Load-balancing and resource-provisioning in large distributed systems

Mathieu Leconte

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2013-ENST-0092

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Mathieu LECONTE

le 18 décembre 2013

Équilibrage de charge et répartition de ressources dans les grands systèmes distribués

Directeur de thèse : **Laurent MASSOULIÉ**
Co-encadrement de la thèse : **Marc LELARGE**

Jury

M. Laurent Massoulié, Directeur du centre MSR - INRIA, Palaiseau	Directeur de Thèse
M. Marc LELARGE, Chargé de Recherche, ENS - INRIA Rocquencourt	Co-encadrant de Thèse
M. David GAMARNIK, Professeur, MIT, Cambridge, USA	Rapporteur
M. James ROBERTS, Chercheur Senior, IRT SystemX, Palaiseau	Rapporteur
M. François BACCELLI, Professeur, University of Texas, Austin, USA	Examinateur
Mme. Nidhi HEGDE, Chercheuse, Technicolor, Paris	Examinatrice
M. Alain JEAN-MARIE, Directeur de Recherche, LIRMM - INRIA Sophia-Antipolis	Examinateur
M. Florent KRZAKALA, Professeur, Université Paris VI et ENS	Examinateur

TELECOM ParisTech
École de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

THÈSE

Remerciements

Mes premiers remerciements vont naturellement à mes directeurs de thèse, Marc Lelarge et Laurent Massoulié, pour leurs conseils et les échanges fructueux que j'ai eus avec eux durant ma thèse. Je remercie chaleureusement Marc pour sa gentillesse, sa patience et sa disponibilité, et bien sûr pour les nombreux sujets de recherche auxquels il m'a exposé. Je suis également très reconnaissant envers Laurent pour sa vision des problèmes scientifiques actuels, qu'il est toujours prêt à partager, ainsi que pour sa patience et sa sympathie. Merci à vous deux également pour la liberté très formatrice que vous m'avez accordée dans la conduite de ma recherche.

I wish to thank James Roberts and David Gamarnik for their careful review of this thesis and their detailed comments, and for the discussions that followed. Je suis également reconnaissant envers François Baccelli, Florent Krzakala, Alain Jean-Marie et Nidhi Hegde pour avoir accepté de faire partie de mon jury, et envers Laurent Viennot et Laurent Decreusefond pour avoir constitué mon jury de mi-parcours.

I am also thankful to R. Srikant, Jian Ni, Xinzhou Wu, Cyril Measson and the Qualcomm team where I interned for a few month. It is the excellent atmosphere and fascinating work there that finally decided me to do a PhD.

Je suis très heureux d'avoir pu travailler au sein de deux équipes excellentes et chaleureuses, l'équipe DYOGENE (anciennement TREC) de l'INRIA et le laboratoire de Technicolor Paris, et je tiens à exprimer ma gratitude envers François Baccelli et Christophe Diot pour m'y avoir accueilli. Pour la bonne ambiance qu'ils savent y faire règner, merci aux membres permanents de ces deux équipes: Ana, Anne, Bartek, Hélène, Marc, et Pierre pour DYOGENE; et Augustin, Fabio, Henrik, Jaideep, Joris, Laurent, Martin, Nidhi, Pascal, Pieter et Theodoros pour Technicolor. Bien évidemment, je remercie tous les doctorants, post-doctorants et stagiaires qui se sont succédés au fil des années et avec qui j'ai pu partager un bout de chemin: Abir, Aleksander, Anastasios, Bruno, Calvin, Chandra, Emilie, Frédéric, Florence, Furcy, Hamed, Justin, Kumar, Lucas, Miodrag, Nadir, Omid, Paul, Rémi, Rui, Stéphane, Thibaut, Thomas, Tien et Yogesh pour DYOGENE; et Anna Kaisa, Abderrahmen, Dan-Cristian, Fernando, Giuseppe, Italo, Lou, Lucas, Marianna, Simon, Stéphane, Stratis et Valentina pour Technicolor. Merci également à Florence Besnard, Nicole Buatu, Isabelle Delais, Joëlle Isnard et Hélène Milome pour m'avoir assisté dans les démarches administratives.

Pour finir, je remercie mes amis et ma famille, et tout particulièrement Lise, pour leur soutien précieux pendant ces années et tout ce qu'ils m'apportent au quotidien.

Résumé

Cette thèse porte principalement sur l'équilibrage de charge dans de grands graphes aléatoires. En informatique, un problème d'équilibrage de charge survient lorsque différentes tâches ont besoin d'accéder à un même ensemble de points de ressources. Il faut alors décider quelles ressources spécifiques seront allouées à quelles tâches. Suivant le contexte, les notions de "tâche" et de "ressource" peuvent avoir différentes interprétations. Afin de prendre des exemples concrets, on se concentrera sur deux applications en particulier:

- un système de hachage à choix multiples (plus précisément, le "cuckoo hashing"). L'objectif est ici d'allouer des cellules d'un tableau à des objets, afin de pouvoir ensuite vérifier facilement la présence d'un objet et récupérer les données associées. Les tâches sont liées aux objets à stocker, et les ressources sont les cellules du tableau.

- un réseau de distribution de contenu distribué, au sens où les contenus peuvent être stockés sur une multitude de petits serveurs aux capacités individuelles très limitées. Ici, les tâches sont des demandes de téléchargement (ou requêtes) pour un contenu et les ressources sont liées aux serveurs et à la façon dont leurs espaces de stockage sont utilisés. Le problème d'équilibrage de charge consiste à décider quel serveur va servir quelle requête.

Les contraintes locales portant sur chaque ressource (en quelle quantité est-elle disponible et pour quelles tâches est-elle convenable?) ainsi que la charge de travail associée avec chaque tâche peuvent être représentées efficacement sur un graphe biparti, avec des contraintes de capacité sur ses sommets et ses arêtes. De plus, en pratique, les systèmes considérés sont souvent de très grande taille (avec parfois des milliers de tâches et de points de ressources différents) et relativement aléatoires (que ce soit par choix ou une conséquence de leur grande taille). Une modélisation à l'aide de grands graphes aléatoires est donc souvent pertinente.

L'ensemble des solutions envisageables pour un problème d'équilibrage de charge donné étant vaste, il est primordial de commencer par déterminer des bornes sur les performances que l'on peut espérer. Ainsi, on considérera dans un premier temps une solution optimale du problème (même si elle ne serait pas réalisable avec des contraintes pratiques). Les performances d'une telle solution peuvent être obtenues en étudiant les appariements de taille maximum dans un grand graphe aléatoire, ce que l'on réalisera à l'aide de la méthode de la cavité. Cette méthode vient de l'étude des systèmes désordonnés en physique statistique, et on s'attachera ici à l'appliquer de manière rigoureuse dans le cadre que l'on considère.

Dans le contexte du cuckoo hashing, les résultats obtenus permettent de calculer le seuil sur la charge du système (le nombre d'objets à insérer par rapport à la taille du tableau) en-dessous duquel on peut construire une table de hachage correcte avec grande probabilité dans un grand système, et également de traiter de manière similaire de variantes de la méthode de hachage basique qui tentent de diminuer la quantité d'aléa nécessaire au système. Au-delà du problème d'équilibrage de charge, dans le cadre des réseaux de distributions de contenu distribués, un second problème se pose: comment décider quel contenu stocker et en quelle quantité,

autrement dit comment répliquer les contenus? