_{l\in\mathcal{B}\setminus i} u_i u_l}{|\mathcal{B}\setminus j|} - \frac{\sum_{l\in\mathcal{B}\setminus j} u_j u_l}{|\mathcal{B}\setminus j|} = (u_i - u_j)\frac{\sum_{l\in\mathcal{B}} u_l}{|\mathcal{B}|}.$$

Thus, for any two relatively un-ranked items i,j, at every iteration

$$\mathbb{E}[\frac{\sum_{l\in[2N]} X(i,l)}{\sum_{l\in[2N]} C(i,l)} - \frac{\sum_{l\in[2N]} X(j,l)}{\sum_{l\in[2N]} C(j,l)}] \geq \Delta_{i,j}\mu_t(\mathcal{B}\setminus\{i,j\}).$$

According to Proposition 3.A.20, we have

$$\mu_t(T\setminus\{i,j\}) \geq \frac{1}{2N}\sum_{l\in[2N]\setminus\{i,j\}} u_l.$$

Thus, items i and j are guaranteed to be relatively ranked by the first iteration where $\sum_{j=1}^{2N} C(i,j) = k_l$ with

$$\frac{1}{2^{l+1}} < \sqrt{\frac{\log(\beta_l)}{k_l}} < \frac{1}{4}\Delta_{i,j}(\frac{1}{2N}\sum_{l\in[2N]\setminus\{i,j\}} u_l) := \frac{1}{4}\tilde{\Delta}_{i,j}.$$

This implies

$$\frac{1}{4}\tilde{\Delta}_{i,j} \leq \frac{1}{2^l}.$$

Thus, we have

$$k_l \leq \frac{32}{\tilde{\Delta}_{i,j}^2}\log(\frac{1}{\delta}) + \frac{32}{\tilde{\Delta}_{i,j}^2}\log\left(2-\log_2(\tilde{\Delta}_{i,j})\right)^2 + \frac{32}{\tilde{\Delta}_{i,j}^2}\log(\frac{2N\pi^2}{3}) := \frac{32}{\tilde{\Delta}_{i,j}^2}\left[\log(\frac{1}{\delta}) + f(\tilde{\Delta}_{i,j})\right] \tag{3.35}$$

Proposition 3.A.21 implies that at every iteration, for any $i \in U$:

$$\frac{\sum_{j\in[2N]} C(i,j)}{t} \geq \frac{4}{9}. \tag{3.36}$$

Recall

$$\mu_{[2N]\setminus\{2k,2k+1\}} = \frac{1}{2N}\sum_{i\in[2N]\setminus\{2k,2k+1\}} u_i$$

and

$$\gamma_{\min} = \min_{k\in[N-1]} \{\mu_{[2N]\setminus\{2k,2k+1\}}\Delta_{2k,2k+1}\}.$$

Equations 3.35 and 3.36 imply that sample complexity of the MATCHING-ID algorithm is upper bounded as:

$$\tau_\delta \leq \frac{72}{\gamma_{\min}^2}\left[\log(\frac{1}{\delta}) + f\left(\gamma_{\min}\right)\right]$$

with probability at least $1-\delta$.

Let $s = \arg\min_{k\in[1,N-1]} \Delta_{2k,2k+1}$, and let $h = \arg\min_{k\in[1,N-1]\setminus s} \Delta_{2k,2k+1}\mu_{[2N]\setminus\{2k,2k+1\}}$. Let τ_2 be the stopping time at which all items except those in pairs s and $s+1$ are ranked. Under the "good event", the number of counted samples for unranked items in pairs s and $s+1$ before τ_2, $\tilde{\tau}_2$ is upper bounded as

$$\tilde{\tau}_2 \leq (\frac{32}{\Delta_{2h,2h+1}\mu_{[2N]\setminus\{2h,2h+1\}}})^2[\log(\frac{1}{\delta}) + f(\Delta_{2h,2h+1}\mu_{[2N]\setminus\{2h,2h+1\}})].$$

In the case where $s \geq 2$, for any $t > \tau_2$, $\mathcal{B} \subset [4]$, thus the following holds:

$$\mu_t(T\setminus\{2s,2s+1\}) \geq \frac{u_1+u_2+u_3+u_4}{4} + \frac{\tilde{\tau}_2}{\tilde{t}}(\frac{1}{2N}\sum_{l\in[2N]\setminus\{2s,2s+1\}} u_l - \frac{u_1+u_2+u_3+u_4}{4})$$

At each iteration of the algorithm, the following bound holds on the number of counted samples for un-ranked items in pairs $s, s+1$ $\tilde{t}$

$$\tilde{t} \leq \frac{32}{(\Delta_{2s,2s+1}\mu_t(T\setminus\{2s,2s+1\})^2}[\log(\frac{1}{\delta}) + f(\Delta_{2s,2s+1}\mu_t(\mathcal{B}\setminus\{2s,2s+1\})]$$

Now, either

$$\tilde{\tau}_\delta \leq \frac{32}{(\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4})^2}[\log(\frac{1}{\delta}) + f(\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4})^2)]$$

which implies a bound on τ_δ by $\tau_\delta \leq 9/4\tilde{\tau}_\delta$, or

$$\sqrt{\frac{\tilde{\tau}_2}{\tilde{\tau}_\delta}} \leq \frac{\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4}}{\Delta_{2h,2h+1}\mu_{[2N]\setminus\{2h,2h+1\}}}\sqrt{\frac{\log(\frac{1}{\delta}) + f\left(\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4}\right)}{\log(\frac{1}{\delta}) + f\left(\Delta_{2h,2h+1}\mu_{[2N]\setminus\{2h,2h+1\}}\right)}}.$$

Since f is a decreasing function, if for some $0<\alpha<1$,

$$\left(\frac{\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4}}{\Delta_{2h,2h+1}\mu_{[2N]\setminus\{2h,2h+1\}}}\right)^2 \leq \alpha$$

then

$$\tau_\delta \leq \frac{1}{(1-\alpha)^2}\frac{72}{(\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4})^2}\left[\log(\frac{1}{\delta}) + f\left((1-\alpha)\Delta_{2s,2s+1}\frac{u_1+u_2+u_3+u_4}{4}\right)\right]$$

which complicates the proof in the case where $s \geq 2$.

In the case when $s=1$, we have

$$\Delta_{1,3}\mu_t(\mathcal{B}\setminus\{1,3\}) \geq \Delta_{2,3}\mu_t(\mathcal{B}\setminus\{2,3\}).$$

Thus, at every iteration of the algorithm, it holds

$$\tilde{t} \leq \frac{32}{\left(\Delta_{2,3}\mu_t\left(\mathcal{B}\setminus\{2,3\}\right)\right)^2}[\log(\frac{1}{\delta}) + f(\Delta_{2,3}\mu_t(\mathcal{B}\setminus\{2,3\})].$$

Therefore, repeating the steps of the previous proof, if for some $0 < \alpha < 1$,

$$\left(\frac{\Delta_{2,3}\frac{u_1}{4}}{\Delta_{2h,2h+1}\frac{1}{2N}\sum_{l\in[2N]\setminus\{2h,2h+1\}}u_l}\right)^2 \leq \alpha$$

then

$$\tau_\delta \leq \frac{1}{(1-\alpha)^2}\frac{72}{(\Delta_{2,3}\frac{u_1}{4})^2}\left[\log(\frac{1}{\delta}) + f\left((1-\alpha)\Delta_{2,3}\frac{u_1}{4}\right)\right].$$

Proof of the lower bound

We prove the following theorem.

Theorem 3.A.22. *Assume that stochastic rewards of item pairs have Gaussian distribution with unit variance. Then, for any δ-PAC algorithm, we have*

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}} \sum_{i\in[2N]:\Delta_i>0} \frac{1}{\sum_{j=1}^{2N} u_j^2}\frac{1}{\Delta_i^2}\log(1/\delta).$$

Moreover, if $s \geq 2$, then

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}}\frac{1}{u_1^2+u_2^2}\frac{1}{\Delta_{2s,2s+1}^2}\log(1/\delta)$$

else, if $s = 1$, then

$$\mathbb{E}[\tau_\delta] \geq c_{\mathfrak{u}}\frac{1}{u_1^2}\frac{1}{\Delta_{2,3}^2}\log(1/\delta).$$

Let us first rewrite the linear program that gives a lower bound on the sampling complexity. Let us define

$$\Lambda(U) = \{\lambda \in [0,1]^{2N} \mid m^*(\lambda) \neq m^*(U)\}/$$

to be the class of alternative models that give a different optimal arms. The expected sampling complexity is lower bounded by the solution of the following linear program:

$$\begin{aligned}&\text{minimize} \sum_{m\in\mathcal{M}} \eta_m \\ &\text{subject to} \sum_{m\in\mathcal{M}} \eta_m d(u,\lambda) \geq d(\delta,1-\delta), \text{ for all } \lambda\in\Lambda(U).\end{aligned}$$

Which is equivalent to

$$\begin{aligned}&\text{minimize } \frac{1}{4N}\sum_{1=1}^{2N}\sum_{j=1}^{2N}\sum_{m\in\mathcal{M}} \eta_m \mathbb{1}_{\{(i,j)\in m\}} \\ &\text{subject to} \sum_{1=1}^{2N}\sum_{j=1}^{2N}\sum_{m\in\mathcal{M}} \eta_m \mathbb{1}_{\{(i,j)\in m\}} d(u_iu_j,\lambda_i\lambda_j) \geq d(\delta,1-\delta), \text{ for all } \lambda\in\Lambda(U).\end{aligned}$$

Note that for any matching sampling vector, the following always holds for some constant c and every $(i,j) \in [2N]^2$:

$$\begin{cases} \sum_{j=1}^{2N} \sum_{m \in \mathcal{M}} \eta_m \mathbb{1}_{\{(i,j) \in m\}} = c \\ \sum_{m \in \mathcal{M}} \eta_m \mathbb{1}_{\{(i,j) \in m\}} = \sum_{m \in \mathcal{M}} \eta_m \mathbb{1}_{\{(j,i) \in m\}} \\ \sum_{m \in \mathcal{M}} \eta_m \mathbb{1}_{\{(i,i) \in m\}} = 0. \end{cases}$$

Thus the solution of the previous linear program is lower bounded by that of the following one, $\mathbf{LP}_m$,

$$\begin{aligned} \text{minimize } & \frac{c}{2} \\ \text{subject to } & \frac{1}{2} \sum_{(i,j) \in [2N]^2} x_{i,j} d(u_i u_j, \lambda_i \lambda_j) \geq d(\delta, 1-\delta), \ \forall \lambda \in \Lambda(U) \\ & \sum_{j=1}^{2N} x_{i,j} = c, \text{ for all } i \in [2N] \\ & x_{i,j} = x_{j,i}, \ x_{i,i} = 0, \text{ for all } (i,j) \in [2N]^2. \end{aligned}$$

In the case where the stochastic rewards are Gaussian random variables with unit variance, $d(u_i u_j, \lambda_i \lambda_j) = (u_i u_j - \lambda_i \lambda_j)^2$.

We define $s = \arg\min_{k \in [2, N-1]} \Delta_{2k,2k+1}$. By considering alternative models $\lambda_{2s} = u_{2s} + \Delta_{2s,2s+1}$ and $\lambda_{2s+1} = u_{2s+1} - \Delta_{2s,2s+1}$, we obtain our first lower bound

$$\mathbb{E}[\tau_\delta] \geq \frac{1}{2} \frac{1}{u_1^2 + u_2^2} \frac{1}{\Delta_{2s,2s+1}^2} \left(\log\left(\frac{1}{\delta}\right) - 1 \right). \tag{3.37}$$

Let us change variables in the linear program $\mathbf{LP}_m$ for $z_{i,j} = u_j x_{i,j}$. By considering the class of alternative models $\lambda_i = u_i \pm \Delta_i$, and dropping the constraint $x_{i,j} = x_{j,i}$ for i, j ... we get that the solution of the following linear program is a lower bound on the sampling complexity

$$\begin{aligned} \text{minimize } & \frac{c}{2} \\ \text{subject to } & \sum_{i=1}^{2N} z_{i,j} \geq \frac{1}{\Delta_j^2} d(\delta, 1-\delta), \quad \text{for all } j \in [2N] \\ & \sum_{j=1}^{2N} z_{i,j} = c u_i^2, \text{ for all } i \in [2N] \\ & z_{i,i} = 0, \text{ for all } (i,j) \in [2N]^2. \end{aligned}$$

Note that $\sum_{i,j} z_{i,j} = c \sum_j u_j^2 \geq \sum_{i:\Delta_i > 0} (1/\Delta_i^2) d(\delta, 1-\delta)$. This gives the second bound on the sampling complexity

$$\mathbb{E}[\tau_\delta] \geq \left(\sum_{i: \in \Delta_i > 0} \frac{1}{\Delta_i^2} \right) \frac{1}{\sum_j u_j^2} \left(\log\left(\frac{1}{\delta}\right) - 1 \right). \tag{3.38}$$

□

Finally, consider alternative model $\lambda_2 = u_2 + \Delta_{2,3}$ and $\lambda_3 = u_3 - \Delta_{2,3}$. In this case, we first obtain the same lower bound as in Equation (3.37) by removing the constraint $x_{2,2} = 0$, of the

linear program $\mathbf{LP}_m$, then note that the obtained lower bound implies the following one

$$\mathbb{E}[\tau_\delta] \geq \frac{1}{4}\frac{1}{u_1^2}\frac{1}{\Delta_{2,3}^2}\left(\log\left(\frac{1}{\delta}\right) - 1\right). \tag{3.39}$$

□

Chapter 4

Decentralized Learning in Online Queuing Systems

Motivated by packet routing in computer networks and resource allocation in radio networks, online queuing systems are composed of queues receiving packets at different rates. Repeatedly, they send packets to servers, each of them treating only at most one packet at a time. In the centralized case, the number of accumulated packets remains bounded (i.e., the system is *stable*) as long as the ratio between service rates and arrival rates is larger than 1. In the decentralized case, individual no-regret strategies ensures stability when this ratio is larger than 2. Yet, myopically minimizing regret disregards the long term effects due to the carryover of packets to further rounds. On the other hand, minimizing long term costs leads to stable Nash equilibria as soon as the ratio exceeds $\frac{e}{e-1}$. Stability with decentralized learning strategies with a ratio below 2 was a major remaining question. We first argue that for ratios up to 2, cooperation is required for stability of learning strategies, as selfish minimization of policy regret, a *patient* notion of regret, might indeed still be unstable in this case. We therefore consider cooperative queues and propose the first learning decentralized algorithm guaranteeing stability of the system as long as the ratio of rates is larger than 1, thus reaching performances comparable to centralized strategies.

Contents

4.1 Introduction

Inefficient decisions in repeated games can stem from both strategic and learning considerations. First, strategic agents selfishly maximize their own individual reward at others' expense. The *price of anarchy* (Koutsoupias and Papadimitriou, 1999) measures this inefficiency as the social welfare ratio between the best possible situation and the worst Nash equilibrium. Although reaching the best collective outcome might be illusory for selfish agents, considering the worst Nash equilibrium might be too pessimistic. In games with external factors, more complex interactions intervene and might lead the agents to the best equilibrium. Instead, the *price of stability* (Schulz and Moses, 2003) measures the inefficiency by the social welfare ratio between the best possible situation and the best Nash equilibrium.

Second, the agents also have to learn their environment, by repeatedly experimenting different outcomes. Learning equilibria in repeated games is at the core of many problems in computer science and economics (Cesa-Bianchi and Lugosi, 2006; Fudenberg et al., 1998). The interaction between multiple agents can indeed interfere in the learning process, potentially converging to no or bad equilibria. It is yet known that in repeated games, if all agents follow no internal regret strategies, their actions converge in average to the set of correlated equilibria (Blum and Monsour, 2007; Hart and Mas-Colell, 2000; Perchet, 2014).

Many related results are known in the classical repeated games (see e.g., Cesa-Bianchi and Lugosi, 2006; Roughgarden, 2010), where a single game is repeated over independent rounds (but the agents strategies might evolve and depend on the history). Motivated by packet routing in computer networks, Gaitonde and Tardos (2020a) introduced a repeated game with a *carryover* feature: the outcome of a round does not only depend on the actions of the agents, but also on the previous rounds. They consider heterogeneous queues sending packets to servers. If several queues simultaneously send packets to the same server, only the oldest packet is treated by the server.

Because of this carryover effect, little is known about this type of game. In a first paper, Gaitonde and Tardos (2020a) proved that if queues follow suitable no-regret strategies, a ratio of 2 between server and arrival rates leads to stability of the system, meaning that the number of packets accumulated by each queue remains bounded. However, the assumption of regret minimization sort of reflects a myopic behavior and is not adapted to games with carryover. Gaitonde and Tardos (2020b) subsequently consider a patient game, where queues instead minimize their asymptotic number of accumulated packets. A ratio only larger than $\frac{e}{e-1}$ then guarantees the stability of the system, while a smaller ratio leads to inefficient Nash equilibria. As a consequence, going below the $\frac{e}{e-1}$ factor requires some level of cooperation between the queues. This result actually holds with perfect knowledge of the problem parameters and it remained even unknown whether decentralized learning strategies can be stable with a ratio below 2.

We first argue that decentralized queues need some level of cooperation to ensure stability with a ratio of rates below 2. Policy regret can indeed be seen as a patient alternative to the regret notion. Yet even minimizing the policy regret might lead to instability when this ratio is below 2. An explicit decentralized cooperative algorithm called ADEQUA (A DEcentralized QUeuing Algorithm) is thus proposed. It is the first decentralized learning algorithm guaranteeing stability when this ratio is only larger than 1. ADEQUA does not require communication between the queues, but uses synchronisation between them to accurately estimate the problem parameters and avoid interference when sending packets. Our main result is given by Theorem 4.1.1 below, whose formal version, Theorem 4.4.9 in Section 4.4, also provides bounds on the number of accumulated packets.

Theorem 4.1.1 (Theorem 4.4.9, informal)**.** *If the ratio between server rates and arrival rates*

is larger than 1 *and all queues follow* ADEQUA*, the system is strongly stable.*

The remaining of the chapter is organised as follows. The model and existing results are recalled in Section 4.2. Section 4.3 argues that cooperation is required to guarantee stability of learning strategies when the ratio of rates is below 2. ADEQUA is then presented in Section 4.4, along with insights for the proof of Theorem 4.1.1. Section 4.5 finally compares the behavior of ADEQUA with no-regret strategies on toy examples and empirically confirms the different known theoretical results.

4.1.1 Additional related work

Queuing theory includes applications in diverse areas such as computer science, engineering, operation research (Shortle et al., 2018). Borodin et al. (1996) for example use the stability theorem of Pemantle and Rosenthal (1999), which was also used by Gaitonde and Tardos (2020b), to study the problem of packet routing through a network. Our setting is the single-hop particular instance of throughput maximization in wireless networks. Motivated by resource allocation in multihop radio problem, packets can be sent through more general routing paths in the original problem. Tassiulas and Ephremides (1990) proposed a first stable centralized algorithm, when the service rates are known *a priori*. Stable decentralized algorithms were later introduced in specific cases (Jiang and Walrand, 2009; Neely et al., 2008; Shah and Shin, 2012), when the rewards $X_k(t)$ are observed before deciding which server to send the packet. The main challenge then is coordination and queues aim at avoiding collisions with each other. The proposed algorithms are thus not adapted to our setting, where both coordination between queues and learning the service rates are required. We refer the reader to (Georgiadis et al., 2006) for an extended survey on resource allocation in wireless networks.

Krishnasamy et al. (2016) first considered online learning for such queuing systems model, in the simple case of a single queue. It is a particular instance of stochastic multi-armed bandits, a celebrated online learning model, where the agent repeatedly takes an action within a finite set and observes its associated reward. This model becomes intricate when considering multiple queues, as they interfere when choosing the same server. It is then related to the multiplayer bandits problem which considers multiple players simultaneously pulling arms. When several of them pull the same arm, a *collision* occurs and they receive no reward (Anandkumar et al., 2010).

The collision model is here different as one of the players still gets a reward. It is thus even more closely related to competing bandits (Liu et al., 2020a,b), where arms have preferences over the players and only the most preferred player pulling the arm actually gets the reward. Arm preferences are here not fixed and instead depend on the packets' ages. While collisions can be used as communication tools between players in multiplayer bandits (Bistritz and Leshem, 2018; Boursier and Perchet, 2019; Mehrabian et al., 2020; Wang et al., 2020), this becomes harder with an asymmetric collision model as in competing bandits. However, some level of communication remains possible (Basu et al., 2021; Sankararaman et al., 2020). In queuing systems, collisions are not only asymmetric, but depend on the age of the sent packets, making such solutions unsuited.

While multiplayer bandits literature considers cooperative players, Boursier and Perchet (2020) showed that cooperative algorithms could be made robust to selfish players. On the other hand, competing bandits consider strategic players and arms as the goal is to reach a bipartite stable matching between them. Despite being cooperative, ADEQUA also has strategic considerations as the queues' strategy converges to a correlated equilibrium of the patient game described in Section 4.2.

An additional difficulty here appears as queues are asynchronous: they are not active at each round, but only when having packets left. This is different from the classical notion of asynchronicity (Bonnefoi et al., 2017), where players are active at each round with some fixed probability. Most strategies in multiplayer bandits rely on synchronisation between the players (Boursier and Perchet, 2019) to avoid collisions. While such a level of synchronisation is not possible here, some lower level of synchronisation is still used to avoid collisions between queues.

4.2 Queuing Model

We consider a queuing system composed of N queues and K servers, associated with vectors of arrival and service rates $\boldsymbol{\lambda}, \boldsymbol{\mu}$, where at each time step $t = 1, 2, \dots$, the following happens:

- each queue $i \in [N]$ receives a new packet with probability $\lambda_i \in [0, 1]$, that is marked with the timestamp of its arrival time. If the queue currently has packet(s) on hold, it sends one of them to a chosen server j based on its past observations.
- Each server $j \in [K]$ attempts to clear the oldest packet it has received, breaking ties uniformly at random. It succeeds with probability $\mu_j \in [0, 1]$ and otherwise sends it back to its original queue, as well as all other unprocessed packets.

At each time step, a queue only observes whether or not the packet sent (if any) is cleared by the server. We note Q_t^i the number of packets in queue i at time t. Given a packet-sending dynamics, the system is **stable** if, for each i in $[N]$, Q_t^i/t converges to 0 almost surely. It is **strongly stable**, if for any $r, t \geq 0$ and $i \in [N]$, $\mathbb{E}[(Q_t^i)^r] \leq C_r$, where C_r is an arbitrarily large constant, depending on r but not t. Without ambiguity, we also say the policy or the queues are (strongly) stable. Naturally, a strongly stable system is also stable (Gaitonde and Tardos, 2020a).

In the following, $x_{(i)}$ will denote the i-th order statistics of a vector $\boldsymbol{x}$, i.e., $\lambda_{(1)} \geq \lambda_{(2)} \geq \dots \geq \lambda_{(N)}$ and $\mu_{(1)} \geq \dots \geq \mu_{(K)}$. Without loss of generality, we assume $K \geq N$ (otherwise, we simply add fictitious servers with 0 service rate). The key quantity of a system is its **slack**, defined as the largest real number η such that:

$$\sum_{i=1}^{k} \mu_{(i)} \geq \eta \sum_{i=1}^{k} \lambda_{(i)}, \ \forall\ k \leq N.$$

We also denote by $\mathcal{P}([K])$ the set of probability distributions on $[K]$ and by Δ the **margin** of the system defined by

$$\Delta \coloneqq \min_{k \in [N]} \frac{1}{k} \sum_{i=1}^{k} (\mu_{(i)} - \lambda_{(i)}). \tag{4.1}$$

Notice that the alternative system where $\tilde{\lambda}_i = \lambda_i + \Delta$ and $\tilde{\mu}_k = \mu_k$ has a slack 1. In that sense, Δ is the largest *margin* between service and arrival rates that all queues can individually get in the system. Note that if $\eta > 1$, then $\Delta > 0$. We now recall existing results for this problem, summarized in Figure 4.1 below.

Theorem 4.2.1 (Marshall et al. 1979)**.** *For any instance, there exists a strongly stable centralized policy if and only if $\eta > 1$.*

Theorem 4.2.2 (Gaitonde and Tardos 2020a, informal)**.** *If $\eta > 2$, queues following appropriate no regret strategies are strongly stable.*
For each $N > 0$, there exists a system and a dynamic s.t. $\eta > 2 - o(1/N)$, all queues follow appropriate no-regret strategies, but they are not strongly stable.

In the above theorem, an *appropriate no regret strategy* is a strategy such that there exists a partitioning of the time into successive windows, for which the incurred regret is $o(w)$ with high probability on any window of length w. This for example includes the EXP3.P.1 algorithm (Auer et al., 2002b) where the k-th window has length 2^k.

The patient queuing game $\mathcal{G} = ([N], (c_i)_{i=1}^n, \boldsymbol{\mu}, \boldsymbol{\lambda})$ is defined as follows. The strategy space for each queue is $\mathcal{P}([K])$. Let $\boldsymbol{p}_{-i} \in (\mathcal{P}([K]))^{N-1}$ denote the vector of fixed distributions for all queues over servers, except for queue i. The cost function for queue i is defined as:

$$c_i(p_i, \boldsymbol{p}_{-i}) = \lim_{t \to +\infty} T_t^i / t,$$

where T_t^i is the age of the oldest packet in queue i at time t. Bounding T_t^i is equivalent to bounding Q_t^i.

Theorem 4.2.3 (Gaitonde and Tardos 2020b, informal)**.** *If $\eta > \frac{e}{e-1}$, any Nash equilibrium of the patient game $\mathcal{G}$ is stable.*

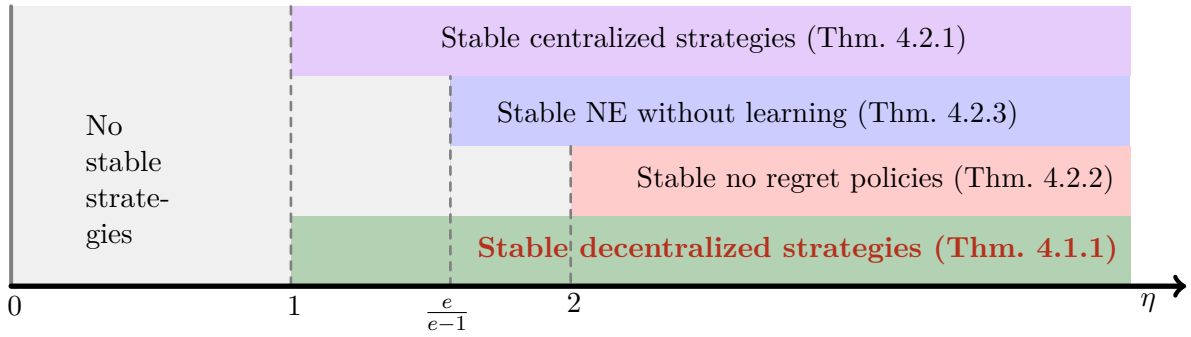


Figure 4.1: Existing results depending on the slack η. Our result is highlighted in red.

4.3 The case for a cooperative algorithm

According to Theorems 4.2.2 and 4.2.3, queues that are patient enough and select a fixed randomization over the servers are stable over a larger range of slack η than queues optimizing their individual regret. A key difference between the two settings is that when minimizing their regret, queues are myopic, which is formalized as follows. Let $a_{1:t} = (a_1, ..., a_t)$ be the vector of actions played by a queue up to time t and let $\nu_t(a_{1:t})$ be the indicator that it cleared a packet at iteration t. Classical (external) regret of queue i over horizon T is then defined as:

$$R_i^{\text{ext}}(T) := \max_{p \in \mathcal{P}([K])} \sum_{t=1}^{T} \mathbb{E}_{\tilde{a}_t \sim p}[\nu_t(a_{1:t-1}, \tilde{a}_t)] - \sum_{t=1}^{T} \nu_t(a_{1:t}).$$

Thus minimizing the external regret is equivalent to maximizing the instant rewards at each iteration, ignoring the consequences of the played action on the state of the system. However, in the context of queuing systems, the actions played by the queues change the state of the system. Notably, letting other queues clear packets can be in the best interest of a queue, as it may give it priority in the subsequent iterations where it holds older packets. Since the objective is to maximize the total number of packets cleared, it seems adapted to minimize a *patient* version of the regret, namely the policy regret (Arora et al., 2012), rather than the external regret, which

is defined by

$$R_i^{\mathrm{pol}}(T) := \max_{p \in \mathcal{P}([K])} \sum_{t=1}^{T} \mathbb{E}_{\tilde{a}_{1:t} \sim \otimes_{i=1}^{t} p}[\nu_t(\tilde{a}_{1:t})] - \sum_{t=1}^{T} \nu_t(a_{1:t}).$$

That is, $R_i^{\mathrm{pol}}(T)$ is the expected difference between the number of packets queue i cleared and the number of packets it would have cleared over the whole period by playing a fixed (possibly random) action, taking into account how this change of policy would affect the state of the system.

However, as stated in Theorem 4.3.1, optimizing this patient version of the regret rather than the myopic one could not guarantee stability on a wider range of slack value. This suggests that adding only patience to the learning strategy of the queues is not enough to go beyond a slack of 2, and that any strategy beating that factor 2 must somewhat include synchronisation between the queues.

Proposition 4.3.1. *Consider the partition of the time $t = 1, 2, \ldots$ into successive windows, where $w_k = k^2$ is the length of the k-th one. For any $N \geq 2$, there exists an instance with $2N$ queues and servers, with slack $\eta = 2 - \mathcal{O}\left(\frac{1}{N}\right)$, s.t., almost surely, each queue's policy regret is $o(w_k)$ on all but finitely many of the windows, but the system is not strongly stable.*

To ease comparison, the formulation in the above proposition matches that of the counter-example used to prove Theorem 4.2.2 (Gaitonde and Tardos 2020a). In that counter-example, a set of system parameters, for which *any* no external regret policies were unstable, was exhibited. Whereas we exhibit in our case a *specific* strategy that satisfies the no policy regret condition, but is unstable.

Sketch of proof. Consider a system with $2N$ queues and servers with $\lambda_i = 1/2N$ and $\mu_i = 1/N - 1/4N^2$ for all $i \in [2N]$. The considered strategy profile is the following. For each $k \geq 0$, the k^{th} time window is split into two stages. During the first stage, of length $\lceil \alpha w_k \rceil$, queues $2i$ and $2i+1$ both play server $2i + t \pmod{2N}$ at iteration t, for all $i \in [N]$. During the second stage of the time window, queue i plays server $i + t \pmod{2N}$ at iteration t. This counter example, albeit very specific, illustrates well how when the queues are highly synchronised, it is better to remain synchronized rather than deviate, even if the synchronisation is suboptimal in terms of stability. The complete proof is provided in Section 4.A.3.

Queues following this strategy accumulate packets during the first stage, and clear more packets than they receive during the second stage. The value of α is tuned so that the queues still accumulate a linear portion of packets during each time window. For those appropriate α, the system is unstable.

Now suppose that queue i deviates from the strategy and plays a fixed action $p \in \mathcal{P}([K])$. In the first stage of each time window, queue i can clear a bit more packets than it would by not deviating. However, during the second stage, it is no longer synchronised with the other queues and collides with them a large number of times. Because of those collisions, it will accumulate many packets. In the detailed analysis, we demonstrate that, in the end, for appropriate values of α, queue i accumulates more packets than it would have without deviating. □

According to Theorem 4.2.3, the factor $\frac{e}{e-1}$ can be seen as the price of anarchy of the problem, as for slacks below, the worst Nash equilibria might be unstable. On the other hand, it is known that for any slack above 1, there exists a centralized stable strategy. This centralized strategy actually consists in queues playing the same joint probability at each time step, independently from the number of accumulated packets. As a consequence, it is also a correlated equilibrium of the patient game and 1 can be seen as the correlated price of stability.

4.4 A decentralized algorithm

This section describes the decentralized algorithm ADEQUA, whose pseudocode is given in Algorithm 22. Due to space constraints, all the proofs are postponed to Section 4.A.4. ADEQUA assumes all queues *a priori* know the number N of queues in the game and have a unique rank or *id* in $[N]$. Moreover, the existence of a shared randomness between all queues is assumed. The *id* assumption is required to break the symmetry between queues and is classical in multiplayer bandits without collision information. On the other side, the shared randomness assumption is equivalent to the knowledge of a common seed for all queues, which then use this common seed for their random generators. A similar assumption is used in multiplayer bandits (Bubeck et al., 2020).

ADEQUA is inspired by the celebrated ε-greedy strategy. With probability $\varepsilon_t = (N+K)t^{-\frac{1}{5}}$, at each time step, queues explore the different parameters λ_i and μ_i as described below. Otherwise with probability $1-\varepsilon_t$, they exploit the servers. Each queue i then sends a packet to a server following a policy solely computed from its local estimates $\hat{\lambda}^i, \hat{\mu}^i$ of the problem parameters λ and μ. The shared randomness is here used so that exploration simultaneously happens for all queues. If exploration/exploitation was not synchronized between the queues, an exploiting queue could collide with an exploring queue, biasing the estimates $\hat{\lambda}^i, \hat{\mu}^i$ of the latter.

Algorithm 22: ADEQUA

input : N (number of queues), $i \in [N]$ (queue *id*)

for $t = 1, \dots, \infty$ **do**
- $\hat{P} \leftarrow \phi(\hat{\lambda}, \hat{\mu})$ and $\hat{A} \leftarrow \psi(\hat{P})$ where ϕ and ψ are resp. defined by Equations (4.3) and (4.4)
- Draw $\omega_1 \sim \text{Bernoulli}((N+K)t^{-\frac{1}{5}})$ and $\omega_2 \sim \mathcal{U}(0,1)$ `// shared randomness`
- **if** $\omega_1 = 1$ **then** EXPLORE(i) `// exploration`
- **else** Pull $\hat{A}(\omega_2)(i)$ `// exploitation`

end

Exploration. When exploring, queues choose either to explore the servers' parameters μ_k or the other queues' parameters λ_i as described in Algorithm 23 below. In the former case, all queues choose different servers at random (if they have packets to send). These rounds are used to estimate the servers means: $\hat{\mu}^i_k$ is the empirical mean of server k observed by the queue i for such rounds. Thanks to the shared randomness, queues pull different servers here, making the estimates unbiased.

In the latter case, queues explore each other in a pairwise fashion. When queues i and j explore each other at round t, each of them sends their **most recent** packet to some server k, chosen uniformly at random, if and only if a packet appeared during round t. In that case, we say that *the queue i explores λ_j* (and vice versa). To make sure that i and j are the only queues choosing the server k during this step, we proceed as follows:

- queues sample a matching π between queues at random. To do so, the queues use the same method to plan an all-meet-all (or round robin) tournament, for instance Berger tables (Berger, 1899), and choose uniformly at random which round of the tournament to play. If the number of queues N is odd, in each round of the tournament, one queue remains alone and does nothing.

- the queues draw the same number $l \sim \mathcal{U}([K])$ with their shared randomness. For each pair of queues (i, j) matched in π, associate $k_{(i,j)} = l + \min(i, j) \pmod K + 1$ to this pair. The queues i and j then send to the server $k_{(i,j)}$.

As we assumed that the server breaks ties in the packets' age uniformly at random, the queue i clears with probability $(1 - \frac{\lambda_j}{2})\bar{\mu}$, where $\bar{\mu} = \frac{1}{K}\sum_{k=1}^{K} \mu_k$. Thanks to this, λ_j is estimated by queue i as:

$$\hat{\lambda}_j^i = 2 - 2\hat{S}_j^i/\tilde{\mu}^i, \tag{4.2}$$

where $\tilde{\mu}^i = \frac{\sum_{k=1}^{K} N_k^i \hat{\mu}_k^i}{\sum_{k=1}^{K} N_k^i}$, N_k^i is the number of *exploration* pulls of server k by queue i and $\hat{S}_j^i$ is the empirical probability of clearing a packet observed by queue i when exploring λ_j.

Algorithm 23: EXPLORE

input : $i \in [N]$ `// queue id`

$k \leftarrow 0$

Draw $n \sim \mathcal{U}([N + K])$ `// shared randomness`

if $n \leq K$ **then** `// explore` μ

  $k \leftarrow n + i \pmod K + 1$

  Pull k ; Update N_k and $\hat{\mu}_k$

else `// explore` λ

  Draw $r \sim \mathcal{U}([N])$ and $l \sim \mathcal{U}([K])$ `// shared randomness`

  $j \leftarrow r^{\text{th}}$ opponent in the all-meet-all tournament planned according to Berger tables

  $k \leftarrow l + \min(i, j) \pmod K + 1$

  if $k \neq 0$ *and packet appeared at current time step* **then** `// explore` λ_j `on server` k

    Pull k with most recent packet ; Update $\hat{S}_j$ and $\hat{\lambda}_j$ according to Equation (4.2)

  end

end

Remark 4.4.1. *The packet manipulation when exploring λ_j strongly relies on the servers tie breaking rules (uniformly at random). If this rule was unknown or not explicit, the algorithm can be adapted: when queue i explores λ_j, queue j instead sends the packet generated at time $t - 1$ (if it exists), while queue i still sends the packet generated at time t. In that case, the clearing probability for queue i is exactly $(1 - \lambda_j)\bar{\mu}$, allowing to estimate λ_j. Anticipating the nature of the round t (exploration vs. exploitation) can be done by drawing $\omega_1 \sim \text{Bernoulli}(\varepsilon_t)$ at time $t - 1$. If $\omega_1 = 1$, the round t is exploratory and the packet generated at time $t - 1$ is then kept apart by the queue j.*

To describe the exploitation phase, we need a few more notations. We denote by $\mathfrak{B}_K$ the set of doubly stochastic matrices (non-negative matrices such that each of its rows and columns sums to 1) and by $\mathfrak{S}_K$ the set of permutation matrices in $[K]$ (a permutation matrix will be identified with its associated permutation for the sake of cumbersomeness).

A **dominant mapping** is a function $\phi : \mathbb{R}^N \times \mathbb{R}^K \to \mathfrak{B}_K$ which, from (λ, μ), returns a doubly stochastic matrix P such that $\lambda_i < (P\mu)_i$ for any $i \in [N]$ if it exists (and the identity matrix otherwise).

A **BvN** (Birkhoff von Neumann) **decomposition** is a function $\psi : \mathfrak{B}_K \to \mathcal{P}(\mathfrak{S}_K)$ that associates to any doubly stochastic matrix P a random variable $\psi(P)$ such that $\mathbb{E}[\psi(P)] = P$; stated otherwise, it expresses P as a convex combination of permutation matrices. For convenience, we will represent this random variable as a function from $[0, 1]$ (equipped with the uniform distribution) to $\mathfrak{S}_K$.

Informally speaking, those functions describe the strategies queues would follow in the centralized case: a dominant mapping gives adequate marginals ensuring stability (since the queue i clears in expectation $(P\mu)_i$ packets at each step, which is larger than λ_i by definition), while a BvN decomposition describes the associated coupling to avoid collisions. Explicitly, the joint strategy is for each queue to draw a shared random variable $\omega_2 \sim \mathcal{U}(0,1)$ and to choose servers according to the permutation $\psi(\phi(\lambda,\mu))(\omega_2)$

Exploitation. In a decentralized system, each queue i computes a mapping $\hat{A}^i := \psi(\phi(\hat{\lambda}^i, \hat{\mu}^i))$ solely based on its own estimates $\hat{\lambda}^i, \hat{\mu}^i$. A shared variable $\omega_2 \in [0,1]$ is then generated uniformly at random and queue i sends a packet to the server $\hat{A}^i(\omega_2)(i)$. If all queues knew exactly the parameters λ, μ, the computed strategies $\hat{A}^i$ would be identical and they would follow the centralized policy described above.

However, the estimates $(\hat{\lambda}^i, \hat{\mu}^i)$ are different between queues. The usual dominant mappings and BvN decompositions in the literature are non-continuous. Using those, even queues with close estimates could have totally different $\hat{A}^i$, and thus collide a large number of times, which would impede the stability of the system. Regular enough dominant mappings and BvN decompositions are required, to avoid this phenomenon. The design of ϕ and ψ is thus crucial and appropriate choices are given in the following Sections 4.4.1 and 4.4.2. Nonetheless, they can be used in some black-box fashion, so we provide for the sake of completeness sufficient conditions for stability, as well as a general result depending on the properties of ϕ and ψ, in Section 4.A.1.

Remark 4.4.2. *The exploration probability $t^{-\frac{1}{5}}$ gives the smallest theoretical dependency in Δ in our bound. A trade-off between the proportion of exploration rounds and the speed of learning indeed appears in the proof of Theorem 4.1.1. Exploration rounds have to represent a small proportion of the rounds, as the queues accumulate packets when exploring. On the other hand, if queues explore more often, the regime where their number of packets decreases starts earlier. A general stability result depending on the choice of this probability is given by Theorem 4.A.1 in Section 4.A.1.*
Yet in Section 4.5, taking a probability $t^{-\frac{1}{4}}$ empirically performs better as it speeds up the exploration.

4.4.1 Choice of a dominant mapping

Recall that a dominant mapping takes as inputs (λ, μ) and returns, if possible, a doubly stochastic matrix P such that

$$\lambda_i < \textstyle\sum_{k=1}^{K} P_{i,k}\mu_k \text{ for all } i \in [N].$$

The usual dominant mappings sort the vector λ and μ in descending orders (Marshall et al., 1979). Because of this operation, they are non-continuous and we thus need to design a regular dominant mapping satisfying the above property. Inspired by the log-barrier method, it is done by taking the minimizer of a strongly convex program as follows

$$\phi(\lambda,\mu) = \operatorname*{arg\,min}_{P\in\mathfrak{B}_K} \max_{i\in[N]} -\ln\left(\sum_{j=1}^{K} P_{i,j}\mu_j - \lambda_i\right) + \frac{1}{2K}\|P\|_2^2. \tag{4.3}$$

Although the objective function is non-smooth because of the max operator, it enforces fairness between queues and leads to a better regularity of the arg min.

Remark 4.4.3. *Computing ϕ requires solving a non-smooth strongly convex minimization problem. This cannot be computed exactly, but a good approximation can be quickly obtained using the scheme described in Section 4.A.2. If this approximation error is small enough, it has no impact on the stability bound of Theorem 4.4.9. It is thus ignored for simplicity, i.e., we assume in the following that $\phi(\lambda, \mu)$ is exactly computed at each step.*

As required, ϕ always returns a matrix P satisfying that $\lambda < P\mu$ if possible, since otherwise the objective is infinite (and in that case we assume that ϕ returns the identity matrix). Moreover, the objective function is $\frac{1}{K}$-strongly convex, which guarantees some regularity of the $\arg\min$, namely local-Lipschitzness, leading to Theorem 4.4.4 below.

Lemma 4.4.4. *For any (λ, μ) with positive margin Δ (defined in Equation (4.1)), if $\|(\hat{\lambda} - \lambda, \hat{\mu} - \mu)\|_\infty \leq c_1\Delta$, for any $c_1 < \frac{1}{2\sqrt{e}+2}$, then*

$$\|\phi(\hat{\lambda}, \hat{\mu}) - \phi(\lambda, \mu)\|_2 \leq \frac{c_2 K}{\Delta}\|(\hat{\lambda} - \lambda, \hat{\mu} - \mu)\|_\infty,$$

where $c_2 = \frac{4}{(1-2c_1)/\sqrt{e}-2c_1}$. Moreover, denoting $\hat{P} = \phi(\hat{\lambda}, \hat{\mu})$, it holds for any $i \in [N]$,

$$\lambda_i \leq \sum_{k=1}^K \hat{P}_{i,k}\mu_k - \left(\tfrac{1-2c_1}{\sqrt{e}} - 2c_1\right)\Delta.$$

The first property guarantees that if the queues have close estimates, they also have close doubly stochastic matrices $\hat{P}$. Moreover, the second property guarantees that any queue should clear its packets with a margin of order Δ, in absence of collisions.

Remark 4.4.5. *An alternative dominant mapping without the regularizing term in Equation (4.3) can also be proposed. Yet, its local Lipschitz bound would also depend on the smallest difference between the λ_i or the μ_i, which can be arbitrarily small. If two parameters λ_i or μ_i are equal, this choice of dominant mapping might lead to unstable policies. Using a regularization term in Equation (4.3) thus avoids this problem, although a smaller dependency might be possible without this regularization term when the parameters λ_i and μ_i are very distinct.*

4.4.2 Choice of a Birkhoff von Neumann decomposition

Given a doubly stochastic matrix $\hat{P}$, Birkhoff algorithm returns a convex combination of permutation matrices $P[j]$ such that $\hat{P} = \sum_j z[j]P[j]$. The classical version of Birkhoff algorithm is non-continuous in its inputs and it even holds for its extensions as the one proposed by Dufossé et al. (2018). Yet it can be smartly modified as in Ordered Birkhoff, described in Algorithm 24, to get a regular BvN decomposition defined as follows for any $\omega \in (0, 1)$:

$$\psi(P)(\omega) = P[j_\omega] \tag{4.4}$$

where $P = \sum_j z[j]P[j]$ is the decomposition returned by Ordered Birkhoff algorithm and j_ω verifies $\sum_{j \leq j_\omega} z[j] \leq \omega < \sum_{j \leq j_\omega + 1} z[j]$.

For a matrix P in the following, its support is defined as $\mathrm{supp}(P) = \{(i,j) \mid P_{i,j} \neq 0\}$.

Algorithm 24: ORDERED BIRKHOFF

input : $\hat{P} \in \mathfrak{B}_K$ (doubly stochastic matrix), $C \in \mathbb{R}^{K\times K}$ (cost matrix)

1 $j \leftarrow 1$
2 **while** $\hat{P} \neq \mathbf{0}$ **do**
3 $\quad C_{i,k} \leftarrow +\infty$ for all $(i,k) \notin \mathrm{supp}(\hat{P})$ `// remove edge` (i,k) `in induced graph`
4 $\quad P[j] \leftarrow \textsc{Hungarian}(C)$ `// matching with minimal cost w.r.t.` C
5 $\quad z[j] \leftarrow \min_{(i,k)\in\mathrm{supp}(P[j])} \hat{P}_{i,k}$
6 $\quad \hat{P} \leftarrow \hat{P} - z[j]P[j]$ and $j \leftarrow j+1$
7 **end**
8 **return** $(z[j], P[j])_j$

Obviously $\mathbb{E}_{\omega\sim\mathcal{U}(0,1)}[\psi(P)(\omega)] = P$ and permutations avoid collisions between queues. The difference with the usual Birkhoff algorithm happens at Line 4. Birkhoff algorithm usually computes any perfect matching in the graph induced by the support of $\hat{P}$ at the current iteration. This is often done with the Hopcroft-Karp algorithm, while it is here done with the Hungarian algorithm with respect to some cost matrix C. Although using the Hungarian algorithm slightly increases the computational complexity of this step (K^3 instead of $K^{2.5}$), it ensures to output the permutation matrices $P[j]$ according to a fixed order defined below.

Definition 4.4.6. *A cost matrix C induces an order $\prec_C$ on the permutation matrices defined, for any $P, P' \in \mathfrak{S}_K$ by*

$$P \prec_C P' \quad \textit{iff} \quad \textstyle\sum_{i,j} C_{i,j}P_{i,j} < \sum_{i,j} C_{i,j}P'_{i,j}.$$

This order might be non-total as different permutations can have the same cost. However, if C is drawn at random according to some continuous distribution, this order is total with probability 1. The order $\prec_C$ has to be the same for all queues and is thus determined beforehand for all queues.

Lemma 4.4.7. *Given matrices $C \in \mathbb{R}^{K\times K}$ and $P \in \mathfrak{B}_K$, ORDERED BIRKHOFF outputs a sequence $(z[j], P[j])_j$ of length at most $K^2 - K + 1$, such that*

$$P = \textstyle\sum_j z[j]P[j], \text{ where for all } j,\ z[j] > 0 \text{ and } P[j] \in \mathfrak{S}_K.$$

Moreover if the induced order $\prec_C$ is total, $z[j]$ is the j-th non-zero element of the sequence $(z_l(P))_{1\le l\le K!}$ defined by

$$z_j(P) = \min_{(i,k)\in\mathrm{supp}(P_j)} \left(P - \sum_{l=1}^{j-1} z_l(P)P_l\right)_{i,k} \tag{4.5}$$

where $(P_j)_{1\le j\le K!}$ is a $\prec_C$-increasing sequence of permutation matrices, i.e., $P_j \prec_C P_{j+1}$ for all j.

Theorem 4.4.7 is crucial to guarantee the regularity of ψ, given by Theorem 4.4.8.

Lemma 4.4.8. *Consider ψ defined as in Equation (4.4) with a cost matrix C inducing a total order $\prec_C$, then for any doubly stochastic matrices P, P'*

$$\int_0^1 \mathbb{1}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right) \mathrm{d}\omega \le 2^{2(K^2-K+1)}\|P - P'\|_\infty.$$

Theorem 4.4.8 indeed ensures that the probability of collision between two queues remains small when they have close estimates. Unfortunately, the regularity constant is exponential in K^2, which yields a similar dependency in the stability bound of Theorem 4.4.9. The existence of a BvN decomposition with polynomial regularity constants remains unknown, even without computational considerations. The design of a better BvN decomposition is left open for future work and would directly improve the stability bounds, using the general result given by Theorem 4.A.1 in Section 4.A.1. The number of accumulated packets yet remain reasonably small in the experiments of Section 4.5, suggesting that the bound given by Theorem 4.4.8 is not tight and might be improved in future work.

4.4.3 Stability guarantees

This section finally provides theoretical guarantees on the stability of the system when all queues follow ADEQUA. The success of ADEQUA relies on the accurate estimation of all problem parameters by the queues, given by Theorem 4.A.7 in Section 4.A.4. After some time τ, the queues have tight estimations of the problem parameters. Afterwards, they clear their packets with a margin of order Δ, thanks to Theorems 4.4.4 and 4.4.8. This finally ensures the stability of the system, as given by Theorem 4.4.9.

Theorem 4.4.9. *For any $\eta > 1$, the system where all queues follow ADEQUA, for any queue i and any $r \in \mathbb{N}$, there exists a constant C_r depending only on r such that*

$$\mathbb{E}[(Q_t^i)^r] \leq C_r K N \left(\frac{N^{\frac{5}{2}} K^{\frac{5}{2}} 2^{5(K^2-K+1)}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}} \Delta^5} \right)^r, \quad \textit{for all } t \in \mathbb{N}.$$

As a consequence, for any $\eta > 1$, this decentralized system is strongly stable.

Despite yielding an exponential dependency in K^2, this anytime bound leads to a first decentralized stability result when $\eta \in (1, \frac{e}{e-1})$, which closes the stability gap left by previous works. Moreover it can be seen in the proof that the asymptotic number of packets is much smaller. It actually converges, in expectation, to the number of packets the queues would accumulate if they were following a stable centralized strategy from the beginning. As already noted by Krishnasamy et al. (2016) for a single queue, the number of packets first increases during the learning phase and then decreases once the queues have tight enough estimations, until reaching the same state as in the perfect knowledge centralized case. This is empirically confirmed in Section 4.5.

4.5 Simulations

Figures 4.2 and 4.3 compare on toy examples the stability of queues, when either each of them follows the no-regret strategy EXP3.P.1, or each queue follows ADEQUA. For practical considerations, we choose the exploration probability $\varepsilon_t = (N+K)t^{-\frac{1}{4}}$ for ADEQUA, as the exploration is too slow with ε_t of order $t^{-\frac{1}{5}}$.

These figures illustrate the evolution of the average queue length on two different instances with $N = K = 4$. The code for the experiments is available at `gitlab.com/f_sen/queuing_systems`.

In the first instance shown in Figure 4.2, for all $i \in [N]$, $\lambda_i = (N+1)/N^2$. Moreover $\mu_1 = 1$ and for all $i \geq 2$, $\mu_i = (N-1)/N^2$. Here $\eta < 2$ and no-regret strategies are known to be unstable (Gaitonde and Tardos, 2020a). It is empirically confirmed as the number of packets in each queue diverges when they follow EXP3.P.1. Conversely, when the queues follow ADEQUA, after a learning phase, the queues reach equilibrium and all succeed in clearing their packets.

In the second instance shown in Figure 4.3, for all $i \in [N]$, $\lambda_i = 0.55 - 0.1 \cdot i$ and $\mu_i = 2.1\lambda_i$. Here $\eta > 2$ and both strategies are known to be stable, which is again empirically confirmed. However, ADeQuA requires more time to learn the different parameters, suggesting that individual no-regret strategies might be better on easy instances where $\eta > 2$.

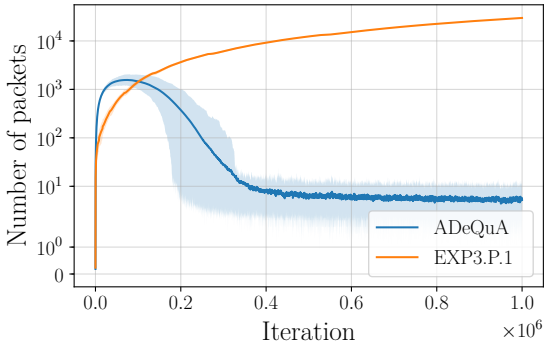


Figure 4.2: Hard instance, $\eta < 2$.

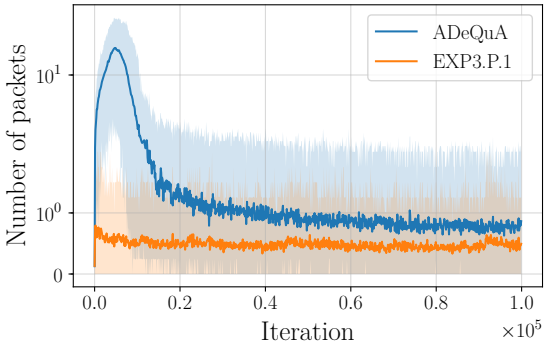


Figure 4.3: Easy instance, $\eta > 2$.

4.6 Conclusion

In this work, we showed that minimizing a more patient version of regret was not necessarily stable when the system's slack is smaller than two and we argued that some level of cooperation was then required between learning queues to reach stability. We presented the first decentralized learning algorithm guaranteeing stability of any queuing system with a slack larger than 1. Our stability bound presents an exponential dependency in the number of queues and remains open for improvement, e.g., through a better dominant mapping/BvN decomposition or a tighter analysis of ours. The proposed algorithm relies heavily on synchronisation between the queues, which all start the game simultaneously and share a common time discretisation. In particular, the shared randomness assumption merely results from this synchronisation when the players use a common random seed. Stability of asynchronous queues thus remains open for future work, for which Glauber dynamics approaches used in scheduling problems might be of interest (see e.g., Shah and Shin, 2012).

Remark : A subsequent work of Freund et al. (2022) managed to fix some of the issues mentioned above. They got rid of the exponential dependency in the number of queues and treat queues leaving and entering the game. To get this result, they designed a different decision rule for the servers and assumed a known lower bound on the slack of the system.

4.A Appendix

4.A.1 General version of Theorem 4.4.9

ADEQUA is described for specific choices of the functions ϕ and ψ given by Sections 4.4.1 and 4.4.2. It yet uses them in a black box fashion and different functions can be used, as long as they verify some key properties. This section provides a general version of Theorem 4.4.9, when the used dominant mapping and BvN decomposition respect the properties given by Assumptions 1 and 2.

Assumption 1 (regular dominant mapping)**.** *There are constants $c_1, c_2 > 0$ and a norm $\|\cdot\|$ on $\mathbb{R}^{K\times K}$, such that if $\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty \leq c_1\Delta$, then*

$$\|\phi(\hat{\lambda},\hat{\mu}) - \phi(\lambda,\mu)\| \leq L_\phi \cdot \|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty.$$

Moreover, $\hat{P} = \phi(\hat{\lambda},\hat{\mu})$ is doubly stochastic and for any $i \in [N]$,

$$\lambda_i \leq \textstyle\sum_{k=1}^K \hat{P}_{i,k}\mu_k - c_2\Delta.$$

Assumption 2 (regular BvN decomposition)**.** *Consider the same norm $\|\cdot\|$ as Assumption 1 on $\mathbb{R}^{K\times K}$. For any doubly stochastic matrices P, P'*

$$\int_0^1 \psi(P)(\omega)\mathrm{d}\omega = P$$
$$\text{and } \int_0^1 \mathbb{1}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right)\mathrm{d}\omega \leq L_\psi \cdot \|P-P'\|.$$

Theorems 4.4.4 and 4.4.8 show that the functions described in Sections 4.4.1 and 4.4.2 verify Assumptions 1 and 2 with the constants L_ϕ and L_ψ respectively of order $\frac{K}{\Delta}$ and 2^{2K^2} with the norm $\|\cdot\|_\infty$. Designing a dominant mapping and a BvN decomposition with smaller constants L_ϕ and L_ψ is left open for future work. It would lead to a direct improvement of the stability bound, as shown by Theorem 4.A.1.

Theorem 4.A.1. *Assume all queues follow ADEQUA, using an exploration probability $\varepsilon_t = xt^{-\alpha}$ with $x > 0, \alpha \in (0,1)$ and functions ϕ and ψ verifying Assumptions 1 and 2 with the constants L_ϕ, L_ψ. The system is then strongly stable and for any $r \in \mathbb{N}$, there exists a constant C_r such that:*

$$\mathbb{E}[(Q_t^i)^r] \leq C_r\left(\frac{x^{r/\alpha}}{\Delta^{r/\alpha}} + KN\left(\frac{N^2KL_\phi^2L_\psi^2}{\min(1,K\bar{\mu})\underline{\lambda}\Delta^2 x}\right)^{\frac{r}{1-\alpha}}\right), \quad \textit{for all } t \in \mathbb{N}$$

The proof directly follows the lines of the proof of Theorem 4.4.9 in Section 4.A.4 and is thus omitted here. From this version, it can be directly deduced that $\alpha = \frac{1}{5}$ gives the best dependency in Δ for ADEQUA. Moreover the best choice for x varies with r. When $r \to \infty$, it actually is $x = N^{\frac{2}{5}}K^{\frac{3}{5}}2^{\frac{4}{5}K^2}$ for ADEQUA. The choice $x = N + K$ is preferred for simplicity and still yields quite similar problem dependent bounds.

4.A.2 Efficient computation of ϕ

As mentioned in Section 4.4.1, computing exactly $\phi(\hat{\lambda},\hat{\mu})$ is not possible. Even efficiently approximating it is not obvious, as the function to minimize is neither smooth nor Lipschitz.

We here describe how an approximation of ϕ can be efficiently computed with guarantees on the approximation error.

First define the empirical estimate of the margin Δ:

$$\hat{\Delta} \coloneqq \min_{k \in [N]} \frac{1}{k} \left(\sum_{i=1}^{k} \hat{\mu}_{(i)} - \hat{\lambda}_{(i)} \right).$$

It can be computed in time $\mathcal{O}\left(N \log(N)\right)$ as it only requires to sort the vectors $\hat{\lambda}$ and $\hat{\mu}$. If $\hat{\Delta} \leq 0$, then the value of the optimization problem is $+\infty$ and any matrix can be returned. Assume in the following $\hat{\Delta} > 0$. Similarly to the proof of Theorem 4.4.4, it can be shown that the value of the optimization problem is smaller than $-\ln(\hat{\Delta}/\sqrt{e})$. Noting by $\mathfrak{B}_K$ the set of $K \times K$ doubly stochastic matrices, the optimization problem given by Equation (4.3) is then equivalent to

$$\arg\min_{P \in \mathcal{X}} g(P), \tag{4.6}$$

where

$$\mathcal{X} = \left\{ P \in \mathfrak{B}_K \mid \ \forall i \in [N], \textstyle\sum_{j=1}^{K} P_{i,j}\mu_j - \lambda_i \geq \frac{\hat{\Delta}}{\sqrt{e}} \right\},$$
$$\text{and } g(P) = \max_{i \in [N]} -\ln(\textstyle\sum_{j=1}^{K} P_{i,j}\mu_j - \lambda_i) + \frac{1}{2K}\|P\|_2^2.$$

Thanks to this new constraint set, the objective function of Equation (4.6) is now $(\frac{\sqrt{e}}{\Delta} + 1)$-Lipschitz. We can now use classical results for Lipschitz strongly convex minimization to obtain convergence rates of order $\frac{1}{t}$ for the projected gradient descent algorithm (see e.g., Bubeck, 2014, Theorem 3.9). These results yet assume that the projection on the constraint set can be exactly computed in a short time. This is not the case here, but it yet can be efficiently approximated using interior point methods (see e.g., Bubeck, 2014, Section 5.3), which has a linear convergence rate. If this approximation is good enough, similar convergence guarantees than with exact projection can be shown similarly to the original proof.

Algorithm 25 then describes how to quickly estimate $\phi(\hat{\lambda}, \hat{\mu})$, where $\hat{\Pi}_{\mathcal{X}}$ returns an approximation of the orthogonal projection on the set $\mathcal{X}$ and ∂g is a sub-gradient of g. It uses an averaged value of the different iterates, as the last iterate does not have good convergence guarantees.

Algorithm 25: Compute ϕ

input : function g, constraint set $\mathcal{X}$, $P^0 \in \mathcal{X}$

$P, \hat{P} \leftarrow P^0$

for $t = 1, \ldots, n$ **do**

  $P \leftarrow \hat{\Pi}_{\mathcal{X}} \left(P - \frac{2N}{(t+1)} \partial g(P) \right)$ `// approximated projection`

  $\hat{P} \leftarrow \frac{t}{t+2}\hat{P} + \frac{2}{t+2}P$

end

return $\hat{P}$

In practice, the approximation can even be computed faster by initializing P^0 in Algorithm 25 with the solution of the previous round $t - 1$.

4.A.3 Unstable No-Policy regret system example

Algorithm 26: Unstable No-policy regret system example

input : w_k, N, α, $\lambda = (1/N, \dots, 1/N)$, $\mu = (2(N-d)/N^2, \dots, 2(N-d)/N^2)$

for $k = 1, \dots, \infty$ **do**
  for $t = 1, \dots, \lceil \alpha w_k \rceil$ **do**
    Queues $2i$ and $2i+1$ play server $2i + t \pmod N$ `// stage 1`
  end
  for $t = \lceil \alpha w_k \rceil + 1, \dots, w_k$ **do**
    Queue i plays server $i + t \pmod N$ `// stage 2`
  end
end

Lemma 4.A.2. *Consider the system where the queues play according to the policy described in Algorithm 26 over successive windows of length $w_k = k^2$. If $\alpha > 1 - \frac{d}{N-d}$, the system is not stable.*

Proof. Note that the system is equivalent to a system where each queue or pair of queue would always pick the same server. For simplicity, the analysis deals with that equivalent system. Also, wlog, we analyse the subsystem with the two first queues and the two first servers. Let $\left\{B_t^i\right\}_{i\in[n],t\geq 1}$ be the independent random variables indicating the arrival of a packet on queue i at time t, $\left\{S_t^i\right\}_{i\in[n],t\geq 1}$ be the indicators that server j would clear a packet at iteration ℓ if one were sent to it. For each queue $i \in [N]$ and $t \geq 0$, we have by Chernoff bound

$$\Pr\left(\left|\sum_{t=1}^{\ell} B_t^i - \lambda_i \ell\right| \geq \sqrt{\ell \ln(\ell)}\right) \leq \frac{2}{\ell^2}.$$

The same holds for each queue, thus the probability that this event happens for queue 1 or queue 2 is at most, $\frac{4}{\ell^2}$. As it is summable in ℓ, The Borel-Cantelli Lemma implies that, for large enough ℓ, almost surely, for any $i \in [2]$:

$$\sum_{\ell=1}^{\ell} B_t^i = \lambda_i \ell \pm \tilde{\mathcal{O}}\left(\sqrt{\ell}\right). \tag{4.7}$$

Let $W_k = \sum_{i=1}^{k} w_i$. Note that $W_k = \Theta\left(k^3\right) = \Theta\left(w_k^{3/2}\right)$. Again by Chernoff bound and Borel-Cantelli, for large enough k, almost surely, for any $i \in \{1, 2\}$:

$$\sum_{t=W_{k-1}}^{W_{k-1}+\lceil \alpha w_k \rceil} S_t^i = \mu_i \alpha w_k \pm \tilde{\mathcal{O}}\left(\sqrt{w_k}\right), \quad \sum_{t=W_{k-1}+\lceil \alpha w_k \rceil}^{W_k} S_t^i = \mu_i (1-\alpha) w_k \pm \tilde{\mathcal{O}}\left(\sqrt{w_k}\right). \tag{4.8}$$

Thus, for any large enough k, the total number of packet in both queues at time W_k is

almost surely lower bounded as:

$$Q^1_{W_k} + Q^2_{W_k} \geq \sum_{t=1}^{W_k}(B^1_t + B^2_t) - \sum_{t=1}^{W_k} S^1_t - \sum_{l=1}^{k}\left(\sum_{t=W_{l-1}+\lceil \alpha w_l\rceil}^{W_l} S^2_t\right) \tag{4.9}$$

$$\geq \left[\frac{2}{N} - \frac{2(N-d)}{N^2} - (1-\alpha)\frac{2(N-d)}{N^2}\right] W_k - \tilde{\mathcal{O}}\left(W_k^{2/3}\right) \tag{4.10}$$

$$\geq \frac{2\left[\alpha(N-d) - (N-2d)\right]}{N^2} W_k - \tilde{\mathcal{O}}\left(W_k^{2/3}\right) \tag{4.11}$$

which is a diverging function of W_k. Note that this result also holds for any two pair of queues $(2i-1, 2i)$, with $i \in [N/2]$.

□

Lemma 4.A.3. *Consider the same setting as in Theorem 4.A.2. For any $i \in [N]$, for any large enough k, queue i clears*

$$\left(\frac{N-d}{N^2} + (1-\alpha)\frac{N-d}{N^2} + o(1)\right) w_k$$

packets almost surely over window w_k.

Proof. The proof starts by showing that for any large enough t, all the queues hold roughly the same number of packets. Then, as they receive roughly the same number of packets over a time window and we can compute the approximate total number of packets cleared, the results follows.

Let T^t_i be the age of the oldest packet in queue i at time t. By Chernoff bound,

$$\mathrm{P}(|T^t_i - NQ^t_i| \geq N\sqrt{t\ln(t)}) \leq \frac{2}{t^2}.$$

Thus, using the Borel-Cantelli lemma, for any queue i, almost surely, for any large enough k and any $t \in [W_{k-1}+1, W_k]$,

$$|T^t_i - NQ^t_i| \leq N\sqrt{t\ln(t)} = \tilde{\mathcal{O}}(w_k^{3/4}). \tag{4.12}$$

For any $(i,j) \in [N]^2$, define

$$\phi^+_t(i,j) := \left(Q^i_t - Q^j_t - 2N\sqrt{t\ln(t)}\right)_+ \text{ and } \phi^-_t(i,j) := \left(Q^i_t - Q^j_t + 2N\sqrt{t\ln(t)}\right)_- .$$

Let C^i_t be the indicator function that queue i clears a packet at iteration t. Note that for any large enough t, $\phi^+_t(i,j)$ is a supermartingale. Indeed,

$$\begin{aligned}\mathbb{E}[\phi^+_{t+1}(i,j)|\phi^+_{1:t}(i,j)] \leq& \phi^+_t(i,j) + \mathbb{E}[B^i_t - B^j_t|\phi^+_{1:t}(i,j)] - \mathbb{E}[C^i_t - C^j_t|\phi^+_{1:t}(i,j)] \\ \leq& \phi^+_t(i,j).\end{aligned}$$

The second inequality comes from Equation (4.12), that implies that for any large enough t, if $\phi^+_t(i,j)$ is strictly positive, queue i holds the oldest packet and thus clears one with higher probability than queue j. By the same arguments, $\phi^-_t(i,j)$ a submartingale. Also, $|\phi^+_{t+1}(i,j) - \phi^+_t(i,j)| \leq 2(N+1)$ for any $t \geq 0$, and the same holds for $\phi^-_t(i,j)$. Let τ_{ij} be the stopping time of the smallest iteration after which Equation (4.12) always holds for queues i

and j. By Azuma-Hoeffding's inequality,

$$\Pr\left(\phi_\ell^+(i,j) - \phi_{\tau_{ij}}^+(i,j) \geq 3(N+1)\sqrt{\ell\ln(\ell)}\right) \leq \frac{2}{\ell^2}$$

and

$$\Pr\left(\phi_\ell^-(i,j) - \phi_{\tau_{ij}}^+(i,j) \leq -3(N+1)\sqrt{\ell\ln(\ell)}\right) \leq \frac{2}{\ell^2}.$$

This, together with a union bound and Borel-Cantelli's Lemma implies that almost surely, for any large enough t, for any $(i,j) \in [N]^2$

$$Q_t^i - Q_t^j = \tilde{\mathcal{O}}\left(\sqrt{t}\right). \tag{4.13}$$

This with Equation (4.9) implies that for any large enough k, for any $i \in [N]$, almost surely,

$$Q_{W_k}^i \geq \frac{[\alpha(N-d) - (N-2d)]}{N^2} W_k - \tilde{\mathcal{O}}\left(W_k^{2/3}\right).$$

This means that for any large enough k, every queue holds at least one packet over the whole window w_k. This and Equation (4.8) is already enough to show that for any time-window w_k, for any large enough k, the total number of packets cleared by any couple of queue $(2i-1, 2i)$, $i \in [N/2]$ is:

$$2\left(\frac{N-d}{N^2} + (1-\alpha)\frac{N-d}{N^2}\right) w_k + \tilde{\mathcal{O}}\left(\sqrt{w_k}\right).$$

During time window w_k, according to Equation (4.7), both every queue receives $\alpha w_k/N + \tilde{\mathcal{O}}\left(w_k^{3/4}\right)$ packets almost surely for any large enough k. Equation (4.13) implies that for any $i \in [N/2]$

$$Q_{W_k}^{2i-1} - Q_{W_k}^{2i} = \tilde{\mathcal{O}}\left(w_k^{3/4}\right) \text{ and } Q_{W_{k-1}}^{2i-1} - Q_{W_{k-1}}^{2i} = \tilde{\mathcal{O}}\left(w_k^{3/4}\right).$$

Therefore, over each time-window w_k, for any large enough k, each queue clears

$$\left(\frac{N-d}{N^2} + (1-\alpha)\frac{N-d}{N^2} + o(1)\right) w_k$$

packets almost surely. □

Lemma 4.A.4. *Consider again the system where the queues play according to the policy described in Algorithm 26 over successive windows of length $w_k = k^2$. If $\alpha < 1 - \frac{1}{N-1}$, the queues have $o(w_k)$ policy regret in all but finitely many of the windows.*

Wlog, let us consider that queue 1 deviates, and plays at every iteration a server chosen from the probability distribution $\mathbf{p} = (p_1, ..., p_N)$, with p_i the probability to play server i. To upper bound the number of packets queue 1 clears over each time window, we can assume it always has priority over queue 2 and ignore it in the analysis.

Before proving Theorem 4.A.4, we prove the following technical one.

Lemma 4.A.5. *Consider that a queue deviates from the strategy considered in Theorem 4.A.4 and plays at every iteration a server chosen from the probability distribution $\boldsymbol{p} = (p_1, ..., p_N)$, with p_i the probability to play server i. For any large enough k, almost surely, the number of*

packets the deviating queue clears of the first stage of the k^{th} window is

$$\left(\frac{1}{2}+\frac{1}{N}\right)\frac{2(N-d)}{N^2}\alpha w_k+\tilde{\mathcal{O}}\left(w_k^{3/4}\right).$$

Proof. The proof starts by showing that for any large enough t, every non-deviating queue holds approximately the same number of packets.

Fist note that for any large enough t, Equation (4.12) still holds surely for any queue i. For any $(i,j)\in\{3,\dots,N\}^2$, define

$$\phi_\ell^+(i,j):=\left(Q^i_{\lceil \ell N\rceil}-Q^j_{\lceil \ell N\rceil}-4N\sqrt{\lceil \ell N\rceil \ln(\lceil \ell N\rceil)}\right)_+$$

and

$$\phi_\ell^-(i,j):=\left(Q^i_{\lceil \ell N\rceil}-Q^j_{\lceil \ell N\rceil}+4N\sqrt{\lceil \ell N\rceil \ln(\lceil \ell N\rceil)}\right)_-.$$

For any interval $[\lceil \ell N\rceil,\lceil(\ell+1)N\rceil]$ where Equation (4.12) holds for queues $1,i$ and j, if $\phi_\ell^+(i,j)$ is strictly positive, then

$$\mathbb{E}\left[\sum_{t=\lceil \ell N\rceil}^{\lceil(\ell+1)N\rceil} C_t^j-C_t^i\middle|\phi^+_{1:t}(i,j)\right]\le 0.$$

Indeed, if $\phi_\ell^+(i,j)$ is strictly positive and Equation (4.12) holds, queue i holds the oldest packets throughout the interval. Also, queue i and queue j collide with queue 1 the same number of times over the interval in expectation, and if at one iteration of the interval, queue 1 holds an older packet than queue i, it holds an older packet than queue j over the whole interval. Thus $\phi_\ell^+(i,j)$ is a submartingale. By the same arguments, $\phi_\ell^+(i,j)$ is a supermartingale. Also, $|\phi^+_{\ell+1}(i,j)-\phi^+_\ell(i,j)|\le 4(N+1)^2$ and the same holds for $\phi_\ell^-(i,j)$. Finishing with the same arguments used to prove Equation (4.13), almost surely, for any $(i,j)\in\{3,\dots,N\}^2$,

$$Q_t^i-Q_t^j=\tilde{\mathcal{O}}\left(\sqrt{t}\right). \tag{4.14}$$

We now show that for any large enough t, queue 1 can not hold many more packets than the non-deviating queues. Define

$$\phi_t^+:=\left(Q_t^1-\max_{i\ge 3}Q_t^i-2N\sqrt{t\ln(t)}\right)_+.$$

Once again, at every iteration where ϕ_t^+ is strictly positive and Equation (4.12) holds, queue 1 holds the oldest packet and thus has priority on whichever server it chooses. This implies that for any large enough t, ϕ_t^+ is a supermartingale. It also holds that for any $t\ge 0$, $|\phi^+_{t+1}-\phi_t^+|\le 2(N+1)$. Thus, with the same arguments used to prove Equation (4.13), almost surely,

$$\left(Q_t^1-\max_{i\ge 3}Q_t^i\right)_+=\tilde{\mathcal{O}}\left(\sqrt{t}\right). \tag{4.15}$$

With that at hand, we prove that for any large enough k, queue 1 does not get priority often over the other queues during the first stage of the k^{th} window. For any $i\in\{2,\dots,N/2\}$, pose:

$$\psi_\ell^i=\frac{1}{2}\left(Q^{2i-1}_{\lceil \ell N\rceil}+Q^{2i}_{\lceil \ell N\rceil}\right)-Q^1_{\lceil \ell N\rceil}-\frac{2(N-d)}{N^3}(\lceil \ell N\rceil-W_{k-1})$$

For any ℓ s.t. $\{\lceil \ell N \rceil; \lceil (\ell+1)N - 1 \rceil\}$ is included in the first phase of a window, we have

$$\begin{aligned}\sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[C_t^1 \middle| \psi_{1:\ell}^+(i,j) \right] &\geq \sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[S_i^t \mathbb{1}_{\{\text{queue 1 and only queue 1 picks server } i\}} \middle| \psi_{1:\ell}^+(i,j) \right] \\ &\geq \frac{N-d}{N} + \frac{2(N-d)}{N^2}\end{aligned}$$

as well as

$$\begin{aligned}\sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[\frac{1}{2}\left(C_t^{2i} + C_t^{2i-1} \right) \middle| \psi_{1:\ell}^+(i,j) \right] &\leq \sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[\frac{1}{2} S_{i+t \pmod N}^t \middle| \psi_{1:\ell}^+(i,j) \right] \\ &\leq \frac{N-d}{N}.\end{aligned}$$

Those two inequalities imply:

$$\begin{aligned}\mathbb{E}[\psi_{\ell+1}^i | \psi_{1:\ell}^+(i,j)] =& \psi_\ell^+(i,j) + \sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[\frac{1}{2}(B_t^{2i} - B_t^{2i-1}) - B_t^1 \middle| \psi_{1:\ell}^+(i,j) \right] \\ &- \sum_{t=\lceil \ell N \rceil}^{\lceil (\ell+1)N \rceil - 1} \mathbb{E}\left[\frac{1}{2}(C_t^{2i} - C_t^{2i-1}) - C_t^1 \middle| \psi_{1:\ell}^+(i,j) \right] - \frac{2(N-d)}{N^2} \\ \geq& \psi_\ell^+(i,j).\end{aligned}$$

Thus, for any ℓ s.t. $\{\lceil \ell N \rceil; \lceil (\ell+1)N - 1 \rceil\}$ is included in the first phase of a window, ψ_ℓ^i is a submartingale. Moreover, for any $\ell \geq 0$, $|\psi_{\ell+1}^i - \psi_\ell^i| \leq 3N$. Thus, by Azuma-Hoeffding's inequality, for any ℓ s.t.$\{\lceil \ell N \rceil; \lceil (\ell+1)N - 1 \rceil\} \subset [W_{k-1}, W_{k-1} + \alpha w_k]$,

$$\Pr\left(\psi_\ell^i - \psi_{W_k}^i \leq -6N\sqrt{\ell N \ln(\ell N)} \right) \leq \frac{1}{(\ell N)^2}.$$

Borel-Cantelli's lemma implies, that for any large enough ℓ s.t.$\{\lceil \ell N \rceil; \lceil (\ell+1)N - 1 \rceil\} \subset [W_{k-1}, W_{k-1} + \alpha w_k]$, almost surely:

$$\psi_\ell^i \geq \psi_{W_k}^i - 6N\sqrt{\ell N \ln(\ell N)}.$$

This and Equation (4.15) applied at $t = W_k$, imply that for any large enough k, for any $t \in [W_{k-1}, W_{k-1} + \alpha w_k]$,

$$\begin{aligned}\frac{1}{2}\left(Q_t^{2i-1} + Q_t^{2i} \right) \geq& Q_{\lceil \ell N \rceil}^1 + \frac{2(N-d)}{N^3}(t - W_{k-1}) + \psi_{W_k}^i - \tilde{O}(\sqrt{t}) \\ \geq& Q_{\lceil \ell N \rceil}^1 + \frac{2(N-d)}{N^3}(t - W_{k-1}) - \tilde{O}(w_k^{3/4}).\end{aligned}$$

This and Equation (4.12) imply that during the first stage of the time window, queue 1 holds younger packets than any other queues $i \geq 3$ after at most $\tilde{\mathcal{O}}(w_k^{3/4})$ iterations.

By Chernoff bound and the Borel-Cantelli lemma again, for any large enough k, almost surely, the number of packets queue 1 clears during the first stage of the k^{th} window on servers

where it does not collide with other queues is:

$$\sum_{t=W_{k-1}+1}^{W_{k-1}+\alpha w_k}\sum_{i=1}^{N} S_i^t \mathbb{1}_{\{\text{queue 1 and only queue 1 picks server } i\}} = (\frac{1}{2}+\frac{1}{N})\frac{2(N-d)}{N^2}\alpha w_k + \tilde{\mathcal{O}}\left(\sqrt{w_k}\right).$$

Since we have shown that for any large enough k, almost surely, queue 1 does not have priority over the other queues after at most $\tilde{\mathcal{O}}(w_k^{3/4})$ iterations, for any large enough k, almost surely, the number of packets queue 1 clears of the first stage of the k^{th} window is

$$\left(\frac{1}{2}+\frac{1}{N}\right)\frac{2(N-d)}{N^2}\alpha w_k + \tilde{\mathcal{O}}\left(w_k^{3/4}\right).$$

□

We are now ready to prove Theorem 4.A.4.

Proof. By Chernoff bound and the Borel-Cantelli lemma, almost surely for any large enough k, the number of packets queue 1 clears during the second stage of the window on servers where it does not collide with other queues is:

$$\sum_{t=W_{k-1}+\alpha w_k}^{W_k-1}\sum_{i=1}^{N} S_i^t \mathbb{1}_{\{\text{queue 1 and only queue 1 picks server } i\}} = \frac{4(N-d)}{N^3}(1-\alpha)w_k + \tilde{\mathcal{O}}\left(\sqrt{w_k}\right). \quad (4.16)$$

Suppose that during the second stage of the window, queue 1 never gets priority over another queue. In that case, according to Equation (4.16) and Theorem 4.A.5, for any large enough k, almost surely, the total number of packets cleared by queue 1 during the time window is

$$\left(\frac{\alpha}{2}+\frac{2-\alpha}{N}\right)\frac{2(N-d)}{N^2}w_k + \tilde{\mathcal{O}}(w_k^{3/4}).$$

For any large enough k, if $\alpha \leq 1-\frac{1}{N-1}$ this is smaller than the number of packets queue 1 would have cleared had it not deviated, according to Theorem 4.A.3.

On the other hand, suppose that queue gets priority over some other queue i at some iteration τ of the second stage of the window. In that case, at that iteration, queue 1 holds the oldest packets, which, according to Equation (4.12), implies

$$Q_1^\tau > Q_i^\tau - \tilde{\mathcal{O}}(w_k^{3/4})$$

During the second stage of the window, for any $i \geq 3$, $\gamma_t^i := \left(Q_i^t - Q_1^t - 2N\sqrt{t\ln(t)}\right)_+$ is a supermartingale with bounded increments for any t where Equation (4.12) holds for queues 1 and i. Indeed, in that case, if γ_t^i is strictly positive, queue i holds an older packet than queue 1, and thus, whether they collide or not, it has a higher probability to clear a packet than queue 1. Thus, by Azuma-Hoeffding and the Borel-cantelli lemma again, for any large enough k, almost surely,

$$Q_i^{W_k} - Q_1^{W_k} \leq Q_i^\tau - Q_1^\tau + \tilde{\mathcal{O}}(w_k^{3/4}).$$

Thus it holds that $Q_1^{W_k} \geq Q_i^{W_k} - \tilde{\mathcal{O}}(w_k^{3/4})$ for any $i \geq 2$. This and Equation (4.15) imply that all the queues clear approximately the same number of packets over those time windows for any

large enough k almost surely. Thus queue 1 clears

$$\left[(2-\alpha)(N-2)+(\alpha+\frac{4-2\alpha}{N})\right]\frac{(N-d)}{(N-1)N^2}w_k+\tilde{\mathcal{O}}\left(w_k^{3/4}\right)$$

packets almost surely, which again is smaller than the number of packets it would have cleared had it not deviated.

Thus, the deviating queue clears almost surely less packets by time window than it would have had it not deviated on all but finitely many of the time windows, which implies that it has no policy regret on all but finitely many of the time windows. □

4.A.4 Proofs of Section 4.4

Proof of Lemma 4.4.4

We want to show that if $\|(\hat{\lambda}-\lambda,\hat{\mu}-\mu)\|_\infty \leq c_1\Delta$, then

$$\|\phi(\hat{\lambda},\hat{\mu})-\phi(\lambda,\mu)\|_2 \leq \frac{c_2K}{\Delta}\|(\hat{\lambda}-\lambda,\hat{\mu}-\mu)\|_\infty, \tag{4.17}$$

with the constants c_1, c_2 given in Theorem 4.4.4.

Recall that ϕ is defined as

$$\phi(\lambda,\mu)=\operatorname*{arg\,min}_{P\in\mathfrak{B}_K} f(P,\lambda,\mu),$$

where $\mathfrak{B}_K$ is the set of $K\times K$ doubly stochastic matrices and f is defined as:

$$f(P,\lambda,\mu)\coloneqq\max_{i\in[N]}-\ln(\sum_{j=1}^{K}P_{i,j}\mu_k-\lambda_i)+\frac{1}{2K}\|P\|_2^2$$

Let P^* and $\hat{P}^*$ be the minimizers of f with the respective parameters (λ,μ) and $(\hat{\lambda},\hat{\mu})$. They are uniquely defined as f is $\frac{1}{K}$ strongly convex.

As the property of Theorem 4.4.4 is symmetric, we can assume without loss of generality that $f(P^*,\lambda,\mu)\geq f(\hat{P}^*,\hat{\lambda},\hat{\mu})$.

Given the definition of Δ, we have the bound

$$-\ln(\Delta)+\frac{1}{2}\geq f(P^*,\lambda,\mu)\geq -\ln(\Delta).$$

The lower bound holds because the term in the ln is at most Δ for at least one i. For the upper bound, some matrix P ensures that the term in the ln is at least Δ for all i and $\|P\|_2^2\leq K$.

Similarly for $\hat{P}^*$, it follows:

$$-\ln((1-2c_1)\Delta)+\frac{1}{2}\geq f(\hat{P}^*,\hat{\lambda},\hat{\mu})\geq -\ln((1+2c_1)\Delta).$$

As a consequence, it holds for any $i \in [N]$:

$$\begin{aligned} -\ln\left(\sum_{j=1}^{K} \hat{P}^*_{i,j}\hat{\mu}_j - \hat{\lambda}_i\right) &\leq f(\hat{P}^*, \hat{\lambda}, \hat{\mu}) \\ &\leq -\ln((1-2c_1)\Delta/\sqrt{e}) \\ \sum_{j=1}^{K} \hat{P}^*_{i,j}\hat{\mu}_j - \hat{\lambda}_i &\geq (1-2c_1)\Delta/\sqrt{e}. \end{aligned}$$

Note that for any $i \in [N]$,

$$\sum_{j=1}^{K} \hat{P}^*_{i,j}\hat{\mu}_j - \hat{\lambda}_i \leq \sum_{j=1}^{K} \hat{P}^*_{i,j}\mu_j - \lambda_i + 2\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty.$$

It then yields the second point of Theorem 4.4.4:

$$\sum_{j=1}^{K} \hat{P}^*_{i,j}\mu_j - \lambda_i \geq \left((1-2c_1)/\sqrt{e} - 2c_1\right)\Delta$$

Moreover, it follows

$$\begin{aligned} -\ln\left(\sum_{j=1}^{K} \hat{P}^*_{i,j}\hat{\mu}_j - \hat{\lambda}_i\right) &\geq -\ln\left(\sum_{j=1}^{K} \hat{P}^*_{i,j}\mu_j - \lambda_i\right) - \ln\left(1 + \frac{2\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty}{\sum_{j=1}^{K} \hat{P}^*_{i,j}\mu_j - \lambda_i}\right) \\ &\geq -\ln\left(\sum_{j=1}^{K} \hat{P}^*_{i,j}\mu_j - \lambda_i\right) - \frac{2\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty}{\left((1-2c_1)/\sqrt{e} - 2c_1\right)\Delta} \end{aligned}$$

Recall that for a $\frac{1}{K}$-strongly convex function g of global minimum x^* and any x:

$$\|x - x^*\|_2 \leq 2K(g(x) - g(x^*))$$

As a consequence, it follows:

$$\begin{aligned} f(\hat{P}^*, \hat{\lambda}, \hat{\mu}) &\geq f(\hat{P}^*, \lambda, \mu) - \frac{2\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty}{\left((1-2c_1)/\sqrt{e} - 2c_1\right)\Delta} \\ &\geq f(P^*, \lambda, \mu) - \frac{2\|(\hat{\lambda}-\lambda, \hat{\mu}-\mu)\|_\infty}{\left((1-2c_1)/\sqrt{e} - 2c_1\right)\Delta} + \frac{1}{2K}\|P^* - \hat{P}^*\|_2. \end{aligned}$$

Equation (4.17) then follows.

Proof of Lemma 4.4.7

The coefficient $C_{i,j}$ is replaced by $+\infty$ as soon as the whole weight $P_{i,j}$ is exhausted. Thanks to this, the HUNGARIAN algorithm does return a perfect matching with respect to the bipartite graph with edges (i, j) where there remains some weight for $P_{i,j}$. Because of this, it can be shown

following the usual proof of Birkhoff algorithm (Birkhoff, 1946) that the sequence $(z[j], P[j])$ is indeed of length at most $K^2 - K + 1$ and is a valid decomposition of P.

Now assume that $\prec_C$ is a total order. At each iteration j of HUNGARIAN algorithm, denote $\tilde{P}^j = P - \sum_{s=1}^{j-1} z[s]P[s]$ the remaining weights to attribute.

Let l_j be such that $P[j] = P_{l_j}$ for any iteration j of HUNGARIAN algorithm.

It can now be shown by induction that

$$\tilde{P}^j = P - \sum_{l=1}^{l_j} z_l(P) P_l.$$

where $z_l(P)$ are defined by Equation (4.5). Indeed, by definition

$$\begin{aligned}\tilde{P}^{j+1} &= \tilde{P}^j - z[j+1]P[j+1] \\ &= \tilde{P}^j - z[j+1]P_{l_{j+1}}\end{aligned}$$

The HUNGARIAN algorithm returns the minimal cost matching with respect to the modified cost matrix C where the coefficients i, k such that $\tilde{P}^j_{i,k} = 0$ are replaced by $+\infty$. Thanks to this, $P_{l_{j+1}}$ is the minimal cost permutation matrix P_l (for $\prec_C$) such that $\tilde{P}^j_{i,k} > 0$ for all $(i,k) \in \mathrm{supp}(P_l)$.

This means that for any $l < l_{j+1}$

$$\min_{(i,k)\in \mathrm{supp}(P_l)} (\tilde{P}^j)_{i,k} = 0.$$

Using the induction hypothesis, this implies that $z_l(P) = 0$ for any $l_j < l < l_{j+1}$. And finally, this also implies that $z_{l_{j+1}}(P) = z[j+1]$.

This finally concludes the proof as $\tilde{P}^j = 0$ after the last iteration.

Proof of Lemma 4.4.8

For z and z' the respective decompositions of P and P' defined in Theorem 4.4.7, then

$$\int_0^1 \mathbb{1}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right) \mathrm{d}\omega = \mathbb{P}_{\omega\sim U(0,1)}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right).$$

In the following, note $A = \psi(P)$ and $A' = \psi(P')$. It follows for P_n defined as in Theorem 4.4.7

$$\begin{aligned}\int_0^1 \mathbb{1}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right) \mathrm{d}\omega &= \sum_{n=1}^{K!} \mathbb{P}(A = P_n \text{ and } A' \neq P_n) \\ &= \frac{1}{2}\sum_{n=1}^{K!} \mathbb{P}(A = P_n \text{ and } A' \neq P_n) + \frac{1}{2}\sum_{n=1}^{K!} \mathbb{P}(A' = P_n \text{ and } A \neq P_n) \\ &= \frac{1}{2}\sum_{n=1}^{K!} \mathrm{vol}\left(\left[\sum_{j=1}^{n-1} z_j(P), \sum_{j=1}^{n} z_j(P)\right] \ominus \left[\sum_{j=1}^{n-1} z_j(P'), \sum_{j=1}^{n} z_j(P')\right]\right),\end{aligned}$$

where vol denotes the volume of a set and $A \ominus B = (A \setminus B) \cup (B \setminus A)$ is the symmetric difference of A and B. The last equality comes from the expression of ψ with respect to the coefficients $z_j(P)$, thanks to Theorem 4.4.7.

It is easy to show that

$$\mathrm{vol}\left([a,b]\ominus[c,d]\right) \leq \left(|c-a|+|d-b|\right)\mathbb{1}\left(b>a \text{ or } c>d\right).$$

The previous equality then leads to

$$\begin{aligned}\int_0^1 \mathbb{1}\left(\psi(P)(\omega)\neq\psi(P')(\omega)\right)\mathrm{d}\omega &\leq \frac{1}{2}\sum_{n=1}^{K!}\left|\sum_{j=1}^{n-1} z_j(P)-z_j(P')\right|\mathbb{1}\left(z_n(P)+z_n(P')>0\right)\\ &\quad+\frac{1}{2}\sum_{n=1}^{K!}\left|\sum_{j=1}^{n} z_j(P)-z_j(P')\right|\mathbb{1}\left(z_n(P)+z_n(P')>0\right)\\ &\leq \sum_{n=1}^{K!}\left|\sum_{j=1}^{n} z_j(P)-z_j(P')\right|\mathbb{1}\left(z_n(P)+z_n(P')>0\right). \quad (4.18)\end{aligned}$$

The last inequality holds because $\sum_{j=1}^{k} z_j(P)-z_j(P')$ is counted twice when $z_k(P)+z_k(P')$ is positive: when $n=k$ and for the next n such that the elements are counted in the sum.

Thanks to Theorem 4.4.7, only $2(K^2-K+1)$ elements $z_j(P)$ and $z_j(P')$ are non-zero. Let k_n be the index of the n-th non-zero element of $(z_s(P)+z_s(P'))_{1\leq s\leq K!}$. Note that $z_s(P')$ can be non-zero while $z_s(P)$ is zero (or conversely). Let also

$$\begin{aligned}(i_{k_n},j_{k_n}) &\in \operatorname*{arg\,min}_{(i,j)\in\mathrm{supp}(P_{k_n})} P_{i,j}-\sum_{l<k_n} z_l(P)\mathbb{1}\left((i,j)\in\mathrm{supp}(P_{k_n})\right),\\ (i'_{k_n},j'_{k_n}) &\in \operatorname*{arg\,min}_{(i,j)\in\mathrm{supp}(P_{k_n})} P'_{i,j}-\sum_{l<k_n} z_l(P')\mathbb{1}\left((i,j)\in\mathrm{supp}(P_{k_n})\right).\end{aligned}$$

It then comes, thanks to Theorem 4.4.7

$$\begin{aligned}z_{k_n}(P)-z_{k_n}(P') &\leq P_{i'_{k_n},j'_{k_n}}-P'_{i'_{k_n},j'k_n}-\sum_{l<k_n}(z_l(P)-z_l(P'))\mathbb{1}\left((i'_{k_n},j'_{k_n})\in\mathrm{supp}(P_{k_n})\right)\\ &\leq P_{i'_{k_n},j'_{k_n}}-P'_{i'_{k_n},j'_{k_n}}-\sum_{l<n}(z_{k_l}(P)-z_{k_l}(P'))\mathbb{1}\left((i'_{k_n},j'_{k_n})\in\mathrm{supp}(P_{k_n})\right)\end{aligned}$$

The second inequality holds, because for $l'\notin\{k_l \mid l<2K^2\}$, the term in the sum is zero by definition of the sequence k_l.

A similar inequality holds for $z_{k_n}(P')-z_{k_n}(P)$, which leads to

$$|z_{k_n}(P)-z_{k_n}(P')|\leq\|P-P'\|_\infty+\sum_{l<n}|z_{k_l}(P)-z_{k_l}(P')|.$$

By induction, it thus holds

$$|z_{k_n}(P)-z_{k_n}(P')|\leq 2^{n-1}\|P-P'\|_\infty.$$

We finally conclude using Equation (4.18)

$$\begin{aligned}
\int_0^1 \mathbb{1}\left(\psi(P)(\omega) \neq \psi(P')(\omega)\right) \mathrm{d}\omega &\leq \sum_{n=1}^{K!} \left|\sum_{j=1}^{n} z_j(P) - z_j(P')\right| \mathbb{1}\left(z_n(P) + z_n(P') > 0\right) \\
&\leq \sum_{n=1}^{2(K^2-K+1)-1} \left|\sum_{j=1}^{k_n} z_j(P) - z_j(P')\right| \\
&\leq \sum_{n=1}^{2(K^2-K+1)-1} \left|\sum_{l=1}^{n} z_{k_l}(P) - z_{k_l}(P')\right| \\
&\leq \sum_{n=1}^{2(K^2-K+1)-1} \sum_{j=1}^{n} 2^{j-1} \|P - P'\|_\infty \\
&\leq 2^{2(K^2-K+1)} \|P - P'\|_\infty.
\end{aligned}$$

In the fourth inequality, the $2(K^2 - K + 1)$-th term of the sum is ignored. It is indeed 0 as z and z' both sum to 1.

Proof of Theorem 4.4.9

First recall below a useful version of Chernoff bound.

Lemma 4.A.6. *For any independent variables $X_1, \ldots, X_n$ in $[0,1]$ and $\delta \in (0,1)$,*

$$\mathbb{P}\left(\sum_{i=1}^{n} X_i \leq (1-\delta) \sum_{i=1}^{n} \mathbb{E}[X_i]\right) \leq e^{-\frac{\delta^2 \sum_{i=1}^{n} \mathbb{E}[X_i]}{2}}.$$

We now prove the following concentration lemma.

Lemma 4.A.7. *For any time $t \geq (N+K)^5$ and $\varepsilon \in (0, \frac{1}{4})$,*

$$\begin{aligned}
\mathbb{P}\left(|\hat{\mu}_k^i(t) - \mu_k| \geq \varepsilon\right) &\leq 3 \exp\left(-\lambda_i \left(t^{\frac{4}{5}} - 1\right) \varepsilon^2\right) \\
\mathbb{P}\left(|\hat{\lambda}_j^i(t) - \lambda_j| \geq \varepsilon\right) &\leq 6 \exp\left(-\lambda_i K \bar{\mu} \frac{t^{\frac{4}{5}} - 1}{145} \varepsilon^2\right).
\end{aligned}$$

Proof.
Concentration for $\hat{\mu}$. Consider agent i in the following and denote by $N_k(t)$ the number of *exploratory pulls* of this agent on server k at time t. By definition, the probability to proceed to an exploratory pull on the server k at round t is at least $\lambda_i \min(t^{-\frac{1}{5}}, \frac{1}{N+K})$. The term λ_i here appears as a pull is guaranteed if a packet appeared at the current time step. Yet the number of exploratory pulls might be much larger in practice as queues should accumulate a large number of uncleared packets at the beginning.

For $t \geq (N+K)^5$, it holds:

$$\begin{aligned}\sum_{n=1}^{t} \min(n^{-\frac{1}{5}}, \frac{1}{N+K}) &= \sum_{n=1}^{(N+K)^5} \frac{1}{N+K} + \sum_{n=(N+K)^5+1}^{t} n^{-\frac{1}{5}} \\ &\geq (N+K)^4 + \int_{(N+K)^5}^{t} x^{-\frac{1}{5}} \mathrm{d}x - 1 \\ &\geq \frac{1}{4}\left(5t^{\frac{4}{5}} - (N+K)^4 - 4\right) \\ &\geq t^{\frac{4}{5}} - 1.\end{aligned}$$

Theorem 4.A.6 then gives for $N_k(t)$:

$$\begin{aligned}\mathbb{P}\left(N_k(t) \leq (1-\delta)\mathbb{E}[N_k(t)]\right) &\leq \exp\left(-\frac{\delta^2 \mathbb{E}[N_k(t)]}{2}\right) \\ \mathbb{P}\left(N_k(t) \leq (1-\delta)\lambda_i\left(t^{\frac{4}{5}} - 1\right)\right) &\leq \exp\left(-\frac{\lambda_i \delta^2\left(t^{\frac{4}{5}} - 1\right)}{2}\right).\end{aligned}$$

Which leads for $\delta = \frac{1}{2}$ to

$$\mathbb{P}\left(N_k(t) \leq \frac{\lambda_i}{2}\left(t^{\frac{4}{5}} - 1\right)\right) \leq \exp\left(-\lambda_i \frac{t^{\frac{4}{5}} - 1}{8}\right). \tag{4.19}$$

The number of exploratory pulls and the observations on the server k are independent. Thanks to this, Hoeffding's inequality can be directly used as follows

$$\mathbb{P}\left(|\hat{\mu}_k^i(t) - \mu_k| \geq \varepsilon \mid N_k(t)\right) \leq 2\exp\left(-2N_k(t)\varepsilon^2\right).$$

Using Equation (4.19) now gives the first concentration inequality for $\varepsilon \leq \frac{1}{4} \leq \frac{1}{2\sqrt{2}}$:

$$\begin{aligned}\mathbb{P}\left(|\hat{\mu}_k^i(t) - \mu_k| \geq \varepsilon\right) &\leq 2\exp\left(-\lambda_i\left(t^{\frac{4}{5}} - 1\right)\varepsilon^2\right) + \exp\left(-\lambda_i \frac{t^{\frac{4}{5}} - 1}{8}\right) \\ &\leq 3\exp\left(-\lambda_i\left(t^{\frac{4}{5}} - 1\right)\varepsilon^2\right).\end{aligned}$$

Concentration for $\hat{\lambda}$. Consider agent i in the following. First show a concentration inequality for $\tilde{\mu}$. Denote by $N(t)$ the total number of exploratory pulls on servers proceeded by player i at round t, i.e., $N(t) = \sum_{k=1}^{K} N_k(t)$. Similarly to Equation (4.19), it can be shown that

$$\mathbb{P}\left(N(t) \leq \lambda_i K \frac{t^{\frac{4}{5}} - 1}{2}\right) \leq \exp\left(-\lambda_i K \frac{t^{\frac{4}{5}} - 1}{8}\right).$$

Theorem 4.A.6 then gives for $\delta \in (0,1)$:

$$\begin{aligned}\mathbb{P}\left(|\tilde{\mu}-\bar{\mu}| \geq \delta\bar{\mu}\right) &\leq 2\exp\left(-\lambda_i K\delta^2\bar{\mu}\frac{t^{\frac{4}{5}}-1}{8}\right)+\exp\left(-\lambda_i K\frac{t^{\frac{4}{5}}-1}{8}\right)\\ &\leq 3\exp\left(-\lambda_i K\delta^2\bar{\mu}\frac{t^{\frac{4}{5}}-1}{8}\right).\end{aligned}$$

Note that $|\tilde{\mu}-\bar{\mu}| \leq \delta\bar{\mu}$ implies $|\frac{1}{\bar{\mu}}-\frac{1}{\tilde{\mu}}| \leq \frac{\delta}{(1-\delta)\bar{\mu}}$. So this gives the following inequality:

$$\mathbb{P}\left(\left|\frac{1}{\bar{\mu}}-\frac{1}{\tilde{\mu}}\right| \geq \frac{\delta}{(1-\delta)\bar{\mu}}\right) \leq 3\exp\left(-\lambda_i K\delta^2\bar{\mu}\frac{t^{\frac{4}{5}}-1}{8}\right). \tag{4.20}$$

A concentration bound on $\hat{S}_j^i$ can be shown similarly for any $\delta \in (0,1)$

$$\mathbb{P}\left(\left|\hat{S}_j^i(t)-(1-\frac{\lambda_j}{2})\bar{\mu}\right| \geq \delta\bar{\mu}\right) \leq 3\exp\left(-\lambda_i K\delta^2\bar{\mu}\frac{t^{\frac{4}{5}}-1}{8}\right). \tag{4.21}$$

Now recall that the estimate of λ_j is defined by $\hat{\lambda}_j = 2-\frac{2\hat{S}_j^i}{\tilde{\mu}}$. We then have the following identity:

$$\hat{\lambda}_j-\lambda_j = 2(\frac{1}{\tilde{\mu}}-\frac{1}{\bar{\mu}})\hat{S}_j^i+\frac{2}{\bar{\mu}}\left((1-\frac{\lambda_j}{2})\bar{\mu}-\hat{S}_j^i\right).$$

Since $\hat{S}_j \in [0,1]$, it yields for $\varepsilon \leq \frac{1}{4}$ and $x \in (0,1)$:

$$\begin{aligned}\mathbb{P}\left(\left|\hat{\lambda}_j^i(t)-\lambda_j\right| \geq \varepsilon\right) &\leq \mathbb{P}\left(\left|2(\frac{1}{\tilde{\mu}}-\frac{1}{\bar{\mu}})\hat{S}_j^i\right| \geq x\varepsilon \text{ or } \left|\frac{2}{\bar{\mu}}\left((1-\frac{\lambda_j}{2})\bar{\mu}-\hat{S}_j\right)\right| \geq (1-x)\varepsilon\right)\\ &\leq \mathbb{P}\left(\left|\frac{1}{\tilde{\mu}}-\frac{1}{\bar{\mu}}\right| \geq \frac{x\varepsilon}{2\hat{S}_j^i} \mid \hat{S}_j^i \leq (1+\frac{1-x}{8})\bar{\mu}\right)+\mathbb{P}\left(\left|\hat{S}_j^i(t)-(1-\frac{\lambda_j}{2})\bar{\mu}\right| \geq \frac{(1-x)\bar{\mu}\varepsilon}{2}\right)\\ &\leq \mathbb{P}\left(\left|\frac{1}{\tilde{\mu}}-\frac{1}{\bar{\mu}}\right| \geq \frac{\delta}{(1-\delta)\bar{\mu}} \text{ for } \delta = \frac{4x\varepsilon}{9}\right)+\mathbb{P}\left(\left|\hat{S}_j^i(t)-(1-\frac{\lambda_j}{2})\bar{\mu}\right| \geq \frac{(1-x)\bar{\mu}\varepsilon}{2}\right)\end{aligned}$$

Taking $x = \frac{9}{17}$ leads to $\frac{4x}{9} = \frac{1-x}{2}$ and thus, using Equations (4.20) and (4.21):

$$\begin{aligned}\mathbb{P}\left(\left|\hat{\lambda}_j^i(t)-\lambda_j\right| \geq \varepsilon\right) &\leq 6\exp\left(-\lambda_i K\left(\frac{8}{17}\right)^2\bar{\mu}\frac{t^{\frac{4}{5}}-1}{32}\varepsilon^2\right)\\ &\leq 6\exp\left(-\lambda_i K\bar{\mu}\frac{t^{\frac{4}{5}}-1}{145}\varepsilon^2\right)\end{aligned}$$

□

In the following, let $c_1 = 0.1$ and $c_2 = \frac{4}{(1-2c_1)/\sqrt{e}-2c_1} \approx 14$. For a problem instance, let the good event $\mathcal{E}_t$ at time t be defined as

$$\mathcal{E}_t \coloneqq \left\{ \|(\hat{\lambda}^i - \lambda, \hat{\mu}^i - \mu)\|_\infty \leq \frac{0.1\Delta^2}{2c_2 2^{2K^2} KN},\ \forall i \in [N] \right\}.$$

As Δ is smaller than 1, the right hand term in the definition of $\mathcal{E}_t$ is smaller than $c_1\Delta$. Thanks to Theorems 4.4.4 and 4.4.8, $\mathcal{E}_t$ then guarantees that any player will collide with another player with probability at most 0.1Δ, i.e., , $\forall i \in [N]$,

$$\mathbb{P}_{\omega\sim\mathcal{U}(0,1)}\left(\exists j \in [N], \hat{A}_t^i(\omega) \neq \hat{A}_t^j(\omega) \mid \mathcal{E}_t\right) \leq 0.1\Delta.$$

Moreover, thanks to Theorem 4.4.4, under $\mathcal{E}_t$,

$$\lambda_i \leq \sum_{k=1}^{K} \hat{P}_{i,k}\mu_k - \left(\frac{1-2c_1}{\sqrt{e}} - 2c_1\right)\Delta.$$

These two last inequalities lead to the following lemma.

Lemma 4.A.8. *For $t \geq \frac{2^5K^5}{0.08^5\Delta^5}$, denote by $\mathcal{H}_t$ the history of observations up to round t. Then*

$$\mathbb{E}\left[S_t^i \mid \mathcal{E}_t, \mathcal{H}_t\right] \geq \lambda_i + 0.1\Delta.$$

Proof. This is a direct consequence of the following decomposition:

$$\begin{aligned}
\mathbb{E}\left[S_t^i \mid \mathcal{E}_t, \mathcal{H}_t\right] &\geq \overbrace{(1-(N+K)t^{-\frac{1}{5}})}^{\text{proba to exploit}} \left(\overbrace{\hat{P}_{i,k}\mu_k}^{\text{proba to clear}} - \overbrace{\mathbb{P}\left(\exists j \in [N], \hat{A}_t^i(\omega) \neq \hat{A}_t^j(\omega) \exists \mid \mathcal{E}_t\right)}^{\text{collision proba}} \right) \\
&\geq (1-(N+K)t^{-\frac{1}{5}})(\lambda_i + \left(\frac{1-2c_1}{\sqrt{e}} - 2c_1\right)\Delta - 0.1\Delta) \\
&\geq (1-(N+K)t^{-\frac{1}{5}})(\lambda_i + 0.18\Delta).
\end{aligned}$$

The last inequality is given by $c_1 = 0.1$ and it leads to

$$\mathbb{E}\left[S_t^i \mid \mathcal{E}_t, \mathcal{H}_t\right] \geq \lambda_i + 0.18\Delta - (N+K)t^{-\frac{1}{5}}.$$

For $t \geq \frac{2^5K^5}{0.08^5\Delta^5}$, the last term is smaller than 0.08Δ, giving Theorem 4.A.8. □

Define the stopping time

$$\tau \coloneqq \min\left\{ t \geq \frac{2^5K^5}{0.08^5\Delta^5} \mid \forall t \geq s, \mathcal{E}_s \text{ holds} \right\}. \tag{4.22}$$

Lemma 4.A.9. *For any integer $r \geq 1$,*

$$\mathbb{E}[\tau^r] = \mathcal{O}\left(KN \left(\frac{N^{\frac{5}{2}}K^{\frac{5}{2}}2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}}\Delta^5} \right)^r \right),$$

where the $\mathcal{O}$ notation hides constant factors that only depend on r.

Proof. Define for this proof $t_0 = \left\lceil \frac{2^5K^5}{0.08^5\Delta^5} \right\rceil$. By definition, if $\mathcal{E}_t$ does not hold for $t > t_0$, then

$\tau \geq t$. As a consequence, for any $t > t_0$ and thanks to Theorem 4.A.7:

$$\begin{aligned}\mathbb{P}(\tau \geq t) &\leq \mathbb{P}(\neg \mathcal{E}_t) \\ &\leq (3eKN + 6eN^2) \exp\left(-ct^{\frac{4}{5}}\right),\end{aligned}$$

where $c = c_0 \frac{\min(1, K\bar{\mu})\underline{\lambda}\Delta^4}{N^2 K^2 2^{4K^2}}$ for some universal constant $c_0 \leq 1$.

We can now bound the moments of τ:

$$\begin{aligned}\mathbb{E}[\tau^r] &= r \int_0^\infty t^{r-1} \mathbb{P}(\tau \geq t)\, \mathrm{d}t \\ &\leq t_0^r + (3eKN + 6eN^2) r \int_0^\infty t^{r-1} e^{-ct^{\frac{4}{5}}}\, \mathrm{d}t.\end{aligned}$$

Using the change of variable $u = ct^{\frac{4}{5}}$, it can be shown that

$$\int_0^\infty t^{r-1} e^{-ct^{\frac{4}{5}}}\, \mathrm{d}t = \frac{5}{4} c^{-\frac{5r}{4}} \Gamma\left(\frac{5r}{4}\right),$$

where Γ denotes the Gamma function. It finally allows to conclude:

$$\begin{aligned}\mathbb{E}[\tau^r] &= \mathcal{O}\left(\frac{K^{5r}}{\Delta^{5r}} + KNc^{-\frac{5r}{4}}\right) \\ &= \mathcal{O}\left(KN\left(\frac{N^{\frac{5}{2}} K^{\frac{5}{2}} 2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}} \Delta^5}\right)^r\right).\end{aligned}$$

□

Let X_t be a random walk biased towards 0 with the following transition probabilities:

$$\begin{aligned}\mathbb{P}(X_{t+1} = X_t + 1) = p,\ \mathbb{P}(X_{t+1} = X_t - 1 | X_t > 0) = q, \\ \mathbb{P}(X_{t+1} = X_t | X_t > 0) = 1 - p - q,\ \mathbb{P}(X_{t+1} = X_t | X_t = 0) = 1 - p,\end{aligned} \tag{4.23}$$

and $X_0 = 0$.

Lemma 4.A.10. *The non-asymptotic moments of the random walk defined by Equation* (4.23) *are bounded. For any $t > 0, r > 0$:*

$$\mathbb{E}\left[(X_t)^r\right] \leq \frac{r!}{\left(\ln(q/p)\right)^r}.$$

Proof: Let π be the stationary distribution of the random walk. It verifies the following system of equations:

$$\begin{cases}\pi(z) = p\pi(z-1) + q\pi(z+1) + (1-p-q)\pi(z),\ \forall z > 0 \\ \pi(0) = (1-p)\pi(0) + q\pi(1) \\ \sum \pi(z) = 1\end{cases}$$

which gives:

$$\pi(z) = \frac{q-p}{q}\left(\frac{p}{q}\right)^z.$$

Equivalently, $\pi(z) = \mathbb{P}(\lfloor Y \rfloor = z)$ with Y an exponential random variable of parameter $\ln(q/p)$. This gives:

$$\mathbb{E}_{X\sim\pi}\left[(X)^r\right] \leq \frac{r!}{\left(\ln\left(q/p\right)\right)^r}.$$

Let $\tilde{X}_t$ be the random walk with the same transition probabilities as X_t and $\tilde{X}_0 \sim \pi$. For any $t > 0$, $\tilde{X}_t \sim \pi$. Moreover, for any $t > 0$, $\tilde{X}_t$ stochasticaly dominates X_t, which terminates the proof. □

Proof of Theorem 4.4.9. For τ the stopping time defined by Equation (4.22), Theorem 4.A.9 bounds its moments as follows

$$\mathbb{E}[\tau^r] = \mathcal{O}\left(KN\left(\frac{N^{\frac{5}{2}}K^{\frac{5}{2}}2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}}\Delta^5}\right)^r\right).$$

Let

$$p_i = \lambda_i(1 - \lambda_i - 0.1\Delta) \text{ and } q_i = (\lambda_i + 0.1\Delta)(1 - \lambda_i).$$

Let X_t^i be the random walk biased towards 0 with parameters p_i and q_i, with $X_t^i = 0$ for any $t \leq 0$. According to Lemma 4.A.8, past time τ, Q_t^i is stochastically dominated by the random process $\tau + X_{t-\tau}^i$. Thus, for any $t > 0$, for any $r > 0$

$$\begin{aligned}
\mathbb{E}[\left(Q_i^t\right)^r] &\leq \max(1, 2^{r-1})\left(\mathbb{E}[\tau^r] + \mathbb{E}[(X_{t-\tau}^i)^r]\right) \\
&= \mathcal{O}\left(KN\left(\frac{N^{\frac{5}{2}}K^{\frac{5}{2}}2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}}\Delta^5}\right)^r + \frac{1}{\ln(q_i/p_i)^r}\right) \\
&= \mathcal{O}\left(KN\left(\frac{N^{\frac{5}{2}}K^{\frac{5}{2}}2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}}\Delta^5}\right)^r + \Delta^{-r}\right) \\
&= \mathcal{O}\left(KN\left(\frac{N^{\frac{5}{2}}K^{\frac{5}{2}}2^{5K^2}}{\left(\min(1, K\bar{\mu})\underline{\lambda}\right)^{\frac{5}{4}}\Delta^5}\right)^r\right).
\end{aligned}$$

□

Part II

Online Algorithms

Chapter 5

Online Matching in Sparse Random Graphs: Non-Asymptotic Performances of Greedy Algorithm

Motivated by sequential budgeted allocation problems, we investigate online matching problems where connections between vertices are not i.i.d., but they have fixed degree distributions – the so-called configuration model. We estimate the competitive ratio of the simplest algorithm, GREEDY, by approximating some relevant stochastic discrete processes by their continuous counterparts, that are solutions of an explicit system of partial differential equations. This technique gives precise bounds on the estimation errors, with arbitrarily high probability as the problem size increases. In particular, it allows the formal comparison between different configuration models. We also prove that, quite surprisingly, GREEDYcan have better performance guarantees than RANKING, another celebrated algorithm for online matching that usually outperforms the former.

Contents

5.1 Introduction

Finding matchings in bipartite graphs $(\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, where $\mathcal{E} \subset \mathcal{U} \times \mathcal{V}$ is a set of edges, is a long-standing problem with different motivations and approaches (Bordenave et al., 2013; Godsil, 1981; Lovász and Plummer, 2009b; Zdeborová and Mézard, 2006). If $\mathcal{U}$ is seen as a set of resources and $\mathcal{V}$ as demands, the objective is to allocate as many resources to demands (an allocation - or a matching - between u and v is admissible if $(u, v) \in \mathcal{E}$) with the constraint that a resource is allocated to only one demand and vice-versa.

Motivated particularly by practical applications to Internet advertising, the *online* variant of this problem is receiving increasing attention (we refer to the excellent survey (Mehta, 2012) for more applications, specific settings, results and techniques). In this case, the set of vertices $\mathcal{U}$ is present at the beginning and the graph unveils sequentially: vertices $v \in \mathcal{V}$ are observed sequentially, one after the other, along with the edges they belong to. An online algorithm must decide, right after observing v_k and its associated set of edges $\mathcal{E}_k := \{(u, v_k) \in \mathcal{E}\}$ to match it to some other vertex $u \in \mathcal{U}$, at the conditions that $(u, v_k) \in \mathcal{E}_k$ and $u \in \mathcal{U}$ has not been matched yet. The performance of an online algorithm is evaluated by its *competitive ratio*, which is the ratio between the size of the matching it has created and the highest possible matching in hindsight (Feldman et al., 2009).

This theoretical setting is particularly well suited for online advertising: $\mathcal{U}$ is the set of campaigns/ads that an advertiser can run and users $v_1, v_2, \dots, v_T$ arrive sequentially (Manshadi et al., 2012; Mehta, 2012). Some of them are eligible for a large subset of campaigns, others are not (usually based on their attributes/features, such as the geographic localization, the browsing history, or any other relevant information). The objective of an advertiser (in this over-simplified model) is to maximize the number of displayed ads. In practice, campaigns/ads are not displayed only once but have a maximal budget of impressions (say, a specific ad can be displayed only 10.000 times each day). A possible trick consists of duplicating the vertices of $\mathcal{U}$ as many times as the budget. However, this results in strong and undesirable correlations between vertices. It is, therefore, more appropriate to consider a bipartite graph with *capacities* and admissible matchings as subsets of edges such that each vertex belongs to several different edges, but not more than their associated capacities $\omega \in \mathbb{N}$ (a vertex $v \in \mathcal{V}$ is matched once while $u \in \mathcal{U}$ can be matched ω_u times).

This online matching problem with capacities has been quite extensively studied. It is known that GREEDY, which matches all incoming vertices to any available neighbor has a competitive ratio of $1/2$ in the worst case, albeit it achieves $1 - 1/e$ as soon as the incoming vertices arrive in Random Order (Goel and Mehta, 2008). The worst-case optimal algorithm is the celebrated RANKING, which achieves $1 - 1/e$ on any instance (Birnbaum and Mathieu, 2008; Devanur et al., 2013; Karp et al., 1990), and also has better guarantees in the Random Order setting (Mahdian and Yan, 2011).

Beyond the adversarial setting, the following stochastic setting has been considered: there exist a finite set of L "base" vertices $v^{(1)}, \dots, v^{(L)}$ associated to base edge-sets $\mathcal{E}^{(1)}, \dots, \mathcal{E}^{(L)}$. When a vertex v_k arrives, its type $\theta_k \in \{1, \dots, L\}$ is drawn iid from some distribution (either known beforehand or not) and then its edge set is set as $\mathcal{E}_k = \mathcal{E}^{(\theta_k)}$. In the context where the distribution is known, algorithms with much better competitive ratios than GREEDY or RANKING were designed (Brubach et al., 2019; Jaillet and Lu, 2014; Manshadi et al., 2012), specifically with a competitive ratio of $1 - 2/e^2$ when the expected number of arrival of each type are integral and 0.706 without this assumption. Notably, those competitive ratios still hold with Poisson arrival rates rather than a fixed number of arrivals.

On a side note, a vast line of work considers online matching in weighted graphs (Devanur et al., 2012; Goel and Mehta, 2008; Mehta, 2012), which is outside the scope of this chapter.

However, it is still worth noting that the unweighted graph is a weighted graph with all weights equal.

This model of the stochastic setting is quite interesting but rather strong: it lacks flexibility and cannot be used to represent some challenging instances (for example when the degrees of each vertex $\mathcal{U}$ increase linearly with the number of vertices in $\mathcal{V}$, or when the set $\mathcal{U}$ of campaigns must be fixed so that the model is well specified, etc...). Another tentative is to consider Erdős-Rényi graphs assuming that each possible edge is present in $\mathcal{U} \times \mathcal{V}$ with some fixed probability and independently of the other edges (see (Mastin and Jaillet, 2013)). The most interesting and challenging setting corresponds to the so-called *sparse* regime where each vertex of $\mathcal{U}$ has an expected degree independent of the size n of $\mathcal{V}$, which amounts to take a probability of connection equal to c/n. Interestingly enough, even the analysis of the simplest GREEDY algorithm is quite challenging and already insightful in those models (Arnosti, 2019; Borodin et al., 2018; Dyer et al., 1993; Mastin and Jaillet, 2013). Unfortunately, although this Erdős-Rényi model is compatible with growing sets $\mathcal{U}$ and $\mathcal{V}$, it also turns out to be quite restrictive. The main reason is that the approximate Poisson degree distribution of the vertices has light-tail and does not allow for the appearance of the so-called *scale-free property* satisfied by many real-world networks (Barabási et al., 2000; Van Der Hofstad, 2016).

We, therefore, consider a more appropriate random graphs generation process called *configuration model*, introduced by (Bender and Canfield, 1978) and (Bollobás, 1980). The optimal matching of this model has been computed in (Bordenave et al., 2013). The configuration model is particularly well suited to handle different situations such as the following one. Assume that campaigns can either be "intensive" (with many eligible users) or "selective/light" (few eligible users), with an empirical proportion of, say, 20%/80%. Then whether an advertiser handles 100 campaigns at the same time or 10.000, it will always have roughly this proportion of intensive vs. light campaigns. Similarly, some users are more valuable than others, and are thus eligible for more campaigns than the others; the proportion of each type being independent of the total population size. The configuration model accommodates these observations by basically drawing iid degrees for vertices $\mathcal{U}$ and $\mathcal{V}$ (accordingly to some different unknown distributions for $\mathcal{U}$ and $\mathcal{V}$) and then by finding a graph such that those degrees distribution are satisfied (up to negligible errors); as a consequence, the graphs generated are *sparse*, in the sense that the number of edges grows linearly with the number of vertices.

Additionally, the configuration model is a well-suited random graph model which mimics a number of properties of real-world complex networks, while being analytically tractable. For instance, choosing power-law distributions for the degrees allows to obtain the so-called scale-free property (often observed in practice, as highlighted for the web by Faloutsos et al. (1999)). The configuration model also displays the so called "small-world phenomenon" (observed for instance in the graph of Facebook by Backstrom et al. (2012)) as its diameter is of logarithmic order.

Main contribution

We investigate the performances (in terms of expected competitive ratio) of the GREEDY matching algorithm in configuration models and we provide explicit quantitative results using stochastic approximation techniques (Wormald, 1995); we prove that the increasing size of the random matching created is arbitrarily close to the solution of some explicit ODE. Solving the latter then gives in turn the solution to the original problem.

The remaining of the chapter is organized as follows. Section 5.2 describes precisely the problem and Theorem 5.2.1 is our first main result: it describes the performances of GREEDY in the capacity-less problem. The proof of Theorem 5.2.1 is delayed to Section 5.4.4, but the main ideas and intuitions are provided in Section 5.3. The online matching with capacities problem

is treated in Section 5.4.1.

5.2 Online Matching Problems; Models and main result

Consider a bipartite graph with capacities $G = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \omega)$ where $\mathcal{U} = \{1, \dots, N\}$ and $\mathcal{V} = \{1, \dots, T\}$ are two finite set of vertices, $\mathcal{E} \subset \Big\{(u, v),\, u \in \mathcal{U},\, v \in \mathcal{V}\Big\}$ is the set of edges and $\omega : \mathcal{U} \to \mathbb{N}_*$ is a capacity function. A matching M on G is a subset of edges $e \in \mathcal{E}$ such that any vertex $v \in \mathcal{V}$ is the endpoint of at most one edge $e \in M$ and any vertex $u \in \mathcal{U}$ is the endpoint of at most ω_u edges in M. We will denote by $\mathcal{M}$ the set of matchings on G; the optimal matching $M^* \in \mathcal{M}$ is the one (or any one) with the highest cardinality, denoted by $|M^*|$.

The batched matching problem consists in finding any optimal matching M^* given a graph with capacities G; the online variant might be a bit more challenging, as the matching is constructed sequentially. Formally, the set of vertices $\mathcal{U}$ and their capacities ω are known from the start, and vertices $v \in \mathcal{V}$ arrive sequentially (with the edges they belong to) and $M_0 = \emptyset$. At stage $t \in \mathbb{N}$ – assuming a matching M_{t-1} has been constructed –, a decision maker observes a new vertex[1] v_t and its associated set of edges $\{(u, v_t); u \in \mathcal{E}\}$. If possible, one of these edges (u_t, v_t) is added to M_{t-1}, with the constraint that $M_t = M_{t-1} \cup \{(u_t, v_t)\}$ is still a matching. The objective is to maximize the size of the constructed matching M_T. The classical way to evaluate the performances of an algorithm is the *competitive ratio*, defined as $|M_T|/|M^*| \in [0, 1]$ (the higher the better).

5.2.1 Structured online matching via Configuration Model

As mentioned before, the online matching problem can be quite difficult without additional structure. We will therefore assume that the vertex degrees in $\mathcal{U}$ and $\mathcal{V}$ have (at least asymptotically in N and T) some given subGaussian[2] distributions $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$, of respective expectation $\mu_{\mathcal{U}}$ and $\mu_{\mathcal{V}}$ and respective proxy-variance $\sigma^2_{\mathcal{U}}$ and $\sigma^2_{\mathcal{V}}$. Those numbers are related in the sense that we assume[3] that $T = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}} N \in \mathbb{N}$. Given those degree distributions, the graphs we consider are random draws from a bipartite configuration model described below; for the sake of clarity, we first consider the capacity-less case (when $\omega_u = 1$ for all $u \in \mathcal{U}$).

Given $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ and $N, T \geq 1$, let $\mathbb{N}\ d_1^{\mathcal{U}}, \dots, d_N^{\mathcal{U}} \in \mathbb{N} \overset{\text{i.i.d.}}{\sim} \pi_{\mathcal{U}}$ and $d_1^{\mathcal{V}}, \dots, d_T^{\mathcal{V}} \in \mathbb{N} \overset{\text{i.i.d.}}{\sim} \pi_{\mathcal{V}}$ be independent random variables; intuitively, those numbers are respectively the number of half-edges attached to vertex in $\mathcal{U}$ and $\mathcal{V}$. Consider also two extra random variables

$$d_{T+1}^{\mathcal{V}} = \max\Big\{ \sum_{i=1}^{N} d_i^{\mathcal{U}} - \sum_{j=1}^{T} d_j^{\mathcal{V}}\ , 0 \Big\} \quad \text{and} \quad d_{N+1}^{\mathcal{U}} = \max\Big\{ \sum_{j=1}^{T} d_j^{\mathcal{V}} - \sum_{i=1}^{N} d_i^{\mathcal{U}}\ , 0 \Big\}$$

so that equality between total degrees holds, i.e., $\sum_{i=1}^{N+1} d_i^{\mathcal{U}} = \sum_{j=1}^{T+1} d_j^{\mathcal{V}}$. Finally, a random (capacity-less) bipartite graph denoted by $\mathbf{CM}(\mathbf{d}^U, \mathbf{d}^V)$ is constructed with a uniform pairing of half-edges of $\mathcal{U} \cup \{N+1\}$ with half-edges of $\mathcal{V} \cup \{T+1\}$ and removing vertices $T+1$ and $N+1$ and their associated edges. These two artificially added vertices are just here to define a pairing between half-edges. Notice that, by the law of large numbers and since $T = (\mu_{\mathcal{U}}/\mu_{\mathcal{V}})N$,

[1] Although the order of arrival is irrelevant to the models we studied, it could have an impact on other models.

[2] X is subGaussian with proxy-variance σ^2 if for any $s \in \mathbb{R}, \mathbb{E}[\exp(sX)] \leq \exp\left(\frac{\sigma^2 s^2}{2}\right)$. Actually, we only need that $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ have some finite moment of order $\gamma > 2$.

[3] In the general case, consider $T = \lfloor N\mu_{\mathcal{U}}/\mu_{\mathcal{V}} \rfloor$. The proof is identical, up to a negligible $1/N$ error term

$d^{\mathcal{V}}_{T+1} = o(N)$ and $d^{\mathcal{U}}_{N+1} = o(N)$ almost surely[4].

The bipartite configuration model $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}})$ is then the random graph obtained by a uniform matching between the half-edges of $\mathcal{U}$ and the half-edges of $\mathcal{V}$, where the random sequences $\mathbf{d}^{\mathcal{U}} = (d^{\mathcal{U}}_i)_i$ and $\mathbf{d}^{\mathcal{V}} = (d^{\mathcal{V}}_j)_j$ are defined as above.

5.2.2 Competitive ratio of greedy algorithm. Main result

The first question to investigate in this structured setting is the computation of the (expected) competitive ratio of the simple algorithm GREEDY. It constructs a matching by sequentially adding any admissible edge uniformly at random. Describing it and stating our results require the following additional notations: for any $e = (u, v) \in E$, $u(e) = u$ (resp. $v(e) = v$) is the extremity of e in $\mathcal{U}$ (resp. $\mathcal{V}$); the generating series of $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ are denoted by $\phi_{\mathcal{U}}$ and $\phi_{\mathcal{V}}$ and are defined as

$$\phi_{\mathcal{U}}(s) := \sum_{k\geq 0} \pi_{\mathcal{U}}(k) s^k \qquad \text{and} \qquad \phi_{\mathcal{V}}(s) := \sum_{k\geq 0} \pi_{\mathcal{V}}(k) s^k.$$

Our first main theorem, stated below, identifies the asymptotic size of the matching generated by GREEDY on the bipartite configuration model we have just defined. As the batched problem (i.e., computing the size of the optimal matching M^*) is well understood (Bordenave et al., 2013), this quantity is sufficient to derive competitive ratios. Again, for the sake of presentation, we first assume that all capacities are fixed, equal to one; the general case is presented in Appendix 5.4.1.

Theorem 5.2.1. ***(Performances of* greedy *in the capacity-less case)***

Given $N \geq 1$ and $T = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}} N$, let M_T be the matching built by GREEDY *on $\mathbf{CM}(\mathbf{d}^U, \mathbf{d}^V)$ then the following convergence in probability holds:*

$$\frac{|\mathrm{M}_T|}{N} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \phi_{\mathcal{U}}(1 - G(1)).$$

where G is the unique solution of the following ordinary differential equation:

$$G'(s) = \frac{1 - \phi_{\mathcal{V}}\left(1 - \frac{1}{\mu_{\mathcal{U}}}\phi'_{\mathcal{U}}\left(1 - G(s)\right)\right)}{\frac{\mu_{\mathcal{V}}}{\mu_{\mathcal{U}}}\phi'_{\mathcal{U}}(1 - G(s))}; \quad G(0) = 0. \tag{5.1}$$

Moreover, for any $s \in [0, 1]$, if $M_T(s)$ is the matching obtained by GREEDY *after seeing a proportion s of vertices of $\mathcal{V}$, then*

$$\frac{|\mathrm{M}_T(s)|}{N} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \phi_{\mathcal{U}}(1 - G(s)). \tag{5.2}$$

Convergence rates are explicit; with probability exponentially large, at least $1 - \zeta N \exp(-\xi N^{c/2})$,

$$\sup_{s\in[0,1]} \left| \frac{|\mathrm{M}_T(s)|}{N} - \Big(1 - \phi_{\mathcal{U}}(1 - G(s))\Big) \right| \leq \kappa N^{-c},$$

where ζ, ξ, κ depend only on the (first two) moments of both $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{U}}$, and c is some universal constant (set arbitrarily as $1/20$ in the proof).

[4] And even $\mathcal{O}(\sqrt{N})$ with probability exponentially large in N as both distributions are sub-Gaussian. So the effects of those additional vertices can be neglected.

Theorem 5.2.1 generalizes to the case with capacities, see Sections 5.4.1 and 5.4.1. The details of the proof of Theorem 5.2.1 are postponed to Appendix 5.4.4, but the main ideas are given in Section 5.3.

5.2.3 Examples, Instantiations and Corollaries

We provide in this section some interesting examples and corollaries that illustrate the powerfulness of Theorem 5.2.1, and how it can be used to compare different situations.

d-regular graphs

The first typical example of random graphs are " d-regular ", for some $d \in \mathbb{N}$, i.e., graphs such that each vertex has an exact degree of d (to avoid trivial examples, we obviously assume $d \geq 2$).

It is non-trivial to sample a d-regular graph at random, yet it is easy to generate random graphs G_N with the configuration model described above, with the specific choices of $\pi_{\mathcal{U}} = \pi_{\mathcal{V}} = \delta_d$, the Dirac mass at d. The downside is that G_N is not exactly a d-regular bipartite random graph (as some vertices might be connected more than once, i.e., there might exist parallel edges). However, conditioned to be *simple*, i.e, without multiple edges and loops, it has the law of a uniform d-regular bipartite random graph. Moreover, the probability of being simple is bounded away from 0 (Van Der Hofstad, 2016); as a consequence, any property holding with probability tending to 1 for G_N, holds with probability tending to 1 for uniform d-regular bipartite random graphs. Finally, we also mention that Hall's Theorem (Frieze and Karoński, 2016) implies that G_N admits a perfect matching, so that $|M^*| = N$.

Instantiating Equation (5.1) to d-regular graphs yields that the competitive ratio of GREEDY converges, with probability 1, to $1 - (1 - G(1))^d$ where G is the solution of the following ODE

$$\frac{(1 - G(s))^{d-1}}{1 - \left(1 - (1 - G(s))^{d-1}\right)^d} G'(s) = \frac{1}{d}. \tag{5.3}$$

As expected, had we taken $d = 1$, then $G(s) = s$ hence the competitive ratio of GREEDY is 1 (but again, $d = 1$-regular graphs are trivial). More interestingly, if $d = 2$, the ODE has a closed form solution: $G(s) = \exp(\frac{s}{2}) - 1$, so that the competitive ratio of GREEDY converges to $4\sqrt{e} - (e + 3) \simeq 0.877 \gg 1 - \frac{1}{e} \simeq 0.632$, where the latter is a standard bound of the competitive ratio of GREEDY (for general, non-regular graphs) (Mehta, 2012).

Solving Equation (5.3) In the general case $d \geq 3$, even if Equation (5.3) does not have a closed form solution, it is still possible to provide some insights. Notice first that the polynomial $P(X) = 1 - (1 - (1 - X)^{d-1})^d$ admits $n := d(d - 1)$ roots, among which there is 1 with multiplicity $d - 1$. If X is another root, then

$$\left(1 - (1 - X)^{d-1}\right)^d = 1 \;\Leftrightarrow\; 1 - (1 - X)^{d-1} = e^{\frac{ik\pi}{d}},\ k = 1, \ldots, d - 1.$$

Therefore,

$$(1 - X)^{d-1} = 1 - e^{\frac{ik\pi}{d}},$$

which admits $d - 1$ distinct solutions for each $k = 1, \ldots, d - 1$. The resulting $n := (d - 1)^2$ distinct complex, denoted $x_1, \ldots, x_n$, are the roots of $P(X)/(1 - X)^{d-1}$, so the ODE reduces to:

$$\frac{y'(t)}{\prod_{1 \leq i \leq n} y(t) - x_i} = \frac{1}{d}. \tag{5.4}$$

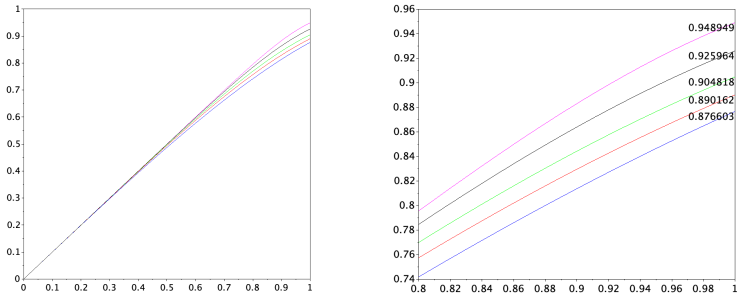


Figure 5.1: Numerical computations (on Scilab, results are almost instantaneous) of GREEDY performances for $d = 2$ (blue), $d = 3$ (red), $d = 4$ (green), $d = 6$ (black) and $d = 10$ (magenta). On the left, global solution, on the right, zoom-in on the end points with final values.

Since the following trivially holds:

$$\frac{1}{\prod_{1\le i\le n}(X - x_i)} = \sum_{1\le i\le n} \frac{1}{\prod_{j\neq i}(x_i - x_j)} \frac{1}{X - x_i} =: \sum_{1\le i\le n} \frac{a_i}{X - x_i}.$$

it is possible to integrate Equation (5.4) in $\sum_{1\le i\le n} a_i \log(y(t) - x_i) = \frac{s}{d} + c$ to finally get

$$\prod_{1\le i\le n} (y(t) - x_i)^{a_i} = C \exp(\frac{s}{d}),$$

and since $y(0) = 0$, it must hold that $C = \prod_{1\le i\le n}(-x_i)^{a_i}$. As a consequence, $y(1)$ solves:

$$\prod_{1\le i\le n} (y(1) - x_i)^{a_i} = e^{1/d} \prod_{1\le i\le n} (-x_i)^{a_i}.$$

Unfortunately, even for $d = 3$, the solution somehow simplifies but has no closed form; on the other hand, numerical computations indicate that the competitive ratio of GREEDY converges to 0.89 when $d = 3$ and N tends to infinity. We provide in Figure 5.3 the numerical solutions of the ODE for d-regular graphs (actually, we draw the functions $1 - \phi_U(1 - G(s))$ that are more relevant) for various values of d; the end-point obtained at $s = 1$ indicates the relative performance of GREEDY. As expected, those functions are point-wise increasing with d (as the problem becomes simpler and simpler for GREEDY when $d \ge 2$).

The Erdős-Rényi case.

In a Erdős-Rényi graph, there is an edge between two vertices $u \in \mathcal{U}$ and $v \in \mathcal{V}$ with some probability $p = \frac{c}{N}$, independently from each others. As N goes to infinity, the number of edges to a vertex follows (approximately) a Poisson law of parameter $c > 1$.

As a consequence, we consider the configuration model where $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ are Poisson laws of parameter c, which yields $\mu = c$, $\phi_{\mathcal{U}}(s) = \mathrm{e}^{c(s-1)}$. In this case, Equation (5.1) becomes:

$$\frac{cG'(s)\,\mathrm{e}^{-cG(s)}}{1 - \mathrm{e}^{-c\,\mathrm{e}^{-cG(s)}}} = 1.$$

The solutions are given by:

$$G(s) = \frac{1}{c}\log\left(\frac{c}{\log(\mathrm{e}^{k-cs}+1)}\right),$$

yielding

$$\phi_X(1-G(s)) = \frac{1}{c}\log\left(\mathrm{e}^{k-cs}+1\right).$$

The initial condition $\phi_{\mathcal{U}}(1-G(0)) = \phi_{\mathcal{U}}(1) = 1$ gives $e^k = e^c - 1$, from which we deduce that the number of matches of GREEDY is asymptotically proportional to

$$1 - \phi_{\mathcal{U}}(1-G(1)) = 1 - \frac{\log\left(2-\mathrm{e}^{-c}\right)}{c},$$

which recovers, as a sanity check, some existing results (Mastin and Jaillet, 2013).

The comparison of different configuration models

Using Gronwall's Lemma, it is possible to show Theorem 5.2.1 can be used to compare different configuration models, as in the following Corollary.

Corollary 5.2.1.1. *Consider two configuration models* $\mathbf{CM}_1(\mathbf{d}_1^U, \mathbf{d}_1^V)$ *and* $\mathbf{CM}_2(\mathbf{d}_2^U, \mathbf{d}_2^V)$*, s.t.* $\mathbf{d}_1^U$ *and* $\mathbf{d}_1^U$ *are both drawn i.i.d. from* π_U*,* $\mathbf{d}_1^V$ *is drawn i.i.d. from* π_V^1 *and* $\mathbf{d}_2^V$ *is drawn i.i.d. from* π_V^2*, with* $\sum_x x\pi_V^1(x) = \sum_x x\pi_V^2(x)$*. If* $\phi_V^1(s) \geq \phi_V^2(s)$ *for any* $s \in (0,1)$*, then by denoting respectively* γ_1 *and* γ_2 *the asymptotic proportion of vertices matched by* GREEDY *in* $\mathbf{CM}_1(\mathbf{d}_1^U, \mathbf{d}_1^V)$ *and* $\mathbf{CM}_2(\mathbf{d}_2^U, \mathbf{d}_2^V)$*, it holds that necessarily* $\gamma_2 \geq \gamma_1$*.*

For instance, let us assume that the degree distribution on the offline side is fixed. Then the matching size obtained by GREEDY is asymptotically larger if vertices on the online side all have exactly the same degree d rather than if those degrees are drawn from a Poisson distribution with expectation d.

A similar result (with a different criterion) holds with fixed degree distribution on the online side and differing one on the offline side.

5.2.4 greedy can outperform ranking !

We recall that the RANKING algorithm, which is the worse case optimal, chooses at random a ranking over $\mathcal{U}$ and uses it to break ties (i.e., if two vertices u and u' can be matched to v_k, then it is the one with the smallest rank that is matched by RANKING). Quite surprisingly, we get that in the configuration model RANKING can have a worse competitive ratio than GREEDY, which advocates again for its thorough study.

Proposition 5.2.2. *Let* γ_R *and* γ_G *be the assymptotic performances of* RANKING *and* GREEDY *on the* 2*-regular graph. The following holds:*

$$\gamma_G > \gamma_R.$$

In other words, GREEDY *outperforms* RANKING *in the* 2*-regular graph.*

We conjecture that the above result actually holds for any $d \geq 2$, and more generally for a wide class of distributions $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ (finding a general criterion would be very interesting). The proof of Proposition 5.2.2 is provided in Section 5.4.7. The main idea is that in the 2-regular graph, RANKING is biased towards selecting as matches vertices with two remaining half-edges

rather than just one. Indeed, vertices with only one remaining half-edge were not selected previously and thus have a higher rank. The vertices with only one remaining half-edge will not get matched in the subsequent iterations, so not picking them as matches is suboptimal. On the other hand, GREEDY picks any match uniformly at random and does not exhibit such bias.

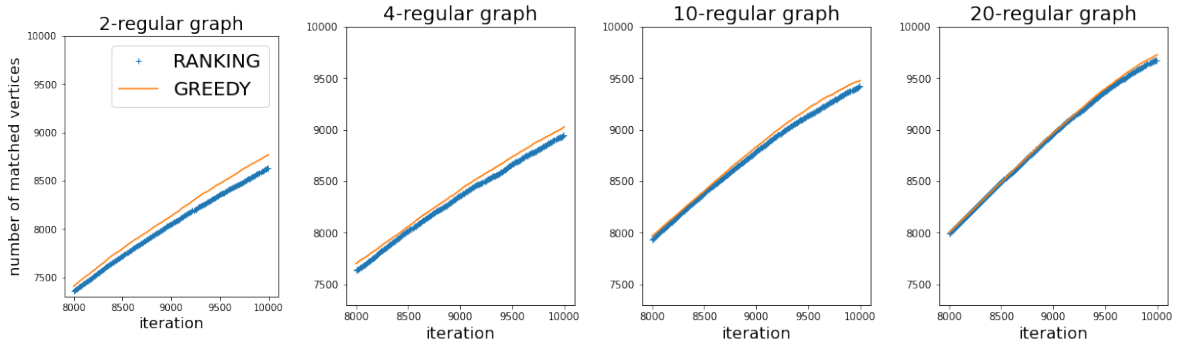


Figure 5.2: Experimental performances of GREEDY vs. RANKING on d-regular graphs

5.3 Ideas of proof of Theorem 5.2.1

The main idea behind the proof of Theorem 5.2.1 (postponed to Section 5.4.4) is to show that the random deterministic evolution of the matching size generated by GREEDY is closely related to the solution of some ODE (this is sometimes called "the differential equation method" (Wormald, 1995) or "stochastic approximations" (Robbins and Monro, 1951)). Computing the solution of the ODE is easier - if not explicitly, at least numerically in intricate cases - than estimating the performances of GREEDY by Monte-Carlo simulations and it provides qualitative, as well as quantitative, properties.

Tracking the matching size is non-trivial because the vertices (in $\mathcal{U}$ and $\mathcal{V}$) have different degrees, hence some of them are more likely to be matched than others. However, in the configuration model, each vertex has the same distribution of degrees before the sequences $\mathbf{d}^{\mathcal{U}}$ and $\mathbf{d}^{\mathcal{V}}$ are fixed. As a consequence, the proof relies on the three following techniques

1. The graph is built sequentially, along with the matching and not beforehand (fixing the "randomness" at the beginning would be very difficult to handle in the analysis). Thankfully, this does not change the law of the graph generated (this is obviously crucial).

2. We are not only going to track the size of the matching built as we need to handle different probabilities of matching (and pairing the graph) for each vertex. As a consequence, we are going to track the numbers of non-matched vertices which have still i half-edges to be paired and the number of already matched vertices that have j half-edges remaining. This will give one different ODE per value of i of j.

   Since $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ are sub-Gaussian, we will prove that with arbitrarily high probability - exponential in N -, there are only a polynomial number of such equations

3. All those differential equations are then "aggregated" to build the final ODE satisfied by the matching size. Interestingly, this aggregated ODE has a simple form, while the full system is on the other hand quite intricate.

In the following sub-sections, we separate the proofs in the different building blocks to provide intuitions; the proof of technical lemmas are deferred to the appendix.

5.3.1 Building the graph together with the matching

The first step in the analysis is to notice that the bipartite configuration model can be constructed by sequentially pairing the half-edges coming from $\mathcal{V}$. The matching generated by GREEDY is then constructed simultaneously with the graph. More precisely, given two sequences[5] of non-negative integers $\mathbf{d}^{\mathcal{U}} = (d_1^{\mathcal{U}}, \dots, d_N^{\mathcal{U}})$ and $\mathbf{d}^{\mathcal{V}} \cup \{d_{T+1}^{\mathcal{V}}\} = (d_1^{\mathcal{V}}, \dots, d_T^{\mathcal{V}}, d_{T+1}^{\mathcal{V}})$, we introduce in the following a generating algorithm that simultaneously build the associated bipartite configuration model $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}})$ together with GREEDY. Recall that the bipartite configuration model is obtained through a uniform matching between the half-edges of U and the half-edges of V. To avoid confusion, we will call a *marked matching* a pairing of two half-edges that corresponds to an edge that will belong to the constructed matching M. This construction pseudo-code is detailed in Algorithm 27.

Algorithm 27: GREEDY MATCHING CONFIGURATION MODEL WITHOUT CAPACITIES

Input: $\mathbf{d}^{\mathcal{U}} = (d_1^{\mathcal{U}}, \dots, d_N^{\mathcal{U}})$ and $\mathbf{d}^{\mathcal{V}} = (d_1^{\mathcal{V}}, \dots, d_T^{\mathcal{V}})$
Initialization. $\mathrm{M}_0 \leftarrow \emptyset$, $\mathcal{E}_0 \leftarrow \emptyset$ and $H_0^{\mathcal{U}} \leftarrow \{$ half-edges of $\mathcal{U}\}$
for $t = 1, \dots, T$ **do**
  Order uniformly at random the edges emanating from v_t: $e_1^t, \dots, e_{k_t}^t$
  for $i = 1, \dots, k_t$ **do**
    Choose uniformly an half-edge $e_i^{\mathcal{U}}$ in $H^{\mathcal{U}}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{u(e_i^{\mathcal{U}}), v_t\}$ // Create an edge between e_i^t and $e_i^{\mathcal{U}}$
    $H^{\mathcal{U}} \leftarrow H^{\mathcal{U}} \setminus \{e_i^{\mathcal{U}}\}$ // Remove the half-edge
    if v_t *and* $u(e_i^{\mathcal{U}})$ *unmatched* **then**
      $\mathrm{M}_t \leftarrow \mathrm{M}_{t-1} \cup \{u(e_i^{\mathcal{U}}), v_t\}$ // v_t is matched
    end
  end
end
$\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}}) \leftarrow (\mathcal{U}, \mathcal{V}, \mathcal{E})$.
Output: Bipartite configuration model $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}})$ and matching M_T on it.

Since each pairing of each half-edge is done uniformly at random, the graph obtained at the end of the algorithm has indeed the law of a bipartite configuration model. Moreover, it is easy to see that M corresponds to the matching constructed by GREEDY MATCHING on $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}})$.

5.3.2 Differential Equation Method - Stochastic Approximation

As mentioned above, several quantities are going to be tracked through time: for all $k \in \{0, \dots, T\}$ and all $i \geq 0$, we define:

- $F_i(k)$ as the number of vertices $u \in \mathcal{U}$ that are not yet matched at the end of step k and whose remaining degree is i, meaning that $d_u - i$ of their initial half-edges have been paired. We will refer them to as *free* vertices.
- $M_i(k)$ as the number of vertices $u \in \mathcal{U}$ already matched at the end of step k and whose remaining degree is i. We will refer them to as *marked* vertices.

Notice that for all $0 \leq k \leq T$, the sum $F_i(k) + M_i(k)$ corresponds to the total number of vertices of $\mathcal{U}$ with remaining degree i at the end of step k. We also define

- $\widehat{F}(k) := \sum_{i \geq 0} i F_i(k)$ is the number of available half-edges attached to free vertices at the end of step k,

[5] Without loss of generality, we assume that the additional extra vertex is always on the $\mathcal{V}$ side.

- $\widehat{M}(k) := \sum_{i\geq 0} iM_i(k)$ is the number of available half-edges attached to marked vertices at the end of step k.

We are going to study the evolution of these quantities along with the one of GREEDY. A major ingredient of the proof is to show that $F_i(k)$ and $M_i(k)$ closely follow the solutions of some ODE. This is the so-called *differential equation method* (Wormald, 1995), stated in Appendix 5.4.3. For instance, it can easily be seen that $\widehat{F}(k) + \widehat{M}(k)$ closely follows the function $t \mapsto \mu_{\mathcal{U}} - t\mu_{\mathcal{V}}$ on $(0, \mu_{\mathcal{U}}/\mu_{\mathcal{V}})$ in the following sense.

Lemma 5.3.1. *For every $\varepsilon > 0$, and for all $0 \leq k \leq T$,*

$$\left| \frac{\widehat{F}(k) + \widehat{M}(k)}{N} - \left(\mu_{\mathcal{U}} - \frac{k}{N}\mu_{\mathcal{V}}\right)\right| \leq \varepsilon.$$

with probability at least $1 - \exp\left(-\frac{N\epsilon^2}{2\sigma_U^2}\right) + \exp\left(-\frac{T\epsilon^2}{2\sigma_V^2}\right)$.

We now turn to each individual quantity F_i (resp. M_i). We can prove a similar result, yet the limit function is not explicit (unlike for the matching size as in Theorem 5.2.1 statement). The following Lemma 5.3.2 states that the discrete sequences of (free and marked) half-edges are closely related to the solutions of some system of differential equations.

Before stating it, we first introduce, for any sequence of non-negative numbers $(x_\ell)_{\ell\geq 0}$ and $(y_\ell)_{\ell\geq 0}$ such that $0 < \sum_\ell \ell(x_\ell + y_\ell) < \infty$, every $i \geq 0$, the following mappings

$$\Phi_i(x_0, x_1, \ldots, y_0, y_1, \ldots) := \frac{-i\mu_{\mathcal{V}}x_i + (i+1)\mu_{\mathcal{V}}x_{i+1} - h\left(\frac{\sum_{\ell\geq 0}\ell y_\ell}{\sum_{\ell\geq 0}\ell(x_\ell+y_\ell)}\right)(i+1)x_{i+1}}{\sum_{\ell\geq 0}\ell(x_\ell + y_\ell)} \tag{5.5}$$

and

$$\Psi_i(x_0, x_1, \ldots, y_0, y_1, \ldots) := \frac{-i\mu_{\mathcal{V}}y_i + (i+1)\mu_{\mathcal{V}}y_{i+1} + h\left(\frac{\sum_{\ell\geq 0}\ell y_\ell}{\sum_{\ell\geq 0}\ell(x_\ell+y_\ell)}\right)(i+1)x_{i+1}}{\sum_{\ell\geq 0}\ell(x_\ell + y_\ell)},$$

where h is the following function, well-defined on $[0, 1]$,

$$h(s) = \frac{1 - \phi_{\mathcal{V}}(s)}{1 - s}.$$

Lemma 5.3.2. *With probability $1 - \zeta N \exp(-\xi N^{c/2})$, there are at most N^c quantities F_i and M_i, and for all $0 \leq k \leq T$ and all $i \geq 0$*

$$\left|\frac{F_i(k)}{N} - f_i\left(\frac{k}{N}\right)\right| \leq \kappa N^{-2c} \quad \text{and} \quad \left|\frac{M_i(k)}{N} - m_i\left(\frac{k}{N}\right)\right| \leq \kappa N^{-2c},$$

where ζ, κ depend only on the (first two) moments of $\pi_{\mathcal{V}}$ and $\pi_{\mathcal{U}}$ and $c = 1/20$.

The continuous mappings f_i and m_i are solutions of the system of differential equations on $[0, \mu_{\mathcal{U}}/\mu_{\mathcal{V}})$

$$\begin{array}{rcl} \frac{\mathrm{d}f_i}{\mathrm{d}t} & = & \Phi_i(f_0, f_1, \ldots, m_0, m_1, \ldots), \\ \frac{\mathrm{d}m_i}{\mathrm{d}t} & = & \Psi_i(f_0, f_1, \ldots, m_0, m_1, \ldots), \\ f_i(0) & = & \pi_{\mathcal{U}}(i), \\ m_i(0) & = & 0. \end{array} \tag{5.6}$$

This system is well defined as stated by the following Lemma 5.3.3.

Lemma 5.3.3. *The system* (5.6) *has a unique solution which is well-defined on* $[0, \mu_{\mathcal{U}}/\mu_{\mathcal{V}})$. *More precisely, denoting by* f *and* m *the generating series of the sequences* $(f_i)_{i\geq 0}$ *and* $(m_i)_{i\geq 0}$,

$$f(t,s) = \sum_{i\geq 0} f_i(t)s^i \qquad \text{and} \qquad m(t,s) = \sum_{i\geq 0} m_i(t)s^i,$$

it holds that:

$$f\left(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}\left(1-\mathrm{e}^{-\mu_{\mathcal{V}}t}\right), s\right) = \phi_{\mathcal{U}}\left((s-1)e^{-\mu_{\mathcal{V}}t} + 1 - F(t)\right), \tag{5.7}$$

and

$$m\left(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}\left(1-\mathrm{e}^{-\mu_{\mathcal{V}}t}\right), s\right) = \int_0^t F'(u)\phi'_{\mathcal{U}}\left((s-1)e^{-\mu_{\mathcal{V}}u} + 1 - F(u)\right)\mathrm{d}u.$$

where F *is a solution of the following ODE*

$$\frac{\frac{1}{\mu_{\mathcal{U}}}\phi'_{\mathcal{U}}(1-F(t))}{1-\phi_{\mathcal{V}}\left(1-\frac{1}{\mu_{\mathcal{U}}}\phi'_{\mathcal{U}}(1-F(t))\right)}F'(t) = \mathrm{e}^{-\mu_{\mathcal{V}}t}.$$

5.3.3 Aggregating solutions to compute greedy performances

To get Theorem 5.2.1, notice that the number of vertices matched by GREEDY is N minus the number of free vertices remaining at the end, which is approximately equal to $Nf(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}, 1)$ by definition of f and because of Lemma 5.3.2. This corresponds to $t = +\infty$ in Equation (5.7), thus the performance of GREEDY is, with arbitrarily high probability, arbitrarily close to

$$N(1-\phi_{\mathcal{U}}(1-F(+\infty)))$$

The statement of Theorem 5.2.1 just follows from a simple final change of variable.

Conclusion

We studied theoretical performances of GREEDY algorithm on matching problems with different underlying structures. Those precise results are quite interesting and raise many questions, especially since GREEDY actually outperforms RANKING in many different situations (in theory for 2-regular graphs, but empirical evidence indicates that this happens more generically).

Our approach has also successfully been used to unveil some questions on the comparison between different possible models. But more general questions are still open; for instance, assuming that the expected degree is fixed, which situation is the more favorable to GREEDY and online algorithm: small or high variance, or more generally this distribution $\pi_{\mathcal{U}}$ or an alternative one $\pi'_{\mathcal{U}}$? The obvious technique would be to compare the solution of the different associated ODE's. Similarly, the questions of stability/robustness of the solution to variation in the distribution $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$ are quite challenging and left for future work.

We believe online matching will become an important problem for the machine learning community in the future. Each year, the complexity of the underlying graphs increases and we are considering adding features to the model in future work (such as random variables on the edges, modeling the interest for a consumer for a given product), or connection modeled via some Kernel between vertices features (say, if users and products/campaigns are embedded in the same space). In this context, machine learning tools will certainly be needed to tackle the problem.

Here, we have focused on the study of specific matching algorithms, namely GREEDY and

RANKING in the configuration model. It remains to be studied how knowledge on the input graph could drive algorithm design, for instance, can the knowledge of the degree distribution be leveraged in an algorithm? An example in that direction, is the work of Aamand et al. (2022), in which they exhibit the optimal algorithm for the bipartite version of a stochastic graph model due to Chung, Lu, and Vu Chung et al. (2003) where the expected values of the offline degrees are known.

5.4 Appendix

5.4.1 General version of the result

The fixed capacity matching problem

We now investigate the case where vertices $u \in \mathcal{U}$ have capacities, which means that they can be matched to several vertices $v \in \mathcal{V}$. Precisely, if the capacity of u is denoted by ω_u, then this vertex can be matched to at most ω_u vertices in $\mathcal{V}$ (but as before, half-edges of u are going to be paired with d_u half-edges originating from $\mathcal{V}$). The graph is still constructed using the configuration model introduced in Section 5.2.1, i.e., the law of d_u is $\pi_{\mathcal{U}}$ (and similarly, degrees of $v \in \mathcal{V}$ are i.i.d., with law $\pi_{\mathcal{V}}$).

For the moment, to simplify the analysis and the results statements, we are going to assume that all vertices $u \in \mathcal{U}$ have the same initial capacity $C \in \mathbb{N}$. We denote the random graph with capacities generated this way by $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}}, C)$

Theorem 5.4.1. ***(Performances of* greedy *with fixed capacities)***
Given $N \geq 1$ and $T = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}} N$, let M_T be the matching built by GREEDY *on $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}}, C)$ then the following convergence in probability holds:*

$$\frac{|\mathrm{M}_T|}{CN} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \sum_{k=0}^{C-1} \frac{1-k/C}{k!} G(1)^k \phi_{\mathcal{U}}^{(k)} \left(1 - G(1)\right).$$

where G is the unique solution of the following ordinary differential equation

$$G'(s) = \frac{1 - \phi_{\mathcal{V}}\left(1 - \frac{1}{\mu_{\mathcal{U}}}\Gamma_U(G(s))\right)}{\frac{\mu_{\mathcal{V}}}{\mu_{\mathcal{U}}}\Gamma_U(G(s))}.$$

where

$$\Gamma_U(g) = \phi'_{\mathcal{U}}(1-g) + \sum_{k=1}^{C-1} \frac{g^k}{k!} \phi_{\mathcal{U}}^{(k+1)}(1-g)$$

Moreover, for any $s \in [0,1]$, if $M_T(s)$ is the matching obtained by GREEDY *after seeing a proportion s of vertices of $\mathcal{V}$, then*

$$\frac{|\mathrm{M}_T(s)|}{CN} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \sum_{k=0}^{C-1} \frac{1-k/C}{k!} G(s)^k \phi_{\mathcal{U}}^{(k)} \left(1 - G(s)\right).$$

The proof of Theorem 5.4.1, in Appendix 5.4.5, has three major differences with the one of Theorem 5.2.1:

1. The first one is that more quantities must be tracked, not just the number of vertices with remaining free half-edges, but the number of such vertices for each possible value of remaining capacity; the total number of equations is roughly speaking multiplied by a factor $(C+1)/2$ (since only $F_i(k)$ are affected by the capacities and not $M_i(k)$). We will therefore denote in the remaining by $F_i^{(c)}(k)$ the number of vertices with i remaining half-edges to be paired and with current capacity equal to c (those vertices can still be matched to c different vertices $v \in \mathcal{V}$).

2. The second major difference lies in the resolution of the system of differential equations. The solution was rather direct without capacities (i.e., $c = 1$). Unfortunately, the evolution of $F_i^{(c)}$ strongly depends on $F_i^{(c+1)}$. As a consequence, the trick is to solve this system

by induction, starting from $c = C$ (this solution is almost identical to that of the case with no capacities) and then to inject this solution in the PDEs defining $F_i^{(C-1)}$ so on so forth. Indeed, the fluid limits of $\sum_i F_i(c)$ and $\sum_i M_i$, that we denote respectively be $f^{(c)}$ and m satisfy the following coupled equations (up to some time change $\theta(t)$ and where $H(t) = h(q(t))$ for some function $q(\cdot)$ introduced in the proof):

$$\partial_t f^{(c)}(\theta(t), s) = \left[-\mu_{\mathcal{V}} s + \mu_{\mathcal{V}} - H(t)\right] \partial_s f^{(c)}(\theta(t), s) + H(t)\partial_s f^{(c+1)}(\theta(t), s),$$

and

$$\partial_t m(\theta(t), s) = \left[-\mu_{\mathcal{V}} s + \mu_{\mathcal{V}}\right] \partial_s m(\theta(t), s) + H(t)\partial_s f^{(1)}(\theta(t), s).$$

3. Finally, the third main difference is how the performances of GREEDY are defined. The upper bound is obviously to create the minimum between CN and T matches (where T is the number of vertices in $\mathcal{V}$). Anyway, those two numbers are within a constant multiplicative factor (recall that $T = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}} N$ for a valid configuration model), hence we arbitrarily chose to normalize GREEDY performances by CN. As a consequence, the (normalized) performances of GREEDY now rewrite as

$$\frac{\sum_{i\geq 0}\left(M_i(T) + \sum_{c=1}^{C}(1 - \frac{c}{C})F_i^{(c)}(T)\right)}{N},$$

where $M_i(k)$ still denotes the number of marked vertices, i.e., those whose capacities have been depleted before step k with i remaining half-edges to be paired.

General case, online matching with capacities

In the general case, we no longer assume that all vertices $u \in \mathcal{U}$ have the same initial capacities, but ω_u can be equal to any value in $\mathbb{N}$ (yet this capacity is independent of the degree). Notice however that the capacities ω_u of vertices could be capped at their degrees d_u (since they would never be depleted otherwise). As a consequence, capacities can be assumed to be bounded by $C < N^\beta$ for some $\beta < 1$ since the maximal degree is also smaller than N^β with arbitrarily high probability.

We therefore denote by $p_c \in [0, 1]$ the fraction of vertices of $\mathcal{U}$ whose initial capacity is exactly $c \in [1, C]$. Notice, we do not need to assume that capacities are drawn i.i.d. accordingly to some distribution, our results hold for any values $(p_c)_c$. We denote by $\mathbf{CM}(\mathbf{d}^{\mathcal{U}}, \mathbf{d}^{\mathcal{V}}, \mathbf{p})$ the random graph with capacities generated.

Quite interestingly, the techniques are exactly the same as in the previous case: we consider the exact same system of differential equations; the only differences are the initial conditions. Similarly, the maximal matching size is no longer NC but $N\mathbb{E}_{\mathbf{p}}[c] := N\sum_c cp_c$. We also denote the cdf of the empirical distribution p_c by $P_c := \sum_{k\leq c} p_c$

Theorem 5.4.2. ***(Performances of* greedy *with different capacities)***

Given $N \geq 1$ and $T = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}} N$, let M_T be the matching built by GREEDY on $\mathbf{CM}(\mathbf{d}^U, \mathbf{d}^V, \mathbf{p})$ then the following convergence holds in probability:

$$\frac{|\mathrm{M}_T|}{N\mathbb{E}_{\mathbf{p}}[c]} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \sum_{k=0}^{C-1} \frac{\sum_{c=1}^{C} cp_{c+k}}{\mathbb{E}_{\mathbf{p}}[c]} \frac{1}{k!} G(1)^k \phi^{(k)}\left(1 - G(1)\right).$$

where G is the unique solution of the following ordinary differential equation

$$G'(s) = \frac{1 - \phi_{\mathcal{V}}\left(1 - \frac{1}{\mu_{\mathcal{U}}}\Gamma_U^{\mathbf{p}}(G(s))\right)}{\frac{\mu_{\mathcal{V}}}{\mu_{\mathcal{U}}}\Gamma_U^{\mathbf{p}}(G(s))}.$$

with

$$\Gamma_U^{\mathbf{p}}(g)) = \phi_U'(1-g) + \sum_{k=1}^{C-1}\left(\frac{(1-P_k)g^k}{k!}\phi_{\mathcal{U}}^{(k+1)}(1-g)\right).$$

Moreover, for any $s \in [0,1]$, if $M_T(s)$ is the matching obtained by GREEDY *after seeing a proportion s of vertices of $\mathcal{V}$, then*

$$\frac{|\mathrm{M}_T(s)|}{N\mathbb{E}_{\mathbf{p}}[c]} \xrightarrow[N\to+\infty]{\mathbf{P}} 1 - \sum_{k=0}^{C-1}\frac{\sum_{c=1}^{C} c p_{c+k}}{\mathbb{E}_{\mathbf{p}}[c]}\frac{1}{k!}G(s)^k\phi^{(k)}\left(1-G(s)\right).$$

As mentioned before, the proof (delayed to Appendix 5.4.6) is rather similar to the previous one; the major difference is that the change of initial condition of the system of PDE makes it a bit more complicated to solve (hence the more intricate formulation of the result).

5.4.2 Additional Numerical Experiments

Further comparisons between the theoretical result and simulations

We provide in Figure 5.3 a comparison between the score predicted by the numerical solutions of the ODE (the functions $1 - \phi_{\mathcal{U}}(1 - G(s))$) for 4-regular graphs and the simulated performance of GREEDY for various values of N. As expected, the deviations of the simulated trajectories remain within $\mathcal{O}(\sqrt{N})$ of the expected theoretical trajectory. Figure 5.4 illustrates the same comparison on an Erdős-Rényi graph whose expected degree equals 4.

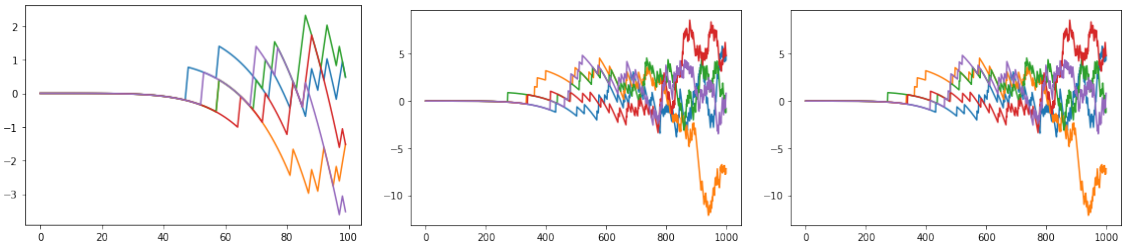


Figure 5.3: Difference between the theoretical performances and simulated performances of the GREEDY algorithm on the d-regular graph ($d = 4$) on 5 independent runs, with $N = 100, 1000, 10000$.

In Figure 5.5, we plot the theoretical performance of the GREEDY algorithm along with its experimental performance on the d-regular graph for various values of d. We also plot the competitive ratio of GREEDY predicted by the ODE as a function of d. As expected, the score increases with d (as the problem becomes simpler and simpler for GREEDY when $d \geq 2$).

greedy vs ranking

We further illustrate in this section the quite surprising fact that, in some configuration models, GREEDY actually outperforms RANKING.

In the adversarial configuration, it is known that the competitive ratio of RANKING is $1 - \frac{1}{e}$ which is bigger than the one of GREEDY, equal to $1/2$, see (Mehta, 2012). In the following

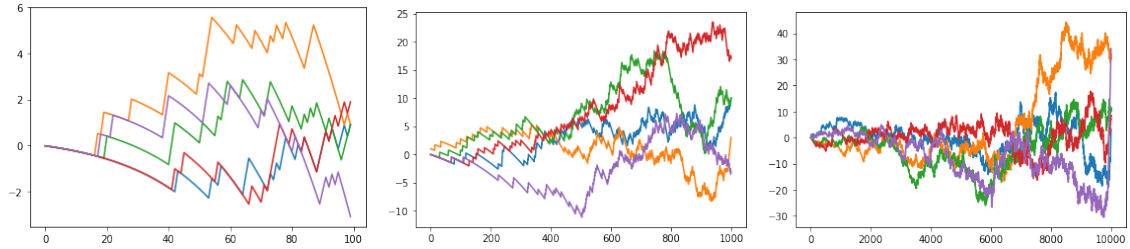


Figure 5.4: Difference between the theoretical value 5.2 and simulated performances of the GREEDY algorithm on the Erdős-Rényi graph, $c = 4$, on 5 independent runs, with $N = 100, 1000, 10000$.

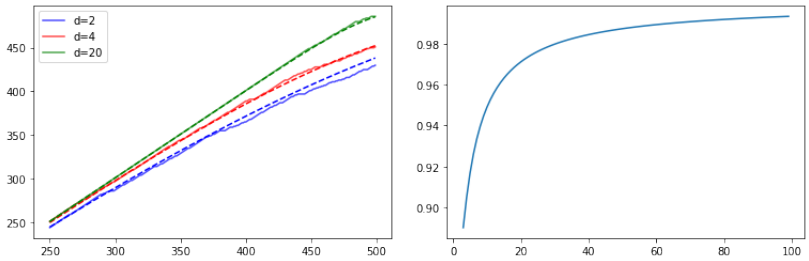


Figure 5.5: On the left, the expected theoretical performance of the GREEDY algorithm (dashed line) along with the simulated performance (full line) for various values of d. On the right, the expected competitive ratio of GREEDY on the d-regular graph as a function of d.

figures, we also plot the performances of two other "algorithms" SMALLEST and HIGHEST, for the sake of comparison; indeed, those are not admissible algorithms as they use the (future) knowledge of the number of half-edges of each vertex $u \in \mathcal{U}$.

More precisely, SMALLEST matches a vertex $v_k \in \mathcal{V}$ to the vertex $u \in \mathcal{U}$ with the smallest number of remaining half-edges (under the constraints obviously that $(u, v_k) \in \mathcal{E}$). As a consequence SMALLEST could be seen as an upper limit for an online algorithm.

HIGHEST does the opposite: it matches v_k to the vertex $u \in \mathcal{U}$ with the highest remaining number of half-edges. So HIGHEST should serve as a lower bound/sanity check for any online algorithm.

In Figure 5.6, the performances of those 4 matching "algorithms" (again SMALLEST and HIGHEST are not admissible as they use extra knowledge) are illustrated on configuration models with $d = 2, 4, 10$ and 20.

As mentioned before, GREEDY surprisingly outperforms RANKING in some configuration models, with a relative performance that decreases with d (which is rather natural on the other hand, since the relative performance of HIGHEST and SMALLEST also decreases).

Figure 5.6 also illustrates the different time steps at which algorithms fail to match new vertices v_k (because all the u they are paired with are already matched with another vertex v_j for some $j < k$). This happens later and later as d increases (as expected), at around half the horizon for $d = 2$ and roughly 82% with $d = 20$.

On the other hand, RANKING and GREEDY have the same performance on Erdos-Renyi graphs, which is a consequence of the memory-less property of those graphs, i.e. the probability of creating a match at each iteration depends only on the number of matched vertices, as shown in Figure 5.7.

In Figure 5.8, we plot the relative performance of RANKING and GREEDY on bi-degrees graph, where half the vertices have degree x, the other half $2x$. The plots illustrate that the

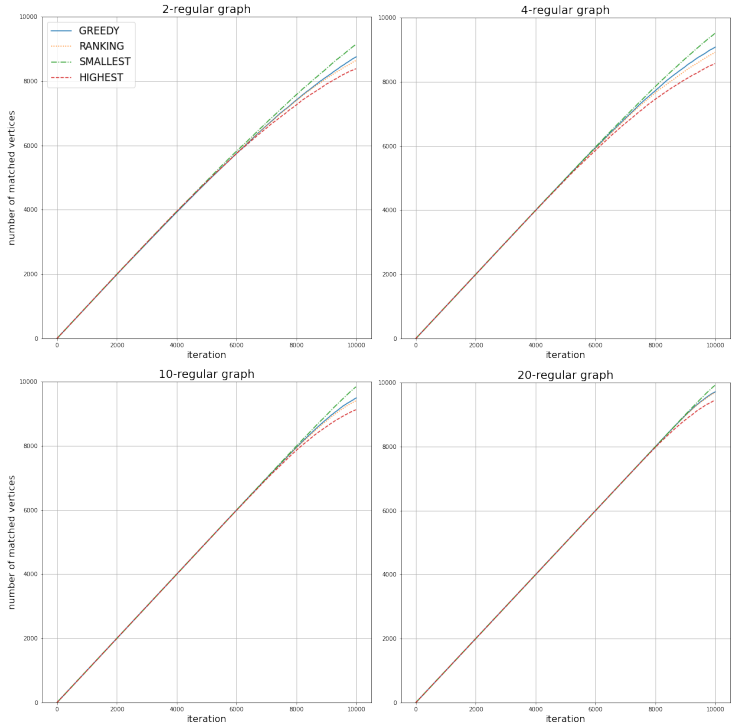


Figure 5.6: GREEDY outperforms RANKING in d-regular graphs

best algorithm is not always teh best one depending on the value of x.

A few vertices with high capacity vs many vertices with low capacity

In this section, we investigate how nodes' capacities affect GREEDY's expected performance. The baseline is its performance on a random graph where all vertices have capacity 1 and the vertices degrees in $\mathcal{U}$ and $\mathcal{V}$ follow the distributions $\pi_{\mathcal{U}}$ and $\pi_{\mathcal{V}}$. The comparison graph with capacity C has $|\mathcal{U}|/C$ "in-place" vertices, each with a capacity C, and their degrees follows the modified distribution $\tilde{\pi}_C^U$ where $\tilde{\pi}_C^U(x = k) = \pi_{\mathcal{U}}(x = k/C)$. Informally, the graph with capacity C is built from the baseline graph by merging C vertices of equal degree d into a single vertex of degree dC.

The results of the simulation illustrate that the GREEDY performs better on graphs with

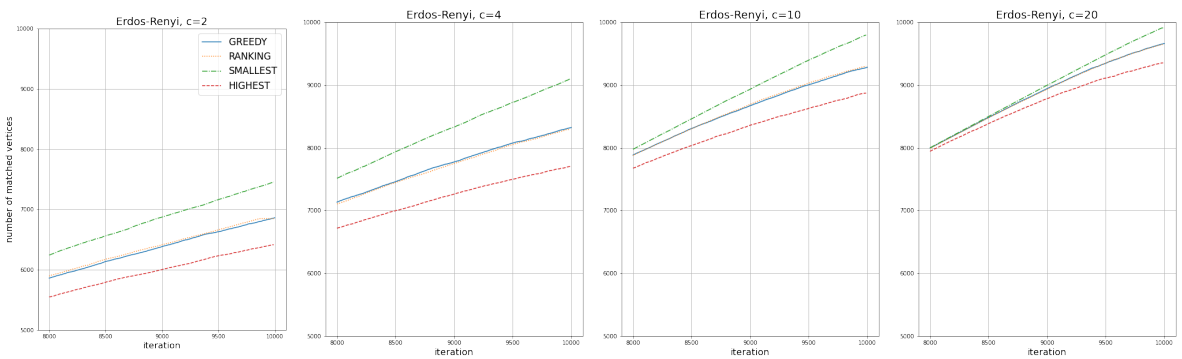


Figure 5.7: Experimental performances of GREEDY vs. RANKING on Erdos-Renyi graphs

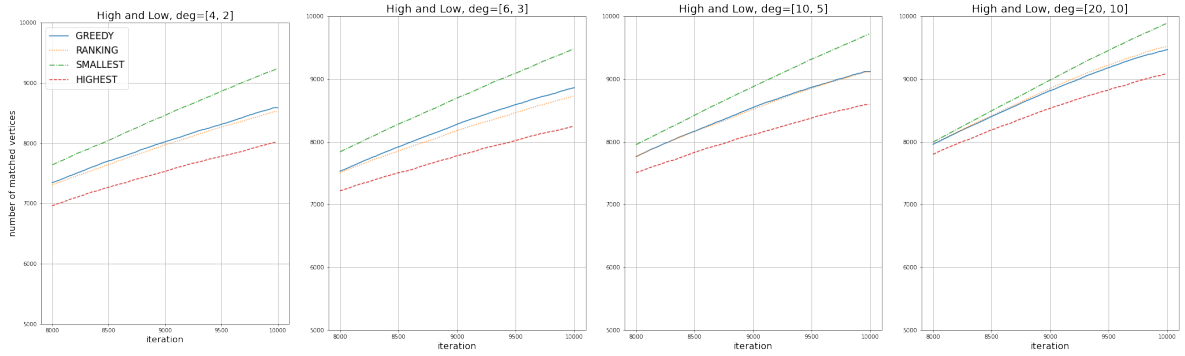


Figure 5.8: Experimental performances of GREEDY vs. RANKING on bi-degrees graphs

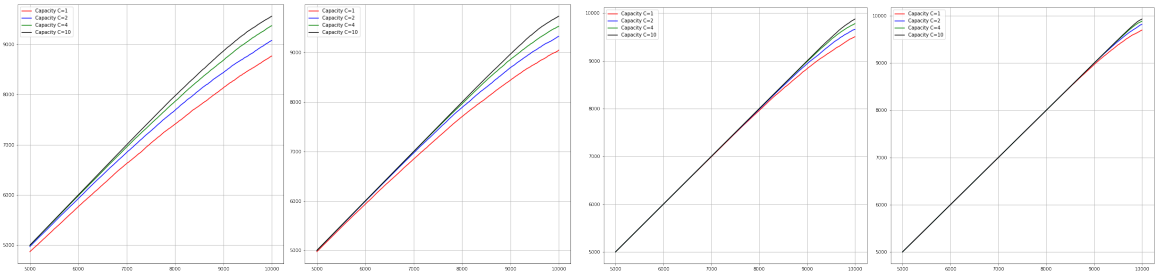


Figure 5.9: GREEDY performs better in high capacity graphs in d-regular graphs, from left to right $d = [2, 4, 10, 20]$

vertices of high capacity.

5.4.3 Stochastic approximation & Differential equation method

The following theorem is an improved version of Wormald's Theorem (Enriquez et al., 2019).

Theorem 5.4.3. *Let $a > 0$. For all $N \geq 1$ and all $1 \leq k \leq N^a$, let $Y_k(i) = Y_k^{(N)}(i)$ be a Markov chain with respect to a filtration $\{\mathcal{F}_i\}_{i\geq 1}$. Suppose that, for all $k \geq 1$, there exists a function f_k such that:*

- $Y_k(0)/N = z_k(0)$;
- $|Y_k(i+1) - Y_k(i)| \leq N^\beta$;
- $\left|\mathbf{E}\left[Y_k(i+1) - Y_k(i) \,\middle|\, \mathcal{F}_i\right] - f_k\left(\frac{i}{N}, \frac{(Y_k(i))_{1\leq k\leq N^a}}{N}\right)\right| \leq cN^{-\lambda}$, *for some constant $c > 0$*

where $0 < \beta < 1/2$, $\lambda > 0$. Suppose that the following infinite system of differential equations with initial conditions $(z_k(0))_{k\geq 1}$ has a unique solution $(z_k)_{k\geq 1}$:

$$\forall k \geq 1, \quad z_k'(t) = f_k(t, (z_k(t))_{k\geq 1}).$$

Then, for all $k \geq 1$, $Y_k(\lfloor tN \rfloor)/N$ converges in probability towards z_k for the topology of uniform convergence.

More precisely, for every $1 < \varepsilon < \frac{1-\beta}{\beta}$, for every $\frac{(1+\varepsilon)\beta}{2} < \alpha < \varepsilon\beta$ and for every $0 \leq i \leq \frac{N}{\omega}$ where $\omega = N^{(1+\varepsilon)\beta}$, it holds that

$$\mathbb{P}\Big(|Y(i\omega) - z(\frac{i\omega}{N})N| \leq i\Big(N^{\alpha+\beta} + cN^{(1+\varepsilon)\beta-\lambda} + N^{2(1+\varepsilon)\beta-2}\Big)\Big) \leq i\exp\Big(-\frac{N^{2\alpha-(1+\varepsilon)\beta}}{2}\Big)$$

5.4.4 Proofs of technical steps of Theorem 5.2.1

Proof of Lemma 5.3.1

It is an application of (maximal) Hoeffding-Azuma inequality since, for every $0 \leq k \leq M-1$,

$$\mathbf{E}\left[\left(\widehat{F}(k+1)+\widehat{M}(k+1)\right)-\left(\widehat{F}(k)+\widehat{M}(k)\right)\Big|\,\mathcal{F}_k\right]=-\mathbf{E}\left[d_k^{\mathcal{V}}\right]=-\mu_{\mathcal{V}}.$$

Proof of Lemma 5.3.2

Since $\pi_{\mathcal{V}}$ is $\sigma_{\mathcal{V}}$ subGaussian, then for any $\beta>0$,

$$\mathbf{P}\left(\exists i\in\{1,\dots,T\},\, d_i^{\mathcal{V}}\geq\mu_{\mathcal{V}}+N^{\beta}\right)\leq T\exp(-\frac{N^{2\beta}}{2\sigma_{\mathcal{V}}^2}).$$

In particular, for some $\beta<1/2$ to be chosen later on, if $\mu_{\mathcal{V}}\leq N^{\beta}/2$, then all degrees are smaller than N^{β} with probability at least $1-T\exp\left(-\frac{N^{2\beta}}{8\sigma_{\mathcal{V}}^2}\right)$; from now on, we will place ourselves on that event.

We also denote by $(\mathcal{F}_k)_{0\leq k\leq M}$ the natural filtration associated to the GREEDY MATCHING algorithm. In order to apply Theorem 5.4.3, it remains to control for every $i\geq 0$ and $0\leq k\leq M-1$,

$$\left|\mathbf{E}\left[F_i(k+1)-F_i(k)\,\Big|\,\mathcal{F}_k\right]-\Phi_i\left(F_0\left(\frac{k}{N}\right),F_1\left(\frac{k}{N}\right),\dots,M_0\left(\frac{k}{N}\right),M_1\left(\frac{k}{N}\right),\dots\right)\right|$$

and

$$\left|\mathbf{E}\left[M_i(k+1)-M_i(k)\,\Big|\,\mathcal{F}_k\right]-\Psi_i\left(F_0\left(\frac{k}{N}\right),F_1\left(\frac{k}{N}\right),\dots,M_0\left(\frac{k}{N}\right),M_1\left(\frac{k}{N}\right),\dots\right)\right|$$

Let $0\leq k\leq T-\frac{2N^{\gamma}}{\mu^{\mathcal{V}}}$, with $\gamma>1/2$ some parameter to be fixed later, so that, according to Lemma 5.3.1, with probability at least $1-\exp(-\frac{N^{2\gamma-1}}{2\sigma_{\mathcal{U}}^2})-\exp(-\frac{\mu_{\mathcal{U}}N^{2\gamma-1}}{2\mu_{\mathcal{V}}\sigma_{\mathcal{V}}^2})$ it holds that $\widehat{F}(k)+\widehat{M}(k)\geq N^{\gamma}$.

Recall that, in the k-th step of the algorithm, half-edges of the k-th vertex of V_N are ordered uniformly at random: $(e_i^k)_i$ for $i=1,\dots,d_k^{\mathcal{V}}$. Then, each of these half-edges is sequentially paired uniformly at random with half-edges of $d^{\mathcal{V}}$ that are not yet paired. Let u_i^k be the vertex to which e_i^k is paired and let I_k be the first integer i such that u_i^k belongs to the free vertices of $\mathcal{U}$ at time k, that is to the vertices that are not yet matched. If such an integer does not exist, that is when all u_i^k are already matched, we set $I_k=+\infty$. As a consequence, we aim at estimating $\mathbf{P}\left(I_k=i\,\Big|\,\mathcal{F}_k\right)$ for the different admissible values, where this probability has the following explicit definition

$$\begin{aligned}\mathbf{P}\left(I_k=i\,\Big|\,\mathcal{F}_k\right)&=\frac{\widehat{M}(k)}{\widehat{F}(k)+\widehat{M}(k)}\frac{\widehat{M}(k)-1}{\widehat{F}(k)+\widehat{M}(k)-1}\cdots\frac{\widehat{M}(k)-(i-2)}{\widehat{F}(k)+\widehat{M}(k)-(i-2)}\frac{\widehat{F}(k)}{\widehat{F}(k)+\widehat{M}(k)-(i-1)}\\&=\frac{\widehat{M}(k)!}{(\widehat{M}(k)-(i-1))!}\frac{(\widehat{F}(k)+\widehat{M}(k)-i)!}{(\widehat{F}(k)+\widehat{M}(k))!}\widehat{F}(k)\end{aligned}$$

First, assume that $\widehat{M}(k) \geq 2N^\theta$ for some parameter $\theta > 2\beta$ to be chosen later, so that those probabilities are all strictly positive. Using Stirling approximation formula, we get that, with $p(k) = \frac{\widehat{M}(k)}{\widehat{F}(k)+\widehat{M}(k)}$ and for any i,

$$0 \geq \frac{\mathbf{P}\left(I_k = i \,\middle|\, \mathcal{F}_k\right) - (1-p(k))^{i-1}p(k)}{(1-p(k))^{i-1}p(k)} \geq -2\frac{N^{2\beta}}{N^\theta} - \frac{N^\beta}{N^\gamma}$$

Second, assume that $\widehat{M}(k) < 2N^\theta$ for some $\theta > \beta$. This immediately implies that, for i,

$$0 \geq \mathbf{P}\left(I_k = i \,\middle|\, \mathcal{F}_k\right) - (1-p(k))^{i-1}p(k) \geq -2\frac{N^\theta}{N^\gamma}$$

Similar inequalities holds for $\mathbf{P}(I_k = +\infty \,\middle|\, \mathcal{F}_k)$, except that it is approximately equal to $\mathbf{E}\left[(1-p(k))^{d_k^{\mathcal{V}}}\right] = \phi_{\mathcal{V}}(1-p(k))$.

It remains to control the evolution of the processes $F_i(k)$ and $M_i(k)$. Notice that, by their very definition, on the event $I_k = x$ for some $1 \leq x \leq d_k^{\mathcal{V}}$, the following happens:

1. The first $x-1$ half-edges $e_k^1, \ldots, e_k^x$ are paired uniformly at random with marked half-edges of $\mathcal{U}$. If the corresponding vertex has a remaining degree equal to i, then M_i decreases by one, and M_{i-1} increases by one.
2. The x-th half-edge e_k^x is paired uniformly at random with free half-edge of $\mathcal{U}$. If the corresponding vertex has a remaining degree i, then F_i decreases by one, and M_{i-1} increases by one.
3. The $d_k^{\mathcal{V}} - x$ remaining half-edges $e_k^{x+1}, \ldots, e_k^{d_k^{\mathcal{V}}}$ are paired uniformly at random with half-edges of $\mathcal{U}$. If the corresponding vertex is free with remaining degree i, then F_i decreases by one and F_{i-1} increases by one. Otherwise, if the corresponding vertex is marked with remaining degree i, then M_i decreases by one and M_{i-1} increases by one.

Notice that, after the pairing of each half-edges, the quantity $\widehat{F}(k)$ (resp. $\widehat{M}(k)$) may decrease (resp. increase) by one. Therefore, working on the event where $d_k^{\mathcal{V}} \leq N^\beta$, we deduce that $\widehat{F}$ and $\widehat{M}$ are affected by an additive term of order at most N^β. The same argument holds on F_i and M_i.

All of these considerations imply that

$$\left|\mathbf{E}[F_i(k+1) - F_i(k) \,\middle|\, \mathcal{F}_k,\, I_t = x] - \left(-\frac{iF_i(k)}{\widehat{F}(k)} + (\mu_{\mathcal{V}} - x)\left(-\frac{iF_i(k)}{\widehat{F}(k)+\widehat{M}(k)} + \frac{(i+1)F_{i+1}(k)}{\widehat{F}(k)+\widehat{M}(k)}\right)\right)\right|$$
$$\leq 2\sigma_{\mu\nu}^2 \frac{N^\beta}{N^\gamma}$$

and similarly

$$\begin{aligned}\Bigg|\mathbf{E}[M_i(k+1) - M_i(k) \,\Big|\, \mathcal{F}_k,\, I_t = x] - \Bigg((x-1)\Big(-\frac{iM_i}{\widehat{M}} + \frac{(i+1)M_{i+1}}{\widehat{M}}\Big) + \frac{(i+1)F_{i+1}}{\widehat{F}}\Bigg) \\ + (\mu_{\mathcal{V}} - x)\left(-\frac{iM_i}{\widehat{F}+\widehat{M}} + \frac{(i+1)M_{i+1}}{\widehat{F}+\widehat{M}}\right)\Bigg)\Bigg| \\ \leq 2\sigma_\mu^2 \frac{N^\beta}{N^\gamma} + 2\sum_{j=1}^{x-1} \frac{j}{\widehat{M}(k) - j}\end{aligned}$$

Finally, the case $I_t = +\infty$ is handled similarly, as by definition

$$\mathbf{E}[F_i(k+1) - F_i(k) \,|\, \mathcal{F}_k,\, I_t = +\infty] = 0.$$

and the following also holds also holds:

$$\left|\mathbf{E}[M_i(k+1) - M_i(k) \,|\, \mathcal{F}_k,\, I_t = +\infty] - \mu_{\mathcal{V}} \left(-\frac{iM_i}{\widehat{M}} + \frac{(i+1)M_{i+1}}{\widehat{M}} \right) \right| \leq 2\sigma_\mu^2 \frac{N^\beta}{N^\gamma} + 2 \sum_{j=1}^{d_k^{\mathcal{V}}-1} \frac{j}{\widehat{M}(k) - j}.$$

It remains to compute the expected variation in $F_i(k)$ and $M_i(k)$. It is a bit simpler for the former, but still, to lighten the notations, we write $p = p_k$ and $q = q_k$ in the following computation.

$$\begin{aligned}
&\mathbf{E}\Big[F_i(k+1) - F_i(k) \,\Big|\, \mathcal{F}_k\Big] \\
&\quad = \mathbf{E}_{d_k^{\mathcal{V}} \sim \pi_{\mathcal{V}}} \left[\sum_{x=1}^{d_k^{\mathcal{V}}} q^{x-1} \left(-\frac{iF_i}{\widehat{F} + \widehat{M}} \right) + p \sum_{x=1}^{d_k^{\mathcal{V}}} q^{x-1} (d_k^{\mathcal{V}} - x) \left(-\frac{iF_i}{\widehat{F} + \widehat{M}} + \frac{(i+1)F_{i+1}}{\widehat{F} + \widehat{M}} \right) \right] + \eta_N \\
&\quad = \frac{1}{\widehat{F} + \widehat{M}} \mathbf{E}_{d_k^{\mathcal{V}} \sim \pi_{\mathcal{V}}} \left[-iF_i \left(\sum_{i=1}^{d_k^{\mathcal{V}}} q^{x-1} + p d_k^{\mathcal{V}} \sum_{x=1}^{d_k^{\mathcal{V}}} q^{x-1} - p \sum_{x=1}^{d_k^{\mathcal{V}}} x q^{x-1} \right) \right. \\
&\qquad\qquad \left. + (i+1)F_{i+1} \left(p d_k^{\mathcal{V}} \sum_{x=1}^{d_k^{\mathcal{V}}} q^{x-1} - p \sum_{x=1}^{d_k^{\mathcal{V}}} x q^{x-1} \right) \right] + \eta_N \\
&\quad = \frac{1}{\widehat{F} + \widehat{M}} \mathbf{E}_{d_k^{\mathcal{V}} \sim \pi_{\mathcal{V}}} \left[-iF_i \left(\frac{1 - q^{d_k^{\mathcal{V}}}}{1-q} + d_k^{\mathcal{V}} (1 - q^{d_k^{\mathcal{V}}}) - \frac{d_k^{\mathcal{V}} q^{d_k^{\mathcal{V}}+1} - (d_k^{\mathcal{V}} + 1) q^{d_k^{\mathcal{V}}} + 1}{1-q} \right) \right. \\
&\qquad\qquad \left. + (i+1)F_{i+1} \left(q(1 - q^{d_k^{\mathcal{V}}}) - \frac{d q^{d_k^{\mathcal{V}}+1} - (d_k^{\mathcal{V}} + 1) q^{d_k^{\mathcal{V}}} + 1}{1-q} \right) \right] + \eta_N \\
&\quad = \frac{1}{\widehat{F} + \widehat{M}} \mathbf{E}_{d_k^{\mathcal{V}} \sim \pi_{\mathcal{V}}} \left[-iF_i d_k^{\mathcal{V}} + (i+1)F_{i+1} \frac{q^{d_k^{\mathcal{V}}} - d_k^{\mathcal{V}} q + (d_k^{\mathcal{V}} - 1)}{1-q} \right] + \eta_N \\
&\quad = \frac{1}{\widehat{F} + \widehat{M}} \mathbf{E}_{d_k^{\mathcal{V}} \sim \pi_{\mathcal{V}}} \left[-iF_i + (i+1) d_k^{\mathcal{V}} F_{i+1} + (i+1)F_{i+1} \frac{1 - q^{d_k^{\mathcal{V}}}}{1-q} \right] + \eta_N \\
&\quad = \frac{-i\mu_{\mathcal{V}} F_i + (i+1)\mu_{\mathcal{V}} F_{i+1} - (i+1)h(q)F_{i+1}}{\widehat{F} + \widehat{M}} + \eta_N,
\end{aligned}$$

which is exactly (5.5), up to error term η_N that satisfies, if $\widehat{M}(k) < 2N^\theta$,

$$|\eta_N| \leq 2\sigma_{\mu_{\mathcal{V}}}^2 \frac{N^\beta}{N^\gamma} + 2\mu_{\mathcal{V}}^2 \frac{N^\theta}{N^\gamma}$$

and, if $\widehat{M}(k) \geq 2N^\theta$,

$$|\eta_N| \leq 3\sigma_{\mu_{\mathcal{V}}}^2 \frac{N^\beta}{N^\gamma} + 2\mu_{\mathcal{V}}^2 \frac{N^{2\beta}}{N^\theta} + \mu_{\mathcal{V}}^2 \frac{N^\beta}{N^\gamma}$$

Computations are quite similar for the difference in $M_i(k)$ and the error term still depends

whether $\widehat{M}(k)$ is bigger, or smaller, than $2N^\theta$:

$$\begin{aligned}
&\mathbf{E}\Big[M_i(k+1) - M_i(k)\,\Big|\,\mathcal{F}_k\Big] \\
&= \mathbf{E}_{d_k^{\mathcal{V}}\sim\pi_{\mathcal{V}}}\left[\left(\frac{d_k^{\mathcal{V}}(1-p)^{d_k^{\mathcal{V}}}}{\widehat{M}} + \sum_{x=1}^{d_k^{\mathcal{V}}} pq^{x-1}\left(\frac{(x-1)}{\widehat{M}} + \frac{d_k^{\mathcal{V}}-x}{\widehat{M}+\widehat{F}}\right)\right)\Big((i+1)M_{i+1} - iM_i\Big)\right] \\
&\qquad + \mathbf{E}_{d_k^{\mathcal{V}}\sim\pi_{\mathcal{V}}}\left[\sum_{x=1}^{d_k^{\mathcal{V}}} pq^{x-1}\frac{(i+1)F_{i+1}}{\widehat{F}}\right] + \varepsilon_N \\
&= \frac{1}{\widehat{M}+\widehat{F}}\mathbf{E}_{d_k^{\mathcal{V}}\sim\pi_{\mathcal{V}}}\left[\left(d_k^{\mathcal{V}}(1-p)^{d_k^{\mathcal{V}}-1} + \sum_{x=1}^{d_k^{\mathcal{V}}}\Big(pq^{x-2}(x-1) + pq^{x-1}(d_k^{\mathcal{V}}-x)\Big)\right)\Big((i+1)M_{i+1} - iM_i\Big)\right] \\
&\qquad + \frac{(i+1)h(q)F_{i+1}}{\widehat{M}+\widehat{F}} + \varepsilon_N \\
&= \frac{\mu_{\mathcal{V}}((i+1)M_{i+1} - iM_i) + (i+1)h(q)F_{i+1}}{\widehat{F}+\widehat{M}} + \varepsilon_N,
\end{aligned}$$

where ε_N satisfies, if $\widehat{M}(k) < 2N^\theta$,

$$|\varepsilon_n| \le 2\sigma_{\mathcal{V}}^2\frac{N^\theta}{N^\gamma} + 2\sigma_{\mathcal{V}}^2\frac{N^\beta}{N^\gamma} + 2\mu_{\mathcal{V}}^2\frac{N^\theta}{N^\gamma};$$

and, if $\widehat{M}(k) \ge 2N^\theta$, it satisfies

$$|\varepsilon_n| \le 3\sigma_{\mathcal{V}}^2\frac{N^\beta}{N^\gamma} + 2\mu_{\mathcal{V}}^2\frac{N^{2\beta}}{N^\theta} + \mu_{\mathcal{V}}^2\frac{N^\beta}{N^\gamma};$$

We used in the above computations (at the third equality) the following observation:

$$kq^{k-1} + \sum_{x=1}^{k} pq^{x-1}\left(\frac{x-1}{q} + k - x\right) = k$$

Summing error terms over all the $2N^\beta$ equations relating F_i to f_i and M_j to m_j, the error terms coming from the differential equation method Theorem 5.4.3, and using the fact that m is $\mu_{\mathcal{U}}$-Lipschitz, we get that the total error, defined by,

$$\mathrm{Err} := \sup_{s\in[0,1]}\left|\frac{|\mathrm{M}_T(s)|}{N} - \Big(1 - \phi_{\mathcal{U}}(1-G(s))\Big)\right|$$

satisfies

$$\mathrm{Err} \le N^\beta N^{(1+\varepsilon)\beta}(N^{\alpha+\beta} + 4(\sigma_{\mathcal{V}}^2+\mu_{\mathcal{V}}^2)N^{(1+\varepsilon)\beta-\gamma/2+\beta} + N^{2(1+\varepsilon)\beta-2}) + \mu_{\mathcal{U}}N^{(1+\varepsilon)\beta}$$

as soon as $\theta = \beta + \frac{\gamma}{2}$.

It remains to pick admissible values for the different parameters, such as the following ones (checking admissibility follows from immediate computations):

$$\beta = 1/20, \epsilon = 10, \gamma = 21/40, \theta = 25/80, \alpha.23/80$$

Those choices ensures that $\mathrm{Err} = \mathcal{O}(N^{-1/20})$.

All those arguments hold with probability at least (summing all the bad event probabilities)

$$1-T\exp(-\frac{N^{2\beta}}{2\sigma_V^2})-\exp(-\frac{N^{2\gamma-1}}{2\sigma_{\mathcal{U}}^2})-\exp(-\frac{\mu_{\mathcal{U}}N^{2\gamma-1}}{2\mu_{\mathcal{V}}\sigma_{\mathcal{V}}^2})-2N^{\beta}N^{(1+\varepsilon)\beta)}\exp(-N^{2\alpha-(1+\varepsilon)\beta}) \geq 1-\mathcal{O}(N\exp(-\zeta N^{1/40})$$

where the equality holds because of the choice of parameters.

Proof of Lemma 5.3.3

Notice that the functions f and m satisfy the following partial differential equations:

$$\partial_t f(t,s) = \frac{1}{\mu_{\mathcal{U}} - t\mu_{\mathcal{V}}}\left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}} - h(q(t))\right]\partial_s f(t,s),$$

and

$$\partial_t m(t,s) = \frac{1}{\mu_{\mathcal{U}} - t\mu_{\mathcal{V}}}\left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}}\right]\partial_s m(t,s) + h(q(t))\partial_s f(t,s),$$

where $q(t) = \partial_s f(t,1)/(\mu^{\mathcal{U}} - t\mu^{\mathcal{V}})$.

To solve these equations, we first perform a time change to get rid of the denominator. Let

$$\theta(t) = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}\left(1 - \mathrm{e}^{-\mu_{\mathcal{V}}t}\right)$$

so that $\theta'(t) = \mu_{\mathcal{U}} - \theta(t)\mu_{\mathcal{V}}$. In order to simplify notations, we set:

$$H(t) := h\left(q(\theta(t))\right).$$

Then, the new functions

$$g(t,s) := f(\theta(t),s) \qquad \text{and} \qquad o(t,s) := m(\theta(t),s)$$

satisfy the following PDEs:

$$\partial_t g(t,s) = \left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}} - H(t)\right]\partial_s g(t,s), \tag{5.8}$$

and

$$\partial_t o(t,s) = \left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}}\right]\partial_s o(t,s) + H(t)\partial_s g(t,s). \tag{5.9}$$

These two equations fall into the classical framework of *transport* differential equation and can be explicitly solved. We give the details for the reader's convenience.

Solution of (5.8)**.** Let s be a solution of the following ODE:

$$s'(t) = \mu_{\mathcal{V}}s(t) - \mu_{\mathcal{V}} + H(t). \tag{5.10}$$

Then, the function g is constant along the curve $(t, s(t))$. Indeed:

$$\frac{\mathrm{d}}{\mathrm{d}t}g(t,s(t)) = \partial_t g(t,s) + s'(t)\partial_s g(t,s) = 0.$$

The differential equation (5.10) admits the following general solutions:

$$s_c(t) = \left[c + e^{-\mu_{\mathcal{V}}t} - 1 + \int_0^t e^{-\mu_{\mathcal{V}}u}H(u)\mathrm{d}u\right]e^{\mu_{\mathcal{V}}t}.$$

Therefore,

$$(t,s) = (t, s_c(t)) \iff c = c(t,s) = (s-1)e^{-\mu_{\mathcal{V}} t} + 1 - \int_0^t e^{-\mu_{\mathcal{V}} u} H(u)\mathrm{d}u,$$

and we deduce that (the initial condition is $g(0,s) = \phi_{\mathcal{U}}(s)$):

$$g(t,s) = g(0, c(t,s)) = \phi_{\mathcal{U}}(c(t,s)) = \phi_{\mathcal{U}}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - \int_0^t e^{-\mu_{\mathcal{V}} u} H(u)\mathrm{d}u\right). \tag{5.11}$$

Solution of (5.9)**.** Let $s_\gamma(t) = \gamma e^{\mu_{\mathcal{V}} t} + 1$. Then, $s'_\gamma(t) = \mu_{\mathcal{V}} s(t) - \mu_{\mathcal{V}}$ and we deduce that, along the curves $(t, s_\gamma(t))$, $o(t,s)$ satisfies the following ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t} o(t, s_\gamma(t)) = \frac{1 - q(t)^d}{1 - q(t)} \partial_s g(t, s_\gamma(t)).$$

Since

$$(t,s) = (t, s_\gamma(t)) \iff \gamma = \gamma(t,s) = (s-1)e^{-\mu_{\mathcal{V}} t},$$

we deduce that:

$$o(t,s) = \int_0^t H(u) \partial_s g(u, (s-1)e^{-\mu_{\mathcal{V}}(t-u)} + 1)\mathrm{d}u. \tag{5.12}$$

We now define the function $F(\cdot)$ as

$$F(t) := \int_0^t \mathrm{e}^{-\mu_{\mathcal{V}} u} H(u)\mathrm{d}u. \tag{5.13}$$

Using Equations (5.11) and (5.12), one can easily deduce that

$$\partial_s g(t,1) = \mathrm{e}^{-\mu_{\mathcal{V}} t} \phi'_{\mathcal{U}}(1 - F(t))$$

and

$$\begin{aligned}\partial_s o(t,1) &= \int_0^t H(u)\, \mathrm{e}^{-\mu_{\mathcal{V}} u}\, \phi''_U(1 - F(u))\mathrm{d}u \\ &= \phi'_U(1) - \phi'_U(1 - F(t)) = \left(\mu_{\mathcal{U}} - \phi'_{\mathcal{U}}(1 - F(t))\right) \mathrm{e}^{-\mu_{\mathcal{V}} t}\,.\end{aligned}$$

In particular,

$$\partial_s g(t,1) + \partial_s o(t,1) = \partial_s f(\theta(t),1) + \partial_s m(\theta(t),1) = \mu_{\mathcal{U}}\, \mathrm{e}^{-\mu_{\mathcal{V}} t}\,.$$

Therefore,

$$H(t) = \frac{1 - \phi_{\mathcal{V}}\left(\frac{\partial_s o(t,1)}{\partial_s g(t,1) + \partial_s o(t,1)}\right)}{1 - \frac{\partial_s o(t,1)}{\partial_s g(t,1) + \partial_s o(t,1)}} = \mu_{\mathcal{U}} \frac{1 - \phi_{\mathcal{V}}\left(1 - \frac{1}{\mu_{\mathcal{U}} \phi'_{\mathcal{U}}(1 - F(t))}\right)}{1 - \phi_{\mathcal{U}}(1 - F(t))},$$

which yields the following ordinary differential equation on F:

$$\frac{\frac{1}{\mu_{\mathcal{U}}} \phi'_{\mathcal{U}}(1 - F(t))}{1 - \phi_{\mathcal{V}}\left(1 - \frac{1}{\mu_{\mathcal{U}}} \phi'_U(1 - F(t))\right)} F'(t) = \mathrm{e}^{-\mu_{\mathcal{V}} t}\,. \tag{5.14}$$

5.4.5 Proof of Theorem 5.4.1

We recall the notations introduced. For all $k \in \{0, \dots, T\}$, all $c \in \{0, \dots, C\}$ and all $i \geq 0$, we define:

- $F_i^{(c)}(k)$ the number of vertices of $\mathcal{U}$ that still have capacity c at the end of step k and whose remaining degree is i. Those vertices are referred to as *free* (with remaining degree i and capacity c at the end of step k).
- $M_i(k)$ the number of vertices of $\mathcal{U}$ that have capacity $c = 0$ at the end of step k and whose remaining degree is i. Those vertices are referred to as *marked* (with remaining degree i at the end of step k).

We also define as before the number of remaining half-edges to respectively free and marked vertices as

$$\widehat{F}(k) = \sum_{c=1}^{C} \sum_{i \geq 0} i F_i^{(c)}(k), \quad \text{and} \quad \widehat{M}(k) = \sum_{i \geq 0} i M_i(k).$$

The normalized performance of GREEDY is the ratio between the matched vertices in $\mathcal{V}$ and its maximal number, equal to CN:

$$A = \frac{\sum_{i \geq 0} \left(C M_i(T) + \sum_{c=1}^{C} (C - c) F_i^{(c)}(T) \right)}{CN}$$

As in the proof of Theorem 5.2.1:

1. we will place ourselves on the event where all vertices have bounded degrees, smaller than N^β for some small $\beta > 0$
2. we will stop the analysis at N^γ steps of the horizon T so that $\widehat{F}(k) + \widehat{M}(k) > N^\gamma$ with arbitrarily high probability
3. we will distinguish the cases where $\widehat{M}(k) > 2N^\theta$ (with $\theta = \beta + \gamma/2$)

As a consequence, the errors are going to be of the same order of magnitude with the same order of probability (up to a multiplicative factor C) (hence those computations are skipped and replace by $\mathcal{O}(\cdot)$ notations). The interesting new component in this proof is the new system of differential equations and their solutions.

The Differential equations

Using the same notations than in the proof of Theorem 5.2.1, we get that for all $0 \leq k \leq T$, $i \geq 0$ and $c \leq C$,

$$\begin{aligned}
&\mathbf{E}\Big[F_i^{(c)}(k+1) - F_i^{(c)}(k) \,\Big|\, \mathcal{F}_k \Big] \\
&\quad = \mathbf{E}_{d_k \sim \pi_{\mathcal{V}}} \left[\sum_{x=1}^{d_k} -q^{x-1} \frac{i F_i^{(c)}}{\widehat{F} + \widehat{M}} + p \sum_{x=1}^{d_k} q^{x-1} (d_k - x) \left(\frac{(i+1) F_{i+1}^{(c)} - i F_i^{(c)}}{\widehat{F} + \widehat{M}} \right) \right] \\
&\qquad\qquad + \mathbf{E}_{d_k \sim \pi_{\mathcal{V}}} \left[\sum_{x=1}^{d_k} q^{x-1} \frac{(i+1) F_{i+1}^{(c+1)}}{\widehat{F} + \widehat{M}} \right] + \mathcal{O}(N^{\theta - \gamma}) \\
&\quad = \frac{\mu_{\mathcal{V}} \left(-i F_i^{(c)} + (i+1) F_{i+1}^{(c)} \right) - (i+1) h(q) F_{i+1}^{(c)} + (i+1) h(q) F_{i+1}^{(c+1)}}{\widehat{F} + \widehat{M}} + \mathcal{O}(N^{\theta - \gamma})
\end{aligned}$$

where the function h is still defined as $h(q) = \frac{1-\phi_{\mathcal{V}}(q)}{1-q}$. Similarly, we can compute the expected increment in M_i as

$$\begin{aligned}
&\mathbf{E}\Big[M_i(k+1) - M_i(k) \,\Big|\, \mathcal{F}_k\Big] \\
&\quad = \mathbf{E}_{d_k\sim\pi_{\mathcal{V}}}\left[\left(\frac{d_k(1-p)^{d_k}}{\widehat{M}} + \sum_{x=1}^{d_k} pq^{x-1}\left(\frac{(x-1)}{\widehat{M}} + \frac{d_k - x}{\widehat{M}+\widehat{F}}\right)\right)\left((i+1)M_{i+1} - iM_i\right)\right] \\
&\qquad\qquad + \mathbf{E}_{d_k\sim\pi_{\mathcal{V}}}\left[\sum_{x=1}^{d_k} pq^{x-1}\frac{(i+1)F_{i+1}^{(1)}}{\widehat{F}}\right] + \mathcal{O}(N^{\theta-\gamma}) \\
&\quad = \frac{\mu_{\mathcal{V}}((i+1)M_{i+1} - iM_i) + (i+1)h(q)F_{i+1}^{(1)}}{\widehat{F}+\widehat{M}} + \mathcal{O}(N^{\theta-\gamma})
\end{aligned}$$

From this, we get the following system of differential equations:

$$\partial_t f^{(c)}(t,s) = \frac{1}{\mu_{\mathcal{U}} - t\mu_{\mathcal{V}}}\left[\left(-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}} - h(q(t))\right)\partial_s f^{(c)}(t,s) + \frac{1}{\mu_{\mathcal{U}} - t\mu_{\mathcal{V}}}h(q(t))\partial_s f^{(c+1)}(t,s)\right], \tag{5.15}$$

and

$$\partial_t m(t,s) = \frac{1}{\mu_{\mathcal{U}} - t\mu_{\mathcal{V}}}\left[\left(-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}}\right)\partial_s m(t,s) + h(q(t))\partial_s f^{(1)}(t,s)\right] \tag{5.16}$$

With those notations, the normalized performances of GREEDY rewrite then into:

$$\begin{aligned}
A &= m(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}, 1) + \sum_{c=1}^{C}(1 - \frac{c}{C})f^{(c)}(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}, 1) \\
&= 1 - \sum_{c=1}^{C}\frac{c}{C}f^{(c)}(\frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}, 1)
\end{aligned}$$

Solving the PDEs

As in the previous section, we start with a time change. Let

$$\theta(t) = \frac{\mu_{\mathcal{U}}}{\mu_{\mathcal{V}}}\left(1 - \mathrm{e}^{-\mu_{\mathcal{V}}t}\right) \tag{5.17}$$

so that $\theta'(t) = \mu_{\mathcal{U}} - \theta(t)\mu_{\mathcal{V}}$. In order to simplify notations, we set:

$$H(t) := h\left(q(\theta(t))\right). \tag{5.18}$$

Then, the new functions

$$g^{(c)}(t,s) := f^{(c)}(\theta(t), s) \qquad \text{and} \qquad o(t,s) := m(\theta(t), s)$$

satisfy the following PDEs:

$$\partial_t g^{(c)}(t,s) = \left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}} - H(t)\right]\partial_s g^{(c)}(t,s) + H(t)\partial_s g^{(c+1)}(t,s),$$

and

$$\partial_t o(t,s) = \left[-\mu_{\mathcal{V}}s + \mu_{\mathcal{V}}\right]\partial_s o(t,s) + H(t)\partial_s g^{(1)}(t,s). \tag{5.19}$$

We distinguish:

$$\partial_t g^{(C)}(t,s) = \left[-\mu_{\mathcal{V}} s + \mu_{\mathcal{V}} - H(t)\right] \partial_s g^{(C)}(t,s) \tag{5.20}$$

We define:

$$F(t) = \int_0^t e^{-\mu_{\mathcal{V}} u} H(u) \mathrm{d}u$$

Solution of (5.20)**.** This equation is the same as the one satisfied by $g(t,s)$, with the same initial conditions. Thus, we can write:

$$g^{(C)}(t,s) = \phi_{\mathcal{U}}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)\right).$$

Solution of (5.20)**.** Lets define the curves:

$$s_{t,s}(u) = \left[(s-1)e^{-\mu_{\mathcal{V}} t} - F(t) + F(u)\right] e^{\mu_{\mathcal{V}} u} + 1.$$

Along those curves, we have:

$$\frac{\mathrm{d}}{\mathrm{d}t} g^{(c)}(u, s_{t,s}(u)) = H(u) \partial_s g^{(c+1)}(u, s_{t,s}(u)).$$

So:

$$g^{(c)}(t,s) = \int_0^t H(u) \partial_s g^{(c+1)}(u, s_{t,s}(u)) \mathrm{d}u$$

Solution for $c = C-1$. We have:

$$\begin{aligned} g^{(c-1)}(t,s) &= \int_0^t H(u) \partial_s g^{(C)}(u, s_{t,s}(u)) \mathrm{d}u \\ &= \int_0^t F'(u) \phi_U'((s-1)e^{-\mu_{\mathcal{V}} u} + 1 - F(u)) \mathrm{d}u \\ &= F(t) \phi_U'((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)) \end{aligned}$$

Solution for $c = C-k$, general formula. We will prove by induction:

$$g^{(C-k)}(t,s) = \frac{1}{k!} (F(t))^k \phi^{(k)}((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t))$$

If it is true for rank k, we have:

$$\partial_s g^{(C-k)}(u, s_{t,s}(u)) = \frac{e^{-\mu_{\mathcal{V}} u}}{k!} (F(u))^k \phi^{(k+1)}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)\right)$$

Which gives:

$$\begin{aligned} g^{(C-(k+1))}(t,s) &= \frac{1}{k!} \left(\int_0^t F'(u) (F(u))^k \mathrm{d}u \right) \phi^{(k+1)}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)\right) \\ &= \frac{1}{(k+1)!} (F(t))^{k+1} \phi^{(k+1)}((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)) \end{aligned}$$

Solution of (5.19)**.** Let's define the curves:

$$\gamma_{s,t}(u) = 1 + (s-1)e^{-\mu_{\mathcal{V}}(t-u)}$$

Along those curves:

$$\frac{\mathrm{d}}{\mathrm{d}u}o(u,\gamma_{t,s}(u)) = H(u)\partial_s g^{(1)}(u,\gamma_{t,s}(u))$$

So:

$$o(t,s) = \int_0^t F'(u)\frac{(F(u))^{(C-1)}}{(C-1)!}\phi^{(C)}\left((s-1)e^{-\mu_{\mathcal{V}}t}+1-F(u)\right)\mathrm{d}u$$

Formula for greedy performances. Recall that the normalized performances of GREEDY are

$$\begin{aligned} A &= 1-\sum_{c=1}^{C}\frac{c}{C}g^{(c)}(+\infty,1) \\ &= 1-\sum_{k=0}^{C-1}\frac{1-\frac{k}{C}}{k!}(F(+\infty))^k\phi^{(k)}\left(1-F(+\infty)\right) \end{aligned}$$

ODE for F

We have as before:

$$F'(t) = H(t)e^{-\mu_{\mathcal{V}}t},\ H(t) = \frac{1-\phi_{\mathcal{V}}(Q(t))}{1-Q(t)}$$

And we also have:

$$Q(t) = \frac{\partial_s o(t,1)}{\partial_s o(t,1)+\sum_{c=1}^{C}\partial_s g^{(c)}(t,1)}$$

According to the previous section :

$$\begin{aligned} \partial_s o(t,1) &= \left(\int_0^t F'(u)\frac{(F(u))^{(C-1)}}{(C-1)!}\phi_{\mathcal{U}}^{(C+1)}\left(1-F(u)\right)\mathrm{d}u\right)e^{-\mu_{\mathcal{V}}t} \\ &= \left(\int_0^{F(t)}\frac{x^{(C-1)}}{(C-1)!}\phi_{\mathcal{U}}^{(C+1)}\left(1-x\right)\mathrm{d}x\right)e^{-\mu_{\mathcal{V}}t} \\ &= \left[\phi_U'(1)-\phi_U'(1-F(t))-\sum_{k=1}^{C-1}\frac{F(t)^K}{k!}\phi_{\mathcal{U}}^{(k+1)}(1-F(t))\right]e^{-\mu_{\mathcal{V}}t} \end{aligned}$$

Which gives:

$$Q(t) = 1-\frac{1}{\mu_{\mathcal{U}}}\left(\phi_U'(1-F(t))+\sum_{k=1}^{C-1}\frac{F(t)^k}{k!}\phi_{\mathcal{U}}^{(k+1)}(1-F(t))\right)$$

We define:

$$\Gamma_U(F(t)) = \frac{1}{\mu_{\mathcal{U}}}\left(\phi_U'(1-F(t))+\sum_{k=1}^{C-1}\frac{F(t)^k}{k!}\phi_{\mathcal{U}}^{(k+1)}(1-F(t))\right)$$

This yields the following differential equation for F:

$$\frac{\Gamma_U(F(t))}{1-\phi_{\mathcal{V}}\left(1-\Gamma_U(F(t))\right)}F'(t) = \mathrm{e}^{-\mu_{\mathcal{V}}t}\,.$$

Theorem 5.4.1 then follows from the same arguments in the proof of Theorem 5.2.1 (except that errors are C times bigger as there are C more equations to handle).

5.4.6 Proof of Theorem 5.4.2

As mentioned in the main text, the only difference with Theorem 5.4.1 is that C could be of the order of N^β (but not bigger on the event where all degrees are smaller than N^β). As a consequence, one must take β even smaller than $1/20$ to have sublinear errors terms (choosing $\beta = 1/40$ is admissible for instance) with exponentially high probability.

Solution of (5.20)**.** This equation is the same as the one satisfied by $g(t,s)$, the new initial condition is $g^{(C)}(t,s) = p_C \phi_{\mathcal{U}}(s)$. Thus, we can write:

$$g^{(C)}(t,s) = p_C \phi_{\mathcal{U}}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)\right).$$

Solution for $c = C-1$. We have:

$$\begin{aligned} g^{(c-1)}(t,s) &= \int_0^t H(u)\partial_s g^{(C)}(u, s_{t,s}(u))\mathrm{d}u + g^{(c-1)}(0, s_{t,s}(0)) \\ &= \int_0^t F'(u)\phi_U'((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t))\mathrm{d}u + p_{(C-1)}\phi_{\mathcal{U}}((s-1)e^{-\mu_{\mathcal{V}} t} - F(t) + 1) \\ &= p_C F(t)\phi_U'((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)) + p_{(C-1)}\phi_{\mathcal{U}}((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)) \end{aligned}$$

Solution for $c = C-2$.

$$\partial_s g^{(C-2)}(u, s_{t,s}(u)) = p_C e^{-\mu_{\mathcal{V}} u} F(u)\phi_{\mathcal{U}}''\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)\right) + p_{(C-1)}e^{-\mu_{\mathcal{V}} u}\phi_{\mathcal{U}}'(s_{t,s}(u))$$

Let's define:

$$c(t,s) = (s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(t)$$

Which gives:

$$\begin{aligned} g^{(C-2)}(t,s) =& p_{(C-1)}\left(\int_0^t F'(u)F(u)\mathrm{d}u\right)\phi^{''}\left(c(t,s)\right) \\ &+ p_{(C-1)}\left(\int_0^t F'(u)e^{\mu_{\mathcal{V}} u}\mathrm{d}u\right)\phi^{'}\left(c(t,s)\right)\mathrm{d}u + p_{(C-2)}\phi_{\mathcal{U}}(s_{t,s}(0)) \\ =& \frac{p_C}{2}(F(t))^2\phi^{''}((s-1)e^{-\mu_{\mathcal{V}} t} \\ &+ 1 - F(t)) + p_{(C-1)}F(t)\phi^{'}((s-1)e^{-\mu_{\mathcal{V}} t} + p_{(C-2)}\phi_{\mathcal{U}}(c(t,s)) \end{aligned}$$

Solution for $c = C-k$, general formula. We will prove by induction:

$$g^{(C-k)}(t,s) = \sum_{l=0}^{k} p_{C-l}\frac{1}{(k-l)!}(F(t))^{k-l}\phi^{(k-l)}(c(t,s))$$

If it is true for rank k, we have:

$$\partial_s g^{(C-k)}(u, s_{t,s}(u)) = \sum_{l=0}^{k} p_{C-l}\frac{1}{(k-l)!}(F(t))^{k-l}e^{-\mu_{\mathcal{V}} u}\phi^{(k+1-l)}(c(t,s))$$

Which gives:

$$g^{(C-(k+1))}(t,s) = p_{(C-(k+1))}\phi_{\mathcal{U}}(c(t,s)) + \sum_{l=0}^{k} p_{(C-l)} \frac{1}{(k-l)!} \left(\int_0^t F'(u)(F(u))^{(k-l)} \mathrm{d}u \right) \phi^{(k+1-l)}\left(c(t,s)\right)$$
$$= \sum_{l=0}^{k+1} p_{(C-l)} \frac{1}{(k+1-l)!} (F(t))^{k+1-l} \phi^{(k+1-l)}(c(t,s))$$

Solution of (5.19)**.**

$$o(t,s) = \sum_{c=1}^{C} p_c \int_0^t F'(u) \frac{(F(u))^{(c-1)}}{(c-1)!} \phi^{(c)}\left((s-1)e^{-\mu_{\mathcal{V}} t} + 1 - F(u)\right) \mathrm{d}u$$

$$g^{(c)}(t,s) = \sum_{k=0}^{C-c} p_{c+k} \frac{1}{k!} (F(t))^k \phi^k(c(t,s))$$

Quantity of interest.

$$A = \frac{\mu_{\mathcal{V}}}{\mu_{\mathcal{U}}} \sum_{c=1}^{C} c(p_c - g^{(c)}(+\infty, 1))$$
$$= \frac{\mu_{\mathcal{V}}}{\mu_{\mathcal{U}}} \left(\sum_{c=1}^{C} c p_c - \sum_{k=0}^{C-1} \left(\frac{1}{k!} (F(+\infty))^k \phi^{(k)}\left(1 - F(+\infty)\right) \sum_{c=1}^{C} c p_{c+k} \right) \right)$$

ODE for the function F.

$$\partial_s o(t,1) = \left(\sum_{c=1}^{C} p_c \int_0^t F'(u) \frac{(F(u))^{(c-1)}}{(c-1)!} \phi_{\mathcal{U}}^{(c+1)}\left(1 - F(u)\right) \mathrm{d}u \right) e^{-\mu_{\mathcal{V}} t}$$
$$= \left[\phi_U'(1) - \phi_U'(1 - F(t)) + \sum_{k=1}^{C-1} \left(\frac{F(t)^k}{k!} \phi_{\mathcal{U}}^{(k+1)}(1 - F(t)) \sum_{c=k+1}^{C} p_c \right) \right] e^{-\mu_{\mathcal{V}} t}$$

Which yields:

$$Q(t) = 1 - \frac{1}{\mu_{\mathcal{U}}} \left(\phi_U'(1 - F(t)) + \sum_{k=1}^{C-1} \left(\frac{F(t)^k}{k!} \phi_{\mathcal{U}}^{(k+1)}(1 - F(t)) \sum_{c=k+1}^{C} p_c \right) \right)$$

We define:

$$\Gamma_U(F(t)) = \frac{1}{\mu_{\mathcal{U}}} \left(\phi_U'(1 - F(t)) + \sum_{k=1}^{C-1} \left(\frac{F(t)^k}{k!} \phi_{\mathcal{U}}^{(k+1)}(1 - F(t)) \sum_{c=k+1}^{C} p_c \right) \right)$$

This yields the following differential equation for F:

$$\frac{\Gamma_U(F(t))}{1 - \phi_{\mathcal{V}}\left(1 - \Gamma_U(F(t))\right)} F'(t) = \mathrm{e}^{-\mu_{\mathcal{V}} t}\,.$$

5.4.7 Proof of Theorem 5.2.2

Lemma 5.4.4. *On the 2-regular graph, the law of the matches generated by the algorithm Ranking equals the law of the matches generated by a biased Greedy algorithm, that chooses a free vertex of degree 2 over one of degree 1 with probability at least 2/3. This is biased as the classical Greedy algorithm chooses it with probability 1/2.*

Proof: Two vertices of the same degree are interchangeable, they are both equally likely to have the smallest rank. Thus Ranking and Greedy behave the same on arriving vertices whose potential neighbors all have the same degree. Let $r(v)$ be the rank of vertex v and $deg(v)$ its residual number of unpaired half-edges.

Let $\mathcal{A}_t(u,2)$ be the following event:

- u had one of its half-edges paired to the incoming vertex v_t at iteration t,
- u was not matched and v_t was instead matched to a vertex of residual degree 2.

Let $\mathcal{A}_t(u,1)$ be the similar event with v_t instead matched to a vertex of residual degree 1.

$$\begin{aligned}\mathbf{P}\left(r(u)\geq k|\deg(u)=1,\, u \text{ free at } t\right) = &\sum_{t'<t}\mathbf{P}\left(r(u)\geq k|\mathcal{A}_{t'}(u,2)\right)\mathbf{P}(\mathcal{A}_{t'}(u,2)|\deg(u)=1,\, u \text{ free at } t)\\ &+\sum_{t'<t}\mathbf{P}\left(r(u)\geq k|\mathcal{A}_{t'}(u,1)\right)\mathbf{P}(\mathcal{A}_{t'}(u,1)|\deg(u)=1,\, u \text{ free at } t).\end{aligned}$$

Now, assume

$$\forall t'<t, \forall a\in[N], \mathbf{P}(r(a)\geq k|\deg(a)=1 \text{ and } a \text{ free at } t)\geq \mathbf{P}(r(a)\geq k|\deg(a)=2). \quad (5.21)$$

Hypothesis 5.21 implies:

$$\forall t'<t, \mathbf{P}(r(u)\geq k|\mathcal{A}_{t'}(u,1))\geq \mathbf{P}(r(u)\geq k|\mathcal{A}_{t'}(u,2)),$$

thus, the following inequality holds:

$$\mathbf{P}(r(u)\geq k|\deg(u)=1 \text{ and } u \text{ free at } t)\geq \mathbf{P}(r(u)\geq k|\cup_{t'<t}\mathcal{A}_{t'}(u,2)).$$

Let a and b be two different numbers randomly chosen in $[n]$. Two vertices with two remaining half-edges were not affected by the run before, and thus could have any rank. It therefore holds:

$$\begin{aligned}\mathbf{P}\left(r(u)\geq k|\deg(u)=1 \text{ and } u \text{ free at } t\right)\geq &\mathbf{P}\left(\max(a,b)\geq k\right)\\ =&1-\frac{\binom{k-1}{2}}{\binom{n}{2}}\\ =&1-\frac{(k-1)(k-2)}{n(n-1)}.\end{aligned}$$

This inequality implies $\mathbf{P}\left(r(u)\geq k|\deg(u)=1 \text{ and } u \text{ free at } t\right)\geq \mathbf{P}\left(r(u)\geq k|\deg(u)=2\right)$, thus 5.21 is true by induction.

Therefore, vertices with only one remaining half-edges are likely to have a higher rank than

those with two remaining half-edges:

$$\begin{aligned}\mathbf{P}(r(b) < r(a) | \deg(b) = 2, \deg(a) = 1) &= \sum_{k=1}^{n} \mathbf{P}(\deg = k-1, \deg(a) \geq k | \deg(b) = 2, \deg(a) = 1) \\ &\geq 1 - \sum_{k=1}^{n} \frac{(k-1)(k-2)}{n^2(n-1)} \\ &\geq \frac{2}{3} + O\left(\frac{1}{n}\right).\end{aligned}$$

□

Let $M_1^G(t)$ and $M_1^R(t)$ be the number of marked vertices of degree 1 by GREEDY and RANKING algorithms respectively. Note that the number of vertices of degree 2 is the same for both algorithms, $F_2^G(t) = F_2^R(t)$. Also, the following always holds

$$F_1^G(t) = 2N - 2t - 2F_2^G(t) - M_1^G(t).$$

This shows that in 2-regular graphs, the number of half-edges in each ensemble is a deterministic quantity of $M_1^R(t+1)$ and $M_1^G(t+1)$.

Suppose it holds at time t that $M_1^G(t) = M_1^R(t) = M_1(t)$ (event $\mathcal{A}$), then

$$\begin{aligned}\mathbb{E}[M_1^R(t+1)|\mathcal{A}] - \mathbb{E}[M_1^G(t+1)|\mathcal{A}] =& \mathbb{E}[\mathbb{1}_{\{\text{RANKING marks a vertex in } F_2^R(t)\}}|\mathcal{A}] \\ &- \mathbb{E}[\mathbb{1}_{\{\text{GREEDY marks a vertex in } F_2^R(t)\}}|\mathcal{A}] \\ =& \frac{1}{6} \cdot \frac{F_1(t)F_2(t)}{2(N-t)} > 0.\end{aligned}$$

The first equality holds since the probability of pairing an half-edge in $M_1^R(t+1)$ and $M_1^G(t+1)$ only depends on the number of half edges in each ensembles, not on the algorithm.

Therefore, by application of Gronwald's lemma, RANKING generates strictly more marked vertices of degree 1. As the probability that an incoming vertice is matched only to non-available vertices increases with M_1, RANKING performs strictly worse than GREEDY on 2-regular graphs.

Chapter 6

Online Matching in Geometric Random Graphs

In online advertisement, ad campaigns are sequentially displayed to users. Both users and campaigns have inherent features, and the former is eligible to the latter if they are "similar enough". We model these interactions as a bipartite geometric random graph: the features of the $2N$ vertices (N users and N campaigns) are drawn independently in a metric space and an edge is present between a campaign and a user node if the distance between their features is smaller than c/N, where $c > 0$ is the parameter of the model.

Our contributions are two-fold. In the one-dimensional case, with uniform distribution over the segment $[0, 1]$, we derive the size of the optimal offline matching in these bi-partite random geometric graphs, and we build an algorithm achieving it (as a benchmark), and analyze precisely its performance.

We then turn to the online setting where one side of the graph is known at the beginning while the other part is revealed sequentially. We study the number of matches of the online algorithm CLOSEST, which matches any incoming point to its closest available neighbor. We show that its performances can be compared to its fluid limit, completely described as a solution of an explicit PDE. From the latter, we can compute the competitive ratio of CLOSEST.

Contents

In online advertisement, ad campaigns are sequentially displayed to users. Both users and campaigns have inherent features, and the former is eligible to the latter if they are "similar enough". We model these interactions as a bipartite geometric random graph: the features of the $2N$ vertices (N users and N campaigns) are drawn independently in a metric space and an edge is present between a campaign and a user node if the distance between their features is smaller than c/N, where $c > 0$ is the parameter of the model.

Our contributions are two-fold. In the one-dimensional case, with uniform distribution over the segment $[0, 1]$, we derive the size of the optimal offline matching in these bi-partite random geometric graphs, and we build an algorithm achieving it (as a benchmark), and analyze precisely its performance.

We then turn to the online setting where one side of the graph is known at the beginning while the other part is revealed sequentially. We study the number of matches of the online algorithm CLOSEST, which matches any incoming point to its closest available neighbor. We show that its performances can be compared to its fluid limit, completely described as a solution of an explicit PDE. From the latter, we can compute the competitive ratio of CLOSEST.

6.1 Introduction

Online matching is motivated, among others, by its application to ad allocation on the internet. Advertising platforms are faced with the following issue: there are on the platform several companies launching their campaigns. These have two constraints, a budget constraint, the maximum number of ads they are willing to pay for, and a targeting constraint, which describes the population they wish to show ads to, based on age, location, etc... From the point of view of the platform, this is a matching problem, where the goal is to maximize the number of valid ad-user allocations whilst respecting the budget constraint. However, users do not arrive all at once, they rather generate web pages sequentially as they surf the web. An ad has to be allocated on the spot to the generated slot, which is otherwise lost. The interaction can be formulated as an online matching problem.

Formally, consider the bipartite graph $G(\mathcal{U}, \mathcal{V}, \mathcal{E})$, with set of vertices $\mathcal{U} \cup \mathcal{V}$ and set of edges $\mathcal{E} \in \mathcal{U} \times \mathcal{V}$. In our context of online matching, the "offline" vertices $\mathcal{U}$ represent the side of the companies, they are present from the beginning, while the vertices $\mathcal{V}$, which represent the users, are revealed sequentially. At each arrival of a vertex $v \in \mathcal{V}$, the edges adjacent to vertex v are revealed. Based on those edges, the vertex can then be matched to a previously unmatched neighbor, and this decision cannot be revoked. The excellent survey Mehta (2012) details applications, results, and techniques of online matching.

Those types of online problems are evaluated through competitive analysis Borodin and El-Yaniv (2005). In the context of online matching, the measure of performance of an algorithm is its competitive ratio, that is the ratio of the size of the matching computed by the algorithm divided by the size of the best matching in hindsight Feldman et al. (2009).

A first line of work studied online matching in the adversarial framework, where the algorithm is evaluated on the worse possible instance with the worst possible arrival order of the vertices. It is folklore that greedy random algorithms, which match incoming vertices to any available neighbor at random have a competitive ratio of $1/2$ in the worst case. However, they achieve $1 - 1/e$ as soon as the incoming vertices arrive in Random Order Goel and Mehta (2008). The RANKING algorithm is the worst-case optimal, it achieves at least $1 - 1/e$ on any instance Birnbaum and Mathieu (2008); Devanur et al. (2013); Karp et al. (1990), and also has a higher competitive ratio in the Random Order setting Mahdian and Yan (2011).

Beyond this worse-case setting, the following stochastic model has been introduced. It is assumed there exists a probability distribution over types of vertices, and at every iteration, the incoming vertex is drawn i.i.d. from that distribution. With the knowledge of that distribution, algorithms with much better competitive ratios than RANKING were designed Brubach et al. (2019); Jaillet and Lu (2014); Manshadi et al. (2012). If the expected number of arrival of each type is integral, the designed algorithm achieves a competitive ratio of $1 - 2/e^2 \approx 0,729$ and 0.706 with that assumption removed. Both those competitive ratios also hold with Poisson arrival rates.

This stochastic model fits some situations and is certainly interesting, but too restrictive to model the aforementioned practical motivating examples. The first criticism is that it fails to handle simple random structures, such as a bipartite Erdos-Renyi random graph; embedding it into the above stochastic model would require an exponential number of types with respect to the number of vertices. As a consequence, another line of work considers standard online algorithms on some classes of random graphs, representing situations where some properties of the underlying graph are known. The seminal example would be online matching in Erdos-Renyi graphs Mastin and Jaillet (2013), or more generally in the configuration model that specifies a law on the degrees of the vertices Noiry et al. (2021). The idea behind the latter is that one can "estimate" the typical attractivity of a campaign (say, some of them target a large number of users while others are more selective).

Unfortunately, these approaches fail to model the following simple fact. Campaigns that are "similar" tend to target the same users, and vice-versa (for instance, luxury products will target users with high incomes, while baby products obviously target families). These interactions can be modeled by assuming that both ads and users are represented by some feature vectors and that an edge is present between two vertices if the features are similar enough. Random geometric graphs are suited to that representation. Vertices are designed by locations in a Euclidean space and an edge is drawn between two vertices if those are at a distance smaller than a threshold (this is an arbitrary choice of the kernel).

As a consequence, we shall in the following introduce and analyze the online matching problem for geometric graphs; for simplicity, we shall focus on the already challenging and interesting one-dimensional geometric graph (the multi-dimensional one would be interesting for future work).

6.1.1 Further Related work

Besides those already mentioned in the introduction, a vast line of works generalizes the online matching problem.

It has been proposed to add capacities to the vertices of the offline side, as well as weights to the edges. The problem in this setting is referred to as Adwords. In the setting with small bids, the competitive ratio of greedy is $1 - 1/e$ Mehta et al. (2007). In the case where all budgets are equal to b and all bids to 1, which is referred to as b-matching, a competitive ratio of $1 - 1/(1 + 1/b)^b$ was obtained for algorithm BALANCE Kalyanasundaram and Pruhs (2000). Under the small bid and i.i.d. arrivals assumption, the same algorithm is $1 - O(1)$ competitive Motwani et al. (2006).

Some works consider the case of edge arrivals instead of vertex arrivals. In the adversarial setting, it has been shown that no algorithm beats the classical greedy algorithm and that the competitive ratio is 1/2 Gamlath et al. (2019). With stochastic arrivals, a slightly better competitive ratio of 0.503 was obtained Gravin et al. (2019).

A line of works relaxes the assumption that the online vertices arrive one by one. For instance, in two-stage matching problems, where the vertices of the online stage arrive in two

batches Feng et al. (2021). The arriving vertices may also be assigned a patience parameter Brubach et al. (2021).

Another related approach is stochastic metric matching. In that context, vertices are also points in a euclidean space. Any vertices on opposite sides of the partition may be matched together, and the cost of the matching is the total distance between the matched points Akbarpour et al. (2021); Gupta et al. (2019).

6.1.2 Bi-partite 1D geometric random graphs

We consider the bipartite geometric graph $\text{Geom}(\mathcal{X}, \mathcal{Y}, c)$, whose set of vertices $\mathcal{X}$ and $\mathcal{Y}$ are two sets of N points drawn independently and uniformly in $[0, 1]$,

$$\mathcal{X} = (x_i)_{i\in[N]} \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0,1] \quad \text{and} \quad \mathcal{Y} = (y_i)_{i\in[N]} \overset{\text{i.i.d.}}{\sim} \mathcal{U}[0,1].$$

There is an edge between $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$ if and only if

$$|x_i - y_j| < \frac{c}{N}.$$

This choice of parametrization ensures that the expected degree of a vertex remains bounded, of order c (neglecting the boundary effects). In particular, the graph remains sparse and the online matching problem is not trivial.

6.1.3 Contributions

First, we derive the size of the maximum matching in this geometric graph, as a function of the parameter c, along with the description of an algorithm constructing this matching. We provide a non-asymptotic convergence bound of the size of the maximum matching to $c/(c + 1/2)$. This bound is obtained through the study of the algorithm, via the construction of a potential function which is then treated as a random walk. More precisely, we shall prove the following

Theorem 6.1.1 (informal)**.** *Let $m^*(\mathcal{X}, \mathcal{Y}, c)$ be the size of the maximum matching in Geom$(\mathcal{X}, \mathcal{Y}, c)$. With probability at least $1 - O\left(\frac{1}{N}\right)$,*

$$m^*(\mathcal{X}, \mathcal{Y}, c) = \frac{c}{c + \frac{1}{2}} N + O\left(\sqrt{N \ln(N)}\right).$$

This informal result is illustrated in Figure 6.1 which shows both the theoretical asymptotic value and the optimal value in several realizations of random graphs for a variety of parameters c.

We then study the size of the matching constructed by an online algorithm, CLOSEST, which matches any incoming vertex to its closest available neighbor. We show the convergence of this quantity to the solution of an explicit PDE. We do that by exhibiting tractable quantities that can be approximated via the Differential Equation Method Enriquez et al. (2019).

More precisely, we shall prove the following Theorem.

Theorem 6.1.2 (informal)**.** *Let $\kappa(c, N)$ be the size of the matching obtained by CLOSEST algorithm on $G(\mathcal{X}, \mathcal{Y}, c)$, then*

$$\kappa(c, N) \xrightarrow[N\to+\infty]{\mathbb{P}} 1 - \int_0^{+\infty} f(x, t)dx$$

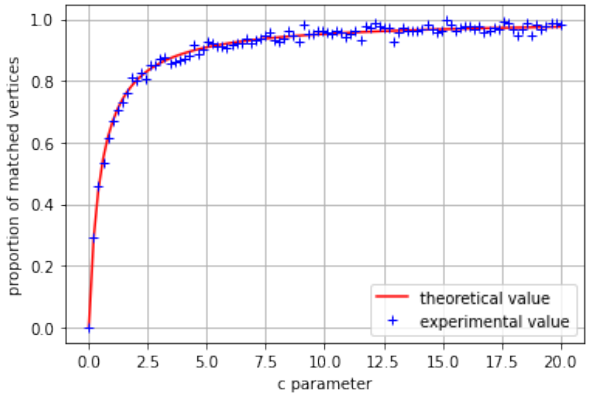


Figure 6.1: The asymptotic optimal offline matching size is displayed in red, as a function of the parameter c. Several simulations (blue crosses) for different values of c and for $N = 100$ vertices illustrate that this limit is reached rapidly.

where f is the solution of some explicit PDE, described later in Equation (6.2).

The difference between the theoretical and the actual sizes of the matchings (as a function of the number of online vertices observed) for different values of N is illustrated in Figure 6.2.

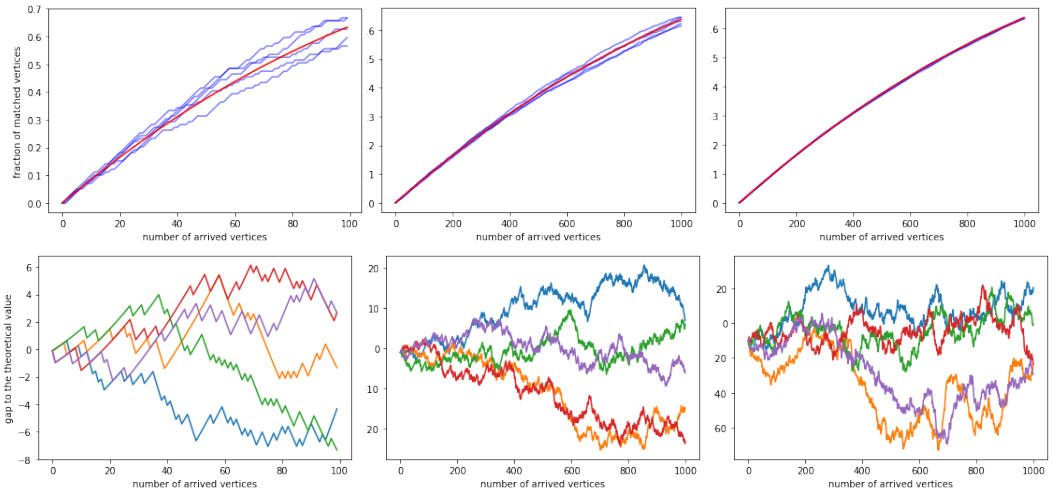


Figure 6.2: First row, from left to right: theoretical (red line) vs. experimental (blue lines) sizes of the online matching in the 1D uniform geometric graph ($c = 1$) as a function of the number of arrived vertices, for $N = 100$, $N = 1.000$ and $N = 10.000$. In the second row, we plot the difference between the above blue and red curves; notice the change in Y-axis: in the first row, it is the average matching size (that is, normalized by N), while on the second row, it is the absolute difference in matching sizes (without normalization by N).

6.1.4 Organization of the chapter

The remaining is organized as follows. The main conceptual contributions are in the two following subsections, followed by some simulations. The remaining sections are more technical.

Section 6.2: This section is dedicated to the offline case. We give the asymptotic formula of the size of the optimal offline matching, given the full bipartite geometric graph. Interestingly,

our approach is algorithmic: we provide a way to construct this optimal matching, and we analyze the latter by carefully studying some random walk.

Section 6.3: This section focuses on the online case. We describe a simple algorithm, CLOSEST, that matches any incoming vertex to the closest available one (as would do a greedy procedure) and we characterize the size of the matching it creates by studying its fluid limit. We shall prove it satisfies some PDE.

Section 6.4: This is the first section dedicated to technical results. This one concerns the offline case, and we study in detail the random walk introduced in Section 6.2.

Section 6.5: This section and the following two, are dedicated to the technicalities of the online case. In Section 6.3, we shall mention that we can "round" the geometric graph to consider more malleable random objects (Poisson processes among others); more precisely, we claim that we can consider the same problem on a circle rather than on a segment. The purpose of this section is to prove such claims.

Section 6.6: In this section, we prove another claim of Section 6.3. Roughly speaking, we said that the distribution of the distances between two successive vertices is “rotation independent” (recall that we embedded the problem on the circle in the previous section) and even “permutation independent”.

Section 6.7: In this final section, apart from the Appendix that recalls standard results, we explain why the fluid limit result, which approximates the behavior of a random discrete process by a deterministic and continuous solution of some PDE, holds.

6.2 Maximum Matching in 1D Uniform Geometric Graph

This section is dedicated to the offline case, where the whole underlying bi-partite 1D geometric graph is known from the beginning. The size of the maximum matching is obtained by analyzing an optimal matching algorithm that computes it. We call it the SMALL-FIRST algorithm (illustrated in Figure Section 6.2), as it iteratively matches the unmatched vertices with the smallest coordinates.

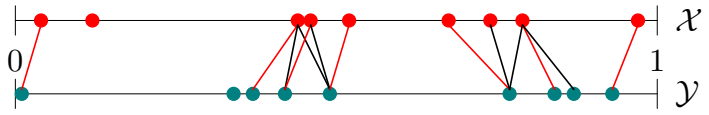


Figure 6.3: Matching constructed with the SMALL-FIRST algorithm

Proposition 6.2.1. *Algorithm* SMALL-FIRST *returns an optimal matching in* 1*D geometric graphs.*

Proof: The proof actually shows that the algorithm constructs a matching with no augmenting path. An augmenting path is a path in the graph starting at an unmatched vertex, then alternating between unmatched and matched edges, then ending at an unmatched vertex. Note that if such a path is found, the size of the matching may be increased by including the two endpoints in the matching and changing the matched edges of the path for the unmatched ones.

By Berge's theorem, a matching is optimal iif it presents no augmenting path Berge (1957). By construction, two edges in the matching m returned by Algorithm SMALL-FIRST cannot cross, that is if $(x_i, y_j), (x_{i'}, y_{j'}) \in m$, then $x_i \leq x_{i'}$ if and only if $y_j \leq y_{j'}$.

Assume that the matching m is not maximal. Then, by Berge's theorem, it admits at least an augmenting path. Consider an augmenting path with minimum length and, without loss of generality, assume that it starts at a point in $\mathcal{X}$. We denote the sequence of consecutive edges in the path by

$$\{(x_{i_1}, y_{j_1}), (y_{j_1}, x_{i_2}), \ldots, (x_{i_\ell}, y_{j_\ell})\},$$

where x_{i_1} and y_{j_ℓ} are not matched and $(y_{j_k}, x_{i_{k+1}}) \in m$, for any $k < \ell$. Note that $\ell \geq 3$, as an augmenting path is always of odd length and $\ell = 1$ would imply that two neighbors are not matched in the optimal matching, which is impossible by definition of optimality.

Since the vertex x_{i_2} is matched by SMALL-FIRST to y_{i_1} while x_{i_1}, which is also a neighbor y_{i_1}, is not matched, then, by definition of SMALL-FIRST:

$$x_{i_1} \geq x_{i_2}.$$

If there were an edge between x_{i_1} and y_{j_2}, the path could be shortened, so

$$y_{j_1} \geq y_{j_2}.$$

Since two edges in m constructed by SMALL-FIRST cannot cross, and $(x_{i_2}, y_{i_1}), (x_{i_3}, y_{i_2})$ belong to m thus

$$x_{i_2} \geq x_{i_3}.$$

With the same arguments, we obtain by induction that for every $k < l$,

$$y_{j_k} \geq y_{j_{k+1}} \text{ and } x_{i_k} \geq x_{i_{k+1}}.$$

In particular, this applies to $k = \ell - 1$, so that $y_{j_\ell} < y_{j_{\ell-1}}$. However, this is impossible as $y_{j_{\ell-1}}$ is matched to x_{i_ℓ}, while there is an edge between x_{i_ℓ} and y_{j_ℓ} and the latter is unmatched by SMALL-FIRST. This is an obvious contradiction as SMALL-FIRST would have matched x_{i_ℓ} to y_{j_ℓ} rather than to $y_{j_\ell - 1}$.

The conclusion is that SMALL-FIRST creates a matching without any augmenting path, which is thus optimal. □

Using the optimality of SMALL-FIRST, we can now prove the first main result.

Proposition 6.2.2. *Let $\mathcal{X}$ and $\mathcal{Y}$ be two independent sets of N identically and uniformly distributed points in $[0,1]$and $M^*(c,N)$ be the expected size of the maximum matching in Geom$(\mathcal{X}, \mathcal{Y}, c)$. Then the following holds:*

$$\lim_{N\to\infty} \frac{M^*(c,N)}{N} = \frac{c}{c+\frac{1}{2}}.$$

Let $m^*(\mathcal{X}, \mathcal{Y}, c)$ be the realized size of the maximum matching in Geom$(\mathcal{X}, \mathcal{Y}, c)$. Then, with probability at least $1 - O\left(\frac{1}{N}\right)$, the following holds:

$$m^*(\mathcal{X}, \mathcal{Y}, c) = \frac{c}{c+\frac{1}{2}} N + O\left(\sqrt{N \ln(N)}\right).$$

Proof: The proof is decomposed into two steps. In the first step, we show that the size of the maximum matching in Geom$(\mathcal{X}, \mathcal{Y}, c)$ is related to the size of the maximum matching in a

geometric graph with a set of vertices generated by Poisson point processes. In the second step, the asymptotic size of the maximum matching in those graphs is derived through the study of a random walk.

Step 1, connection with Poisson point processes. Let $\Phi_{\mathcal{U}}^N$ be an independent homogeneous Poisson point process on the segment $[0,1]$ of intensity N. The definition and some standard properties of Poisson Point Processes (PPP) are reported in Appendix 6.A.1 for the sake of completeness. Let $\mathcal{U}^N \sim \Phi^N$ and $\mathcal{V}^N \sim \Phi^N$ denote two independent PPP, and $\gamma^*(c,N)$ be the expected size of the maximum matching in $\mathrm{Geom}(\mathcal{U}^N, \mathcal{V}^N, c)$.

Lemma 6.2.3. *Under the above notations, the following holds:*

$$|\gamma^*(c,N) - M^*(c,N)| \leq 4(1+\sqrt{N \ln N}).$$

Proof: Let $N_{\mathcal{U}} = |\mathcal{U}|$ and $N_{\mathcal{V}} = |\mathcal{V}|$ denote the size of the vertex sets. By Chernoff bound:

$$P\left\{|N_{\mathcal{U}} - N| \geq 2\sqrt{N \ln N}\right\} \leq \frac{2}{N} \tag{6.1}$$

and the same holds for $N_{\mathcal{V}}$. We now explain how recover the construction of $\mathrm{Geom}(\mathcal{X}, \mathcal{Y}, c)$ from those two PPP (as they do not have the same vertex set sizes). From $\mathcal{U}$, define a new set $\tilde{\mathcal{U}}$ of N vertices as follows. If $N_{\mathcal{U}} > N$, delete uniformly at random $N_{\mathcal{U}} - N$ points from $\mathcal{U}^N$. If $N > N_{\mathcal{U}}$, add $N - N_{\mathcal{U}}$ points independently and uniformly distributed in $[0,1]$ to $\mathcal{U}^N$. The set $\tilde{\mathcal{V}}$ is constructed from $\mathcal{V}$ similarly. This construction ensures that $\mathrm{Geom}(\tilde{\mathcal{U}}, \tilde{\mathcal{V}}, c)$ and $\mathrm{Geom}(\mathcal{X}, \mathcal{Y}, c)$ have the same law. Moreover, the transformation affects the size of the matching at most by the number of added and removed points. □

Step 2, deriving $\gamma^*(c,N)$. A possible way to draw sets $\mathcal{U}^N$ from ϕ^N is through a renewal process with exponential holding times. Precisely, let $\{F_k\}_{k\in\mathbb{N}}$ be a sequence of independent, identically distributed exponential random variables of parameter N. Ensemble $\mathcal{U}^N$ is defined as:

$$\mathcal{U}^N = \left\{ u_k = \sum_{i=1}^{k} F_i \text{ for } k \text{ s.t. } \sum_{i=1}^{k} F_i < 1 \right\}.$$

The same holds from $\mathcal{V}^N$ (see Appendix 6.A.1).

To compute the size of the maximum matching in a given graph, we introduce a modified version of Algorithm SMALL-FIRST, that generates the graph together with a matching. Algorithm SMALL-FIRST-GENERATIVE (Algorithm 18) proceeds as follows. The positions of the first points in $\mathcal{U}^N$ and $\mathcal{V}^N$ are drawn independently from two exponential distributions of parameter N. At iteration t, note $u_{x(t)}$ and $v_{x(t)}$ the position of the last generated points in $\mathcal{U}^N$ and $\mathcal{V}^N$ respectively. Define the potential

$$\psi(t) := u_{x(t)} - v_{x(t)}.$$

Until all the points on one side have been drawn, the following operations are iteratively performed:

- if $|\psi(t)| < \frac{c}{N}$, edge $(u_{x(t)}, v_{x(t)})$ is added to the matching and the next points on both sides of the graph are generated.
- if $\psi(t) > c$, the next point in $\mathcal{V}^N$ is generated.
- if $\psi(t) < -c$, the next point in $\mathcal{U}^N$ is generated.

Algorithm 28: SMALL-FIRST-GENERATIVE

input : N, c;
Draw $u_1 \sim \text{Exp}(N)$ and $v_1 \sim \text{Exp}(N)$;
Initialize $x(1) \leftarrow 1$, $y(1) \leftarrow 1$ and $m \leftarrow \emptyset$;
Define for $t = 1, \ldots$, $\psi(t) \leftarrow u_{x(t)} - v_{x(t)}$;
while $u_{x(t)} < 1$ *and* $v_{x(t)} < 1$ **do**
  if $|\psi(t)| < c$ **then**
    $m \leftarrow m \cup (u_{x(t)}, v_{x(t)})$;
    $x(t+1) \leftarrow x(t) + 1$ and $y(t+1) \leftarrow y(t) + 1$;
    Draw $u_{x(t+1)} - u_{x(t)} \sim \text{Exp}(N)$ and $v_{y(t+1)} - v_{y(t)} \sim \text{Exp}(N)$;
  end
  if $\psi(t) > c$ **then**
    $x(t+1) \leftarrow x(t)$ and $y(t+1) \leftarrow y(t) + 1$;
    Draw $v_{y(t+1)} - v_{y(t)} \sim \text{Exp}(N)$;
  end
  if $\psi(t) < -c$ **then**
    $x(t+1) \leftarrow x(t) + 1$ and $y(t+1) \leftarrow y(t)$;
    Draw $u_{x(t+1)} - u_{x(t)} \sim \text{Exp}(N)$
  end
end

The potential function $\psi(t)$ is a Markov chain with the following transition probability:

$$\psi(t+1) - \psi(t) = \begin{cases} x_t - y_t \text{ if } |\psi(t)| \leq c \\ x_t \text{ if } \psi(t) \leq -c \\ -y_t \text{ if } \psi(t) \geq c \end{cases}$$

with x_t and y_t independent exponential random variables of parameter N. The rest of the proof consists in studying this random walk, its stationary distribution, and its convergence. Those technical details are postponed to Section 6.4.

□

6.3 Match to the closest point algorithm

In this section, we study the performances on the 1D Geometric Graph of the online matching algorithm CLOSEST, that matches the incoming vertex to its closest available neighbor if there is one. The following theorem states that w.h.p., $\kappa(c, N)$, the size of the matching obtained by CLOSEST algorithm on $G(\mathcal{X}, \mathcal{Y}, c)$, is closely related to the solution of an explicit PDE.

Theorem 6.3.1. *Let $\kappa[c, N](t)$ be the size of the matching obtained by* CLOSEST *on $G(\mathcal{X}, \mathcal{Y}, c)$ when t vertices have arrived, then*

$$\kappa[c, N](t) \xrightarrow[N \to +\infty]{\mathbb{P}} 1 - \int_0^{+\infty} f(x, t) dx$$

with $f(x,t)$ the solution of the following differential equation

$$\frac{\partial f(x,t)}{\partial t} = -\min(x,2c)f(x,t) - \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{+\infty}\min(x',2c)f(x',t)dx' f(x,t) + \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{x}\min(x',2c)f(x',t)f(x-x',t)dx' \tag{6.2}$$

with the following initial conditions

$$f(x,0) = e^{-x}.$$

Proof structure: We first show that the score of CLOSEST in $G(\mathcal{X},\mathcal{Y},c)$ is closely related to its score in $G(\tilde{\mathcal{U}},\mathcal{Y},c)$, where the vertices of the offline side are generated through a Poisson point process and have their coordinates rounded to a discrete grid (Section 6.3.1). We then show that the score of CLOSEST on the modified graph is closely related to the solution of a PDE through the differential equation method Enriquez et al. (2019)(Section 6.3.2). □

Remark: It holds that

$$\frac{\partial}{\partial t}\int_x f(x,t)dx = -\int_x \min(x,2c)f(x,t)dx,$$

which rewrites as

$$\frac{\partial}{\partial t}\int_x f(x,t)dx = -2c\int_x f(x,t)dx + \int_0^{2c} xf(x,t)dx.$$

The second term gets negligible in front of the first one as c goes to 0; yet it is non-negative, which implies that $\int_x f(x,t)dx \geq \exp(-2ct)$ and thus the size of the matching obtained by CLOSEST is smaller than $1-\exp(-2ct)$. This is not really helpful for the analysis, as we aim at lower-bounding the performance of an algorithm, but it provides insights on the approximate performance for small values of c.

6.3.1 Graph rounding

Let $\mathcal{X}$ be an ensemble of N points drawn i.i.d. uniformly in $[0,1]$, and $\mathcal{Y}$ be an independent ensemble of N points also drawn i.i.d. uniformly in $[0,1]$, with the vertices in the ensemble indexed according to the order in which they are drawn. We define $\mathrm{MC}(G(\mathcal{X},\mathcal{Y},c))$ as the number of matched vertices by CLOSEST in the graph $G(\mathcal{X},\mathcal{Y},c)$ when the vertices in $\mathcal{Y}$ arrived in the order prescribed by their indexes.

The GRAPH-ROUNDING procedure, illustrated in Figure 6.4, associates to an ensemble $\mathcal{X}$ the rounded ensemble $\tilde{\mathcal{U}}$ through the following steps:

- **Poissonization step:** Let $N_0 \sim \mathrm{Poi}(N)$. If $N_0 > N$, then let $(u_i)_{i=1}^{N_0-N}$ be $N_0 - N$ points drawn uniformly and independently in $[0,1]$ and define $\mathcal{U} = \mathcal{X} \cup \{u_i \mid i \in [N_0 - N]\}$. If $N_0 < N$, $\mathcal{U}$ is obtained be removing $N - N_0$ points selected uniformly at random from $\mathcal{X}$. This is the same procedure used in Section 6.2.

- **Rounding step:** Transform $\mathcal{U}$ in a new ensemble $\lfloor\mathcal{U}\rfloor$ by rounding the coordinate of each point $u \in \mathcal{U}$ to $\frac{\lfloor uNk\rfloor}{Nk}$,

$$\lfloor\mathcal{U}\rfloor := \left\{ \frac{\lfloor uNk\rfloor}{Nk} \;\middle|\; u \in \mathcal{U} \right\}.$$

- **Discarding step:** For any $\ell \in [Nk]$, if multiple vertices have their coordinates rounded to $\frac{\ell}{Nk}$, a random vertex among those is selected, and all others are removed from the graph. This gives the final ensemble $\tilde{\mathcal{U}}$.
- **Gluing step:** The interval $[0,1]$ is mapped to the unit circle of circumference one. Formally, the distance is replaced with the following one:

$$d(x,y) = \min(|x-y|, |x+1-y|).$$

  We also add a vertex at coordinate 0 to $\tilde{\mathcal{U}}$ if it is not already in it.

This final step produces graph $G_{glued}\left(\tilde{\mathcal{U}}, \mathcal{Y}, c, N, k\right)$ as illustrated in Figure 6.4. Note that after

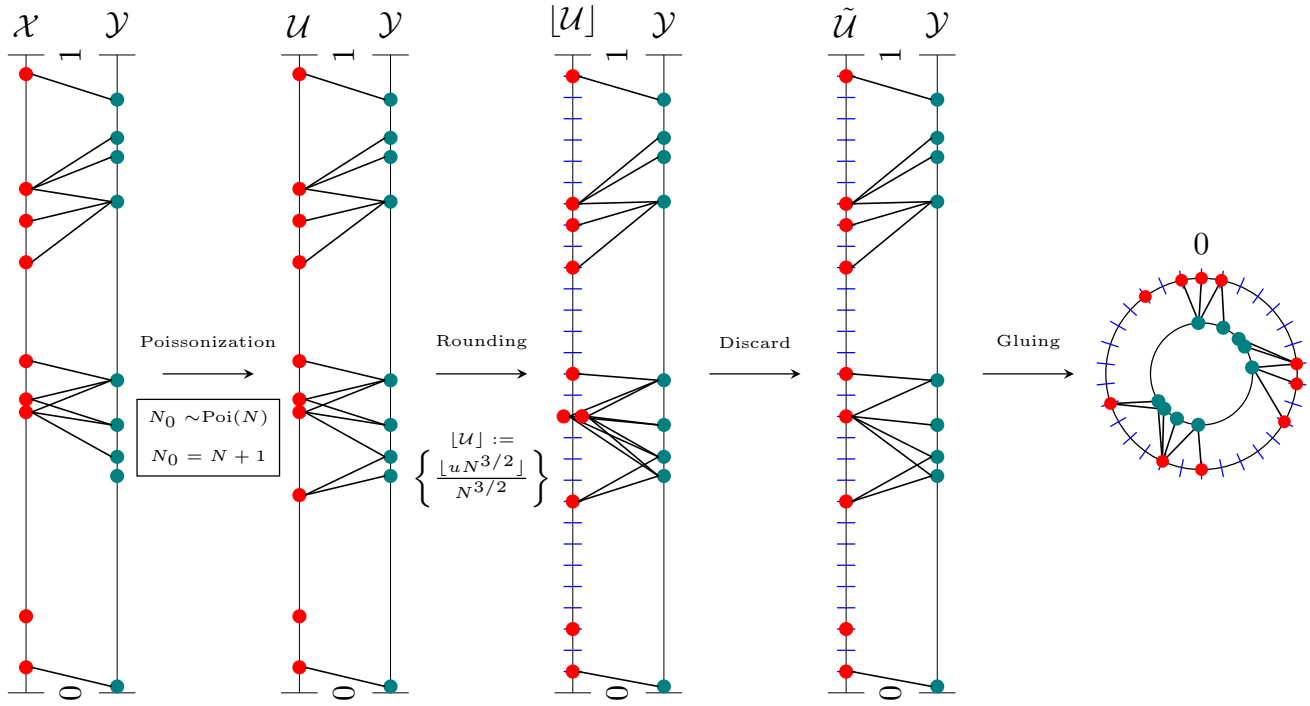


Figure 6.4: Graph Rounding

the discarding step, the law of $\mathcal{U}$ is that of an ensemble of points drawn from a Poisson point process of intensity N in $[0,1]$. Let $u_\ell = \ell/(Nk)$. For each $\ell \in [Nk]$, $\mathbb{P}(u_\ell \in \tilde{\mathcal{U}}) = 1 - e^{-1/k}$, and the events $\left(\{u_\ell \in \tilde{\mathcal{U}}\}\right)_{\ell \in [Nk]}$ are independent of each other.

The following proposition states that this construction does not impact too much the size of the matchings.

Proposition 6.3.2. *With probability at least* $1 - 8\exp(-\min(\sqrt{c}/5, 1/4)\sqrt{N})$*, we have:*

$$\left| MC\left(G\left(\mathcal{X}, \mathcal{Y}, c, N\right)\right) - MC\left(G_{glued}\left(\tilde{\mathcal{U}}, \mathcal{Y}, c, N, k\right)\right) \right| \leq 49c\frac{N}{k} + 5N^{3/4}.$$

Proof: The proposition is a consequence of the four following lemmas that bound the impact in matching sizes incurred in each step.

Lemma 6.3.3 (Poissonization)**.** *With probability at least* $1 - 2\exp(-\sqrt{N}/4)$*, it holds:*

$$\left| MC\left(G\left(\mathcal{X}, \mathcal{Y}, c, N\right)\right) - MC\left(G\left(\mathcal{U}, \mathcal{Y}, c, N\right)\right) \right| \leq N^{3/4}.$$

Lemma 6.3.4 (Rounding). *With probability at least* $1 - 2\exp(-\min(\sqrt{c}/5, 2)\sqrt{N}) - \exp(-N)$, *it holds:*

$$\left| MC\Big(G\left(\lfloor \mathcal{U} \rfloor, \mathcal{Y}, c, N\right)\Big) - MC\left(G\left(\mathcal{U}, \mathcal{Y}, c, N, k\right)\right) \right| \leq 48c\frac{N}{k} + 2\sqrt{N}.$$

Lemma 6.3.5 (Discard). *With probability at least* $1 - 3\exp(-\sqrt{N}/4)$, *it holds:*

$$\left| MC\Big(G\left(\tilde{\mathcal{U}}, \mathcal{Y}, c, N\right)\Big) - MC\Big(G\left(\lfloor \mathcal{U} \rfloor, \mathcal{Y}, c, N, k\right)\Big) \right| \leq \frac{N}{k} + 2N^{3/4}.$$

Lemma 6.3.6 (Gluing). *With probability at least* $1 - \exp(-\sqrt{N})$, *it holds:*

$$\left| MC\Big(G\left(\tilde{\mathcal{U}}, \mathcal{Y}, c, N\right)\Big) - MC\Big(G_{glued}\left(\tilde{\mathcal{U}}, \mathcal{Y}, c, N, k\right)\Big) \right| \leq 2\sqrt{N} + 1.$$

The proofs of these technical lemmas are postponed to Section 6.5. □

6.3.2 Analysis of closest on the modified graph

This section is dedicated to the analysis of the score of CLOSEST on the graphs $G_{\text{glued}}(\tilde{\mathcal{U}}, \mathcal{Y}, c, N, k)$.

Let N_t be the number of free vertices at iteration t. At time t, let $u_t(i)$ be the coordinate of the i^{th} free vertex, with the vertices enumerated according to their coordinates, and the convention $u_t(N_t + 1) = u_t(1)$. For $\ell \in [Nk]$, define

$$F_{k,N}(\ell, t) := \left| \left\{ i \in [N_t] \text{ s.t. } N\left(u_t(i+1) - u_t(i)\right) = \frac{\ell}{k} \right\} \right|,$$

Note that at iteration t, the number of free vertices is equal to $\sum_{l \in [N^{3/2}]} F_N(\ell, t)$. Define also

$$M_{k,N}(\ell_-, \ell_+, t) := \left| \left\{ i \in [N_t] \text{ s.t. } N\left(u_t(i) - u_t(i-1)\right) = \frac{\ell_-}{k} \text{ and } N\left(u_t(i+1) - u_t(i)\right) = \frac{\ell_+}{k} \right\} \right|.$$

Let $\mathcal{F}_t$ be the σ containing information of t the values of the sizes of the gaps up to time t, $\left(F_{k,N}\left(\ell, t'\right)\right)_{\ell, t' \leq t}$. The following lemma describes how many expected times two gaps of a certain size follow each other at iteration t, conditionally on $\mathcal{F}_t$.

Lemma 6.3.7 (Gaps repartition). *For all* $t \in [N]$, *for all* $\ell_-, \ell_+ \in (N^{3/2})^2$,

$$\mathrm{E}\left[M_{k,N}(\ell_-, \ell_+, t) \middle| \mathcal{F}_t \right] = F_{k,N}(\ell_-, t) \left(\mathbb{1}_{\{l_- \neq \ell_+\}} \frac{F_{k,N}(\ell_+, t)}{N_t - 1} + \mathbb{1}_{\{\ell_- = \ell_+\}} \frac{F_{k,N}(\ell_+, t) - 1}{N_t - 1} \right).$$

Sketch of Proof: This lemma is implied by a stronger result: conditionally on $\mathcal{F}_t$, the gaps ordering is uniformly random. This is proved by induction in Section 6.6 □

Note that the expression on the right-hand side is exactly the expectation obtained by drawing uniformly without replacements two successive gaps among all gaps at time t.

This lemma entails explicit computation of the expected evolution of the gaps, still conditionally on $\mathcal{F}_t$.

Lemma 6.3.8 (Evolution law)**.** *For all $t \in [N]$, for all $l \in \mathbf{N}$,*

$$N\mathrm{E}\left[F_{k,N}(\ell,t+1)-F_{k,N}(\ell,t)\middle|\mathcal{F}_t\right] = -\left(\min(2c,\ell/k)+\sum_{\ell'}\min(2c,\ell'/k)\frac{F_{k,N}(\ell',t)-\mathbb{1}_{\ell=\ell'}}{N_t-1}\right)F_{k,N}(\ell,t)$$
$$+\sum_{\ell'=0}^{l}\min(2c,\ell'/k)\frac{F_{k,N}(\ell',t)\left(F_{k,N}(\ell-\ell',t)-\mathbb{1}_{\{\ell-\ell'=\ell'\}}\right)}{N_t-1}.$$

Proof: At every iteration, at most one vertex u_t is matched. If $u_t \in M_{k,N}(\ell_-,\ell_+,t)$, then

- $F_{k,N}(\ell_-,t+1) = F_{k,N}(\ell_-,t)-1$,
- $F_{k,N}(\ell_+,t+1) = F_{k,N}(\ell_+,t)-1$,
- $F_{k,N}(\ell_++\ell_-,t+1) = F_{k,N}(\ell_++\ell_-,t)+1$.

At every iteration,

$$\mathbb{P}\left(v_t \text{ matched to } u_t \in M_N(\ell_-,\ell_+,t)\right) =: m(\ell_-,\ell_+,t)$$
$$= \frac{\min(c,\frac{\ell_-}{2k})+\min(c,\frac{\ell_+}{2k})}{N}.$$

Furthermore, the following chain of equalities holds:

$$\begin{aligned}
\mathrm{E}\left[F_{k,N}(\ell,t+1)-F_{k,N}(\ell,t)\middle|\mathcal{F}_t\right] &= -\sum_{\ell'} m(\ell,\ell',t)M_{k,N}(\ell,\ell',t)+m(\ell',\ell,t)M_{k,N}(\ell',\ell,t)\\
&\quad+\sum_{\ell'\le\ell} m(\ell',\ell-\ell',t)M_{k,N}(\ell',\ell-\ell',t)\\
&= -2\sum_{\ell'}\frac{\min(c,\frac{\ell}{2k})+\min(c,\frac{\ell'}{2k})}{N}M_{k,N}(\ell,\ell',t)\\
&\quad+2\sum_{\ell'\le\ell}\frac{\min(c,\frac{\ell'}{2k})}{N}M_{k,N}(\ell',\ell-\ell',t)\\
&= -\frac{\min(2c,\ell/k)}{N}F_{k,N}(\ell,t)-F_{k,N}(\ell,t)\sum_{\ell'}\frac{\min(2c,\ell'/k)}{N}\frac{F_{k,N}(\ell',t)-\mathbb{1}_{\ell=\ell'}}{N_t-1}\\
&\quad+\sum_{\ell'=0}^{\ell}\frac{\min(2c,\ell'/k)}{N}\frac{F_{k,N}(\ell',t)(F_{k,N}(\ell-\ell',t)-\mathbb{1}_{\{\ell-\ell'=\ell'\}})}{N_t-1}.
\end{aligned}$$

□

The proof of the previous lemma, along with other technical ones, is given in Section 6.7. We can now apply the Differential Equation Method, yielding the following result.

Lemma 6.3.9. *For any $t \in [0,1]$,*

$$\frac{1}{N}\sum_{\ell=0}^{kN}F_{k,N}\left(\ell,\frac{\lfloor tN\rfloor}{N}\right)\xrightarrow[N\to+\infty]{\mathbb{P}}\sum_{\ell=0}^{+\infty}f_k(\ell,t).$$

where $f_k(\ell,t)$ are the solutions of the following system of differential equations:

$$\frac{\partial f_k(\ell,t)}{\partial t} = -\min(\frac{\ell}{k},2c)f_k(\ell,t) - \frac{1}{\sum_{\ell'=0}^{+\infty} f_k(\ell',t)}\left[\sum_{\ell'=0}^{+\infty}\min(\frac{\ell}{k},2c)f_k(\ell',t)\right]f_k(\ell,t)$$
$$+\frac{1}{\sum_{\ell'=0}^{+\infty} f_k(\ell',t)}\sum_{\ell'=0}^{+\infty}\min(\frac{\ell'}{k},2c)f_k(\ell',t)f_k(\ell-\ell',t),$$

with the initial conditions:

$$f_k(\ell,0) = k(1-e^{-\frac{1}{k}})^2 e^{-\frac{\ell}{k}}.$$

The proof of this lemma can also be found in Section 6.7.

The last step of the proof is to link the functions $f_k(\ell,t)$ to a single function f, independent from N.

Lemma 6.3.10. *For any $t\in[0,1]$, it holds:*

$$||\ f(x,t) - kf_k(\lfloor kx\rfloor,t)\ ||_{L_1} \leq \frac{\omega}{k}.$$

with ω a constant depending only on c and $f(x,t)$ the solution of the following PDE

$$\frac{\partial f(x,t)}{\partial t} = -\min(x,2c)f(x,t) - \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{+\infty}\min(x',2c)f(x',t)dx' f(x,t) \qquad (6.3)$$
$$+\frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{x}\min(x',2c)f(x',t)f(x-x',t)dx'.$$

with initial conditions

$$f(x,0) = e^{-x}.$$

The proof of this lemma is postponed to Section 6.7.

6.4 Study of the Random Walk

In this section, we study in detail the aforementioned random walk, from which we can compute the performances of the optimal matching.

Lemma 6.4.1. *The Markov chain described in Section 6.2 admits the following stationary distribution:*

$$\mu(t) = \begin{cases} \frac{1}{2c+2} \text{ if } |x|\leq c \\ \frac{e^{x+c}}{2c+2} \text{ if } x\leq -c \\ \frac{e^{-(x-c)}}{2c+2} \text{ if } x\geq c. \end{cases}$$

Proof: Let us denote by Π the transition kernel of random walk ψ; then for any $x\in[-c,c]$:

$$(2c+2)\int_{-\infty}^{+\infty}\Pi(x,y)\mu(y)dy = \int_{-\infty}^{-c} e^{-(x-y)}e^{y+c}dy + \int_c^{+\infty} e^{-(y-x)}e^{-y+c}dy$$
$$+\int_{-c}^{x}\frac{1}{2}e^{-(x-y)}dy + \int_x^c\frac{1}{2}e^{x-y}dy$$
$$=1.$$

Similarly, for any $x \leq -c$:

$$\begin{aligned}(2c+2)\int_{-\infty}^{+\infty}\Pi(x,y)\mu(y)dy =& \int_{-\infty}^{x} e^{-(x-y)}e^{y+c}dy + \int_{c}^{+\infty} e^{-(y-x)}e^{-y+c}dy \\ &+ \int_{-c}^{c}\frac{1}{2}e^{x-y}dy \\ =& e^{x+c}.\end{aligned}$$

By symmetry, the computation also holds $\forall x \geq c$. □

Let τ_N number of iterations in Algorithm SMALL-FIRST-GENERATIVE run with input parameter N, and

$$p_N := \frac{1}{\tau_N}\sum_{t=1}^{\tau_N}\mathbb{1}_{\{|\psi(t)\leq c|\}}.$$

By Equation (6.1), with probability at least $1 - 2/N$,

$$\tau_N \geq N - 2\sqrt{N\ln N}.$$

Thus, by the ergodic theorem, the following convergence holds:

$$\gamma^*(c,N) = \mathbb{E}\left[\frac{2\sum_{t=1}^{\tau_N}\mathbb{1}_{\{|\psi(t)\leq c|\}}}{2\sum_{t=1}^{\tau_N}\mathbb{1}_{\{|\psi(t)\leq c|\}} + \sum_{t=1}^{\tau_N}\mathbb{1}_{\{|\psi(t)\geq c|\}}}\right] \xrightarrow{N\to+\infty} \frac{2\mu(|x|>c)}{2\mu(|x|>c)+\mu(|x|<c)} = \frac{c}{c+\frac{1}{2}}.$$

We can also derive high probability bounds. Indeed, for any $y \in \mathbf{R}$,

$$\int_{-\infty}^{+\infty}\Pi(x,y)\mu(y)dy \leq \frac{1}{2}$$

and for any $x \in \mathbf{R}$,

$$\int_{-\infty}^{+\infty}\Pi(x,y)\mu(x)dx \leq \frac{3}{4}.$$

Thus by Schur test's lemma, the operator norm of the kernel is bounded as $||\Pi||_\mu \leq \sqrt{3/8}$. By a version of Hoeffding's inequality adapted to Markov chains Miasojedow (2014), we get

$$\mathbb{P}\left(|p_N - \mu(|x|<c)| \geq \epsilon \middle| \tau_N\right) \leq 4(c+1)e^{-\frac{\epsilon^2\tau_N}{5}}$$

Moreover, by Chernoff bound, with probability at least $1 - e^{-\frac{N}{4}}$,

$$\tau_N \geq \frac{N}{2}.$$

Thus, the following holds:

$$\mathbb{P}\left(|p_N - \mu(|x|<c)| \geq \epsilon \middle| \tau_N\right) \leq 5(c+1)e^{-\frac{\epsilon^2 N}{8}}.$$

This implies that with probability of at least $1 - O(\frac{1}{N})$,

$$m^*\left(\mathcal{U}^N/N, \mathcal{V}^N/N, c/N\right) = \frac{c}{c+\frac{1}{2}}N + O\left(\sqrt{N\ln(N)}\right).$$

Combining this with Equation (6.1) ends the proof of Lemma 6.2.2.

6.5 Proof of the auxiliary Lemmas for graph-rounding

Proof of Lemma 6.3.3 (Poissonisation) The proof unfolds in two steps: we first prove that adding or removing a vertex to the ensemble of offline vertices modifies the score of the algorithm by at most one. We then show that w.h.p. the **Poissonization step** adds or removes a small number of vertices.

Lemma 6.5.1. *Adding or removing a vertex to the offline side in the graph modifies the score of* CLOSEST *by at most one.*

Proof: Let us compare the runs of the CLOSEST algorithm in

$$G(\mathcal{X}, \mathcal{Y}, c/N) \text{ and } G(\mathcal{X} \cup x_0^+, \mathcal{Y}, c/N)$$

when the vertices in $\mathcal{Y}$ arrive in the same order. Let m_t and m_t^+ be the number of matched vertices at iteration t, and $\mathcal{X}_t$ and $\mathcal{X}_t^+$ the sets of free vertices at iteration t, in $G(\mathcal{X}, \mathcal{Y}, c/N)$ and $G(\mathcal{X} \cup x_0^+, \mathcal{Y}, c/N)$ respectively. We will show by induction that at every iteration, one of the following properties holds:

- (P1): $m_t = m_t^+$ and $\exists\ x_t^+ \in \mathcal{X} \cup x_0^+$ s.t. $\mathcal{X}_t^+ = \mathcal{X}_t \cup x_t^+$,
- (P2): $m_t + 1 = m_t^+$ and $\mathcal{X}_t^+ = \mathcal{X}_t$.

If (P2) is true at some iteration t, it remains true until the end of the run and the proof is over. If (P1) is true at iteration t, the following cases are possible:

1. the incoming vertex y_t has no neighbor in $\mathcal{X}_t^+$, in which case it is unmatched in both graphs,
2. y_t's closest neighbor in $\mathcal{X}_t^+$ is not x_t^+, in which case y_t is matched to the same vertex in both graphs,
3. y_t's closest neighbor in $\mathcal{X}_t^+$ is x_t^+ and x_{t+1}^+ in $\mathcal{X}_t$, in which case it is matched to x_t^+ in $G(\mathcal{X} \cup x_0^+, \mathcal{Y}, c/N)$ and to x_{t+1}^+ in $G(\mathcal{X}, \mathcal{Y}, c/N)$,
4. y_t's only neighbor in $\mathcal{X}_t^+$ is x_t^+, in which case it is matched to x_t^+ in $G(\mathcal{X} \cup x_0^+, \mathcal{Y}, c/N)$ and unmatched in $G(\mathcal{X}, -\mathcal{Y}, c/N)$.

Cases 1 to 3 imply that (P1) remains true at iteration $t+1$, case 4 implies that (P2) is true at iteration $t+1$. (P1) is true at iteration 0, thus either (P1) or (P2) hold at iteration N. □

Now, by concentration of Poisson random variables,

$$\begin{aligned}\mathbb{P}\left(|N - N_0| \geq N^{3/4}\right) \leq & 2e^{-\frac{N^{3/2}}{2(N+N^{3/4})}} \\ \leq & 2e^{-\frac{N^{1/2}}{4}}\end{aligned}$$

Thus, with probability at least $1 - 2e^{-\frac{N^{1/2}}{4}}$, less than $N^{3/4}$ vertices are added or removed at the **Poissonization step**. Applying Lemma 6.5.1 terminates the proof.

□

Proof of Lemma 6.3.4 (Rounding) The CLOSEST algorithm can also be run through the following process:

1. Assign to each vertex $y \in \mathcal{Y}$ a priority list l_y containing y's neighbors ranked according to their distance to y,

2. Match the incoming vertex y to its lowest-ranked available neighbor in l_y.

Lemma 6.5.2. *Modifying priority list l_y for some $y \in \mathcal{Y}$ affects the number of matched vertices by the CLOSEST algorithm by at most 2.*

Proof: Let us compare the runs of the algorithm on the graphs with the original list l_y and the modified one l'_y. Until vertex y arrives, the run is the same on both graphs. If y is matched differently in both graphs, the set of free vertices in the two graphs after y has been matched may differ by at most two vertices. Lemma 6.5.1 terminates the proof. □

A point $y \in \mathcal{Y}$'s priority list is modified if:

- it gains or loses at least one neighbor after the rounding of the coordinates, which implies:

$$\exists u \in \mathcal{U} \text{ s.t.} |y-u| \in [c - \frac{1}{Nk}; c + \frac{1}{Nk}],$$

- two or more vertices in its neighbors' list switch places, which implies

$$\exists u, u' \in l_y \text{ s.t.} \Big| |y-u| - |y-u'| \Big| \in [0; c + \frac{2}{Nk}].$$

Define

$$p_1 = \int_0^1 \mathbb{1}_{\{\exists u \in \mathcal{U} \text{ s.t. } |u-y| \in [c-\frac{1}{Nk}, c+\frac{1}{Nk}]\}} dy$$

and

$$p_2 = \int_0^1 \mathbb{1}_{\{\exists u,u' \in \mathcal{U} \text{ s.t. } \Big||y-u|-|y-u'|\Big| \in [0; c+\frac{2}{Nk}]\}} dy.$$

With $N_0 = |\mathcal{U}|$, we have $p_1 \leq \frac{4c}{Nk} N_0$. By concentration of poisson random variables,

$$\begin{aligned} \mathbb{P}\left(p_1 \geq \frac{8c}{k}\right) =& \mathbb{P}\left(N_0 \geq 2N\right) \\ \leq& \exp(-N). \end{aligned} \tag{6.4}$$

We also have:

$$p_2 \leq \frac{2}{Nk} \Big| \{u, u' \in \mathcal{U} \text{ s.t. } |u-u'| \leq \frac{4c}{N}\} \Big| \tag{6.5}$$

Note that $|\{u, u' \in \mathcal{U} \text{ s.t. } |u-u'| \leq \frac{4c}{N}\}|$ is exactly the number of edges in a random geometric graph generated by a Poisson point process of intensity N in $[0,1]$, where two vertices x, y are connected if $|x-y| \leq \frac{4c}{N}$. This number is known to concentrate around its expectation (Bachmann and Peccati (2015)), which is:

$$\mathbf{E}\left[\Big| \{u, u' \in \mathcal{U} \text{ s.t. } |u-u'| \leq \frac{4c}{N}\} \Big|\right] = N4c - 8c^2$$

Following the upper tail bound on this number obtained in section 6.2 of Bachmann and Peccati (2015), we have:

$$\mathbb{P}(p_2 \geq \frac{16c}{k}) \leq \exp(-\frac{\sqrt{cN}}{5}).$$

Let

$$\mathcal{B} = \left\{ p_1 + p_2 \leq \frac{24c}{k} \right\}.$$

The points in $\mathcal{Y}$ are N points i.i.d. uniformly in $[0,1]$, thus the events $(\{l_y \text{ is modified} \,|\, \mathcal{U}, \mathcal{B}\})_{y\in\mathcal{Y}}$ are independent, and we also have:

$$\mathbb{P}\left(\{l_y \text{ is modified} \,\middle|\, \mathcal{U}, \mathcal{B}\} \right) \leq \frac{24c}{k}.$$

Let

$$N_{\text{modified}} := \sum_{y\in\mathcal{Y}} \mathbb{1}_{\{l_y \text{ is modified}\}}.$$

By Hoeffding's inequality,

$$\mathbb{P}\left(N_{\text{modified}} \geq 24c\frac{N}{k} + \sqrt{N} \,\middle|\, \mathcal{U}, \mathcal{B} \right) \leq \exp(-2\sqrt{N}).$$

Thus,

$$\mathbb{P}\left(N_{\text{modified}} \geq 24c\frac{N}{k} + \sqrt{N} \right) \leq \exp(-2\sqrt{N}) + \mathbb{P}(\overline{\mathcal{B}}).$$

According to equations 6.4 and 6.5, $\mathbb{P}\left(\overline{\mathcal{B}}\right) \leq \exp(-\frac{\sqrt{cN}}{5}) - \exp(-N)$. This together with Lemma 6.5.2 terminates the proof. □

Proof of Lemma 6.3.5 (Discard) For any $\ell \in [kN]$, let

$$n_\ell := \left\{ \frac{\lfloor uNk \rfloor}{Nk} = \frac{\ell}{Nk} \,\middle|\, u \in \mathcal{U} \right\}.$$

The points in $\mathcal{U}$ are generated through a Poisson point process of intensity N in $[0,1]$, thus, for any $\ell \in [Nk]$:

$$\mathbb{P}(n_\ell > 0) = 1 - e^{-1/k},$$

and the $(n_\ell)_{\ell\in[Nk]}$ are independent of each other. The number of points in $\tilde{\mathcal{U}}$ is

$$|\tilde{\mathcal{U}}| = \sum_{\ell=1}^{Nk} \mathbb{1}_{n_\ell > 0}.$$

We have

$$\begin{aligned} \mathbf{E}\left[|\tilde{\mathcal{U}}|\right] =& Nk(1 - e^{-1/k}) \\ \geq& N - \frac{N}{k}. \end{aligned}$$

By Chernoff bound,

$$\mathbb{P}(|\tilde{\mathcal{U}}| \leq N - \frac{N}{k} - N^{3/4}) \leq \exp(-\sqrt{N}/2)$$

We have

$$N_{\text{removed}} := |\mathcal{U}| - |\tilde{\mathcal{U}}|.$$

We have already obtained by concentration of Poisson random variables

$$\mathbb{P}\left(\left|N - |\tilde{\mathcal{U}}|\right| \geq N^{3/4}\right) \leq 2e^{-\frac{N^{1/2}}{4}}.$$

Thus,

$$\mathbb{P}\left(N_{\text{removed}} \geq 2N^{3/4} + \frac{N}{k}\right) \leq 3e^{-\frac{N^{1/2}}{4}}.$$

Combining this with Lemma 6.5.1 terminates the proof. □

Proof of Lemma 6.3.6 (Gluing) By Lemma 6.5.1, adding the vertex at coordinate 0 modifies the score of the algorithm by at most one.

The priority list of some vertex $y \in \mathcal{Y}$ may be modified by the gluing step only if $y < c/N$ or $y > 1 - c/N$.

$$N_{\text{gluing}} := \sum_{y \in \mathcal{Y}} \mathbb{1}_{\{l_y \text{ is modified during the gluing step}\}}.$$

By Chernoff bound,

$$\mathbb{P}(N_{\text{gluing}} \geq 2c + \sqrt{N}) \leq e^{-\sqrt{N}}.$$

Lemma 6.5.2 concludes the proof. □

6.6 Proof of Lemma 6.3.7 (Gaps repartition)

We note $|H|$ the length of a sequence H and $S([N])$ the ensemble of all permutations over $[N]$. For a sequence $H = (h_i)_{i \in [K]}$ let $\mathcal{A}_t(H)$ be the event:

$$\left\{N_t = |H|, \exists \sigma \in S([N_t]) \text{ s.t. } \forall i \in [N_t], u_t(i+1) - u_t(i) = \frac{h_{\sigma(i)}}{Nk}\right\},$$

and $\mathcal{A}_t(\sigma, H)$ the event:

$$\left\{N_t = |H|, \forall i \in [N_t], u_t(i+1) - u_t(i) = \frac{h_{\sigma(i)}}{Nk}\right\}.$$

In other words, $\mathcal{A}_t(H)$ is the event that unordered values of the gaps between the free vertices at iteration t is sequence H. Event $\mathcal{A}_t(\sigma, H)$ is the more constrained event that those values are ordered following permutation σ. Note that

$$\mathcal{A}_t(H) = \cup_{\sigma \in S([H])} \mathcal{A}_t(\sigma, H).$$

Also note that it is possible for events σ, σ' to be duplicates of each others if there are duplicate values in the sequence H. For instance if all values in H are equal, then all $\mathcal{A}_t(\sigma, H)$ are equal to event $\mathcal{A}_t(H)$. For a list of sequences $H_1, \ldots, H_t$, we note $\mathcal{A}_{1:t}(H_{1:t})$ the event that

$A_s(H_s)$ hold for all $s \leq t$:

$$\mathcal{A}_{1:t}\left(H_{1:t}\right) := \cap_{1\leq s\leq t} A_s(H_s).$$

Lemma 6.3.7 is a consequence of the following stronger Lemma.

Lemma 6.6.1. *For any sequence H, any iteration t, any two permutations $\sigma, \sigma' \in S\left(\left[|H|\right]\right)$,*

$$\mathbb{P}\left(\mathcal{A}_t\left(\sigma, H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right) = \mathbb{P}\left(\mathcal{A}_t\left(\sigma', H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right). \tag{6.6}$$

Proof: We prove this lemma by induction. First, let $p_k = 1 - e^{-1/k}$ so that, by design of the rounding procedure, for each $\ell \in [1, Nk-1]$,

$$\mathbb{P}\left(\frac{\ell}{Nk} \in \tilde{\mathcal{U}}\right) = p_k,$$

and the events $\left(\left\{\frac{\ell}{Nk} \in \tilde{\mathcal{U}}\right\}\right)_{\ell\in[1,Nk-1]}$ are independent of each other (and we still have $0 \in \tilde{\mathcal{U}}$).

Let $K < Nk$ be a non-negative integer and let $H = (h_i)_i$ be any sequence of integers s.t.

$$\sum_{i=1}^{|H|} \frac{h_i}{Nk} = 1.$$

Since $u_0(1) = 0$, the knowledge of the sizes and the ordering of the gaps determines the position of the points. Thus, for any $\sigma \in S\left(\left[|H|\right]\right)$:

$$\mathbb{P}\left(|H| = N_0, \forall i \in [N_0],\ u_0(i+1) - u_0(i) = \frac{h_{\sigma(i)}}{Nk}\right) = p_N^{N_0-1}(1-p_N)^{Nk-N_0}.$$

This does not depend on the choice of permutation σ. Thus for any sequence H s.t. $\sum_{i=1}^{|H|} \frac{h_i}{Nk} = 1$ and any two permutations $(\sigma, \sigma') \in S\left(\left[|H|\right]\right)^2$:

$$\mathbb{P}\left(\mathcal{A}_0\left(\sigma, H\right) \middle| \mathcal{A}_0\left(H\right)\right) = \mathbb{P}\left(\mathcal{A}_0\left(\sigma', H\right) \middle| \mathcal{A}_0\left(H\right)\right). \tag{6.7}$$

We just proved that Equation (6.6) holds at iteration 0. Let us assume that Equation (6.6) holds at all iterations until the t-th one. We aim at showing that this implies that it also holds at iteration $t+1$. There are two cases possible, depending on whether or not a vertex is matched.

We first show the implication in the case where the incoming vertex is not matched, i.e. it lays at a distance larger than c/N of any free vertex. Note y_t the incoming vertex at iteration t. For any $\sigma \in S([N_t])$, any H_t a sequence of length N_t s.t. $\sum_{i=1}^{|H|} \frac{h_i}{Nk} = 1$,

$$\mathrm{P}\left(y_t \text{ is not matched} \;\middle|\; \mathcal{A}_t\left(\sigma, H_t\right)\right) = 1 - \sum_{i=1}^{N_t} \frac{\min(2c, h_i/k)}{N} \tag{6.8}$$

The following therefore holds:

$$\mathrm{P}\left(\mathcal{A}_{t+1}\left(\sigma, H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right) = \mathrm{P}\left(y_t \text{ is not matched}, \mathcal{A}_t\left(\sigma, H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right).$$

By the induction hypothesis and Equation (6.8), the right term does not depend on σ, thus, for any σ, σ' in $S([N_t])$,

$$\mathrm{P}\left(\mathcal{A}_{t+1}\left(\sigma, H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right) \cap \mathcal{A}_{t+1}\left(H_t\right)\right) = \mathrm{P}\left(\mathcal{A}_{t+1}\left(\sigma', H_t\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right) \cap \mathcal{A}_{t+1}\left(H_t\right)\right).$$

We now turn to the case where y_t is matched. We define an admissible sequence for sequence H and a couple $j < j' \leq |H|$ as a sequence $\tilde{H}(j, j')$ s.t. for any $i \in [|H| - 1]$

$$\tilde{h}_i^{j,j'} = \begin{cases} h_i & \text{if } i < j' \text{ and } i \neq j \\ h_j + h_{j'} & \text{if } i = j \\ h_{i+1} & \text{if } i \geq j'. \end{cases}$$

Note it is the sequence of gaps obtained when event $\mathcal{A}_t\left(H\right)$ is true and a vertex $u_t(i)$ with $u_t(i) - u_t(i-1) = h_j$ and $u_t(i+1) - u_t(i) = h_{j'}$ is matched.

For any $H_{1:t}$, any $j < j' \leq |H_t|$, any $\sigma \in S([|H_t| - 1])$ define event $\mathcal{B}_t(\sigma, H_t, i, j, j')$ as:

$$\begin{aligned} &\left\{u_t(i) - u_t(i-1) = \tfrac{h_j}{Nk} \text{ and } u_t(i+1) - u_t(i) = \tfrac{h_{j'}}{Nk}\right\} \\ &\cup \left\{u_t(i) - u_t(i-1) = \tfrac{h_{j'}}{Nk} \text{ and } u_t(i+1) - u_t(i) = \tfrac{h_j}{Nk}\right\}, \\ &\cap \{\forall k < i-1, u_t(k+1) - u_t(k) = \tfrac{\tilde{h}^{j,j'}_{\sigma(k)}}{Nk}\}, \\ &\cap \{\forall k > i, u_t(k+1) - u_t(k) = \tfrac{\tilde{h}^{j,j'}_{\sigma(k-1)}}{Nk}\}. \end{aligned}$$

For any $H_{1:t}$, any $j < j' \leq |H_t|$, any $\sigma \in S([|H_t| - 1])$, it holds that:

$$\begin{aligned} &\mathrm{P}\left(\mathcal{A}_{t+1}\left(\sigma, \tilde{H}_t(j, j')\right) \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right) \\ &\qquad = \sum_{i \text{ s.t. } \tilde{h}_{\sigma(i-1)} = \tilde{h}_j} \mathrm{P}\left(u_t(i) \text{ is matched}, \mathcal{B}_t(\sigma, H_t, i, j, j') \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right) \\ &\qquad = \frac{\min(c, \frac{h_j}{k}) + \min(c, \frac{h_{j'}}{k})}{N} \sum_{i \text{ s.t. } \tilde{h}_{\sigma(i-1)} = \tilde{h}_j} \mathrm{P}\left(\mathcal{B}_t(\sigma, H_t, i, j, j') \;\middle|\; \mathcal{A}_{1:t}\left(H_{1:t}\right)\right). \end{aligned}$$

By the induction property, the right term does not depend on σ. This implies that the induction property remains true when a vertex is matched as well, which implies that Equation (6.6) holds for all $t \in [N]$.

□

We now show that Lemma 6.6.1 implies Lemma 6.3.7. Let $\mathcal{F}_t$ be the event associated with the values $\left(F_{k,N}(\ell, t')\right)_{\ell \in [Nk]}$ for all $t' \leq t$, which also determines the value of N_t. Let H be

any sequence of length N_t s.t. for any $\ell \in [Nk]$, $|\{h_i = \ell | i \in [N_t]\}| = F_{k,N}(\ell, t)$. Equation (6.6) implies:

$$\begin{aligned}
\mathrm{E}\left[M_{k,N}(\ell_-, \ell_+, t)\middle|\mathcal{F}_t\right] =& \frac{1}{|S([N_t])|}\sum_{\sigma\in S([N_t])}\sum_{i=1}^{N_t}\mathbb{1}_{\{h_{\sigma(i)}=\ell_-, h_{\sigma(i+1)}=\ell_+\}} \\
=& \sum_{i=1}^{N_t}\frac{1}{|S([N_t])|}\sum_{\sigma\in S([N_t])}\mathbb{1}_{\{h_{\sigma(i)}=\ell_-, h_{\sigma(i+1)}=\ell_+\}} \\
=& \sum_{i=1}^{N_t}\frac{|S([N_t-2])|}{|S([N_0])|}\left[\mathbb{1}_{\{\ell_-\neq\ell_+\}}F_{k,N}(\ell_-,t)F_{k,N}(\ell_+,t)\right] \\
&+\sum_{i=1}^{N_t}\frac{|S([N_t-2])|}{|S([N_0])|}\left[\mathbb{1}_{\{\ell_-=\ell_+\}}F_{k,N}(\ell_-,t)\left(F_{k,N}(\ell_+,t)-1\right)\right] \\
=& F_{k,N}(\ell_-,t)\left(\mathbb{1}_{\{\ell_-\neq\ell_+\}}\frac{F_{k,N}(\ell_+,t)}{N_t-1}+\mathbb{1}_{\{\ell_-=\ell_+\}}\frac{F_{k,N}(\ell_+,t)-1}{N_t-1}\right).
\end{aligned}$$

□

6.7 Application of the Differential Equation Method

Lemma 6.7.1 (Initial Conditions)**.** *With probability at least* $1-4kN\exp(-\sqrt{N}/4)$*, for any* $\ell < kN$*,*

$$\left|F_{N,k}(\ell,0) - Nke^{-\frac{\ell}{k}}\left(1-e^{-\frac{1}{k}}\right)^2\right| \leq 3N^{3/4}.$$

Proof: Consider the process of placing a sequence of vertices $(v_i)_{i=1}^{+\infty}$ in $\mathbb{R}$, placing v_0 at zero, and having

$$\mathbb{P}\left(v_{i+1}-v_i = \frac{\ell}{Nk}\right) = e^{-\frac{\ell-1}{k}}(1-e^{-\frac{1}{k}}).$$

Note that we have:

$$\mathbb{E}[v_{i+1}-v_i] = \frac{1}{Nk(1-e^{-1/k})}.$$

By Chernoff bound for sums of Bernoulli random variables,

$$\mathbb{P}(v_{Nk(1-e^{-1/k})+N^{3/4}} < 1) \leq \exp(-\sqrt{N}/4) \text{ and } \mathbb{P}(v_{Nk(1-e^{-1/k})-N^{3/4}} > 1) \leq \exp(-\sqrt{N}/4) \tag{6.9}$$

and

$$\mathbb{P}\left(\left|\sum_{i=1}^{Nk(1-e^{-1/k})+N^{3/4}}\mathbb{1}_{\{v_{i+1}-v_i=\frac{\ell}{Nk}\}} - Nke^{-\frac{\ell-1}{k}}(1-e^{-\frac{1}{k}})^2\right| > 2N^{3/4}\right) \leq 2e^{-\sqrt{N}/4}.$$

The law of the vertices placed before 1 is exactly the law of the vertices in $\tilde{\mathcal{U}}$. By Equation (6.9), with probability at least $1-2\exp(-\sqrt{N}/4)$, there are between $N-N^{3/4}$ and $N+N^{3/4}$ vertices places before 1. A union bound over $(\ell)_{\ell=1}^{kN}$ terminates the proof. □

Any vertex $u \in \tilde{\mathcal{U}}$ that has no neighbor in $\mathcal{Y}$ is never matched. We call such vertices *blocking* vertices. We will show that with high probability there is a linear number of such blocking vertices. Let N_b be the number of *blocking* vertices.

Lemma 6.7.2 (Minimum number of vertices)**.** *W.h.p. there is a linear number of blocking vertices:*

$$\mathbb{P}\left(N_b \leq \frac{(1-2e^{-4c})\left(1-e^{-2c}\right)}{8c}N - 2N^{3/4}\middle|\mathcal{C}\right) \leq 2\exp\left(-\min(1,c)N^{1/2}/4\right).$$

Proof: Define

$$f_N(\ell,t) = \frac{F_N(\ell,t)}{N}.$$

Let us split the interval $[0,1]$ in $\frac{N}{4c}$ small intervals of length $\frac{4c}{N}$. There is a blocking vertex in a small interval if there is no vertex $y \in \mathcal{Y}$ in it and at least a vertex $u \in \mathcal{U}$ at a distance larger than c from its border. For any $i \in [\frac{N}{4c}]$,

$$\begin{aligned}\mathbb{P}\left(\tilde{\mathcal{U}} \cap \left[\frac{4ci+c}{N}, \frac{4c(i+1)-c}{N}\right] \neq \emptyset\right) =& 1-(1-p_k)^{2ck}\\ =& 1-e^{-2c}.\end{aligned}$$

Define

$$\mathcal{C} := \left\{\left|\left\{\tilde{\mathcal{U}} \cap \left[\frac{4ci+c}{N}, \frac{4c(i+1)-c}{N}\right] \neq \emptyset \;\middle|\; i \in [1;N]\right\}\right| \leq \frac{\left(1-e^{-2c}\right)}{8c}N\right\}.$$

By Hoeffding's inequality,

$$\mathbb{P}(\mathcal{C}) \leq \exp\left(-\frac{(1-e^{-2c})^2 N}{32c^2}\right).$$

The probability that at least a vertex $y \in \mathcal{Y}$ falls in one of the small intervals is:

$$1-\left(1-\frac{4c}{N}\right)^N \geq 1-2e^{-4c}.$$

Thus, the expected number of blocking vertices conditioned on event $\mathcal{C}$ is lower bounded as :

$$\mathbb{E}\left[N_b \mid \mathcal{C}\right] \geq \frac{(1-2e^{-4c})\left(1-e^{-2c}\right)}{8c}N$$

Combining Hoeffding's inequality with a Poissonization argument on the vertices (to get the independence of the arrival on each small interval) we obtain:

$$\mathbb{P}\left(N_b \leq \frac{(1-2e^{-4c})\left(1-e^{-2c}\right)}{8c}N - 2N^{3/4} \mid \mathcal{C}\right) \leq 2\exp\left(-\min(1,c)N^{1/2}/4\right).$$

□

In the rest of the proof, we use the shorthand:

$$\Phi_{k,\ell}\left(\left(f_k(\ell',t)\right)_{\ell'}\right) = -\min(\frac{\ell}{k},2c)f_k(\ell,t) - \frac{1}{\sum_{\ell'=0}^{+\infty} f_k(\ell',t)}\left[\sum_{\ell'=0}^{+\infty}\min(\frac{\ell}{k},2c)f_k(\ell',t)\right] f_k(\ell,t)$$
$$+ \frac{1}{\sum_{\ell'=0}^{+\infty} f_k(\ell',t)}\sum_{\ell'=0}^{+\infty}\min(\frac{\ell'}{k},2c)f_k(\ell',t)f_k(\ell-\ell',t),$$

Lemma 6.7.3 (Total length invariant)**.** *With f_k the solution of the system of ODE in Lemma 6.3.9, for any $t \in [0,1]$,*

$$\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f_k(\ell,t) = 1.$$

Proof: This is true for $t=0$ by definition of the initial condition. We now show that this quantity is an invariant of the system of ODEs. We have:

$$\frac{\partial\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f_k(\ell,t)}{\partial t} = \sum_{\ell}\Phi_{k,\ell}\left(\left(f_k(\ell',t)\right)_{\ell'}\right).$$

So:

$$\begin{aligned}
\frac{\partial\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f_k(\ell,t)}{\partial t} =& -\sum_{\ell=1}^{+\infty}\frac{\ell}{k}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\\
&- \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f(\frac{\ell}{k},t)\\
&+ \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\sum_{\ell'=0}^{+\infty}\min(\frac{\ell'}{k},2c)f_k(\ell',t)(\ell-\ell')f_k(\ell-\ell',t)\\
&+ \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\sum_{\ell'=0}^{+\infty}\ell'\min(\frac{\ell'}{k},2c)f_k(\ell',t)f_k(\ell-\ell',t)\\
=& -\sum_{\ell=1}^{+\infty}\frac{\ell}{k}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\\
&- \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f(\frac{\ell}{k},t)\\
&+ \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\sum_{\ell=1}^{+\infty}\frac{\ell}{k}f(\frac{\ell}{k},t)\\
&+ \frac{1}{\sum_{\ell=1}^{+\infty}f_k(\ell,t)}\sum_{\ell=1}^{+\infty}\frac{\ell}{k}\min(\frac{\ell}{k},2c)f(\frac{\ell}{k},t)\sum_{\ell=1}^{+\infty}f_k(\ell,t)\\
=&0.
\end{aligned}$$

□

We also have that any $t\in[0,1]$, $\sum_{\ell=1}^{Nk}\frac{\ell}{Nk}F_{k,N}(\ell,\frac{\lfloor tN\rfloor}{N}) = 1$. This and the previous Lemma

imply that for any $\eta > 0$:

$$\sum_{\ell=\frac{k}{\eta}}^{Nk} F_{N,k}(\ell,t) \leq \eta N \quad \text{and} \quad \sum_{\ell=\frac{k}{\eta}}^{+\infty} f_k(\ell,t) \leq \eta. \tag{6.10}$$

It always holds that $N_t > N_b$. Now, for all $t \in [N]$, for all $\ell \in \mathbb{N}$, following Lemma 6.3.8,

$$\begin{aligned}
&\mathbb{E}\left[F_{k,N}(\ell,t+1) - F_{k,N}(\ell,t)\middle|\mathcal{F}_t, N_t \geq c\left(1-e^{-2c}\right)N\right] \\
&= -\left(\min\left(2c,\frac{\ell}{k}\right) + \sum_{\ell'}\min\left(2c,\frac{\ell'}{k}\right)\frac{f_{k,N}(\ell',t)}{N_t/N}\right) f_N(\ell,t) \\
&\quad + \sum_{\ell'=0}^{\ell}\min\left(2c,\frac{\ell'}{k}\right)\frac{f_{k,N}(\ell',t)\left(f_{k,N}(\ell-\ell',t)\right)}{N_t/N} + O\left(\frac{1}{N}\right).
\end{aligned}$$

The end of the proof follows exactly the steps of the proof of Theorem 4.3 in Enriquez et al. (2019) □

6.7.1 Proof of the link with the continuous equation (Lemma 6.3.10)

It holds that

$$\begin{aligned}
\frac{\partial \int_0^{+\infty} f(x,t)dx}{\partial t} &= -\int_0^{+\infty}\min(x,2c)f(x,t)dx - \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{+\infty}\min(x',2c)f(x',t)dx'\int_0^{+\infty} f(x,t)dx \\
&\quad + \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{+\infty}\int_0^{x}\min(x',2c)f(x',t)f(x-x',t)dx'dx \\
&= -\int_0^{+\infty}\min(x,2c)f(x,t)dx - \int_0^{+\infty}\min(x',2c)f(x',t)dx' \\
&\quad + \frac{1}{\int_0^{+\infty} f(x',t)dx'}\int_0^{+\infty}\min(x',2c)f(x',t)dx'\int_0^{+\infty} f(x',t)dx' \\
&= -\int_0^{+\infty}\min(x,2c)f(x,t)dx \\
&\geq -2c\int_0^{+\infty} f(x,t)dx.
\end{aligned}$$

This implies that for any $t \in [0,1]$,

$$e^{-2c} \leq e^{-2ct} \leq \int_0^{+\infty} f(x,t)dx \leq \int_0^{+\infty} f(x,0)dx = 1.$$

Define

$$\mathcal{L}_c := \left\{g \in L_1 \text{ s.t. } e^{-4c} \leq || g ||_{L_1} \leq 1 \text{ and } g \geq 0\right\}.$$

Let us show that the application $A : \mathcal{L}_c \to \mathcal{L}_c$

$$A(f)(x) = -\min(x, 2c)f(x,t) - \frac{1}{\int_0^{+\infty} f(x',t)dx'} \int_0^{+\infty} \min(x', 2c)f(x',t)dx' f(x,t) + \frac{1}{\int_0^{+\infty} f(x',t)dx'} \int_0^{x} \min(x', 2c)f(x',t)f(x-x',t)dx'$$

is Lipschitz with respect to the L_1 norm, and derive the Lipschitz constant. First, for any two functions $f, f' \in \mathcal{L}_c^2$,

$$\left| \frac{1}{\int_0^{+\infty} f(x,t)dx} - \frac{1}{\int_0^{+\infty} f'(x,t)dx} \right| \leq e^{4c} \mid\mid f - f' \mid\mid_{L_1} .$$

Also, noting $\tilde{f}$ the function $\min(x, 2c)f(x)$, we have $\mid\mid \tilde{f}(x) - \tilde{f'}(x) \mid\mid_{L_1} \leq 2c \mid\mid f - f' \mid\mid_{L_1}$, and

$$\begin{aligned} \mid\mid \tilde{f} * f - \tilde{f'} * f' \mid\mid_{L_1} &\leq \mid\mid (\tilde{f} - \tilde{f'}) * f \mid\mid_{L_1} + \mid\mid \tilde{f'} * (f' - f) \mid\mid_{L_1} \\ &\leq \mid\mid \tilde{f} - \tilde{f'} \mid\mid_{L_1} \mid\mid f \mid\mid_{L_1} + \mid\mid \tilde{f'} \mid\mid_{L_1} \mid\mid f' - f \mid\mid_{L_1} \\ &\leq 4c \mid\mid f' - f \mid\mid_{L_1}, \end{aligned}$$

where the second line comes from Young's inequality. Putting everything together, we get:

$$\begin{aligned} \mid\mid A(f) - A(f') \mid\mid_{L_1} &\leq \mid\mid \tilde{f'} - \tilde{f} \mid\mid_{L_1} + (e^{4c} + 2c) \mid\mid f - f' \mid\mid_{L_1} \\ &\quad + e^{4c} \mid\mid f - f' \mid\mid_{L_1} \mid\mid f * \tilde{f} \mid\mid_{L_1} + e^{2c} \mid\mid \tilde{f} * f - \tilde{f'} * f' \mid\mid_{L_1} \\ &\leq \left(4c + (2c+1)\, e^{4c} + 4ce^{2c}\right) \mid\mid \tilde{f'} - \tilde{f} \mid\mid_{L_1} . \end{aligned}$$

We note this Lipschitz constant lip_c. Interpolating the function in Lemma 6.3.9 as:

$$f_k(x,t) = f\left(\frac{\lceil kx \rceil}{k}, t\right),$$

yields

$$\mid\mid f(.,0) - f_k(.,0) \mid\mid_{L_1} \leq \frac{2}{k}.$$

Thus, by application of Gronwall's Lemma, for any $t \in [0,1]$

$$\mid\mid f(1,t) - f_k(1,t) \mid\mid_{L_1} \leq \frac{2}{k} e^{\text{lip}_c t}.$$

□

6.A Appendix

6.A.1 Poisson Point Processes

The following definitions and properties of point processes come from lecture notes Blaszczyszyn (2017), and are reported here for clarity.

Definition 6.A.1. *(Homogeneous Poisson point process). A point process Φ on $[0,1]$ is an homogeneous Poisson point process of intensity λ if the following two conditions are satisfied:*

1. *For any $(a,b] \in [0,1], \Phi(a,b]$, the number of points in interval $(a,b]$, is a Poisson random*

variable of intensity $\lambda(b-a)$, i.e.;

$$P\{\Phi(a,b]=n\}=\frac{[\lambda(b-a)]^n}{n!}e^{-\lambda(b-a)}.$$

2. *The number of points in any two disjoint intervals are independent of each other, and this extends to any finite number of disjoint intervals, i.e.;*

$$P\left\{\Phi\left(a_i,b_i\right]=n_i,i=1,\ldots,k\right\}=\prod_{i=1}^{k}\frac{\left[\lambda\left(b_i-a_i\right)\right]^{n_i}}{n_i!}e^{-\lambda(b_i-a_i)},$$

for any integer $k\geq 2$, any $a_1<b_1\leq a_2\ldots<b_k$.

Note that the second point of the definition implies the following property: given there are n points of the homogeneous Poisson process in the window B, these points are independently and uniformly distributed in B.

Let Φ^N be a Poisson point procees of intensity N on $[0,1]$, and $\mathcal{U}^N\sim\Phi^N$. Let us enumerate the points of the point process $\mathcal{U}^N$ according to their coordinates. The sequence $\{u_k\}$ can be constructed as a renewal process with exponential holding times, i.e., $u_k=\sum_{i=1}^{k}F_i$ for $k\geq 1$, where $\{F_k:k=1,\ldots\}$ is a sequence of independent, identically distributed exponential random variables of parameter N. Indeed,

$$\mathbb{P}\left\{F_1>t\right\}=\mathbb{P}\left\{u_1>t\right\}=\mathbb{P}\{\Phi((0,t])=0\}=e^{-Nt},$$

and, for $k\geq 2$ by independence (second point of the definition),

$$\begin{aligned}\mathbb{P}\left\{F_k>t\mid F_1,\ldots,F_{k-1}\right\}&=\mathbb{P}\left\{u_k-u_{k-1}>t\mid u_1,\ldots,u_{k-1}\right\}\\&=\mathbb{P}\left\{\Phi\left(u_{k-1},u_{k-1}+t\right]=0\mid u_{k-1}\right\}\\&=e^{-Nt}.\end{aligned}$$

Chapter 7

On Preemption and Learning in Stochastic Scheduling

In this work, we study single-machine scheduling of jobs, each belonging to a job type that determines its duration distribution. We start by analyzing the scenario where the type characteristics are known and then move to two learning scenarios where the types are unknown: non-preemptive problems, where each started job must be completed before moving to another job; and preemptive problems, where job execution can be paused in the favor of moving to a different job. In both cases, we design algorithms that achieve sublinear excess cost, compared to the performance with known types, and prove lower bounds for the non-preemptive case. Notably, we demonstrate, both theoretically and through simulations, how preemptive algorithms can greatly outperform non-preemptive ones when the durations of different job types are far from one another, a phenomenon that does not occur when the type durations are known.

Contents

7.1 Introduction

Single Machine Scheduling is a longstanding problem with many variants and applications Pinedo (2012). In this problem, a set of N jobs must be processed on one machine, each of a different 'size' – processing time required for its completion. An algorithm is a policy assigning jobs to the machine, and performance is usually measured by *flow time* – the sum of the times when jobs have finished. If one has access to the size of each job, then scheduling the jobs by increasing size is optimal Schrage (1968). Unfortunately, for most applications, this knowledge is unavailable; yet, oftentimes, some structure or knowledge on the jobs can still be leveraged.

In this paper, we focus on scheduling problems where jobs are grouped by types that determine their duration distribution. This model approximates many real-world scenarios. For example, when scheduling patients for surgery, patients may be grouped by expected procedure time Magerlein and Martin (1978). The model is also relevant in computing problems, where jobs with similar features are expected to have a similar processing time Li et al. (2006). Lastly, in calendar learning, where an agent advises the user on how to organize its day based on the tasks to be done, similar tasks can be assumed to have a similar duration White and Hassan Awadallah (2019).

In practice, when encountering a new scheduling task, we usually know the type of each job, but have little-to-no information on the expected duration under each type. Then, the scheduling algorithm must *learn* the characteristics of each type to be able to utilize this information. This must be done concurrently with the scheduling of tasks, which poses an extra challenge – to be useful, learning must be done as early as possible; however, wrong scheduling allocation at the beginning delays all jobs and causes large penalties.

In this work, we show how learning can be efficiently done in scheduling problems with job types, characterized by exponential distributions, in two different settings – the non-preemptive setting, where once a job started running, it must be completed, and the preemptive setting, where jobs can be put on hold. We present two algorithms in each setting and show that the preemptive setting has a clear advantage when the type durations have to be learned. This comes in contrast to the case of known types, where under reasonable assumptions, the optimal algorithm is non-preemptive.

While our algorithms resemble classic bandit methods, the scheduling objective requires different analysis approaches. In particular, in the context of scheduling, the quality of an algorithm is measured by the *ordering* of jobs. In stark contrast, regret-minimization objectives measure the *number of plays* from each arm (job type). Indeed, in scheduling problems, the number of pulls from each job type is always the same – by the end of the interaction, we would finish all the n jobs of all types. Thus, both our algorithmic design and analysis will be comparative – focus on the number of jobs evaluations from a bad type before the completion of jobs of a good type.

Our contributions are as follows. **(1)** We present the scheduling setting with unknown job types. **(2)** We analyze the optimal algorithm for the case of known job types, called Follow-the-Perfect-Prediction (FTPP), and bound its competitive ratio (CR). **(3)** We present explore-then-commit (ETC) and upper confidence bound (UCB) algorithms for the preemptive and non-preemptive settings and bound their performance, compared to FTPP. In particular, our bounds show that the non-preemptive algorithms have worse dependence on the durations of the longest job types. **(4)** We complement this by proving lower bounds to the non-preemptive case. **(5)** We end by simulating our suggested algorithms and show that their empirical behavior is consistent with our theoretical findings.

7.2 Related Work

Scheduling problems. The scheduling literature and problem zoology are large. We focus on static scheduling on a single machine where the objective is to minimize the flow time. Possible generalizations include dynamic scheduling, where jobs arrive at different times Becchetti and Leonardi (2004); weighted flow time Bansal and Dhamdhere (2007), where different jobs have different weights; multiple machines Lawler and Labetoulle (1978)); and many more Dürr et al. (2020); Tsung-Chyan et al. (1997). While we only tackle some versions of this problem, we believe that our approaches can be adapted or extended to other settings.

Clairvoyant and non-clairvoyant scheduling. In clairvoyant scheduling, job sizes are assumed to be known, and scheduling the shortest jobs first gives the lowest flow time Schrage (1968). In non-clairvoyant scheduling, job sizes are arbitrary and unknown. The Round Robin (RR) algorithm, which gives the same amount of computing time to all jobs, is the best deterministic algorithm with a competitive ratio of $2 - \frac{2}{N+1} = 2 + o(N)$ Motwani et al. (1994). The best randomized algorithm has a competitive ratio of $2 - \frac{4}{N+3} = 2 + o(N)$ Motwani et al. (1994).

Stochastic scheduling. Stochastic scheduling covers a middle ground where job sizes are known random variables. The field of *optimal stochastic scheduling* aims to design optimal algorithms for stochastic scheduling (see Cai et al. 2014 for a review). When distributions have a non-decreasing hazard rate, scheduling the shortest mean first is optimal (see Cai et al. 2014, Corollary 2.1).

In this work, we consider exponential job sizes (which have a non-decreasing hazard rate), as frequently assumed in the scheduling literature Cai and Zhou (2000, 2005); Cunningham and Dutta (1973); Glazebrook (1979); Hamada and Glazebrook (1993); Kämpke (1989); Pinedo and Weiss (1985) and similarly for the presence of different types of jobs Hamada and Glazebrook (1993); Marbán et al. (2011); Mitzenmacher (2020). Yet, in contrast to most of the literature on stochastic scheduling, the means of the exponential sizes are unknown to the scheduler and are learned as the algorithm runs. Nonetheless, we later present algorithms whose CR asymptotically converges to the optimal value, obtained in stochastic scheduling with known job means.

The problem of learning in scheduling has received some attention lately. Specifically, Levi et al. (2019) consider a setting where it is possible to 'test' jobs to learn about their attributes, which comes at a cost. In Krishnasamy et al. (2018), the authors propose an algorithm to learn the $c\mu$ rule (a rule to balance different holding costs per job) in the context of dynamic queues. Perhaps closest to our setting, in Lee and Vojnovic (2021), job types are also considered, but the length of the jobs is assumed to be known, and the goal is to deal with the uncertainty on the holding costs, which are noisily observed at each iteration. In the last two papers, no explicit exploration is needed, which stands in contrast with our setting.

The problem we tackle was previously studied in a Bayesian setting Marbán et al. (2011), under the assumption of two job types, and a Bayesian algorithm, called LSEPT, was presented. When run with an uninformative prior (the same for all job types), LSEPT is reduced to a greedy algorithm; whenever a job finishes, it runs until completion a job whose type has the lowest expected belief on its mean size (computed across jobs that have been processed so far). The author proved it has better performance in expectation than fully non-adaptive methods, but provided no other guarantee. In Section 7.A.4, we empirically evaluate this algorithm and show it has a behavior typical of greedy algorithms: it has a very large variance, and its CR does not converge to the optimal CR, in contrast to our suggested methods.

Setting and Notations

We consider scheduling problems of N jobs on a single machine, each belonging to one of K job types. We assume that $N = nK$, i.e, there are n jobs of each type. The different sizes (also called processing times) of the jobs of type k are denoted $(P_i^k)_{i\in[n]}$,[1] where $P_i^k \sim \mathcal{E}(\lambda_k)$ are independent samples from an exponential variable of parameter λ_k. By extension, $\mathrm{E}[P_i^k] = \lambda_k$, and we call λ_k the mean size of type k. We assume without loss of generality that the mean sizes of the K types are in an increasing order $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_K$ and denote $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$. With slight abuse of notations, we sometimes ignore the job types and denote the job durations by P_i for $i \in [N]$.

Next, denote b_i^k and e_i^k the beginning and end dates of the computation of the i^{th} job of type k. We define the cost of an algorithm ALG, also called *flow time*, as the sum of all completion times: $C_{\mathrm{ALG}} = \sum_{k=1}^{K} \sum_{i\in n} e_i^k$. Given knowledge of the job size realizations, the cost is minimized by an algorithm that computes them in increasing order, which we term as OPT.

Preemption is the operation of pausing the execution of one job in the favor of running another one. Thus, *preemptive algorithms* are ones that support preemption, while *non-preemptive algorithms* do not allow it and must run each started job until completion.

7.3 Benchmark: Follow The Perfect Prediction

We compare our algorithms to a baseline that completes each job by increasing expected sizes, called *Follow-The-Perfect-Prediction* (FTPP). For exponential job sizes, this strategy is optimal between all algorithms without access to job size realizations (see Cai et al. 2014, Corollary 2.1). Thus, with learning, we aim at designing algorithms approaching the performance of FTPP, whilst mitigating the cost of learning (i.e., mitigating the cost of exploration). In the rest of this section, we analyze the performance of FTPP.

First, we evaluate the performance of non-clairvoyant algorithms that do not exploit the job type structure. We then compare the performance of the best of those algorithms against that of FTPP and show the clear advantage of using the structure of job types.

7.3.1 Non-clairvoyant Algorithms

An algorithm A is said non-clairvoyant if it does not have any information on the job sizes, including the job type structure. Recall that RR is the algorithm that computes all unfinished jobs in parallel and is the optimal deterministic algorithm in the adversarial setting. The following proposition states that in our setting, it is the optimal algorithm among all non-clairvoyant ones.

Proposition 7.3.1. *For any $\boldsymbol{\lambda}$ and any (deterministic or randomized) non-clairvoyant algorithm A, there exists a job ordering such that* $\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{RR}}]$.

Proof sketch (full proof in Appendix 7.A.1). Consider T_{ij}^A, the amount of time that job i and job j delay each other. As the algorithm is unaware of the expected job size order, its run is independent of whether the expected size of job i is smaller or greater than that of j. This holds because a non-clairvoyant algorithm has no information on job expected sizes nor on the existence of job types. As an adversary, we can therefore choose the job order so that the algorithm incurs the largest flow time. A careful analysis then provides $\mathrm{E}[T_{ij}^A] \geq 2\mathrm{E}[T_{ij}^{OPT}]$

[1] For clarity of exposition, we assume that there are exactly n jobs per type. When types have different numbers of jobs $n_1, n_2, \ldots, n_K$, all algorithms can run with $n = \max_\ell n_\ell$, and all bounds hold with this same parameter n.

where OPT is the optimal realization-aware algorithm. A similar reasoning is made in the case of randomized algorithms. We conclude by observing that RR achieves such delay. □

Unfortunately, even though our setting is not adversarial, the CR of RR is bounded from below (Lemma 7.A.3):

$$\text{for any } \boldsymbol{\lambda}, \quad \frac{\mathrm{E}\left[C_{\mathrm{RR}}\right]}{\mathrm{E}\left[C_{\mathrm{OPT}}\right]} \geq 2 - \frac{4}{n+3}. \tag{7.1}$$

7.3.2 Performance of FTPP

The first statement establishes that FTPP outperforms RR on any instance.

Lemma 7.3.2. *For any n and $\boldsymbol{\lambda}$,*

$$\mathrm{E}[C_{\mathrm{FTPP}}] \leq \mathrm{E}[C_{\mathrm{RR}}].$$

The inequality is strict if $\lambda \neq (a, , \ldots, a)$ for some $a > 0$.

The proof is a straightforward computation, done in Section 7.A.1.

This indicates that when information on the job types is available, it is always advantageous to use it. In the rest of the section, we quantify the improvement this extra information brings. More precisely, we show that on a wide variety of instances, the CR of FTPP is much smaller than that of RR. We first present such a bound when $K = 2$.

Proposition 7.3.3. *The CR of* FTPP *with $K = 2$ types of jobs with n jobs per type with $\lambda_1 = 1$ and $\lambda_2 = \lambda > 1$ satisfies:*

$$\frac{\mathrm{E}[C_{\mathrm{FTPP}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} \leq 2 - 4\frac{\lambda - 1}{(1+\lambda)^2 + 4\lambda}.$$

Proof sketch (full proof in Theorem 7.A.4). The total expected flow time of any algorithm is given by the sum of the expected time spent computing all jobs and the expected time lost waiting as jobs delay each other. In the case of OPT, the expected flow time can be computed in closed form using that job i and j delay each other by $\mathrm{E}[\min(X_i, X_j)] = \frac{\mathrm{E}[X_i]\mathrm{E}[X_j]}{\mathrm{E}[X_i]+\mathrm{E}[X_j]}$. The CR $\frac{\mathrm{E}[C_{\mathrm{FTPP}}]}{\mathrm{E}[C_{\mathrm{OPT}}]}$ can then be calculated and is upper bounded to yield the result. □

In the case $K = 2$, there exists values of λ for which the CR of FTPP is lower than 1.71. In the general case, Theorem 7.A.5 in Section 7.A.1 shows that there exist values of K and $\boldsymbol{\lambda}$ for which the CR is as low as 1.274.[2]

7.4 Non-Preemptive Algorithms

After establishing FTPP as the baseline for learning algorithms, we move to tackle learning in the non-preemptive setting, where once started, job execution cannot be stopped (see Algorithm 29). This is relevant, for example, to settings where switching tasks is very costly (e.g., in running time or memory) or even impossible (e.g., in medical applications, where treatment of a patient cannot be stopped). We show how algorithms from the bandit literature can be adapted to the scheduling setting and bound their excessive cost, compared to FTPP. Specifically, by treating each job type as an 'arm', we adapt explore-then-commit and optimism-based strategies to the scheduling setting.

[2] An exact expression for the CR of FTPP is given at Equation (7.7) in the appendix and is omitted for clarity reasons.

Algorithm 29: Non-Preemptive Algorithms routine

```
Init: type set 𝒰 = [K], active jobs i_k = 1, ∀k ∈ [K];
while 𝒰 is not empty do
    Use a type selection subroutine to select a type k ∈ 𝒰;
    Run job i_k until completion;
    Set i_k ← i_k + 1;
    if All jobs of type k are completed then
        Remove type k from 𝒰;
    end
end
```

7.4.1 Description of ETC-U and UCB-U

In the following, we describe the type selection mechanism for ETC-U and UCB-U. The full pseudo-code of both algorithms is available in Section 7.A.2.

Let $\mathcal{U}$ be the set of all job types with at least one remaining job.

ETC-U type selection. While ETC-U runs, it maintains a set of types $\mathcal{A}$ that are candidates for having the lowest mean size among the incomplete types $\mathcal{U}$. At each iteration, ETC-U chooses a job of type k of the minimal number of completed jobs in $\mathcal{A}$ and executes it to completion. Then, $\mathcal{U}$ and $\mathcal{A}$ are updated and the procedure repeats until no more jobs are available in $\mathcal{U}$.

We now describe the mechanism of maintaining the candidate type set $\mathcal{A}$. At a given iteration, denote by m_k and m_ℓ, the number of jobs of type k and ℓ that have been computed up to that iteration. Letting

$$\hat{r}_{k,\ell}^{\min(m_k,m_\ell)} = \frac{\sum_{i=1}^{\min(m_k,m_\ell)} \mathbb{1}\left\{P_i^k < P_i^\ell\right\}}{\min(m_k, m_\ell)} \quad \text{and}$$
$$\delta_{k,\ell}^{\min(m_k,m_\ell)} = \sqrt{\frac{\log(2n^2K^3)}{2\min(m_k, m_\ell)}},$$

a type ℓ is excluded from $\mathcal{A}$ if there exists a type k such that

$$\hat{r}_{k,l}^{\min(m_k,m_\ell)} - \delta_{k,\ell}^{\min(m_k,m_\ell)} > 0.5. \tag{7.2}$$

In the proof, we show that this condition implies w.h.p. that $\lambda_k < \lambda_\ell$. Thus, when it holds, job type ℓ is no longer a candidate for the remaining job type with the smallest expectation, and we say that type k eliminates type ℓ. Once a job type is eliminated, it remains so until $\mathcal{A}$ is empty, at which point all job types in $\mathcal{U}$ are reinstated to $\mathcal{A}$.

Finally, whenever $\mathcal{A}$ contains only one type k, all jobs of this type are run to completion, and after all jobs from type k are finished, it is removed from $\mathcal{U}$ and therefore from $\mathcal{A}$. This means that types that were eliminated by k can be candidates again.

UCB-U type selection. At every iteration, the algorithm computes an index for each job type and plays a type with the minimal index from the incomplete types $\mathcal{U}$. Specifically, if m_k

jobs were completed from type k, the index of the type is defined as

$$\underline{\lambda}_k^{m_k} = \frac{2\sum_{i=1}^{m_k} X_i^k}{\chi^2_{2m_k}(1-\frac{1}{2n^2K^2})},$$

where $\chi^2_m(\delta)$ is the δ-percentile of a χ^2 distribution with m degrees of freedom. In the proof, we show that these indices are a lower bound of the job means w.h.p., so choosing the minimal index corresponds to choosing the type with the optimistic shortest duration.

7.4.2 Cost Analysis

Proposition 7.4.1. *The following bounds hold:*

$$\begin{aligned}\mathrm{E}[C_{\text{ETC-U}}] \leq\ & \mathrm{E}[C_{\text{FTPP}}] + \frac{1}{n}\mathrm{E}[C_{\text{OPT}}] \\ & + \sum_{k\in[K]}\left[\frac{1}{2}(k-1)(2K-k) + (K-k)^2\right]\lambda_k n\sqrt{8n\log(2n^2K^3)}\end{aligned}$$

and

$$\begin{aligned}\mathrm{E}[C_{\text{UCB-U}}] \leq & \mathrm{E}[C_{\text{FTPP}}] \\ & + n(K-1)\sqrt{3n\ln(2n^2K^2)}\sum_{k=1}^{K}\lambda_k \\ & + \frac{2}{n}\mathrm{E}[C_{\text{OPT}}].\end{aligned}$$

Proof sketch (full proof in Section 7.A.2). The above proposition is a concatenation of propositions 7.A.10 and 7.A.12.

The proof of both bounds starts with the decomposition of the cost with the following Lemma (proven in Appendix 11).

Lemma 7.4.2 (Cost of non-preemptive algorithms)**.** *Any non-preemptive algorithm A has a cost*

$$\begin{aligned}\mathrm{E}[C_A] = & \mathrm{E}[C_{\text{FTPP}}] \\ & + \sum_{\substack{(\ell,k)\in[K^2],k>\ell \\ (i,j)\in[n]^2}} (\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_i^k \leq b_j^\ell\right\}\right].\end{aligned}$$

This lemma is obtained by computing explicitly the expected cost of algorithm FTPP and using the fact that the realized length of the jobs conditioned on their type is independent of their start date.

Then the two proofs diverge.

For ETC-U, the first step is to prove that condition (7.2) implies w.h.p that $\lambda_k \leq \lambda_\ell$, which implies that the type in $\mathcal{U}$ with the smallest mean is never eliminated. Then, for the sake of the analysis, the run of the algorithm is divided into phases. In phase number ℓ, type ℓ is the job type remaining with the smallest mean. We then bound the total number of samples before an arm with a large mean is eliminated at phase ℓ.

The proof for UCB also starts by showing that w.h.p., arm indices lower bound the true means. Then, under the condition that the bounds hold, we upper bound the number of times an arm of type $k \geq \ell$ can be pulled while type ℓ is still active. □

The bounds in 7.4.1 hold for any value of the parameters. When the parameter values are far from each other, tighter bounds hold. We give here these tighter bounds for $K = 2$ when $\lambda_2 \geq 3\lambda_1$. A more general version of this bound is given in the Appendix, propositions 7.A.10 and 7.A.12.

Lemma 7.4.3. *If $K = 2$ and $\lambda_2 \geq 3\lambda_1$, the following bounds hold:*

$$\begin{aligned}\mathrm{E}[C_{\text{ETC-U}}] \leq &\mathrm{E}[C_{\text{FTPP}}] + 12\lambda_2 n \log(2n^2K^3) \\ &+ \frac{2}{n}\mathrm{E}[C_{\text{OPT}}].\end{aligned}$$

and:

$$\begin{aligned}\mathrm{E}[C_{\text{UCB-U}}] \leq &\mathrm{E}[C_{\text{FTPP}}] + \frac{9}{2}\lambda_2 n \ln\left(2n^2K^2\right) \\ &+ \frac{4}{n}\mathrm{E}[C_{\text{OPT}}].\end{aligned}$$

The bounds of this section seem quite discouraging – they imply that the existence of even one type of extremely large duration has grave implications on the cost of any algorithm. Unfortunately, for any non-preemptive algorithm, an extra cost w.r.t. FTPP scaling as $n\lambda_K$ is unavoidable. Indeed, in the beginning, no information on the mean types is available, and any started job will be fully computed, delaying all remaining $nK - 1$ jobs (see Line 11 for a formal proof).

7.4.3 Lower bound

We end this section by analyzing lower bounds for any non-preemptive scheduling algorithm. In particular, we focus on the dependency of the excessive cost, compared to FTPP, as a function of n. We focus on lower bounds for the case of $K = 2$, and provide a lower bound when λ_1 and λ_2 are close to each other and show that in this case, the excess cost increases as $n\sqrt{n}$.

Proposition 7.4.4 (Dependency in n)**.** *For any λ_1, λ_2, the flow time of any non-preemptive algorithm A satisfies:*

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\text{FTPP}}] + (\lambda_2 - \lambda_1)n^2/2\frac{\exp(-n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2})}{4}$$

In particular, for any $\lambda_2 \leq \lambda_1\left(1 + \frac{1}{\sqrt{n}}\right)$,

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\text{FTPP}}] + (\lambda_1 + \lambda_2)n\sqrt{n}\frac{e^{-1/4}}{24}.$$

Proof sketch (full proof in Line 11). We start with the decomposition of Theorem 7.4.2 and look at the event

$$E = \mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\left\{e_i^1 < b_j^2\right\} \geq n^2/2\right\}.$$

Then, using standard information-theoretic tools, we lower bound the probability that either E occurs in the original scheduling problem or $\bar{E}$ occurs in a problem where the type order has been switched. In both problems, the relevant event causes an excess cost of $\Omega(n^2)$, and substituting the exact probability of this failure case concludes the proof. □

Algorithm 30: Preemptive Algorithms routine

Init: type set $\mathcal{U} = [K]$, active jobs $i_k = 1, \forall k \in [K]$;
while $\mathcal{U}$ *is not empty* **do**
  Use a type selection subroutine to select a type $k \in \mathcal{U}$;
  Run job i_k for Δ time units;
  if i_k *was completed* **then**
    Set $i_k \leftarrow i_k + 1$;
  end
  if *All jobs of type* k *are completed* **then**
    Remove type k from $\mathcal{U}$;
  end
end

7.5 Preemptive Algorithms

In this section, we show how to leverage preemption to get better theoretical and practical performances.

In practice, we allow preemption by discretizing the computation time into small time slots of length Δ. Then, at every iteration, one or multiple job types are selected depending on some algorithm-specific criteria. The current running job(s) of the selected type is allocated computation time Δ instead of being run to completion. As before, we employ both an explore-then-commit strategy and an optimism-based strategy. In both cases, the only dependence of the resulting algorithm on the discretization size is due to the discretization error (the time between the end of a job and the end of a window), which decreases with the discretization step. We omit that discretization error of at most ΔN is the bounds.

Note that in practice, any implementation of RR proceeds in a similar manner. For instance, in Motwani et al. (1994), the discretization step is assumed much smaller than the length of the jobs. In particular, when we say we run jobs in parallel, in practice, they cyclically run in a RR manner with a small discretization step.

7.5.1 ETC-RR and UCB-RR

ETC-RR type selection. As ETC-U, ETC-RR maintains a set of types $\mathcal{A}$ that are candidates for lowest mean size among the set $\mathcal{U}$ of types with at least one remaining jobs. The main difference is that the job type selected is the one in $\mathcal{A}$ with the lowest total run-time (not the one with the lowest number of computed jobs).

The statistics needed to construct $\mathcal{A}$ are different from the ones used in ETC-U. At a given time, $\beta_{k,\ell}$ is the number of times a job of type k has finished while ℓ and k were both active. Moreover, we define

$$\hat{r}_{k,\ell} = \frac{\beta_{k,\ell}}{\beta_{k,\ell} + \beta_{\ell,k}} \quad \text{and} \quad \delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^3)}{2(\beta_{k,\ell} + \beta_{\ell,k})}}.$$

The elimination rule is the same as the one of ETC-U, using these modified statistics.

Reducing the number of algorithm updates: In practice, both the statistics and the chosen types are not updated at every iteration; active jobs run in parallel (meaning in a round-robin style), and the statistics are updated every time a job terminates. This formulation of the algorithm is the one we implement(see pseudo-code in Section 7.A.3).

UCB-RR type selection. For each job type $k \in [K]$, we introduce $T_k(t)$, the number of times job type k has been chosen up to iteration t, and the random variables $(x_k^s)_s$ s.t.:

$$x_k^s = \sum_t \mathbb{1}\{a(t) = k, T_k(t) = s \text{ and the job finishes}\}.$$

It is the indicator that a job of type k is completed when this type is picked for the s^{th} time by the algorithm. We define the empirical means as:

$$\hat{\mu}_k(T) := \frac{1}{T}\sum_{s=1}^{T} x_k^s,$$

and define the index for each arm k as

$$u_k(t) = \max\left\{\tilde{\mu} \in [0,1] : d\left(\hat{\mu}_k(T_k(t)), \tilde{\mu}\right) \leq \frac{\log n^2}{T_k(t)}\right\},$$

with $d(x, y)$ the Kullback-Leibler divergence between x and y. A job type with the largest index is selected.

Reducing the number of algorithm updates: As for ETC-RR, the running jobs and statistics are not updated at every iteration. Suppose type k^* is chosen at iteration type. If k^* is the last remaining type, it is run until the end. Otherwise, let ℓ be the type with the second largest index. We define

$$\tilde{\mu}_k^\gamma(T) := \frac{1}{T + 2^\gamma}\sum_{s=1}^{T} x_k^s,$$

and

$$\begin{aligned}&\tilde{u}_k^\gamma(t)\\ &\quad = \max\left\{\tilde{\mu} \in [0,1] : d\left(\tilde{\mu}_k^\gamma(T_k(t)), \tilde{\mu}\right) \leq \frac{\log n^2}{T_k(t) + 2^\gamma}\right\}.\end{aligned}$$

This would be the new index of arm k, were it to run for additional 2^γ iterations with no job terminating during this additional iterations. Then, we set

$$\gamma^* = \arg\max_\gamma \tilde{u}_{k^*}^\gamma(t) \geq u_\ell(t),$$

and type k^* is allocated 2^{γ^*} iterations with no statistics update.

7.5.2 Cost Analysis

Proposition 7.5.1. *The following bounds hold:*

$$\begin{aligned}\mathrm{E}[C_{\text{ETC-RR}}] \leq &\mathrm{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathrm{E}[C^{OPT}]\\ &+ 4n\sqrt{n\log(2n^2K^3)}\sum_{k=1}^{K-1}(K-k)^2\lambda_k.\end{aligned}$$

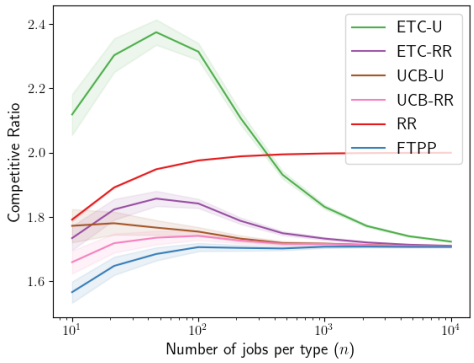


Figure 7.1: CR of all algorithms with varying number of jobs, $\lambda_1 = 1$, $\lambda_2 = 0.25$, averaged over 400 seeds.

and for any $\Delta \leq \frac{\lambda_1}{4}$ *and* $n \geq \max(20, 10\ln(K))$,

$$\mathrm{E}[C_{\text{UCB-RR}}] \leq \mathrm{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathrm{E}[C^{OPT}] + 6n\sqrt{2n\log(2n^2K^2) + 2}\sum_{k=1}^{K-1}(K-k)\lambda_k.$$

Proof sketch (full proof in Section 7.A.3). The above proposition is a combination of Propositions 7.A.16 and 7.A.17.

Both algorithms belong to the following family of type-wise non-preemptive algorithms.

Definition 7.5.2. *Recall that* b_i^k *and* e_i^k *are the beginning and end dates of the computation of the* i^{th} *job of type* k*. A* ***type-wise non-preemptive algorithm*** *is an algorithm that computes jobs of the same type one after another, i.e.,* $\forall i \in [n], \forall k \in [K], e_i^k \leq b_{i+1}^k$.

The following Lemma, proven in Appendix 7.A.2 bounds the expected cost of any type-wise non-preemptive algorithms.

Lemma 7.5.3 (Cost of type-wise non-preemptive algorithms)**.** *Any type-wise non-preemptive algorithm* A *has the following upper bound on its cost:*

$$\mathrm{E}[C_A] \leq \mathrm{E}[C_{\text{FTPP}}] + \sum_{\substack{(\ell,k)\in[K^2],k>\ell \\ (i,j)\in[n]^2}} (\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{k=1}^{K}\lambda_k.$$

The proof of this Lemma again involves computing explicitly the cost of FTPP and using that the realization of a job length is independent of its start date. A first upper bound is obtained by noting that a job started before another delays the former in expectation by at most its expected length. The second element to the proof is the fact that at every job termination, at most a job of each other type is currently active. This observation leads to the upper bound on the additional cost of preemption and the last term of the expression of the Lemma. Note that this last term implies that our upper bound will include a term scaling as $n\lambda_K$, which

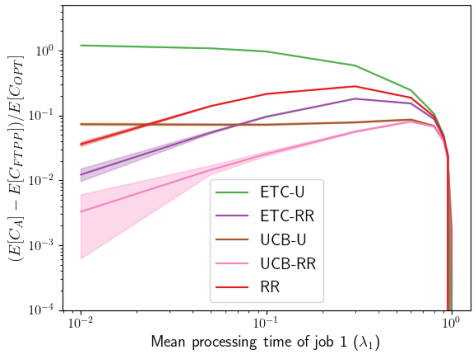


Figure 7.2: Normalized excess cost of all algorithms w.r.t. FTPP with varying value of λ_1, for $\lambda_2 = 1$ and $n = 50$, averaged over $5,000$ seeds.

would indicate that preemptive algorithms have an extra learning cost scaling as the highest mean type. However, we strongly believe this to be an artefact of the analysis.

Given the decomposition, the two proofs diverge.

The analysis of ETC-RR is split into phases, as the analysis of ETC-U. However, the bound on the number of 'bad' jobs computed in each phase requires more care because of independence arguments. Specifically, the upper bound is derived from concentration bounds on the computed statistics, and an additional bound on the number of successful jobs of each type when two types are run in parallel. The details on how to deal with those two non-independent events can be found in Appendix 7.

For UCB-RR, the first step is to distinguish two types of 'failures' of the index. In the first failure case, the index deviates below the true mean. We show that this happens with probability $O(1/n^2)$ (Theorem 7.A.20), independently of Δ. The second type of failure is when the index of a sub-optimal arm is much larger than its true mean. Here, we show that the upper bound on the number of iterations where this happens does diverge as Δ goes to zero. However, the algorithm only incurs a cost on the 'bad pull' of a type when the selected job terminates. The probability of job termination decreases as Δ decreases, which compensates for the rise in the upper bound (Equation (7.32)).

□

7.6 Experiments

In this section, we design synthetic experiments to compare ETC-U, ETC-RR, UCB-RR,UCB-U, RR and FTPP. All code is written in Python. We use matplotlib Hunter (2007) for plotting, and numpy Harris et al. (2020) for array manipulations. The above libraries use open-source licenses. Computations are run on a laptop.

The first experiment plots the CR of each algorithm for two types of jobs and fixed values of λ_1, λ_2 as n varies (see Figure 7.1). Even though all our suggested algorithms have the same asymptotic performance, their non-asymptotic behavior drastically varies. As predicted by theory, the preemptive versions of the algorithm consistently outperform the non-preemptive ones.

In the second experiment, $n = 50$ and $\lambda_2 = 1$ are fixed, while λ_1 varies in $(0, 1)$ (see Figure 7.2). To be able to discern performance gaps when λ_1 is small, we plot the difference between the CR of different algorithms and FTPP at a logarithmic scale. Here, for small values of λ, both preemptive methods outperform the non-preemptive ones. This corresponds with the improvement in the dominant error term of the preemptive cost upper bounds, as a function of

λ_2.

7.6.1 Discussion

Preemptive vs. Non-Preemptive The competitive ratio of all algorithms is asymptotically the one of FTPP. Indeed, it always holds that $\mathrm{E}[C_{\mathrm{OPT}}] \geq (\lambda_1 + \lambda_2)\frac{n^2}{4}$ (Equation (7.6)), so by Propositions 7.5.1 and 7.4.1, for any algorithm A among ETC-U, ETC-RR, UCB-U and UCB-RR:

$$\mathrm{CR}_A = \mathrm{CR}_{\mathrm{FTPP}} + \mathcal{O}\left(\sqrt{\frac{\log(n)}{n}}\right).$$

On the one hand, the leading term in the cost is the same for all algorithms. On the other hand, the error term can be much smaller in the case of preemptive algorithms.

To illustrate this claim, let us consider the case where there are two types of jobs of expected sizes λ_1 and λ_2, respectively. Instantiating the bounds of Propositions 7.4.1 and 7.5.1 to this setting, we get:

$$\begin{aligned}\mathrm{E}[C_{\text{ETC-U}}] \leq \mathrm{E}[C_{\mathrm{FTPP}}] + n(\lambda_1 + \lambda_2)\sqrt{8n\log(2n^2K^3)} \\ + \frac{8}{n}\mathrm{E}[C_{\mathrm{OPT}}],\end{aligned} \tag{7.3}$$

and

$$\begin{aligned}\mathrm{E}[C_{\text{ETC-RR}}] \leq &\mathrm{E}[C_{\mathrm{FTPP}}] + 2n\lambda_1(\sqrt{4n\log(2n^2K^3)} + 1) \\ &+ \frac{16}{n}\mathrm{E}[C_{\mathrm{OPT}}].\end{aligned} \tag{7.4}$$

If $\lambda_2 \gg \lambda_1$ the bound in Equation (7.3) is much larger than the bound in Equation (7.4), which is consistent with what we observe in Figure 7.2. In particular, one can observe that for small λ_1, non-preemptive algorithms converge to a strictly positive error (due to the unavoidable dependence in $\lambda_2 = 1$), while the error of the preemptive algorithms diminishes. This empirically supports our claim that the $n\lambda_K$-dependence, as appears in the preemptive cost decomposition of Theorem 7.5.3, is only due to a proof artefact.

Optimism-based vs. explore-then-commit. In the simulations, we see that optimism-based algorithms perform much better than their ETC counterparts. In traditional bandit settings, it is well known that the regret of ETC strategies is a constant-times larger than that of optimism-based strategies. Here, we believe that in addition to that, a second phenomenon, not reflected in the analysis, renders the optimism-based strategies better than the other ones. Because of the structure of the cost, a pull of a 'bad job' at the beginning is much more expensive than the same pull done later in the interaction (as it delays more jobs). Optimism-based strategies explore continuously as they run, whereas ETC strategies have all the exploration at the beginning, when it is more expensive. Again, this phenomenon stands in contrast with traditional bandits, where only the number of 'bad pulls' matter, and not their position.

7.7 Conclusion and Future Work

In this work, we designed and analyzed a family of algorithms for static scheduling on a single machine in the presence of job types. The special cost structure of this problem differs from

that of traditional bandit problems, and early mistakes carry much more weight than late ones, as they delay more jobs. This modified cost directly impacts the performance of algorithms; although all suggested algorithms asymptotically have the same CR as the optimal algorithm that knows job type sizes (FTPP), their non-asymptotic performances differ.

When preemption is allowed, algorithms that explore job types with a strategy inspired by the worst-case optimal deterministic algorithm RR have a clear advantage over non-preemptive learning algorithms. Thus, because of the cost structure, the performance is impacted not only by the number of exploratory steps but also by the nature of the exploratory steps.

Due to the ubiquitousness of scheduling problems, we believe that our results could be extended to many other variants of this setting. In particular, it would be interesting to take our algorithmic principles and test them on real-world scheduling problems.

Moreover, we believe that elements from our works can be taken to other online learning settings outside the scope of scheduling. Specifically, we believe that the notion of types serves as a reasonable approximation that allows the integration of learning to many online problems. We also think that the study of cost functions that are sensitive the early exploration is of great interest.

When jobs can be preempted, preemptive policies outperform non-preemptive ones. It would be interesting to study what kind of trade-offs apear in the presence of switching costs.

7.A Appendix

7.A.1 Benchmark FTPP

In this section, for all jobs $i \in [N]$, we call P_i the job size of job i. Jobs are ordered in increasing order of their expected size (Notation P_i and $P_{i \pmod n}^{\lceil i/n \rceil}$ denote the same job). For any algorithm A, we note T_{ij}^A for each $(i,j) \in [N]^2$ the amount of time job i and job j delay each other under algorithm A.

Cost of OPT and FTPP, CR of RR

Let us express the expected cost of any algorithm in terms of $T_{i,j}^A$ for $k \in [K]$:

$$\mathrm{E}[C_A] = \mathbb{E}\left[\sum_{i=1}^{N} P_i + \sum_{i=1}^{N}\sum_{j=i+1}^{N} T_{i,j}^A\right] \tag{7.5}$$

Lemma 7.A.1 (Cost of OPT)**.** *The cost of OPT is given by*

$$\mathrm{E}[C_{\mathrm{OPT}}] = n^2\left(\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + \frac{3n}{4}\sum_{\ell=1}^{K}\lambda_\ell.$$

Note that this lemma implies the following inequality, which will be used in other proofs:

$$\mathrm{E}[C_{\mathrm{OPT}}] \geq \frac{n^2}{4}\left(\sum_{\ell=1}^{K}\lambda_\ell\right) \tag{7.6}$$

Proof. We apply Equation (7.5) with $A = \mathrm{OPT}$. In that case, for any to jobs $(i,j) \in [N]$, $i \neq j$, as the shortest job is scheduled first, we have

$$\mathrm{E}[T_{ij}^A] = \mathrm{E}[\min(P_i, P_j)].$$

So $\mathrm{E}[T_{ij}^A] = \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}$ if job i is of type k and job j is of type ℓ.

$$\begin{aligned}
\mathrm{E}[C_{\mathrm{OPT}}] &= \mathbb{E}\left[\sum_{i=1}^{N} P_i + \sum_{i=1}^{N}\sum_{j=i+1}^{N} T_{i,j}^A\right] \\
&= \sum_{\ell=1}^{K}\sum_{i=1}^{n}\left(\lambda_\ell + \sum_{j=i+1}^{n}\frac{\lambda_\ell}{2} + \sum_{k=\ell+1}^{K}\sum_{j=1}^{n}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) \\
&= \sum_{\ell=1}^{K}\left(n\lambda_\ell + \frac{n(n-1)}{2}\frac{\lambda_\ell}{2} + n^2\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) \\
&= n^2\left(\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + \frac{3n}{4}\sum_{k=1}^{K}\lambda_k.
\end{aligned}$$

□

Lemma 7.A.2 (Cost of FTPP)**.** *The cost of FTPP is given by:*

$$\mathrm{E}[C_{\mathrm{FTPP}}] = n^2 \left(\frac{1}{2} \sum_{\ell=1}^{K} \lambda_\ell + \sum_{\ell=1}^{K} (K-\ell)\lambda_\ell \right) + n \left(\frac{\sum_{\ell=1}^{K} \lambda_\ell}{2} \right)$$

Proof. We apply Equation (7.5) with $A = \mathrm{OPT}$, so that $\mathrm{E}[T_{ij}^A] = \min(\lambda_k, \lambda_\ell)$ if job i is of type k and job j is of type ℓ

$$\begin{aligned}
\mathrm{E}[C_{\mathrm{FTPP}}] &= \mathbb{E}\left[\sum_{i=1}^{N} P_i + \sum_{i=1}^{N} \sum_{j=i+1}^{N} T_{i,j}^A \right] \\
&= \sum_{\ell=1}^{K} \sum_{i=1}^{n} \left(\lambda_\ell + \sum_{j=i+1}^{n} \lambda_\ell + \sum_{k=\ell+1}^{K} \sum_{j=1}^{n} \lambda_\ell \right) \\
&= \sum_{\ell=1}^{K} \left(n\lambda_\ell + \frac{n(n-1)}{2} \lambda_\ell + n^2 \sum_{k=\ell+1}^{K} \lambda_\ell \right) \\
&= n^2 \left(\frac{1}{2} \sum_{\ell=1}^{K} \lambda_\ell + \sum_{\ell=1}^{K} (K-\ell)\lambda_\ell \right) + n \left(\frac{\sum_{\ell=1}^{K} \lambda_\ell}{2} \right).
\end{aligned}$$

□

Lemma 7.A.3 (CR of RR)**.** *For any For any* $\boldsymbol{\lambda}$*, the following lower bound holds:*

$$\frac{\mathrm{E}[C_{\mathrm{RR}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} \geq 2 - \frac{4}{n+3}.$$

Proof. For RR, any two jobs are run in parallel until one terminates, thus:

$$\mathrm{E}[T_{ij}^{\mathrm{RR}}] = 2\mathrm{E}[\min(P_i, P_j)].$$

Thus, by equation 7.5:

$$\mathrm{E}[C_{\mathrm{RR}}] = \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{j=1}^{N} 2\mathrm{E}[\min(P_i, P_j)].$$

On the other hand:

$$\mathrm{E}[C_{\mathrm{OPT}}] = \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{j=1}^{N} \mathrm{E}[\min(P_i, P_j)].$$

Thus:

$$\begin{aligned}
\frac{\mathrm{E}[C_{\mathrm{RR}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} &= 2 - \frac{\sum_{i=1}^{N} P_i}{\mathrm{E}[C_{\mathrm{OPT}}]} \\
&= 2 - \frac{n \sum_{\ell=1}^{K} \lambda_\ell}{n^2 \left(\sum_{\ell=1}^{K} \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \frac{\lambda_k \lambda_\ell}{\lambda_k + \lambda_\ell} \right) + \frac{3n}{4} \sum_{\ell=1}^{K} \lambda_\ell} \\
&\geq 2 - \frac{n \sum_{\ell=1}^{K} \lambda_\ell}{n^2 \left(\sum_{\ell=1}^{K} \frac{1}{4}\lambda_\ell \right) + \frac{3n}{4} \sum_{\ell=1}^{K} \lambda_\ell} \\
&= 2 - \frac{4}{n+3}.
\end{aligned}$$

With the second line obtained by Lemma 7.A.1. □

CR of FTPP

CR with K types

Proposition 7.A.4 (Upper bound on the CR in function of $\boldsymbol{\lambda}$)**.** *The CR of FTPP with K types of jobs with n jobs per type satisfies:*

$$\frac{\mathrm{E}[C_{\mathrm{FTPP}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} \leq 2 - f_K(\boldsymbol{\lambda})$$

where $f_K(\boldsymbol{\lambda}) = \frac{2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} - \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell}{\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}}$

Note that instantiating this bound with $K = 2$ types of jobs, n jobs per type, $\lambda_1 = 1$ and $\lambda_2 = \lambda > 1$, we get Theorem 7.3.3:

$$\frac{\mathrm{E}[C_{\mathrm{FTPP}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} \leq 2 - 4\frac{\lambda - 1}{(1+\lambda)^2 + 4\lambda}.$$

Proof of Theorem 7.A.4. Compute $\mathrm{E}[C_{\mathrm{OPT}}]$ using Theorem 7.A.1, $\mathrm{E}[C_{\mathrm{FTPP}}]$ using Theorem 7.A.2.

The competitive ratio of FTPP is given by:

$$\mathrm{CR}_{\mathrm{FTPP}} = \frac{\mathrm{E}[C_{\mathrm{FTPP}}]}{\mathrm{E}[C_{\mathrm{OPT}}]} = \frac{n^2\left(\frac{1}{2}\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell\right) + n(\frac{1}{2}\sum_{\ell=1}^{K}\lambda_\ell)}{n^2\left(\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + n(\frac{3}{4}\sum_{\ell=1}^{K}\lambda_\ell)} \tag{7.7}$$

For any values $a, b, c, d \in \mathbb{R}_+^4$,

$$\text{if } a > c > 0 \text{ and } d > b > 0, \text{ then } \frac{a+b}{c+d} \leq \frac{a}{c}. \tag{7.8}$$

Now, we have $\frac{1}{2} \leq \frac{3}{4}$ and

$$\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} \leq \frac{1}{2}\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell.$$

This implies:

$$\begin{aligned}\mathrm{CR}_{\mathrm{FTPP}} &\leq \frac{\frac{1}{2}\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell}{\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}} \\ &= 2 - \underbrace{\frac{2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} - \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell}{\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{k=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}}}_{f_K(\boldsymbol{\lambda})}.\end{aligned}$$

□

Upper bound on the CR for particular values of $\boldsymbol{\lambda}$

Proposition 7.A.5.

$$\forall K > 1, \exists \boldsymbol{\lambda}, 0 < \lambda_1 \leq \cdots \leq \lambda_K = 1, \mathrm{CR}_{\mathrm{FTPP}}(\boldsymbol{\lambda}) \leq \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K}$$

with $H_K = \sum_{k=1}^{K} \frac{1}{k}$, $B_K = \sum_{k=1}^{K} \frac{1}{k^2}$ *and* $A_K = \sum_{k=1}^{K} \sum_{\ell=1}^{k-1} \frac{1}{k^2+\ell^2}$.

Furthermore $\lim_{K\to\infty} \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K} = \frac{4}{\pi} \approx 1.273$ which implies that there exists some value of K for which $\mathrm{CR}_{\mathrm{FTPP}}(\boldsymbol{\lambda}) \leq 1.274$.

Proof. A way to prove such a result would be to find the minimum of $\mathrm{CR}_{\mathrm{FTPP}}$ with respect to λ. But this is difficult. We propose another point $\tilde{\boldsymbol{\lambda}}$.

$$\tilde{\lambda}_k = \frac{1}{(K-k+1)^2}. \tag{7.9}$$

We express the competitive ratio using $\tilde{\boldsymbol{\lambda}}$:

$$\begin{aligned}
\mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) &= \frac{\sum_{k=1}^{K}(\frac{1}{2} + K - k)\tilde{\lambda}_k}{\sum_{k=1}^{K}\left(\frac{1}{4}\tilde{\lambda}_k + \sum_{\ell=k+1}^{K} \frac{\tilde{\lambda}_k\tilde{\lambda}_\ell}{\tilde{\lambda}_k+\tilde{\lambda}_\ell}\right)} \\
&= \frac{\sum_{k=1}^{K}(\frac{1}{2} + K - k)\frac{1}{(K-k+1)^2}}{\sum_{k=1}^{K}\left(\frac{1}{4}\frac{1}{(K-k+1)^2} + \sum_{\ell=k+1}^{K} \frac{1}{(K-k+1)^2+(K-\ell+1)^2}\right)} \\
&= \frac{\sum_{k=1}^{K}(\frac{1}{2} + k - 1)\frac{1}{k^2}}{\sum_{k=1}^{K}\left(\frac{1}{4}\frac{1}{k^2} + \sum_{\ell=1}^{k-1} \frac{1}{k^2+\ell^2}\right)} \\
&= \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K} \\
\text{with} \quad H_K &= \sum_{k=1}^{K} \frac{1}{k}, \quad B_K = \sum_{k=1}^{K} \frac{1}{k^2}, \quad \text{and} \quad A_K = \sum_{k=1}^{K} \sum_{\ell=1}^{k-1} \frac{1}{k^2+\ell^2}.
\end{aligned}$$

This shows the first part of the lemma. The fact that $\frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K} \to \frac{4}{\pi}$ follows from Theorem 7.A.6. □

Lemma 7.A.6.

$$\lim_{K\to\infty} \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K} = \frac{4}{\pi}$$

with $H_K = \sum_{k=1}^{K} \frac{1}{k}$, $B_K = \sum_{k=1}^{K} \frac{1}{k^2}$ *and* $A_K = \sum_{k=1}^{K} \sum_{\ell=1}^{k-1} \frac{1}{k^2+\ell^2}$

Proof. We now focus on the behavior of $\frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K}$ as K goes to ∞.

We know that for the harmonic number $H_K = \Theta(\log(K))$, and that for the partial sum of the Basel problem $0 \leq B_K \leq \sum_{k=1}^{\infty} k^{-2} = \pi^2/6 = \mathcal{O}(1)$. Let us bound A_k. Using the fact that for $y > 0$ and $x > 0$ the function $f : (x, y) \mapsto (x^2 + y^2)^{-1}$ is decreasing in x, for $(k, \ell) \in [K]^2$ we

have

$$\int_{\ell}^{\ell+1} \frac{1}{k^2+t^2}dt \leq \frac{1}{k^2+\ell^2} \leq \int_{\ell-1}^{\ell} \frac{1}{k^2+t^2}$$
$$\frac{1}{k^2}\int_{\ell}^{\ell+1} \frac{1}{(t/k)^2+1}dt \leq \frac{1}{k^2+\ell^2} \leq \frac{1}{k^2}\int_{\ell-1}^{\ell} \frac{1}{(t/k)^2+1}dt$$
$$\frac{1}{k}(\arctan(\frac{\ell+1}{k}) - \arctan(\frac{\ell}{k})) \leq \frac{1}{k^2+\ell^2} \leq \frac{1}{k}(\arctan(\frac{\ell}{k}) - \arctan(\frac{\ell-1}{k})).$$

Hence by summing for $1 \leq \ell < k \leq K$:

$$\sum_{k=1}^{K} \frac{1}{k}(\arctan(1) - \arctan(\frac{1}{k})) \leq A_K \leq \sum_{k=1}^{K} \frac{1}{k}(\arctan(\frac{k-1}{k}) - \arctan(0)),$$
$$\sum_{k=1}^{K} \frac{1}{k}(\frac{\pi}{4} - \arctan(\frac{1}{k})) \leq A_k \leq \sum_{k=1}^{K} \frac{1}{k}\arctan(\frac{k-1}{k}).$$

For the right-hand-side we use that arctan is increasing, thus

$$A_K \leq \sum_{k=1}^{K} \frac{1}{k}\arctan(\frac{k-1}{k}) \leq \frac{\pi}{4}H_K.$$

Using that $\arctan(x) \leq x$ for $x \geq 0$, we have

$$A_K \geq \sum_{k=1}^{K} \frac{1}{k}(\frac{\pi}{4} - \frac{1}{k}) = \frac{\pi}{4}H_K - B_K.$$

Combining everything we obtain the following inequality:

$$\frac{H_K - \frac{1}{2}B_K}{\frac{\pi}{4}H_K + \frac{1}{4}B_K} \leq \mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) \leq \frac{H_K - \frac{1}{2}B_K}{\frac{\pi}{4}H_K - \frac{3}{4}B_K}.$$

Therefore

$$\lim_{K\to\infty} \mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) = \frac{4}{\pi} \approx 1.273.$$

□

The cost of FTPP is lower than the cost of RR Let us order all jobs $i \in [N]$ in order of their increasing expected size, and denote P_i, the size of job i. An alternative notation to P_i is $P_{i \pmod n}^{\lceil i/n \rceil}$, where the first is used in this proof for convenience. We consider here the most general setting where $K = N$.

We have

Lemma 7.A.7.

$$\mathrm{E}[C_{\mathrm{FTPP}}] \leq \mathrm{E}[C_{\mathrm{RR}}]$$

Proof. The cost of FTPP with $K = N$ and $n = 1$ is given by

$$\begin{aligned}\mathrm{E}[C_{\mathrm{FTPP}}] &= \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathrm{E}\left[T_{ij}^{\mathrm{FTPP}}\right] \\ &= \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \min(\lambda_i, \lambda_j)\end{aligned}$$

where T_{ij}^{FTPP} is the amount of time job i and job j delay each other in FTPP which verifies $\mathrm{E}\left[T_{ij}^{\mathrm{FTPP}}\right] = \min(\lambda_i, \lambda_j)$

Similarly, using $T_{ij}^{\mathrm{RR}} = 2\min(P_i, P_j)$ which implies $\mathrm{E}[T_{ij}^{\mathrm{RR}}] = 2\frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$, we get

$$\begin{aligned}\mathrm{E}[C_{\mathrm{RR}}] &= \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathrm{E}\left[T_{ij}^{\mathrm{RR}}\right] \\ &= \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} 2\frac{\lambda_i\lambda_j}{\lambda_i + \lambda_j}\end{aligned}$$

Then we write

$$\begin{aligned}2\frac{\lambda_i\lambda_j}{\lambda_i + \lambda_j} &= \frac{2}{\frac{1}{\lambda_i} + \frac{1}{\lambda_j}} \\ &\geq \frac{2}{\frac{1}{\min(\lambda_i,\lambda_j)} + \frac{1}{\min(\lambda_i,\lambda_j)}} \\ &\geq \min(\lambda_i, \lambda_j)\end{aligned}$$

We conclude that $C_{\mathrm{FTPP}} \leq C_{\mathrm{RR}}$. □

Lower bound: Proof of Proposition 7.3.1

Let us order all jobs $i \in [N]$ in order of their increasing expected size, and denote P_i, the size of job i. An alternative notation to P_i is $P_{i \pmod n}^{\lceil i/n \rceil}$, where the first is used in this proof for convenience. We consider here the most general setting where $K = N$. Any algorithm has a cost:

$$\mathrm{E}[C^A] = \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathrm{E}[T_{ij}^A]$$

where $T_{ij}^A = D_{ij}^A + D_{ji}^A$ where D_{ij}^A is the amount of time job i delay job j.

Lemma 7.A.8. *Consider $K = N$ jobs where job $i \in [N]$ has mean size λ_i and $\lambda_1 \leq \cdots \leq \lambda_N$. Consider any algorithm A and let T_{ij}^A the total amount of time spent by A on i or j while both jobs are alive.*

$$\mathrm{E}[T_{ij}^A] \geq 2\mathrm{E}[T_{ij}^{OPT}]$$

where OPT *is the optimal offline algorithm*

Proof of Lemma 7.A.8. Let us first prove our proposition for any deterministic algorithm A. We denote $i(t)$ amount of time that A allocates to job i after a time $t < T_{ij}^A$ is allocated to job i or j.

$$\begin{aligned}
&\mathrm{E}[T_{ij}^A] \\
&= \int_{t=0}^{+\infty} \mathbf{P}(T_{ij}^A \geq t)dt \\
&= \int_{t=0}^{+\infty} \mathbf{P}\left(P_i \geq i(t)\right) \mathbf{P}\left(P_j \geq t - i(t)\right) dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\frac{i(t)}{\lambda_i}\right) \exp\left(-\frac{t-i(t)}{\lambda_j}\right) dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\frac{i(t)+t/2-t/2}{\lambda_i}\right) \exp\left(-\frac{t-(i(t)+t/2-t/2)}{\lambda_j}\right) dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\left(\frac{1}{\lambda_i}+\frac{1}{\lambda_j}\right)\frac{t}{2}\right) \exp\left(-\left(\frac{1}{\lambda_i}-\frac{1}{\lambda_j}\right)\left(i(t)-\frac{t}{2}\right)\right) dt.
\end{aligned}$$

Calling $f(t) = \exp\left(-\left(\frac{1}{\lambda_i}+\frac{1}{\lambda_j}\right)\frac{t}{2}\right)$ and $g(t) = |\frac{1}{\lambda_i}-\frac{1}{\lambda_j}|\left(i(t)-\frac{t}{2}\right)$ it holds that either

$$\int_{t=0}^{\infty} f(t)\exp(-g(t))dt \geq \int_{t=0}^{\infty} f(t)dt$$

or

$$\int_{t=0}^{\infty} f(t)\exp(g(t))dt \geq \int_{t=0}^{\infty} f(t)dt.$$

Otherwise, we would have

$$\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t))+\exp(g(t)))dt < \int_{t=0}^{\infty} f(t)dt$$

which cannot be true since $\forall t$, $\frac{1}{2}(\exp(-t)+\exp(t)) \geq 1$.

Therefore an adversary knowing $i(t)$ can always chose the order of λ_i and λ_j such that

$$\mathrm{E}[T_{ij}^A] \geq \int_{t=0}^{+\infty} \exp(-(\frac{1}{\lambda_i}+\frac{1}{\lambda_j})\frac{t}{2})dt = 2\frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$$

The optimal delay is

$$\mathrm{E}[T_{ij}^{OPT}] = \mathrm{E}[\min(P_i, P_j)] = \frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$$

so our Lemma is proven for any deterministic algorithm A.

Consider a randomized algorithm R which can be seen as a probabilistic distribution over the set of deterministic algorithms. Therefore A, $i(t)$ and $g(t)$ are now seen as random variables. By the tower rule, the amount of time job i and j delay each other in R is such that:

$$\begin{aligned}
\mathrm{E}[T_{ij}^R] &= \mathrm{E}[\mathrm{E}[T_{ij}^A|A]] \\
&= \mathrm{E}[\int_{t=0}^{+\infty} f(t)\exp(\mathrm{sign}(\lambda_i-\lambda_j)g(t))dt]
\end{aligned}$$

By the same argument as in the deterministic case, it holds that either

$$\mathrm{E}[\int_{t=0}^{\infty} f(t)\exp(-g(t))dt] \geq \int_{t=0}^{\infty} f(t)dt$$

or

$$\mathrm{E}[\int_{t=0}^{\infty} f(t)\exp(g(t))dt] \geq \int_{t=0}^{\infty} f(t)dt$$

Otherwise, we would have

$$\mathrm{E}[\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t)) + \exp(g(t)))dt] < \int_{t=0}^{\infty} f(t)dt$$

which implies that there exists a deterministic function g such that

$$\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t)) + \exp(g(t)))dt < \int_{t=0}^{\infty} f(t)dt$$

which cannot be true as shown in the deterministic case. The rest of the argument is the same as in the deterministic case and therefore omitted. □

Now we are ready to prove Proposition 7.3.1.

Proof of Proposition 7.3.1. Take any algorithm A

$$\mathrm{E}[C_A] = \sum_{i=1}^{N} \mathrm{E}[P_i] + \sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathrm{E}[T_{ij}^{A}] \tag{7.10}$$

$$\geq \sum_{i=1}^{N} \lambda_i + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathrm{E}[T_{ij}^{OPT}] \tag{7.11}$$

where (7.11) comes from Lemma 7.A.8.

Observe that applying RR on the same data would yield an expected completion time:

$$\begin{aligned}\mathrm{E}[C_{\mathrm{RR}}] &= \sum_{i=1}^{N} \mathrm{E}[P_i] + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathrm{E}[\min(P_i, P_j)] \\ &= \sum_{i=1}^{N} \mathrm{E}[P_i] + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathrm{E}[T_{ij}^{OPT}] \\ &\leq \mathrm{E}[C_A]\end{aligned}$$

which concludes the proof. □

7.A.2 Analysis of Non-Preemptive Learning algorithms

Full Algorithmic Details

In this appendix, we present a full description of ETC-U and UCB-U.

Algorithm 31: Explore-Then-Commit Uniform (ETC-U)]

Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types);
For all pairs of different types k, ℓ initialize $\delta_{k,\ell} = 0$, $\hat{r}_{k,\ell} = 0$ and $h_{k,\ell} = 0$;
For all types k, set $m_k = 0$;
while $\mathcal{U}$ *is not empty* **do**
  $\mathcal{U}$ is the set of types with at least one remaining job;
  if $\mathcal{A}$ *is empty* **then**
    $\mathcal{A} = \{\ell \in \mathcal{U}, \forall k \in \mathcal{U}, k \neq \ell, \ \hat{r}_{k,\ell} - \delta_{k,\ell} \leq 0.5\}$;
  end
  Select the type ℓ with the lowest number of finished jobs $\ell = \arg\min_{k \in \mathcal{A}} m_k$ and run one job of type ℓ yielding a size $P^{\ell}_{m_\ell + 1}$.;
  $m_\ell = m_\ell + 1$;
  for k, ℓ *in* $\mathcal{A}$, $k \neq \ell$ **do**
    $h_{k,\ell} = \sum_{i=1}^{\min(m_k, m_\ell)} \mathbb{1}\{P^k_i < P^\ell_i\}$;
    $\delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^4)}{2\min(m_k, m_\ell)}}$;
    $\hat{r}_{k,\ell} = \frac{h_{k,\ell}}{\min(m_k, m_\ell)}$;
    if $\hat{r}_{k,\ell} - \delta_{k,\ell} \geq 0.5$ *or* $m_\ell = n$ **then**
      Remove ℓ from $\mathcal{A}$;
    end
  end
end

Algorithm 32: Upper-Confidence-Bound-Uniform (UCB-U)

Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types);
For all types $k \in [K]$, set $m_k = 0$;
Set $\mathcal{U} = [K]$;
For all types $k \in [K]$, compute the lower bound $\underline{\lambda}^{m_k}_k$ using Equation (7.16);
while $\mathcal{U}$ *is empty* **do**
  Select $k^* = \arg\min_{k \in \mathcal{U}} \underline{\lambda}^{m_k}_k$;
  Set $m_{k^*} = m_{k^*} + 1$;
  Compute a job of type k^* until completion and record its size $P^{m_{k^*}}_{k^*}$;
  Update the lower bound $\underline{\lambda}^{m_{k^*}}_{k^*}$ using again Equation (7.16);
  If $m_{k^*} = n$, remove k^* from $\mathcal{U}$;
end

Cost Decomposition

In this appendix, we analyze the non-preemptive learning algorithms presented in our chapter - ETC-U and UCB-U. We start by presenting a general cost decomposition result that relates the cost of any non-preemptive algorithm to the one of FTPP. We will use this result to derive the bounds of both our suggested algorithms.

Lemma 7.A.9 (Cost of non-preemptive algorithms)**.** *Any non-preemptive algorithm A has a cost*

$$\mathrm{E}[C_A] = \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right]$$

Proof. Denote P_i, the size of the job i, and $T_{ij}^A = D_{ij}^A + D_{ji}^A$, where D_{ij}^A is the amount of time a job i delays job j. For any algorithm, we have:

$$C_A = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N}\sum_{b=a+1}^{N} T_{ab}^A$$

For non-preemptive algorithms, $T_{ab}^A = P_a$ if job a is scheduled before b and P_b otherwise so that we can write

$$C_A = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N}\sum_{b=a+1}^{N}\left(P_a\mathbb{1}\{P_a \text{ computed before } P_b\} + P_b\mathbb{1}\{P_b \text{ computed before } P_a\}\right)$$

Now assume w.l.o.g. that $(P_a)_{a\in[N]}$ are in the order chosen by FTPP, i.e., P_a is the a^{th} executed task by FTPP and if $a \le b$ then $\mathrm{E}[P_a] \le \mathrm{E}[P_b]$. Under this convention, we get:

$$C_{\mathrm{FTPP}} = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N}\sum_{b=a+1}^{N} P_a$$

and recalling that

$$\mathbb{1}\{P_a \text{ computed before } P_b\} = 1 - \mathbb{1}\{P_b \text{ computed before } P_a\}$$

we have

$$C_A = C_{\mathrm{FTPP}} + \sum_{a=1}^{N}\sum_{b=a+1}^{N}(P_b - P_a)\mathbb{1}\{P_b \text{ computed before } P_a\}$$

Reindexing the job without changing the order, where P_i^k is now the i-th job of type k, we have:

$$C_A = C_{\mathrm{FTPP}} + \sum_{\ell=1}^{K}\sum_{j=1}^{n}\sum_{k=\ell+1}^{K}\sum_{i=1}^{n}(P_i^k - P_j^\ell)\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}$$

Taking the expectation finishes the proof. □

Upper bound for ETC-U

Proposition 7.A.10. *The following upper bounds hold:*

$$\mathrm{E}[C_{\text{ETC-U}}] \le \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{1}{n}\mathrm{E}[C_{\mathrm{OPT}}] + \sum_{k\in[K]}\left[\frac{1}{2}(k-1)(2K-k) + (K-k)^2\right]\lambda_k n\sqrt{8n\log(2n^2K^3)}.$$

and

$$\mathrm{E}[C_{\text{ETC-U}}] \leq \mathrm{E}[C_{\text{FTPP}}] + \frac{1}{n}\mathrm{E}[C_{\text{OPT}}] + \sum_{k\in[K]}\sum_{\ell=1}^{k-1}(K-\ell)\frac{(\lambda_k+\lambda_\ell)^2}{(\lambda_k-\lambda_\ell)}n8\log(2n^2K^3).$$

We start with the following technical lemma, isolated to be reused in other proofs. Pick some $\alpha \in \mathbb{N}$. Let $(X_i^1)_{i\in[\alpha n]}$ and $(X_i^2)_{i\in[\alpha n]}$ be independent exponential variables of parameters λ_1 and λ_2 respectively. Define for any $m \in [\alpha n]$:

$$\hat{r}^m = \frac{1}{m}\sum_{i=1}^{m}\mathbb{1}_{X_i^1<X_i^2}$$

and

$$\delta^{(m,n)} = \sqrt{\frac{\log(2n^2K^3)}{2m}}.$$

Let r denote the expectation $r := \mathrm{E}\left[\mathbb{1}_{X_i^1<X_i^2}\right] = \frac{\lambda_2}{\lambda_1+\lambda_2}$.

Lemma 7.A.11. *For any $m \in [\alpha n]$, the estimator $\hat{r}^m$ is within $\delta^{(m,n)}$ of its expectation w.h.p:*

$$\mathbf{P}\left(\exists m \in [\alpha n] \text{ s.t. } |\hat{r}^m - r| \geq \delta^{(m,n)}\right) \leq \frac{\alpha}{nK^3}.$$

Proof. By Hoeffding's inequality:

$$\forall m \in [\alpha n],\ \mathbf{P}\left(|\hat{r}^m - r| \geq \sqrt{\frac{\log(2n^2K^3)}{2m}}\right) \leq \frac{1}{n^2K^3}$$

The lemma is then obtained by a union bound over the αn possible values of m. □

We are now ready to prove Proposition 7.A.10.

Proof. Recall that we assumed without loss of generality that $\lambda_1 \leq \cdots \leq \lambda_K$. Recall also the definition for any $(k,\ell) \in [K]^2$, for any $(m_\ell, m_k) \in [n]^2$, of:

$$\hat{r}_{k,\ell}^{\min(m_k,m_\ell)} = \frac{1}{\min(m_k,m_\ell)}\sum_{i=1}^{\min(m_k,m_\ell)}\mathbb{1}_{P_i^k<P_i^\ell}.$$

Let us define the good event $\mathcal{E}$ as:

$$\mathcal{E} := \left\{\forall(k,\ell)\in[K]^2, \forall m \in [n], |\hat{r}_{k,\ell}^m - \mathrm{E}[r_{k,\ell}^m]| < \delta^{(m,n)}\right\}$$

By Lemma 7.A.11 applied with $\alpha = 1$, for any couple (ℓ, k) it holds that :

$$\mathbf{P}\left(\exists m \in [n] \text{ s.t. } |\hat{r}_{k,\ell}^m - \mathrm{E}[r_{k,\ell}^m]| > \delta^{(m,n)}\right) \leq \frac{1}{n}.$$

A union bound over the $\frac{K(K-1)}{2}$ possible pairs gives the following bound:

$$\mathbf{P}\left(\overline{\mathcal{E}}\right) \leq \frac{1}{2nK}. \tag{7.12}$$

With the help of Theorem 7.A.9, the cost of ETC-U can be decomposed using the event $\mathcal{E}$ as follows:

$$\mathrm{E}[C_{\text{ETC-U}}] = \mathrm{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right]$$

(Theorem 7.A.9)

$$\leq \mathrm{E}[C_{\text{FTPP}}] + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\mathcal{E}\right]}_{(i)} + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\overline{\mathcal{E}}\right]\mathbf{P}\left(\overline{\mathcal{E}}\right)}_{(ii)}. \tag{7.13}$$

Bounding (ii)**.** Recall that by assumption, if $k \geq \ell$, then $\lambda_k \geq \lambda_\ell$. Therefore, we have that

$$\begin{aligned}
(ii) &= \mathbf{P}\left(\overline{\mathcal{E}}\right)\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\underbrace{(\lambda_k-\lambda_\ell)}_{\geq 0}\mathrm{E}\left[\underbrace{\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}}_{\leq 1}|\mathcal{E}\right] \\
&\leq \mathbf{P}\left(\overline{\mathcal{E}}\right)\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell) \\
&= n^2\mathbf{P}\left(\overline{\mathcal{E}}\right)\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell) \\
&= n^2\mathbf{P}\left(\overline{\mathcal{E}}\right)\left(\sum_{k=1}^{K}(k-1)\lambda_k - \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell\right) \\
&\leq n^2K\mathbf{P}\left(\overline{\mathcal{E}}\right)\sum_{k=1}^{K}\lambda_k \\
&\leq 4K\mathrm{E}[C_{\text{OPT}}]\mathbf{P}\left(\overline{\mathcal{E}}\right) && \text{(Equation (7.6))} \\
&\leq \frac{2}{n}\mathrm{E}[C_{\text{OPT}}], && (7.14)
\end{aligned}$$

where the last inequality is by Equation (7.12).

Bounding (i)**.** Consider any couple $(k,\ell) \in [K]^2$ s.t. $\ell \leq k$. Let $m^*_{\ell,k}$ be the number of comparisons performed between jobs of type ℓ and k before the algorithm detects that $\lambda_\ell \leq \lambda_k$. A first obvious upper bound is $m^*_{\ell,k} \leq n$. A second upper is obtained by noting that $m^*_{\ell,k}$ is smaller than any m' s.t.

$$\delta^{(m',n)} < \frac{1}{2}\left|\frac{\lambda_k}{\lambda_k+\lambda_\ell} - 0.5\right|.$$

For this value of $\delta^{(m',n)}$, the event $\mathcal{E}$ ensures that if $\lambda_k \geq \lambda_\ell$, then

$$\hat{r}^{m'}_{\ell,k} - \delta^{(m',n)} \overset{\text{Under } \mathcal{E}}{\geq} \mathrm{E}[r^{m'}_{\ell,k}] - 2\delta^{(m',n)} > \frac{\lambda_k}{\lambda_k+\lambda_\ell} - \left(\frac{\lambda_k}{\lambda_k+\lambda_\ell} - \frac{1}{2}\right) = \frac{1}{2},$$

and type k would be eliminated. This implies the following upper bound on $m^*_{\ell,k}$:

$$m^*_{\ell,k} \leq \min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2 \log\left(2n^2K^3\right)\right). \tag{7.15}$$

On the other hand, notice that under the good event $\mathcal{E}$, a type ℓ will never be eliminated due to a type k of greater expected duration $\lambda_k \geq \lambda_\ell$, since

$$\hat{r}^{m'}_{k,\ell} - \delta^{(m',n)} \overset{\text{Under } \mathcal{E}}{\leq} \left(\mathrm{E}[r^{m'}_{\ell,k}] + \delta^{(m',n)}\right) - \delta^{(m',n)} = \frac{\lambda_\ell}{\lambda_k+\lambda_\ell} \leq \frac{1}{2}.$$

We decompose the run of the algorithm into (up to K) phases. For each $\ell \in [K]$, we call phase ℓ the iterations at which jobs of type ℓ are the jobs with the smallest mean still not terminated. Note that during phase ℓ, job type ℓ is always active, as the contrary would mean event $\mathcal{E}$ does not hold. This implies that the number of jobs of any type $k > \ell$ computed during phase ℓ is lower than $m^*_{\ell,k}$. We have the following bound:

$$\begin{aligned}
(i) &= \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\mathcal{E}\right] \\
&\overset{(1)}{\leq} \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before phase } \ell+1\right\}|\mathcal{E}\right] \\
&\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{i\in[n]} n(\lambda_k-\lambda_\ell)\mathrm{E}\left[\sum_{o=1}^{\ell}\mathbb{1}\left\{P_i^k \text{ computed during phase } o\right\}|\mathcal{E}\right] \\
&\overset{(2)}{\leq} \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}\mathrm{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_\ell) \\
&\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}\mathrm{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_o) \\
&\overset{(3)}{=} \sum_{k=1}^{K}\sum_{o=1}^{k-1}\sum_{l=o}^{k-1}\mathrm{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_o) \\
&= \sum_{k=1}^{K}\sum_{o=1}^{k-1}\mathrm{E}[m^*_{o,k}|\mathcal{E}]n(k-o)(\lambda_k-\lambda_o) \\
&\overset{(4)}{\leq} \sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\mathrm{E}[m^*_{\ell,k}|\mathcal{E}]n(K-\ell)(\lambda_k-\lambda_\ell) \\
&\overset{(5)}{\leq} \sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2\log\left(2n^2K^3\right)\right)n(K-\ell)(\lambda_k-\lambda_\ell).
\end{aligned}$$

(1) is since by the beginning of phase $\ell+1$, all jobs of type ℓ were completed. (2) is since during phase o, the o^{th} type was not eliminated, so there cannot be more than $m^*_{o,k}$ jobs of type k in this phase. In (3), we changed the summation order and in (4), we replaced $o \to \ell$. Finally, (5) is due to the bound of Equation (7.15), which holds under $\mathcal{E}$.

Next, for any $\lambda_k \geq \lambda_\ell$, we have:

$$(\lambda_k - \lambda_\ell) \min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2 \log\left(2n^2K^3\right)\right) \leq (\lambda_k+\lambda_\ell)\sqrt{8n\log(2n^2K^3)},$$

since $\min\{a,b\} \leq \sqrt{ab}$ for any $a, b \geq 0$. This implies that

$$\begin{aligned}(i) &\leq \sum_{k=1}^{K}\sum_{\ell=1}^{k-1} n(K-\ell)(\lambda_k+\lambda_\ell)\sqrt{8n\log(2n^2K^3)} \\ &= \sum_{k=1}^{K}\left[\frac{1}{2}(k-1)(2K-k)+(K-k)^2\right]\lambda_k n\sqrt{8n\log(2n^2K^3)}.\end{aligned}$$

Substituting this and the bound of Equation (7.14) into the decomposition of Equation (7.13) gives the first bound of the proposition.

The second bound is obtained by upper bounding:

$$(\lambda_k - \lambda_\ell) \min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2 \log\left(2n^2K^3\right)\right) \leq 8\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}\log\left(2n^2K^3\right),$$

□

Upper bound for UCB-U

Proposition 7.A.12. *The expected cost of* UCB-U *is upper bounded by:*

$$\mathrm{E}[C_{\text{UCB-U}}] \leq \mathrm{E}[C_{\text{FTPP}}] + n(K-1)\sqrt{3n\ln\left(2n^2K^2\right)}\sum_{k=1}^{K}\lambda_k + \frac{2}{n}\mathrm{E}[C_{\text{OPT}}],$$

and:

$$\mathrm{E}[C_{\text{UCB-U}}] \leq \mathrm{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}3n\ln\left(2n^2K^2\right) + \frac{2}{n}\mathrm{E}[C_{\text{OPT}}].$$

Concentration of exponential distribution If $X \sim \mathcal{E}(\lambda)$, then $\frac{2}{\lambda}X \sim \mathcal{E}(2) = \chi^2_2$ (χ^2 with 2 degrees of freedom). It follows that if $\forall i \in [m], X_i \sim \mathcal{E}(\lambda)$, then $\frac{2}{\lambda}\sum_{i=1}^{m} X_i \sim \chi^2_{2m}$. Denote $\chi^2_{2m}(\alpha)$ the α-th percentile, we have with probability $1-\delta$ that

$$\frac{2\sum_i X_i}{\chi^2_{2m}(1-\delta/2)} \leq \lambda \leq \frac{2\sum_i X_i}{\chi^2_{2m}(\delta/2)}$$

Setting $\delta = \frac{1}{n^2K^2}$, we get the following formula for a lower bound:

$$\underline{\lambda}^m_k = \frac{2\sum_{i=1}^{m} X^k_i}{\chi^2_{2m}(1-\frac{1}{2n^2K^2})} \tag{7.16}$$

and another formula for the upper bound

$$\overline{\lambda}_k^m = \frac{2\sum_{i=1}^m X_i^k}{\chi^2_{2m}(\frac{1}{2n^2K^2})}$$

If a job k is wrongly scheduled before a job of type ℓ, then the decision rule is misleading meaning that:

$$\underline{\lambda}_k^{m_k} = \frac{2\sum_{i=1}^{n_k} X_i^k}{\chi^2_{2n_k}(1-\frac{1}{2n^2K^2})} < \frac{2\sum_{i=1}^{n_\ell} X_i^\ell}{\chi^2_{2n_\ell}(1-\frac{1}{2n^2K^2})} = \underline{\lambda}_\ell^{m_\ell}$$

even though $\lambda_\ell < \lambda_k$.

Bounding the cost From Theorem 7.A.9, the cost of any non preemptive algorithm A writes

$$\mathrm{E}[C_A] = \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right] \tag{7.17}$$

$$\tag{7.18}$$

Let us then introduce the GOOD event which is:

$$\mathcal{E} = \{\forall i\in[n], \forall k\in[K], \underline{\lambda}_k^i \le \lambda_k \le \overline{\lambda}_k^i\}$$

With a union bound, it is easy to show that $\mathcal{E}$ holds with probability $1-\frac{1}{nK}$ and that the contradictory event $\overline{\mathcal{E}}$ happens with probability $\frac{1}{nK}$.

Using the same method as in the proof of Proposition 7.A.10 (the decomposition using $\mathcal{E}$ and $\overline{\mathcal{E}}$ as done in Equation (7.13) and the derivation of Equation (7.14)), we can upper bound the cost of UCB-U as:

$$\mathrm{E}[C_{UCB-U}] \le \mathrm{E}[C_{\mathrm{FTPP}}] + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\mathcal{E}\right]}_{(i)}$$
$$+ \mathrm{E}[C_{OPT}]\underbrace{4KP(\overline{\mathcal{E}})}_{4/n}$$

Furthermore, P_i^k computed before P_j^ℓ implies that $\underline{\lambda}_k^i < \underline{\lambda}_\ell^j$ and therefore

$$(i) \le \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbf{P}(\underline{\lambda}_k^i < \underline{\lambda}_\ell^j|\mathcal{E})$$

Under $\mathcal{E}$, we have $\underline{\lambda}_\ell^j \le \lambda_\ell$. Moreover, it holds that $\overline{\lambda}_k^i \ge \lambda_k$, and by the definition of $\underline{\lambda}_k^i$, and $\overline{\lambda}_k^i$,

$$\underline{\lambda}_k^i = \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})}\overline{\lambda}_k^i \ge \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})}\lambda_k.$$

Combined, under $\mathcal{E}$ we can bound $\left\{\underline{\lambda}_k^i < \underline{\lambda}_\ell^j\right\} \subseteq \left\{\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}$ and therefore write

$$\begin{aligned}(i) &\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2} \mathbb{1}\left\{\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\} \\ &= \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n \max\left\{i\in[n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}(\lambda_k-\lambda_\ell)\end{aligned}$$

So finally we have

$$\mathrm{E}[C_{\text{UCB-U}}] \leq \mathrm{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n \max\left\{i\in[n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}(\lambda_k-\lambda_\ell) + \frac{4}{n}\mathrm{E}[C_{\text{OPT}}] \tag{7.19}$$

Bounding the ratio. We now focus on bounding the maximum term in Equation (7.19). By Lemma 1 of (Laurent and Massart, 2000), if $U \sim \chi^2_D$, then

$$\mathbf{P}\left(U \geq D + 2\sqrt{Dx} + 2x\right) \leq \exp(-x), \qquad \text{and} \qquad \mathbf{P}\left(U \leq D - 2\sqrt{Dx}\right) \leq \exp(-x), \tag{7.20}$$

and in particular,

$$\chi^2_D(\delta) \geq D - 2\sqrt{D\ln\frac{1}{\delta}}, \qquad \text{and} \qquad \chi^2_D(1-\delta) \leq D + 2\sqrt{D\ln\frac{1}{\delta}} + 2\ln\frac{1}{\delta}. \tag{7.21}$$

Thus, for any $i \in [n]$, a necessary condition to the inequality $\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell$ is

$$\begin{aligned}&\frac{2i - 2\sqrt{2i\ln\left(2n^2K^2\right)}}{2i + 2\sqrt{2i\ln\left(2n^2K^2\right)} + 2\ln\left(2n^2K^2\right)} < \frac{\lambda_\ell}{\lambda_k} \\ \Rightarrow &\left(1-\frac{\lambda_\ell}{\lambda_k}\right) i - \sqrt{2\ln\left(2n^2K^2\right)}\left(1+\frac{\lambda_\ell}{\lambda_k}\right)\sqrt{i} - \frac{\lambda_\ell}{\lambda_k}\ln\left(2n^2K^2\right) < 0 \\ \Rightarrow &(\lambda_k-\lambda_\ell)\, i - \sqrt{2\ln\left(2n^2K^2\right)}\,(\lambda_k+\lambda_\ell)\sqrt{i} - \lambda_\ell \ln\left(2n^2K^2\right) < 0 \\ \Rightarrow &\sqrt{i} < \frac{\sqrt{2\ln\left(2n^2K^2\right)}\,(\lambda_k+\lambda_\ell) + \sqrt{2\ln\left(2n^2K^2\right)(\lambda_k+\lambda_\ell)^2 + 4\ln\left(2n^2K^2\right)\lambda_\ell\,(\lambda_k-\lambda_\ell)}}{2\,(\lambda_k-\lambda_\ell)} \\ \Rightarrow &\sqrt{i} < \sqrt{2\ln\left(2n^2K^2\right)}\frac{(\lambda_k+\lambda_\ell) + \sqrt{(\lambda_k+\lambda_\ell)^2 + 2\lambda_\ell\,(\lambda_k-\lambda_\ell)}}{2\,(\lambda_k-\lambda_\ell)}\end{aligned}$$

Now, using the fact that $2\lambda_\ell \leq \lambda_\ell + \lambda_k$, we get the simplified bound

$$\sqrt{i} < \sqrt{2\ln\left(2n^2K^2\right)}\frac{(\lambda_k+\lambda_\ell) + \sqrt{2\lambda_k\,(\lambda_k+\lambda_\ell)}}{2\,(\lambda_k-\lambda_\ell)} \leq \sqrt{2\ln\left(2n^2K^2\right)}\left(1+\sqrt{2}\right)\frac{\lambda_k+\lambda_\ell}{2\,(\lambda_k-\lambda_\ell)}\ , \tag{7.22}$$

or $i \leq 3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2$. Since we also know that $i \in [n]$, we can write

$$\max\left\{i \in [n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\} \leq \min\left\{3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2, n\right\}$$
$$\leq \sqrt{3n\ln\left(2n^2K^2\right)}\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell},$$

where the second inequality is since $\min\{a,b\} \leq \sqrt{ab}$ for $a,b>0$. Substituting back into Equation (7.19), we get the first bound in the proposition:

$$\begin{aligned}
\mathrm{E}[C_{\text{UCB-U}}] &\leq \mathrm{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\sqrt{3n\ln\left(2n^2K^2\right)}\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}(\lambda_k-\lambda_\ell) + \frac{4}{n}\mathrm{E}[C_{\text{OPT}}] \\
&= \mathrm{E}[C_{\text{FTPP}}] + n\sqrt{3n\ln\left(2n^2K^2\right)}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k+\lambda_\ell) + \frac{4}{n}\mathrm{E}[C_{\text{OPT}}] \\
&= \mathrm{E}[C_{\text{FTPP}}] + n\sqrt{3n\ln\left(2n^2K^2\right)}\left(\sum_{k=1}^{K}(k-1)\lambda_k + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell\right) + \frac{4}{n}\mathrm{E}[C_{\text{OPT}}] \\
&= \mathrm{E}[C_{\text{FTPP}}] + n(K-1)\sqrt{3n\ln\left(2n^2K^2\right)}\sum_{k=1}^{K}\lambda_k + \frac{4}{n}\mathrm{E}[C_{\text{OPT}}].
\end{aligned}$$

The second bound is obtained through the upper bound:

$$(\lambda_k-\lambda_\ell)\min\left\{3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2, n\right\} \leq 3\ln\left(2n^2K^2\right)\left(\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}\right).$$

Lower bounds for Non-Preemptive Algorithms

Small Differences (Theorem 7.4.4)

Proof of Theorem 7.4.4. Assume $K=2$ and take any non-preemptive algorithm A. Call P_i^1 the i-th job of type 1 and P_j^2 the j-th job of type 2. According to Theorem 7.A.9, A has a cost

$$\mathrm{E}[C_A] = \mathrm{E}[C_{\text{FTPP}}] + (\lambda_2-\lambda_1)\mathrm{E}\left[\sum_{(i,j)\in[n]^2}\mathbb{1}\left\{P_i^1 \text{ computed before } P_j^2\right\}\right]$$

if $\lambda_2 > \lambda_1$ (the role of λ_2 and λ_1 are reversed if $\lambda_2 < \lambda_1$).

We then follow the same approach as in chapter 15 in Lattimore and Szepesvári (2020). Consider situation 1 where $\lambda_1 = a, \lambda_2 = b$ and situation 2 where $\lambda_1 = b$ and $\lambda_2 = a$ with $a < b$ and assumes that the adversary chooses the situation based on the algorithm A. Call $\mathbf{P}_{\nu_1}$ the joint probability over the scheduling decisions and job sizes in situation 1 following the policy prescribed by algorithm A and $\mathbf{P}_{\nu_2}$ the same probability in situation 2. Call $P_{a_t}(x_t)$ the probability that the job of type a_t chosen at time t is of size x_t. Calling KL the KL divergence, we have following the Lemma 15.1 in Lattimore and Szepesvári (2020): $KL(\mathbf{P}_{\nu_1}, \mathbf{P}_{\nu_2}) = n(KL(X_a, X_b) + KL(X_b, X_a))$ where X_a is an exponential random variable of expectation a and X_b is an exponential random variable of expectation b.

Note right away that $KL(X_a, X_b) = \frac{a}{b} - 1 - \log(\frac{a}{b})$ (e.g., Calin and Udrişte, 2014, Example

4.2.1), therefore $KL(X_a, X_b) + KL(X_b, X_a) = \frac{a}{b} - 2 + \frac{b}{a} = \frac{(b-a)^2}{ab}$ so

$$KL(\mathbf{P}_{\nu_1}, \mathbf{P}_{\nu_2}) \leq n\frac{(b-a)^2}{ab}.$$

The cost of algorithm A in situation 1 is lower bounded as:

$$\mathrm{E}_{\nu_1}[C_A] \geq \mathrm{E}_{\nu_1}[C_{\mathrm{FTPP}}] + (b-a)\mathrm{E}_{\nu_1}\left[\mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\left\{P_i^1 \text{ computed before } P_j^2\right\} \geq n^2/2\right\}\right] n^2/2.$$

The cost of algorithm A in situation 2 is lower bounded as:

$$\mathrm{E}_{\nu_1}[C_A] \geq \mathrm{E}_{\nu_2}[C_{\mathrm{FTPP}}] + (b-a)\mathrm{E}_{\nu_2}\left[\mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\left\{P_j^2 \text{ computed before } P_i^1\right\} \geq n^2/2\right\}\right] n^2/2.$$

Introduce the event $E = \mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\left\{P_i^1 \text{ computed before } P_j^2\right\} \geq n^2/2\right\}$, we have that

$$\begin{aligned}
\mathrm{E}[C_A] &= \max_{\nu\in\{\nu_1,\nu_2\}} \mathrm{E}_\nu[C_A] \\
&\geq \frac{\mathrm{E}_{\nu_1}[C_A] + \mathrm{E}_{\nu_2}[C_A]}{2} \\
&\geq \frac{\mathrm{E}_{\nu_1}[C_{\mathrm{FTPP}}] + \mathrm{E}_{\nu_2}[C_{\mathrm{FTPP}}]}{2} + (b-a)n^2/2\frac{\mathbf{P}_{\nu_1}(E) + \mathbf{P}_{\nu_2}(\overline{E})}{2}
\end{aligned}$$

First, let us notice that $\mathrm{E}[C_{\mathrm{FTPP}}] = \frac{\mathrm{E}_{\nu_1}[C_{\mathrm{FTPP}}]+\mathrm{E}_{\nu_2}[C_{\mathrm{FTPP}}]}{2}$. Then, using Bretagnolle–Huber inequality (Th 14.2 in Lattimore and Szepesvári (2020)), we get $\mathbf{P}_{\nu_1}(E)+\mathbf{P}_{\nu_2}(\overline{E}) \geq \frac{1}{2}\exp(-KL(\mathbf{P}_{\nu_1}, \mathbf{P}_{\nu_2}))$ and since $KL(\mathbf{P}_{\nu_1}, \mathbf{P}_{\nu_2}) \leq n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2}$, we have

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + (b-a)n^2/2\frac{\exp(-n\frac{(b-a)^2}{ab})}{4}$$

At this stage, we can rewrite the equation assuming $\lambda_2 \geq \lambda_1$ and so that we get

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + (\lambda_2-\lambda_1)n^2/2\frac{\exp(-n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2})}{4},$$

which proves the first result of the proposition. In particular, taking $\lambda_2 \leq \lambda_1\left(1 + \frac{1}{\sqrt{n}}\right)$ gives its second result

$$\begin{aligned}
\mathrm{E}[C_A] - \mathrm{E}[C_{\mathrm{FTPP}}] &\geq \lambda_1 n\sqrt{n}/2\frac{\exp(-n\frac{1/n}{(1+1/\sqrt{n})^2})}{4} \\
&\geq \lambda_1 n\sqrt{n}\frac{e^{-1/4}}{8} \\
&\geq (\lambda_1+\lambda_2)n\sqrt{n}\frac{e^{-1/4}}{24}.
\end{aligned}$$

□

Large Differences

Proposition 7.A.13. *For any non-preemptive algorithm, there exists a problem instance with expected type durations of $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_K$ such that*

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}(2k - K - 1)\lambda_k.$$

In particular, for $K = 2$ and $\lambda_2 \geq 3\lambda_1$, it holds that

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{4}(\lambda_1 + \lambda_2),$$

Let p_k be the probability that a non-preemptive algorithm completes a job of type k at its first. Notice that this distribution cannot depend on the expected duration of any of the types, since no data was gathered. Thus, types can be arbitrarily ordered without affecting this distribution. In particular, we assume w.l.o.g. that $p_1 \leq p_2 \leq \ldots p_K$ and choose a problem instance where job types are ordered in an increasing duration $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_K$. Then, the expected cost of the algorithm can be bounded according to Theorem 7.A.9, by

$$\begin{aligned}
\mathrm{E}[C_A] &= \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right] \\
&\geq \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{j\in[n]}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_1^k \text{ computed before } P_j^\ell\right\}\right] \\
&\geq \mathrm{E}[C_{\mathrm{FTPP}}] + n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{P_1^k \text{ was the first job}\right\}\right] \\
&= \mathrm{E}[C_{\mathrm{FTPP}}] + n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k - \lambda_\ell)p_k \\
&= \mathrm{E}[C_{\mathrm{FTPP}}] + n\sum_{k=1}^{K}p_k\sum_{\ell=1}^{k-1}(\lambda_k - \lambda_\ell).
\end{aligned}$$

Now, one can be easily convinced that between all probability vectors with non-decreasing components, this bound is minimized by the uniform distribution $p_k = 1/K$. To see this, observe that the sum $\sum_{\ell=1}^{k-1}(\lambda_k - \lambda_\ell)$ increases with k. Therefore, if p is non-uniform, then $p_K > 1/K$, and there would exist a coordinate $k < K$ to which we could move weight from p_K, which would decrease the bound.

Substituting $p_k = 1/K$ we then get

$$\begin{aligned}
\mathrm{E}[C_A] &\geq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}\sum_{\ell=1}^{k-1}(\lambda_k - \lambda_\ell) \\
&= \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}(2k - K - 1)\lambda_k.
\end{aligned}$$

In particular, if $K = 2$, we get

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{2}(\lambda_2 - \lambda_1)$$

and, for $\lambda_2 \geq 3\lambda_1$, we have $\lambda_2 - \lambda_1 \geq \frac{\lambda_1+\lambda_2}{2}$ and thus

$$\mathrm{E}[C_A] \geq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{n}{4}(\lambda_1 + \lambda_2).$$

7.A.3 Analysis of Preemptive Learning algorithms

Full Algorithmic Details

In this appendix, we present a full description of ETC-RR and UCB-RR.

Algorithm 33: Explore-Then-Commit-Round-Robin (ETC-RR)

Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types);
For all pairs of different types k, ℓ initialize $\delta_{k,\ell} = 0$, $\hat{r}_{k,\ell} = 0$ and $h_{k,\ell} = 0$;
For all types k, set $c_k = 0$;
while $\mathcal{U}$ *is not empty* **do**
  $\mathcal{U}$ is the set of types with at least one remaining job;
  if $\mathcal{A}$ *is empty* **then**
    $\mathcal{A} = \{\ell \in \mathcal{U}, \forall k \in \mathcal{U}, k \neq \ell, \ \hat{r}_{k,\ell} - \delta_{k,\ell} \leq 0.5\}$;
  end
  Run jobs $(P^k_{c_k+1})_{k\in\mathcal{A}}$ in parallel until a job finishes and denote ℓ the type of this job;
  $c_\ell = c_\ell + 1$;
  for $k \in \mathcal{A}, k \neq \ell$ **do**
    $\beta_{\ell,k} = \beta_{\ell,k} + 1$;
    $\delta_{\ell,k} = \delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^4)}{2(\beta_{\ell,k}+\beta_{k,\ell})}}$;
    $\hat{r}_{\ell,k} = \frac{\beta_{\ell,k}}{\beta_{k,\ell}+\beta_{\ell,k}}$;
    $\hat{r}_{k,\ell} = \frac{\beta_{k,\ell}}{\beta_{k,\ell}+\beta_{\ell,k}}$;
    if $\hat{r}_{\ell,k} - \delta_{\ell,k} \geq 0.5$ **then**
      Remove k from $\mathcal{A}$;
    end
    if $\hat{r}_{k,\ell} - \delta_{k,\ell} \geq 0.5$ *or* $c_\ell = n$ **then**
      Remove ℓ from $\mathcal{A}$;
    end
  end
end

Algorithm 34: Upper-Confidence-Bound-Round-Robin (UCB-RR)

Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types), discretization step Δ;
while $\mathcal{U}$ *is not empty* **do**
  $\mathcal{U}$ is the set of types with at least one remaining job;
  Calculate type indices u_k for all jobs $k \in \mathcal{U}$ according to Equation (7.29);
  Choose type $\ell \in \arg\max_{\ell\in\mathcal{U}} u_\ell$;
  Run a job of type ℓ for Δ time units;
end

Cost Decomposition

We start with a cost decomposition, which relates the performance of preemptive algorithms to the one of FTPP. We limit ourselves to the natural family of preemptive algorithms that do not simultaneously run two tasks of the same type and is formally defined as follows.

Definition 7.A.14. *Denote b_i^ℓ and e_i^ℓ the beginning and end dates of the computation of the i^{th} job of type ℓ. A **type-wise non-preemptive algorithm** is an algorithm that computes jobs of the same type one after another, i.e., $\forall i \in [n], \forall k \in [K], e_i^\ell \leq b_{i+1}^\ell$.*

This property is very natural, as for exponential durations without knowledge of the real execution times, there is no advantage in simultaneously running two tasks of the same type. Specifically, all of our suggested algorithms fall under this definition.

For such algorithms, the cumulative cost can be compared to FTPP using the following lemma.

Lemma 7.A.15 (Cost of type-wise non-preemptive algorithms)**.** *Any type-wise non-preemptive algorithm A has the following upper bound on its cost:*

$$\mathrm{E}[C_A] \leq \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell.$$

Proof. Recall that if D_{ij}^A is the amount of time a job i delays job j when running algorithm A, then we can write the cost of algorithm A as

$$C_A = \sum_{i=1}^{N} P_j + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(D_{ij}^A + D_{ji}^A\right).$$

Moreover, if b_i,e_i are the start (end) time of job i, it always holds that $D_{ij}^A \leq P_i\mathbb{1}\{b_i < e_j\}$. Using this inequality and dividing the summation into types, we get

$$\begin{aligned}\mathrm{E}[C_A] \leq &\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\mathrm{E}\left[P_i^\ell\mathbb{1}\left\{b_i^\ell < e_j^k\right\}\right] + \mathrm{E}\left[P_j^k\mathbb{1}\left\{b_j^k < e_i^\ell\right\}\right]\right)\\ &+\sum_{\ell=1}^{K}\left(\sum_{i=1}^{n}\mathrm{E}[P_i^\ell] + \sum_{j=i+1}^{n}\mathrm{E}[P_i^\ell\mathbb{1}\left\{b_i^\ell < e_j^\ell\right\}] + \mathrm{E}[P_j^\ell\mathbb{1}\left\{b_j^\ell < e_i^\ell\right\}]\right).\end{aligned}$$

Since jobs are independent, the expected duration of a job of type ℓ is λ_ℓ, independently of its start time. Also, as the algorithm is type-wise non-preemptive, for all $\ell \in [K], j > i$, we have $\mathbb{1}\left\{b_j^\ell < e_i^\ell]\right\} = 0$ and $\mathbb{1}\left\{b_i^\ell \leq e_j^\ell]\right\} = 1$. Thus,

$$\begin{aligned}\mathrm{E}[C_A] \leq &\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell\mathrm{E}\left[\mathbb{1}\left\{b_i^\ell < e_j^k\right\}\right] + \lambda_k\mathrm{E}\left[\mathbb{1}\left\{b_j^k < e_i^\ell\right\}\right]\right) + \sum_{\ell=1}^{K}\left(\sum_{i=1}^{n}\lambda_\ell + \sum_{j=i+1}^{n}\lambda_\ell\right)\\ =&\sum_{\ell=1}^{K}\lambda_\ell\frac{n(n+1)}{2} + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell\mathrm{E}\left[\mathbb{1}\left\{b_i^\ell < e_j^k\right\}\right] + \lambda_k\mathrm{E}\left[\mathbb{1}\left\{b_j^k < e_i^\ell\right\}\right]\right)}_{(i)}. \quad (7.23)\end{aligned}$$

We can now decompose the event that job i of type ℓ started before job j of type k finished:

$$\mathbb{1}\left\{b_i^\ell < e_j^k\right\} = \mathbb{1}\left\{b_i^\ell < e_j^k \leq e_i^\ell\right\} + \mathbb{1}\left\{e_i^\ell < e_j^k\right\} = \mathbb{1}\left\{b_i^\ell < e_j^k \leq e_i^\ell\right\} + \mathbb{1}\left\{e_i^\ell < b_j^k\right\} + \mathbb{1}\left\{b_j^k \leq e_i^\ell < e_j^k\right\}.$$

$\left\{e_i^\ell < b_j^k\right\}$ is the event that job i of type ℓ was fully computed before job j of type k started. $\left\{b_i^\ell < e_j^k \leq e_i^\ell\right\}$ is the event that job i of type ℓ was running when job j of type k finished, and reciprocally for $\left\{b_j^k \leq e_i^\ell < e_j^k\right\}$. This gives the following decomposition:

$$
\begin{aligned}
(i) = & \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell \mathrm{E}\left[\mathbb{1}\left\{e_i^\ell < b_j^k\right\}\right] + \lambda_k \mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right]\right)}_{(ii)} \\
& + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\lambda_\ell \mathrm{E}\left[\mathbb{1}\left\{b_i^\ell < e_j^k \leq e_i^\ell\right\}\right] + \mathbb{1}\left\{b_j^k \leq e_i^\ell < e_j^k\right\}}_{(iii)} \\
& + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\lambda_k \mathrm{E}\left[\mathbb{1}\left\{b_j^k < e_i^\ell \leq e_j^k\right\} + \mathbb{1}\left\{b_i^\ell \leq e_j^k < e_i^\ell\right\}\right]}_{(iv)}
\end{aligned}
$$

Since the algorithm is type-wise non-preemptive, a single job of each type may run at a given time. This implies that any job of type ℓ cannot be in a middle of two different jobs of type k, namely

$$
\forall(\ell,k)\in[K]^2,\ \ell\neq k,\ \forall(i,j)\in[n]^2,\ \sum_{j=1}^{n}\mathbb{1}\left\{b_j^k \leq e_i^\ell < e_j^k\right\} \leq 1.
$$

The same conclusion similarly holds for all other sums in terms (iii) and (iv), and therefore implies the following bound:

$$
(iii) + (iv) \leq n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_\ell + \lambda_k) = (K-1)n\sum_{\ell=1}^{K}\lambda_\ell.
$$

We also have $\mathbb{1}\left\{e_j^k < b_i^\ell\right\} \leq 1 - \mathbb{1}\left\{b_i^\ell < e_j^k\right\}$, thus

$$
\begin{aligned}
(ii) &\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell + (\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right]\right) \\
&= n^2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(K-\ell)\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right].
\end{aligned}
$$

Combining everything into Equation (7.23), we get:

$$
\begin{aligned}
\mathrm{E}[C_A] &\leq \sum_{\ell=1}^{K}\lambda_\ell\frac{n(n+1)}{2} + n^2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(K-\ell)\lambda_\ell \\
&\quad + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell \\
&= \mathrm{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell,
\end{aligned}
$$

where the last equality is by Theorem 7.A.2. □

Upper bound for ETC-RR

Proposition 7.A.16. *The following bound holds:*

$$\mathrm{E}[C_{\text{ETC-RR}}] \leq \mathrm{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathrm{E}[C^{OPT}] + 4n\sqrt{n\log(2n^2K^3)}\sum_{\ell=1}^{K}(K-\ell)^2\lambda_\ell. \tag{7.24}$$

Good Event. For any couple (k,ℓ), if at some iteration $\beta_{k,\ell} + \beta_{\ell,k} = m$, we define the more precise notation for $\hat{r}_{\ell,k}$ at that iteration as $\hat{r}^m_{\ell,k}$. Notice that m represents the number of times that jobs of either type were completed while both were active. Therefore, m can have $2n$ different values, where the extreme case $m = 2n-1$ is, for example, when $n-1$ jobs of type k are first completed and only then all n jobs of type ℓ are completed.

Let us start by showing that the estimators $\hat{r}_{\ell,k}$ well-concentrate around their expectations. Exponential random variables are memory-less, i.e., if $X_i \sim Exp(\lambda_i)$, the law of X_i conditionally on it being larger than a constant is unchanged. In particular, 'resetting' (replacing by an independent copy) an exponential random variable at any time that precedes its activation does not affect its distribution. We employ reset when one of the types is completed, or when either of the types is removed from $\mathcal{A}$ (and then we discard the comparison between types k,ℓ), and say that way a comparison is triggered, it is taken from an i.i.d. sequence of comparisons. Specifically, given a deterministic number of comparisons m between types k,ℓ, we write

$$\hat{r}^m_{\ell,k} \overset{\mathcal{L}}{=} \frac{1}{m}\sum_{i=1}^{m}\mathbb{1}\{X_i^\ell < X_i^k\},$$

with $(X_i^\ell)_{i\in[m]}$ and $(X_i^k)_{i\in[m]}$ independent exponential variables of parameters λ_ℓ and λ_k respectively.

Finally, $\mathcal{E}$ be the event that all comparisons between k,ℓ are well-concentrated, namely,

$$\mathcal{E} = \left\{\forall(k,\ell)\in[K]^2, \forall m\in[2n], \left|\hat{r}^m_{\ell,k} - \frac{\lambda_k}{\lambda_\ell+\lambda_k}\right| < \delta^{(m,n)}\right\}$$

with $\delta^{(m,n)} = \sqrt{\frac{\log(2n^2K^3)}{2m}}$, and recall that for any $m\in[2n]$, $\mathrm{E}[\hat{r}^m_{\ell,k}] = \frac{\lambda_k}{\lambda_\ell+\lambda_k}$. By Lemma 7.A.11 applied with $\alpha = 2$, and a union bound over the $\frac{K(K-1)}{2}$ possible pairs, we have:

$$\mathbf{P}(\overline{\mathcal{E}}) \leq \frac{1}{nK}. \tag{7.25}$$

Notice that under $\mathcal{E}$ and type ℓ will never be eliminated by a type k with $\lambda_k \geq \lambda_\ell$, since

$$\hat{r}_{k,\ell} - \delta_{\ell,k} < \left(\frac{\lambda_\ell}{\lambda_k+\lambda_\ell} + \delta_{\ell,k}\right) - \delta_{\ell,k} = \frac{\lambda_\ell}{\lambda_k+\lambda_\ell} \leq \frac{1}{2},$$

so if k is the minimal type in $\mathcal{A}$, then under the good event, it will never be eliminated. Moreover, type k can only be compared to a type ℓ with $\lambda_\ell \leq \lambda_k$ at most

$$m^{\max}_{\ell,k} \leq \min\left\{2n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2\log(2n^2K^3)\right\} := m^*_{\ell,k} \tag{7.26}$$

since clearly $m \le 2n$ and if $m \ge 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2 \log(2n^2K^3)$, then $\delta_{\ell,k} \le \frac{1}{4}\frac{\lambda_k-\lambda_\ell}{\lambda_k+\lambda_\ell}$ and

$$\hat{r}_{\ell,k} - \delta_{\ell,k} > \left(\frac{\lambda_k}{\lambda_k+\lambda_\ell} - \delta_{\ell,k}\right) - \delta_{\ell,k} \ge \frac{\lambda_k}{\lambda_k+\lambda_\ell} - 2\cdot\frac{1}{4}\frac{\lambda_k-\lambda_\ell}{\lambda_k+\lambda_\ell} = \frac{1}{2}.$$

Cost Analysis. Assume that the active type set $\mathcal{A}$ can only change at discrete times $t \in \{0, \Delta, 2\Delta, \dots\} \triangleq \mathcal{T}$ for some $\Delta > 0$. We will later take the limit $\Delta \to 0$, which coincides with the following Algorithm 33. In the following, we denote by $\mathcal{A}(t)$ the active type set at time interval $[t, t+\Delta)$, and $\mathcal{U}(t)$ incomplete type set at $[t, t+\Delta)$. Also, let $b_i^\ell \in \mathcal{T}$ be the start time of the i^{th} job of the ℓ^{th} type and $e_i^\ell \in \mathcal{T}$ be its end-time (w.l.o.g., if a task ended at a time $t \notin \mathcal{T}$, we delay its ending to $\lceil \frac{t}{\Delta} \rceil \Delta$). Starting from the cost decomposition of Theorem 7.A.15, we have

$$\begin{aligned}
\mathrm{E}[C_A] &\le \mathrm{E}[C_{\mathrm{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] \\
&\le \mathrm{E}[C_{\mathrm{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\underbrace{\sum_{j=1}^{n}\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_n^\ell\right\}\right]}_{(*)}. \qquad (7.27)
\end{aligned}$$

We focus our attention on bounding term $(*)$. For any $t \in \mathcal{T}$ and every $o \in [K]$, define the events

$$\mathcal{F}_o(t) = \left\{o \in \mathcal{A}(t), \forall p < o : p \notin \mathcal{A}(t)\right\}, \qquad \bar{\mathcal{F}}_o(t) = \left\{\forall p \le o : p \notin \mathcal{A}(t)\right\}.$$

These events capture the notion of *phases*, namely, when $\mathcal{F}_o$ is active, the o is the type of the smallest mean that has not been finished. Then, we can write

$$\begin{aligned}
(*) &= \sum_{j=1}^{n}\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_n^\ell\right\}\right] \\
&= \sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, e_j^k < b_n^\ell\right\}\right] \\
&\overset{(1)}{\le} \sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \ell \in \mathcal{U}(t)\right\}\right] \\
&\le \sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \mathcal{F}_o(t)\right\}\right] + \sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \ell \in \mathcal{U}(t), \bar{\mathcal{F}}_\ell(t)\right\}\right] \\
&\overset{(2)}{\le} \sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \mathcal{F}_o(t)\right\}\right] + \sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \bar{\mathcal{E}}\right\}\right] \\
&\overset{(3)}{\le} \underbrace{\sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t+\Delta, \mathcal{F}_o(t)\right\}\right]}_{(i)} + n\mathbf{P}(\bar{\mathcal{E}}).
\end{aligned}$$

(1) is true since the event $\left\{e_j^k = t+\Delta, e_j^k < b_n^\ell\right\}$ implies that $b_n^\ell > t$, so type ℓ was not completed at time t. (2) holds since under $\mathcal{E}$, the type of the minimal duration expected is never eliminated, so if $\ell \in \mathcal{U}(t)$, it is impossible that $p \notin \mathcal{A}(t)$ for all $p \le \ell$ (either there is a type $p < \ell, p \in \mathcal{U}(t)$

that should not have been eliminated, or type ℓ should not have been eliminated since it is still incomplete). Finally, (3) is since every job can only end at one time point.

We now further continue to bound term (i). To do so, observe that if $e_j^k = t + \Delta$, then $k \in \mathcal{A}(t)$ and the task j of this type was completed at the interval $[t, t + \Delta)$. Moreover, since the job durations are exponential, the completion of any job in $\mathcal{A}(t)$ at interval $[t, t + \Delta)$ is independent of the events that occurred until time t. Taking into consideration that only one job of any type can run in every interval, the two following equalities hold:

$$\begin{aligned}
&\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t + \Delta, \mathcal{F}_o(t)\right\}\right] = (1 - \exp(-\Delta/\lambda_k))\,\mathrm{E}\left[\mathbb{1}\{\mathcal{F}_o(t), k \in \mathcal{A}(t)\}\right] \\
&\mathrm{E}\left[\mathbb{1}\left\{e_j^k = t + \Delta \text{ or } e_j^o = t + \Delta, \mathcal{F}_o(t), k \in \mathcal{A}(t)\right\}\right] \\
&= (1 - \exp(-\Delta/\lambda_k - \Delta/\lambda_o))\,\mathrm{E}\left[\mathbb{1}\{\mathcal{F}_o(t), k \in \mathcal{A}(t)\}\right].
\end{aligned}$$

Thus, (i) can be written as

$$\begin{aligned}
(i) &= \sum_{o=1}^{\ell} \frac{(1 - \exp(-\Delta/\lambda_k))}{(1 - \exp(-\Delta/\lambda_k - \Delta/\lambda_o))} \sum_{j=1}^{n} \sum_{t \in \mathcal{T}} \mathrm{E}\left[\mathbb{1}\left\{e_j^k = t + \Delta \text{ or } e_j^o = t + \Delta, \mathcal{F}_o(t), k \in \mathcal{A}(t)\right\}\right] \\
&\leq \underbrace{\sum_{o=1}^{\ell} \frac{(1 - \exp(-\Delta/\lambda_k))}{(1 - \exp(-\Delta/\lambda_k - \Delta/\lambda_o))} \mathrm{E}\left[\sum_{j=1}^{n} \sum_{t \in \mathcal{T}} \mathbb{1}\left\{e_j^k = t + \Delta \text{ or } e_j^o = t + \Delta, \mathcal{F}_o(t), k \in \mathcal{A}(t), \mathcal{E}\right\}\right]}_{(ii)} \\
&\quad + \underbrace{\sum_{o=1}^{\ell} \sum_{j=1}^{n} \sum_{t \in \mathcal{T}} \mathrm{E}\left[\mathbb{1}\left\{e_j^k = t + \Delta \text{ or } e_j^o = t + \Delta, \mathcal{F}_o(t), \overline{\mathcal{E}}\right\}\right]}_{(iii)}
\end{aligned}$$

The bad-event term can be bounded by

$$\begin{aligned}
(iii) &\leq \mathrm{E}\left[\mathbb{1}\left\{\overline{\mathcal{E}}\right\} \sum_{o=1}^{\ell} \sum_{j=1}^{n} \sum_{t \in \mathcal{T}} \mathbb{1}\left\{e_j^k = t + \Delta \text{ or } e_j^o = t + \Delta, \mathcal{F}_o(t)\right\}\right] \\
&\leq (\ell + 1) n \mathbf{P}(\overline{\mathcal{E}}).
\end{aligned}$$

For the second term, recall that the ℓ^{th} phase, in which type ℓ is the smallest one that was not completed, is represented by the event $\mathcal{F}_\ell(t)$. Furthermore, given the good event, the smallest type is never eliminated. so once $\mathcal{F}_\ell(t)$ becomes active, it would end only when all jobs of type ℓ are completed. In other words, the time indices in which $\mathcal{F}_\ell(t)$ hold form a (possibly empty) interval $\mathcal{I}_\Delta(\ell)$, which represents the ℓ^{th} phase. Thus, term (ii) counts the expected number of times that jobs of either type k or o could finish in this interval while type k is still active.

Now, we take the limit $\Delta \to 0$ (and denoting the limit interval $\mathcal{I}_\Delta(\ell) \to \mathcal{I}(\ell)$). Notice that the limit and expectation are interchangeable by the bounded convergence theorem, as the number of times a job of either type k or o can be completed is bounded by $2n$.

$$\begin{aligned}
(ii) &\underset{\Delta \to 0}{\to} \sum_{o=1}^{\ell} \frac{\lambda_o}{\lambda_k + \lambda_o} \mathrm{E}\left[\sum_{t \in \mathcal{I}(o)} \sum_{j=1}^{n} \mathbb{1}\left\{e_j^k \in \mathcal{I}(o) \text{ or } e_j^o \in \mathcal{I}(o), k \in \mathcal{A}(t), \mathcal{E}\right\}\right] \\
&\leq \sum_{o=1}^{\ell} \frac{\lambda_o}{\lambda_k + \lambda_o} m_{o,k}^*.
\end{aligned}$$

The inequality holds since under the good event, at any interval where both types k and o with $\lambda_o \leq \lambda_k$ are active, there can be at most $m^*_{o,k}$ comparisons. Substituting (ii) and (iii) back into (i), we get

$$(i) \leq \sum_{o=1}^{\ell} \frac{\lambda_o}{\lambda_k + \lambda_o} m^*_{o,k} + (\ell+1) n \mathbf{P}(\overline{\mathcal{E}}),$$

and yet again, substituting this, through $(*)$, back into Equation (7.27), yields

$$\begin{aligned}
&\mathrm{E}[C_A] - \mathrm{E}[C_{\mathrm{FTPP}}] \\
&\quad \leq (K-1)n \sum_{\ell=1}^{K} \lambda_\ell + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\sum_{o=1}^{\ell} \frac{\lambda_o}{\lambda_k + \lambda_o} m^*_{o,k} + (\ell+2) n \mathbf{P}(\overline{\mathcal{E}}) \right) \\
&\quad \leq (K-1)n \sum_{\ell=1}^{K} \lambda_\ell + 2Kn^2 \mathbf{P}(\overline{\mathcal{E}}) \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \\
&\qquad + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \sum_{o=1}^{\ell} \frac{\lambda_o}{\lambda_k + \lambda_o} \min \left\{ 2n, 8 \left(\frac{\lambda_k + \lambda_o}{\lambda_k - \lambda_o} \right)^2 \log(2n^2 K^3) \right\} \quad \text{(Equation (7.26))} \\
&\quad \leq (K-1)n \sum_{\ell=1}^{K} \lambda_\ell + 2K^2 n^2 \mathbf{P}(\overline{\mathcal{E}}) \sum_{\ell=1}^{K} \lambda_\ell \\
&\qquad + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{o=1}^{\ell} n(\lambda_k - \lambda_\ell) \frac{\lambda_o}{\lambda_k + \lambda_o} 4 \sqrt{n \log(2n^2 K^3)} \frac{\lambda_k + \lambda_o}{\lambda_k - \lambda_o} \qquad (\min\{a,b\} \leq \sqrt{ab}, \forall a, b \geq 0) \\
&\quad \leq \left(2\frac{K-1}{n} + \frac{4K}{n} \right) \mathrm{E}[C^{OPT}] \qquad \text{(Equation (7.25) and (7.6))} \\
&\qquad + 4n \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{o=1}^{\ell} \lambda_o \sqrt{n \log(2n^2 K^3)} \qquad (\lambda_o \leq \lambda_\ell) \\
&\quad \leq \frac{6K}{n} \mathrm{E}[C^{OPT}] + 4n \sqrt{n \log(2n^2 K^3)} \sum_{\ell=1}^{K} \sum_{o=1}^{\ell} (K - \ell) \lambda_o \\
&\quad \leq \frac{6K}{n} \mathrm{E}[C^{OPT}] + 4n \sqrt{n \log(2n^2 K^3)} \sum_{\ell=1}^{K} (K-\ell)^2 \lambda_\ell
\end{aligned}$$

□

Analysis of UCB-RR

Proposition 7.A.17. *The following bound holds for any $\Delta \leq \frac{\lambda_1}{4}$ and $n \geq \max(20, 10\ln(K))$:*

$$\mathrm{E}[C_{\text{UCB-RR}}] \leq \mathrm{E}[C_{\mathrm{FTPP}}] + \frac{12K}{n} \mathrm{E}[C^{OPT}] + 6n \sqrt{2n \log(2n^2 K^2)} \sum_{\ell=1}^{K} (K-\ell) \lambda_\ell. \tag{7.28}$$

Assume discretization of the time to units of Δ, as was done in the analysis of ETC-RR. Specifically, assume that the active job only changes at times $t \in \{0, \Delta, 2\Delta, \dots\} \triangleq \mathcal{T}$ for some $\Delta > 0$. We then denote the index of the discretization step by $h = \frac{t}{\Delta} + 1 \in \{1, 2, \dots\}$.

For each job type $\ell \in [K]$, we introduce $T_\ell(h)$, the number of times job type ℓ has been chosen up to iteration h. Due to the fact that job durations are exponential, their increments our independent, and increments of length Δ of jobs of type ℓ have a termination probability of $\mu_\ell = 1 - e^{-\frac{\Delta}{\lambda_k}}$. Leveraging this, let $(x^s_\ell)_{s \geq 1}$ be sequences of i.i.d Bernoulli random variables of

mean μ_ℓ,. We then fix our probability space for the analysis s.t. when choosing a job of type ℓ for the s^{th} time, it is terminated if $x_\ell^s = 1$. Notice that while we allow the sequence $(x_\ell^s)_{s\geq 1}$ to have more than n job terminations, it is of no consequence of the analysis, as the algorithm will never choose a job type after its n^{th} job was terminated.

Next, define the empirical means after running m discretized intervals of type-ℓ jobs as $\hat{\mu}_\ell(m) := \frac{1}{m}\sum_{s=1}^m x_\ell^s$, and the index at iteration h as

$$u_\ell(h) = \max\left\{\tilde{\mu}\in[0,1] : d\left(\hat{\mu}_\ell(T_\ell(t-1)),\tilde{\mu}\right) \leq \frac{\log\left(n^2K^2\right)}{T_\ell(t-1)}\right\}. \tag{7.29}$$

Starting from the cost decomposition of Theorem 7.A.15, we have

$$\begin{aligned}
\mathrm{E}[C_A] &\leq \mathrm{E}[C_{\mathrm{FTPP}}] + (K-1)n\sum_{\ell=1}^K \lambda_\ell + \sum_{\ell=1}^K\sum_{k=\ell+1}^K \sum_{(i,j)\in[n]^2} (\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] \\
&\leq \mathrm{E}[C_{\mathrm{FTPP}}] + (K-1)n\sum_{\ell=1}^K \lambda_\ell + \underbrace{\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)\sum_{j=1}^n \mathrm{E}\left[\mathbb{1}\left\{e_j^k < e_n^\ell\right\}\right]}_{(*)}.
\end{aligned} \tag{7.30}$$

Denote $a(h)\in[K]$, the type of job chosen at iteration h, and let $\varepsilon_{\ell,k}>0$ be some constant that will be determined later in the proof. Notice that if $e_j^k < e_n^\ell$, then there must be an iteration where type ℓ was not completed and a the j^{th} job of type k were played and completed:

$$\begin{aligned}
(*) &\leq \sum_{h=1}^\infty\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{a(h)=k,\ \ell\in\mathcal{U}(h), x_k^{T_k(h)}=1\right\}\right] \\
&= \sum_{h=1}^\infty\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{a(h)=k,\ \ell\in\mathcal{U}(h), u_\ell(h)\leq u_k(h), x_k^{T_k(h)}=1\right\}\right] \\
&\leq \sum_{h=1}^\infty\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{a(h)=k,\ \ell\in\mathcal{U}(h), u_k(h)\geq u_\ell(h)\geq \mu_\ell-\varepsilon_\ell, x_k^{T_k(h)}=1\right\}\right] \\
&\quad + \sum_{h=1}^\infty\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)\mathrm{E}\left[\mathbb{1}\left\{a(h)=k,\ \ell\in\mathcal{U}(h), u_\ell(h)\leq \mu_\ell-\varepsilon_{\ell,k}, x_k^{T_k(h)}=1\right\}\right] \\
&\overset{(1)}{\leq} \underbrace{\sum_{h=1}^\infty\sum_{\ell=1}^K\sum_{k=\ell+1}^K n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\mathrm{E}\left[\mathbb{1}\{a(h)=k, u_k(h)\geq\mu_\ell-\varepsilon_\ell\}\right]}_{(i)} \\
&\quad + \underbrace{\sum_{\ell=1}^K\sum_{k=\ell+1}^K n^2(\lambda_k-\lambda_\ell)\mathbf{P}\left(\exists h \text{ s.t. } u_\ell(h)\leq\mu_\ell-\varepsilon_\ell\right)}_{(ii)}
\end{aligned} \tag{7.31}$$

In (1), we get the first line by the memoryless property of exponential random variables, noting that all the events inside the indicator are determined before the beginning of the h^{th} iteration. The second line of this relation uses the fact that all tasks will eventually be completed, so $\sum_{h=1}^\infty \mathrm{E}\left[\mathbb{1}\left\{a(h)=k, x_k^{T_k(h)}=1\right\}\right] = n$.

Bounding term (i)**.** We now bound the first term of the decomposition in Equation (7.31).

$$\begin{aligned}(i) :=& \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\mathrm{E}\left[\mathbb{1}\{a(h)=k, u_k(h)\geq \mu_\ell-\varepsilon_\ell\}\right]\\ =& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{h=1}^{\infty}\mathrm{E}\left[\mathbb{1}\{a(h)=k, u_k(h)\geq \mu_\ell-\varepsilon_\ell\}\right].\end{aligned}$$

Denoting $\underline{d}(p,q) = d(p,q)\mathbb{1}\{p\leq q\}$, we have

$$\begin{aligned}\{\mu_\ell-\varepsilon_{\ell,k}\leq u_k(h)\} \implies \{\hat{\mu}_k(T_k(h))\leq \mu_\ell-\varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\}\\ \text{or } \{\hat{\mu}_k(T_k(h))\geq \mu_\ell-\varepsilon_{\ell,k}\},\end{aligned}$$

which is equivalent to $\left\{\underline{d}(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right\}$. Thus, we can bound

$$\begin{aligned}(i) \leq& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{h=1}^{\infty}\mathrm{E}\left[\mathbb{1}\left\{a(h)=k, \underline{d}(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right\}\right]\\ \leq& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{s=1}^{\infty}\mathrm{E}\left[\underline{d}(\hat{\mu}_k(s),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right],\end{aligned}$$

where the second inequality is since $T_k(h)$ increases by 1 every time that $a(h)=k$. This can be naturally bounded using the following lemma.

Lemma 7.A.18. *Let $X_1, X_2, \ldots$ be a sequence of Bernoulli independent random variables with mean μ, and let $\hat{\mu}_s = \frac{1}{s}\sum_{t=1}^{s} X_t$ be the sample mean. Further, let $a>0, \mu'>\mu$ and define $\kappa = \sum_{s=1}^{\infty}\mathbb{1}\left\{\underline{d}(\hat{\mu}_s,\mu')\leq \frac{a}{s}\right\}$. Then,*

$$\mathrm{E}[\kappa]\leq \inf_{\varepsilon\in(0,\mu'-\mu)}\left(\frac{a}{d(\mu+\varepsilon,\mu')}+\frac{1}{d\left(\mu+\varepsilon,\mu\right)}\right).$$

Proof. The proof closely follows the one of (Lattimore and Szepesvári, 2020, Lemma 10.8). For completeness, we now state the well-known Chernoff bound.

Lemma 7.A.19 (Chernoff's bound, e.g., Lattimore and Szepesvári (2020), Lemma 10.3)**.** *Let $X_1, X_2, \ldots, X_T$ be a sequence of Bernoulli independent random variables with mean μ, and let $\hat{\mu} = \frac{1}{T}\sum_{t=1}^{T} X_t$ be the sample mean. Then, for $a\geq 0$:*

$$\mathbb{P}(d(\hat{\mu},\mu)\geq a, \hat{\mu}\leq \mu)\leq \exp(-Ta).$$

Let $\epsilon \in (0, \mu' - \mu)$ and $u = \frac{a}{d(\mu+\varepsilon,\mu')}$. Then, it holds that

$$\begin{aligned}
\mathrm{E}[\kappa] =& \sum_{s=1}^{\infty} \mathbf{P}\left\{\underline{d}(\hat{\mu}_s, \mu') \leq \frac{a}{s}\right\} \\
=& \sum_{s=1}^{\infty} \mathbf{P}\left\{\hat{\mu}_s \geq \mu' \text{ or } d(\hat{\mu}_s, \mu') \leq \frac{a}{s}\right\} \\
\leq& \sum_{s=1}^{\infty} \mathbf{P}\left\{\hat{\mu}_s \geq \mu + \varepsilon \text{ or } d(\hat{\mu}_s, \mu') \leq \frac{a}{s}\right\} && (\mu' > \mu + \epsilon) \\
\leq& \sum_{s=1}^{\infty} \mathbf{P}\left\{\hat{\mu}_s \geq \mu + \varepsilon \text{ or } d(\mu + \varepsilon, \mu') \leq \frac{a}{s}\right\} && (d(\cdot, \mu') \text{ is decreasing in } [0, \mu']) \\
\leq& u + \sum_{s=1}^{\infty} \mathbf{P}\left\{\hat{\mu}_s \geq \mu + \varepsilon\right\} \\
\leq& u + \sum_{s=1}^{\infty} \sum_{s=1}^{\infty} \exp\left(-s d(\mu + \epsilon, \mu)\right) && \text{(Chernoff's bound)} \\
\leq& \frac{a}{d(\mu + \varepsilon, \mu')} + \frac{1}{d\left(\mu + \varepsilon, \mu\right)},
\end{aligned}$$

and the proof is concluded by taking the infimum over all $\varepsilon \in (0, \mu' - \mu)$. □

Now, assume w.l.o.g. that $\lambda_k > \lambda_\ell$ (or, equivalently, $\mu_\ell < \mu_k$) for all $k > \ell$; otherwise, terms where $\lambda_k = \lambda_\ell$ in (i) will be equal to 0. Then, letting $\kappa_{k,\ell} = \sum_{s=1}^{\infty} \mathbb{1}\left\{\underline{d}(\hat{\mu}_k(s), \mu_\ell - \varepsilon_{\ell,k}) \leq \frac{\log\left(n^2K^2\right)}{s}\right\}$, the last lemma implies that

$$\begin{aligned}
(i) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)(1 - e^{-\frac{\Delta}{\lambda_k}})\mathbb{E}[\kappa_{k,\ell}] \\
\leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)(1 - e^{-\frac{\Delta}{\lambda_k}}) \left(\frac{\log\left(n^2K^2\right)}{d(\mu_k + \varepsilon_{k,\ell}, \mu_\ell - \varepsilon_\ell)} + \frac{1}{d\left(\mu_k + \varepsilon_{k,\ell}, \mu_k\right)}\right) \qquad (7.32)
\end{aligned}$$

for some $\varepsilon_{k,\ell} \in (0, \mu_\ell - \varepsilon_{\ell,k} - \mu_k)$ that will be determined later.

Bounding term (ii)**.** We next focus on bounding the probabilities at the second summation of Equation (7.31). To do so, we prove the following lemma, which bounds each of the summands of term (ii).

Lemma 7.A.20. *The following bound holds:* $\mathbf{P}\left(\exists h \text{ s.t. } u_\ell(h) \leq \mu_\ell - \varepsilon_{\ell,k}\right) \leq \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}$.

Proof. Define $S_\ell^{\max} := \sum_{h=1}^{\infty} \mathbb{1}\{a(h) = \ell\}$, the number of iterations ℓ is picked by the algorithm.

We have:

$$\begin{aligned}
&\mathbf{P}\left(\exists h \text{ s.t. } u_\ell(h) \le \mu_\ell - \varepsilon_{\ell,k}\right) \\
&\quad = \mathbf{P}\left(\exists h \text{ s.t. } \hat{\mu}_\ell(T_\ell(h)) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(T_\ell(h)), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{T_\ell(h)}\right) \\
&\quad = \mathbf{P}\left(\exists s \le S_\ell^{\max} \text{ s.t. } \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{s}\right) \\
&\quad \le \mathbf{P}\left(\exists 1 \le s < \infty \text{ s.t. } \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{s}\right).
\end{aligned}$$

Now, observe that the empirical means $\hat{\mu}_\ell$ decrease in intervals without successes. Namely, if $a < b$ are time indices such that $x_\ell^a = 1, x_\ell^b = 1$ and for all $s \in [a+1, b-1]$, $x_\ell^s = 0$, then for any $s \in [a, b-1]$, it holds that $\hat{\mu}_\ell(s) \ge \hat{\mu}_\ell(b-1)$. We thus have:

$$\begin{aligned}
&\mathbb{P}\left(\exists 1 \le s < \infty : d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s},\ \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k}\right) \\
&\qquad = \mathbb{P}\left(\exists 1 \le s < \infty : d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s},\ \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k},\ x_\ell^{s+1} = 1\right).
\end{aligned}$$

Using the union bound, this implies

$$\begin{aligned}
&\mathbf{P}\left(\exists h \text{ s.t. } u_\ell(h) \le \mu_\ell - \varepsilon_{\ell,k}\right) \\
&\quad \le \sum_{s=1}^{\infty} \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k}, \text{ and } x_\ell^{s+1} = 1\right) \\
&\quad = \sum_{s=1}^{\infty} \mathbb{P}\left(x_\ell^{s+1} = 1\right) \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k} | x_\ell^{s+1} = 1\right) \\
&\quad = \sum_{s=1}^{\infty} \mu_\ell \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) < \mu_\ell - \varepsilon_{\ell,k}\right) \\
&\quad \le \sum_{s=1}^{\infty} \mu_\ell \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu\right) > \frac{\log\left(n^2K^2\right)}{s} + d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell), \hat{\mu}_\ell(s) < \mu_\ell\right),
\end{aligned}$$

where we used the fact that the sequence x_ℓ^s is independent and the last inequality is by (Lattimore and Szepesvári, 2020, Lemma 10.2, (c)). Next, using Chernoff's bound (Theorem 7.A.19), we

get

$$\begin{aligned}\mathbf{P}\left(\exists h \text{ s.t. } u_\ell(h) \leq \mu_\ell - \varepsilon_{\ell,k}\right) &\leq \mu_\ell \sum_{s=1}^{\infty} \exp\left(-s\left(d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell) + \frac{\log\left(n^2K^2\right)}{s}\right)\right) \\ &\leq \frac{\mu_\ell}{n^2K^2} \sum_{s=1}^{\infty} \exp\left(-sd(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)\right) \\ &\leq \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)},\end{aligned}$$

which concludes the proof of Theorem 7.A.20. □

Finally, substituting back into (ii) leads to the bound

$$\begin{aligned}(ii) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n^2(\lambda_k - \lambda_\ell) \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)} \\ =& \frac{1}{K^2} \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \frac{\mu_\ell(\lambda_k - \lambda_\ell)}{d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}. \end{aligned} \tag{7.33}$$

Combining both bounds.

$$\begin{aligned}(*) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\frac{\mu_k \log\left(n^2K^2\right)}{d(\mu_k + \varepsilon_{k,\ell}, \mu_\ell - \varepsilon_{\ell,k})} + \frac{\mu_k}{d\left(\mu_k + \varepsilon_{k,\ell}, \mu_k\right)}\right) \\ &+ \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \frac{\mu_\ell}{K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}.\end{aligned}$$

We now use a local refinement of Pinsker's inequality Garivier et al. (2019):

$$d(p, q) \geq \frac{1}{2\max(p, q)}(p - q)^2.$$

This implies:

$$\begin{aligned}(*) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\frac{2\mu_k\left(\mu_\ell - \varepsilon_{\ell,k}\right)\log\left(n^2K^2\right)}{(\mu_\ell - \varepsilon_{\ell,k} - \mu_k - \varepsilon_{k,\ell})^2} + \frac{2\mu_k(\mu_k + \varepsilon_{k,\ell})}{\varepsilon_{k,\ell}^2}\right) \\ &+ \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2}.\end{aligned}$$

Case 1: Assume $\mu_\ell \geq 5\mu_k$. Setting $\varepsilon_{k,\ell} = \mu_k$, we obtain,

$$\begin{aligned}(*) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\frac{2\mu_k(\mu_\ell - \varepsilon_{\ell,k})\log\left(n^2K^3\right)}{(\mu_\ell - \varepsilon_{\ell,k} - 2\mu_k)^2} + 4\right) + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2} \\ \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\frac{2\mu_k(\mu_\ell - \varepsilon_{\ell,k})\log\left(n^2K^3\right)}{(\frac{3}{5}\mu_\ell - \varepsilon_{\ell,k})^2} + 4\right) + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2}\end{aligned}$$

Setting $\varepsilon_{\ell,k} = \frac{1}{5}\mu_\ell$, we get:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\left(\frac{10\mu_k \log\left(n^2K^2\right)}{\mu_\ell} + 4\right) + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k - \lambda_\ell)\frac{50}{K^2}.$$

We have $\mu_k = 1 - e^{-\frac{\Delta}{\lambda_k}} \leq \frac{\Delta}{\lambda_k}$, and if $\Delta \leq \frac{1}{4}\lambda_\ell$, $\frac{1}{\mu_\ell} \leq 1.13\frac{\lambda_\ell}{\Delta}$, this implies:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\lambda_\ell 11.3 \log\left(n^2K^2\right) + \sum_{\ell=1}^{K}\lambda_\ell\left(\frac{50}{K} + 4nK\right).$$

Since $K \geq 2$, this implies:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\lambda_\ell 11.3 \log\left(n^2K^2\right) + \sum_{\ell=1}^{K}\lambda_\ell\left(12.5 + 4n\right)K.$$

Case 2: Assume $\mu_\ell \leq 5\mu_k$. Setting $\varepsilon_{k,\ell} = (\mu_\ell - \mu_k)/4$ and $\varepsilon_{\ell,k} = (\mu_\ell - \mu_k)/4$, we obtain,

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\frac{\mu_k^2}{(\mu_\ell - \mu_k)^2}\left(32\log\left(n^2K^2\right) + 64\right) + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k - \lambda_\ell)\frac{32\mu_\ell^2}{K^2(\mu_\ell - \mu_k)^2}.$$

It also holds that $(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n^2(\lambda_k - \lambda_\ell)$. Thus:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\frac{4\mu_k}{(\mu_\ell - \mu_k)}\sqrt{2n\left(\log\left(n^2K^3\right) + 4\right)} + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k - \lambda_\ell)\frac{4\sqrt{2}\mu_\ell n}{K(\mu_\ell - \mu_k)}.$$

If $\Delta \leq \frac{1}{4}\lambda_\ell$, we have:

$$\frac{1}{\mu_\ell - \mu_k} \leq 1.46\frac{\lambda_k\lambda_\ell}{(\lambda_k - \lambda_\ell)}.$$

We thus obtain:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} 6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right) + 4\right)} + \sum_{\ell=1}^{K} 9n\lambda_\ell.$$

For any $n \geq \max(10, 10\log(K))$, we have:

$$\ln(n^2K^2) \leq \frac{1}{2}n$$

which implies $6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right) + 4\right)} \geq 11.3\log\left(n^2K^2\right)$. Thus for any $n \geq \max(10, 10\log(K))$ and $\Delta \leq \frac{1}{4}\lambda_\ell$,

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} 6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right) + 4\right)} + \frac{10K}{n}\mathrm{E}[C_{OPT}].$$

7.A.4 Additional experiments

We implemented the Bayesian approach of Marbán et al. (2011) that we call LSEPT. We used an uninformative prior $\alpha = 1$, $w = 0$ (the same for all job types). LSEPT is then in essence a greedy algorithm. Whenever a job finishes, it runs until completion a job whose type has the lowest empirical mean size (computed across jobs that have been processed so far).

We ran all algorithms with $K = 2$, where jobs of type 1 have a mean size $\lambda_1 = 0.8$ and jobs of type 2 have a mean $\lambda_2 = 1$.

As can be seen in Figure 7.3, LSEPT has better mean performance than RR, a non-adaptive method. However, it has a large variance and its performance does not improve with n. This is typical of the performance of greedy algorithms: since the algorithm commits very early, it can either get very good or very bad performances. We plot the mean over 200 seeds.

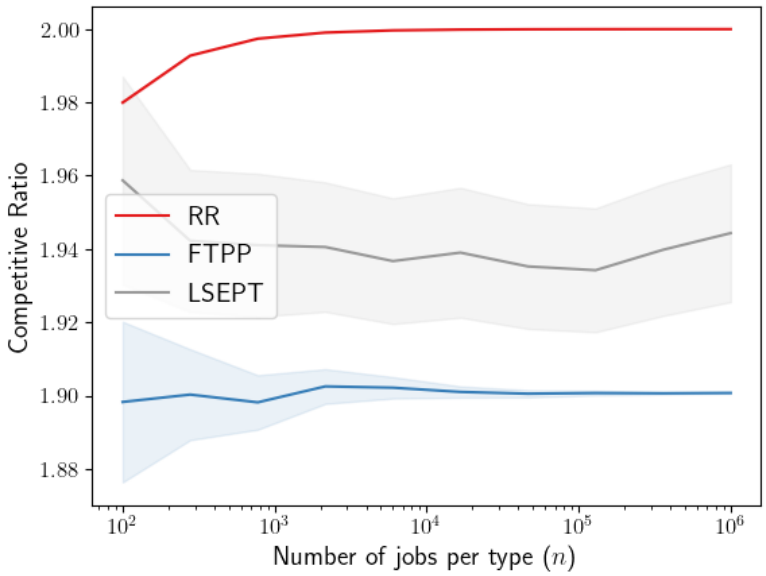


Figure 7.3: **CR on jobs with 2 different types**. $K = 2$, $\lambda_2 = 1$ and $\lambda_1 = 0.8$, n takes a grid of values.

Bibliography

Aamand, A., Chen, J., and Indyk, P. (2022). (optimal) online bipartite matching with degree information. *Advances in Neural Information Processing Systems*, 35:5724–5737.

Akbarpour, M., Alimohammadi, Y., Li, S., and Saberi, A. (2021). The value of excess supply in spatial matching markets.

Anandkumar, A., Michael, N., and Tang, A. (2010). Opportunistic spectrum access with multiple users: Learning under competition. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE.

Arnosti, N. (2019). Greedy matching in bipartite random graphs. *working paper*.

Arora, R., Dekel, O., and Tewari, A. (2012). Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1747–1754.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Bachmann, S. and Peccati, G. (2015). Concentration bounds for geometric poisson functionals: Logarithmic sobolev inequalities revisited.

Backstrom, L., Boldi, P., Rosa, M., Ugander, J., and Vigna, S. (2012). Four degrees of separation.

Bansal, N. and Dhamdhere, K. (2007). Minimizing weighted flow time. *ACM Transactions on Algorithms (TALG)*, 3(4):39–es.

Barabási, A.-L., Albert, R., and Jeong, H. (2000). Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: statistical mechanics and its applications*, 281(1-4):69–77.

Basu, S., Sankararaman, K. A., and Sankararaman, A. (2021). Beyond $\log^2(t)$ regret for decentralized bandits in matching markets. *arXiv preprint arXiv:2103.07501*.

Becchetti, L. and Leonardi, S. (2004). Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM*, 51(4):517–539.

Bender, E. A. and Canfield, E. R. (1978). The asymptotic number of labeled graphs with given degree sequences. *J. Combinatorial Theory Ser. A*, 24(3):296–307.

Berge, C. (1957). Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43(9):842–844.

Berger, J. (1899). Schach-jahrbuch fur 1899/1900 : fortsetzung des schach-jahrbuches fur 1892/93. *Verlag von Veit*.

Birkhoff, G. (1946). Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucuman, Ser. A*, 5:147–154.

Birnbaum, B. and Mathieu, C. (2008). On-line bipartite matching made simple. *SIGACT News*, 39(1):80–87.

Bistritz, I. and Leshem, A. (2018). Distributed multi-player bandits-a game of thrones approach. *Advances in Neural Information Processing Systems (NeurIPS)*.

Blaszczyszyn, B. (2017). Lecture Notes on Random Geometric Models — Random Graphs, Point Processes and Stochastic Geometry. Lecture.

Blum, A. and Monsour, Y. (2007). Learning, regret minimization, and equilibria. *Algorithmic Game Theory*.

Bollobás, B. (1980). A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European J. Combin.*, 1(4):311–316.

Bonnefoi, R., Besson, L., Moy, C., Kaufmann, E., and Palicot, J. (2017). Multi-armed bandit learning in iot networks: Learning helps even in non-stationary settings. In *International Conference on Cognitive Radio Oriented Wireless Networks*, pages 173–185. Springer.

Bordenave, C., Lelarge, M., and Salez, J. (2013). Matchings on infinite graphs. *Probability Theory and Related Fields*, 157(1-2):183–208.

Borodin, A. and El-Yaniv, R. (2005). *Online computation and competitive analysis*. cambridge university press.

Borodin, A., Karavasilis, C., and Pankratov, D. (2018). Greedy bipartite matching in random type poisson arrival model. *arXiv preprint arXiv:1805.00578*.

Borodin, A., Kleinberg, J., Raghavan, P., Sudan, M., and Williamson, D. P. (1996). Adversarial queueing theory. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 376–385.

Boursier, E. and Perchet, V. (2019). SIC-MMAB: Synchronisation involves communication in multiplayer multi-armed bandits. In *NIPS Proceedings*.

Boursier, E. and Perchet, V. (2020). Selfish robustness and equilibria in multi-player bandits. In *Conference on Learning Theory*, pages 530–581. PMLR.

Brubach, B., Grammel, N., Ma, W., and Srinivasan, A. (2021). Follow your star: New frameworks for online stochastic matching with known and unknown patience. In Banerjee, A. and Fukumizu, K., editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2872–2880. PMLR.

Brubach, B., Sankararaman, K. A., Srinivasan, A., and Xu, P. (2019). Online stochastic matching: New algorithms and bounds.

Bubeck, S. (2014). Convex optimization: Algorithms and complexity. *arXiv preprint arXiv:1405.4980*.

Bubeck, S., Budzinski, T., and Sellke, M. (2020). Cooperative and stochastic multi-player multi-armed bandit: Optimal regret with neither communication nor collisions. *arXiv preprint arXiv:2011.03896*.

Cai, X., Wu, X., and Zhou, X. (2014). *Optimal Stochastic Scheduling*, volume 4. Springer.

Cai, X. and Zhou, X. (2000). Asymmetric earliness and tardiness scheduling with exponential processing times on an unreliable machine. *Annals of Operations Research*, 98(1):313–331.

Cai, X. and Zhou, X. (2005). Single-machine scheduling with exponential processing times and general stochastic cost functions. *Journal of Global Optimization*, 31(2):317–332.

Calin, O. and Udrişte, C. (2014). *Geometric modeling in probability and statistics*, volume 121. Springer.

Cappé , O., Garivier, A., Maillard, O.-A., Munos, R., and Stoltz, G. (2013). Kullback–leibler upper confidence bounds for optimal sequential allocation. *The Annals of Statistics*, 41(3).

Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

Cesa-Bianchi, N. and Lugosi, G. (2012). Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422.

Chen, S., Lin, T., King, I., Lyu, M. R., and Chen, W. (2014). Combinatorial pure exploration of multi-armed bandits. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 379–387.

Chung, F., Lu, L., and Vu, V. (2003). Spectra of random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 100(11):6313–6318.

Combes, R., Shahi, M. S. T. M., Proutiere, A., et al. (2015). Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pages 2116–2124.

Cunningham, A. A. and Dutta, S. K. (1973). Scheduling jobs, with exponentially distributed processing times, on two machines of a flow shop. *Naval Research Logistics Quarterly*, 20(1):69–81.

Cuvelier, T., Combes, R., and Gourdin, E. (2021). Statistically efficient, polynomial time algorithms for combinatorial semi bandits.

Degenne, R. and Perchet, V. (2016). Combinatorial semi-bandit with known covariance. In *NIPS 2016 (Conference on Neural Information Processing Systems)*.

Devanur, N., Jain, K., and Kleinberg, R. (2013). Randomized primal-dual analysis of ranking for online bipartite matching.

Devanur, N. R., Sivan, B., and Azar, Y. (2012). Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, page 388–404, New York, NY, USA. Association for Computing Machinery.

Dufossé, F., Kaya, K., Panagiotas, I., and Uçar, B. (2018). Further notes on birkhoff–von neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 554:68–78.

Dürr, C., Erlebach, T., Megow, N., and Meißner, J. (2020). An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675.

Dyer, M., Frieze, A., and Pittel, B. (1993). The average performance of the greedy matching algorithm. *The Annals of Applied Probability*, pages 526–552.

Enriquez, N., Faraud, G., Ménard, L., and Noiry, N. (2019). Depth first exploration of a configuration model. *arXiv preprint arXiv:1911.10083*.

Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '99, page 251–262, New York, NY, USA. Association for Computing Machinery.

Feldman, J., Mehta, A., Mirrokni, V., and Muthukrishnan, S. (2009). Online stochastic matching: Beating 1-1/e. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126. IEEE.

Feng, Y., Niazadeh, R., and Saberi, A. (2021). *Two-stage Stochastic Matching with Application to Ride Hailing*, pages 2862–2877.

Freund, D., Lykouris, T., and Weng, W. (2022). Efficient decentralized multi-agent learning in asymmetric queuing systems. In *Conference on Learning Theory*, pages 4080–4084. PMLR.

Frieze, A. and Karoński, M. (2016). *Introduction to random graphs*. Cambridge University Press.

Fudenberg, D., Drew, F., Levine, D. K., and Levine, D. K. (1998). *The theory of learning in games*, volume 2. MIT press.

Gaitonde, J. and Tardos, E. (2020a). Stability and learning in strategic queuing systems. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 319–347.

Gaitonde, J. and Tardos, E. (2020b). Virtues of patience in strategic queuing systems. *arXiv preprint arXiv:2011.10205*.

Gamlath, B., Kapralov, M., Maggiori, A., Svensson, O., and Wajc, D. (2019). Online matching with general arrivals. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 26–37. IEEE.

Garivier, A. and Kaufmann, E. (2016). Optimal best arm identification with fixed confidence.

Garivier, A., Ménard, P., and Stoltz, G. (2019). Explore first, exploit next: The true shape of regret in bandit problems. *Mathematics of Operations Research*, 44(2):377–399.

Georgiadis, L., Neely, M. J., and Tassiulas, L. (2006). *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc.

Glazebrook, K. D. (1979). Scheduling tasks with exponential service times on parallel processors. *Journal of Applied Probability*, 16(3):685–689.

Godsil, C. D. (1981). Matchings and walks in graphs. *Journal of Graph Theory*, 5(3):285–297.

Goel, G. and Mehta, A. (2008). Online budgeted matching in random input models with applications to adwords. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, page 982–991, USA. Society for Industrial and Applied Mathematics.

Gravin, N., Tang, Z. G., and Wang, K. (2019). Online stochastic matching with edge arrivals.

Gupta, A., Guruganesh, G., Peng, B., and Wajc, D. (2019). Stochastic online metric matching.

Hamada, T. and Glazebrook, K. D. (1993). A bayesian sequential single machine scheduling problem to minimize the expected weighted sum of flowtimes of jobs with exponential processing times. *Operations Research*, 41(5):924–934.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hart, S. and Mas-Colell, A. (2000). A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150.

Huang, Z., Shu, X., and Yan, S. (2022). The power of multiple choices in online stochastic matching. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 91–103, New York, NY, USA. Association for Computing Machinery.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95.

Jaillet, P. and Lu, X. (2014). Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646.

Jiang, L. and Walrand, J. (2009). A distributed csma algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking*, 18(3):960–972.

Johari, R., Kamble, V., Krishnaswamy, A. K., and Li, H. (2018). Exploration vs. exploitation in team formation. *CoRR*, abs/1809.06937.

Kalyanasundaram, B. and Pruhs, K. R. (2000). An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1-2):319–325.

Kämpke, T. (1989). Optimal scheduling of jobs with exponential service times on identical parallel processors. *Operations Research*, 37(1):126–133.

Karande, C., Mehta, A., and Tripathi, P. (2011). Online bipartite matching with unknown distributions. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 587–596, New York, NY, USA. Association for Computing Machinery.

Karp, R. M., Vazirani, U. V., and Vazirani, V. V. (1990). An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, page 352–358, New York, NY, USA. Association for Computing Machinery.

Katariya, S., Kveton, B., Szepesvári, C., Vernade, C., and Wen, Z. (2017a). Bernoulli rank-1 bandits for click feedback. IJCAI'17, page 2001–2007. AAAI Press.

Katariya, S., Kveton, B., Szepesvari, C., Vernade, C., and Wen, Z. (2017b). Stochastic Rank-1 Bandits. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 392–401, Fort Lauderdale, FL, USA.

Kaufmann, E., Cappé, O., and Garivier, A. (2016). On the complexity of best-arm identification in multi-armed bandit models. *The Journal of Machine Learning Research*, 17(1):1–42.

Koutsoupias, E. and Papadimitriou, C. (1999). Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer.

Krishnasamy, S., Arapostathis, A., Johari, R., and Shakkottai, S. (2018). On learning the cμ rule in single and parallel server networks. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 153–154. IEEE.

Krishnasamy, S., Sen, R., Johari, R., and Shakkottai, S. (2016). Regret of queueing bandits. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Lai, T. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22.

Lattimore, T. and Szepesvári, C. (2020). *Bandit Algorithms*. Cambridge University Press.

Laurent, B. and Massart, P. (2000). Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338.

Lawler, E. L. and Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM (JACM)*, 25(4):612–619.

Lee, D. and Vojnovic, M. (2021). Scheduling jobs with stochastic holding costs. *Advances in Neural Information Processing Systems*, 34:19375–19384.

Levi, R., Magnanti, T., and Shaposhnik, Y. (2019). Scheduling with testing. *Management Science*, 65(2):776–793.

Li, H., Chen, J., Tao, Y., Gro, D., and Wolters, L. (2006). Improving a local learning technique for queuewait time predictions. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, volume 1, pages 335–342. IEEE.

Liu, L. T., Mania, H., and Jordan, M. (2020a). Competing bandits in matching markets. In *International Conference on Artificial Intelligence and Statistics*, pages 1618–1628. PMLR.

Liu, L. T., Ruan, F., Mania, H., and Jordan, M. I. (2020b). Bandit learning in decentralized matching markets. *arXiv preprint arXiv:2012.07348*.

Lovász, L. and Plummer, M. D. (2009a). *Matching theory*, volume 367. American Mathematical Soc.

Lovász, L. and Plummer, M. D. (2009b). *Matching theory*, volume 367. American Mathematical Soc.

Magerlein, J. M. and Martin, J. B. (1978). Surgical demand scheduling: a review. *Health services research*, 13(4):418.

Mahdian, M. and Yan, Q. (2011). Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. pages 597–606.

Manshadi, V. H., Gharan, S. O., and Saberi, A. (2012). Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573.

Marbán, S., Rutten, C., and Vredeveld, T. (2011). Learning in stochastic machine scheduling. In *International Workshop on Approximation and Online Algorithms*, pages 21–34. Springer.

Marshall, A. W., Olkin, I., and Arnold, B. C. (1979). *Inequalities: theory of majorization and its applications*, volume 143. Springer.

Mastin, A. and Jaillet, P. (2013). Greedy online bipartite matching on random graphs. *arXiv preprint arXiv:1307.2536*.

Mehrabian, A., Boursier, E., Kaufmann, E., and Perchet, V. (2020). A practical algorithm for multiplayer bandits when arm means vary among players. In *International Conference on Artificial Intelligence and Statistics*, pages 1211–1221. PMLR.

Mehta, A. (2012). Online matching and ad allocation. *Theoretical Computer Science*, 8(4):265–368.

Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. (2007). Adwords and generalized online matching. *J. ACM*, 54(5):22–es.

Miasojedow, B. (2014). Hoeffding's inequalities for geometrically ergodic markov chains on general state space. *Statistics & Probability Letters*, 87:115–120.

Mitzenmacher, M. (2020). Scheduling with predictions and the price of misprediction. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Mitzenmacher, M. and Vassilvitskii, S. (2020). Algorithms with predictions.

Motwani, R., Panigrahy, R., and Xu, Y. (2006). Fractional matching via balls-and-bins. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 487–498. Springer.

Motwani, R., Phillips, S., and Torng, E. (1994). Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47.

Neely, M. J., Modiano, E., and Li, C.-P. (2008). Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions On Networking*, 16(2):396–409.

Noiry, N., Sentenac, F., and Perchet, V. (2021). Online matching in sparse random graphs: Non-asymptotic performances of greedy algorithm. In *NeurIPS*.

Pemantle, R. and Rosenthal, J. S. (1999). Moment conditions for a sequence with negative drift to be uniformly bounded in lr. *Stochastic Processes and their Applications*, 82(1):143–155.

Perchet, V. (2014). Approachability, regret and calibration: Implications and equivalences. *Journal of Dynamics & Games*, 1(2):181.

Perchet, V. and Rigollet, P. (2013). The multi-armed bandit problem with covariates. *The Annals of Statistics*, 41(2).

Perrault, P., Boursier, E., Perchet, V., and Valko, M. (2020). Statistical efficiency of thompson sampling for combinatorial semi-bandits. In *NeurIPS*.

Pinedo, M. and Weiss, G. (1985). Scheduling jobs with exponentially distributed processing times and intree precedence constraints on two parallel machines. *Operations Research*, 33(6):1381–1388.

Pinedo, M. L. (2012). *Scheduling*, volume 29. Springer.

Polyanskiy, Y. and Wu, Y. (2014). Lecture notes on information theory. *Lecture Notes for ECE563 (UIUC) and*, 6(2012-2016):7.

Rejwan, I. and Mansour, Y. (2020). Top-k combinatorial bandits with full-bandit feedback. In Kontorovich, A. and Neu, G., editors, *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 752–776, San Diego, California, USA. PMLR.

Robbins, H. (1952). Some aspects of the sequential design of experiments.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2004). Kidney exchange. *The Quarterly journal of economics*, 119(2):457–488.

Roughgarden, T. (2010). Algorithmic game theory. *Communications of the ACM*, 53(7):78–86.

Sankararaman, A., Basu, S., and Sankararaman, K. A. (2020). Dominate or delete: Decentralized competing bandits with uniform valuation. *arXiv preprint arXiv:2006.15166*.

Schrage, L. (1968). A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3):687–690.

Schulz, A. S. and Moses, N. E. S. (2003). On the performance of user equilibria in traffic networks. In *SODA*, volume 3, pages 86–87.

Shah, D. and Shin, J. (2012). Randomized scheduling algorithm for queueing networks. *The Annals of Applied Probability*, 22(1):128–171.

Shortle, J. F., Thompson, J. M., Gross, D., and Harris, C. M. (2018). *Fundamentals of queueing theory*, volume 399. John Wiley & Sons.

Tassiulas, L. and Ephremides, A. (1990). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pages 2130–2132. IEEE.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294.

Trinh, C., Kaufmann, E., Vernade, C., and Combes, R. (2020). Solving Bernoulli rank-one bandits with unimodal Thompson sampling. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 862–889, San Diego, California, USA.

Tsung-Chyan, L., Sotskov, Y. N., Sotskova, N. Y., and Werner, F. (1997). Optimal makespan scheduling with given bounds of processing times. *Mathematical and Computer Modelling*, 26(3):67–86.

Van Der Hofstad, R. (2016). *Random graphs and complex networks*, volume 1. Cambridge university press.

Wang, P.-A., Proutiere, A., Ariu, K., Jedra, Y., and Russo, A. (2020). Optimal algorithms for multiplayer multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 4120–4129. PMLR.

Wang, S. and Chen, W. (2021). Thompson sampling for combinatorial semi-bandits.

Wheaton, W. C. (1990). Vacancy, search, and prices in a housing market matching model. *Journal of political Economy*, 98(6):1270–1292.

White, R. W. and Hassan Awadallah, A. (2019). Task duration estimation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 636–644.

Wormald, N. C. (1995). Differential equations for random processes and random graphs. *The annals of applied probability*, 5(4):1217–1235.

Yue, Y., Broder, J., Kleinberg, R., and Joachims, T. (2012). The k-armed dueling bandits problem. *J. Comput. Syst. Sci.*, 78(5):1538–1556.

Zdeborová, L. and Mézard, M. (2006). The number of matchings in random graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(05):P05003.

Zoghi, M., Tunys, T., Ghavamzadeh, M., Kveton, B., Szepesvari, C., and Wen, Z. (2017). Online learning to rank in stochastic click models. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4199–4208, International Convention Centre, Sydney, Australia.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre: Apprentissage et Algorithmes pour le Matching Séquentiel

Mots clés: Algorithmes, Apprentissage, Matching, Séquentiel

Résumé: Cette thèse se concentre principalement sur les problèmes de matching séquentiels, où des ensembles de ressources sont alloués de manière séquentielle à des flux de demandes. Nous les traitons à la fois du point de vue de l'apprentissage séquentiel et de l'analyse compétitive, toujours dans le cas où l'entrée est stochastique.
Du côté de l'apprentissage séquentiel, nous étudions comment la structure de matching spécifique influence l'apprentissage dans la première partie, puis comment les effets de report dans le système affectent ses performances.
Du côté de l'analyse compétitive, nous étudions le problème de matching séquentiel dans des classes spécifiques de graphes aléatoires, dans le but de nous éloigner de l'analyse du pire des cas.
Enfin, nous explorons comment l'apprentissage peut être utilisé dans le problème de scheduling.

Title: Learning and Algorithms for Online Matching

Keywords: Online; Algorithms; Learning; Matching

Abstract: This thesis focuses mainly on online matching problems, where sets of resources are sequentially allocated to demand streams. We treat them both from an online learning and a competitive analysis perspective, always in the case when the input is stochastic.
On the online learning side, we study how the specific matching structure influences learning in the first part, then how carry-over effects in the system affect its performance.
On the competitive analysis side, we study the online matching problem in specific classes of random graphs, in an effort to move away from worst-case analysis.
Finally, we explore how learning can be leveraged in the scheduling problem.

Institut Polytechnique de Paris
91120 Palaiseau, France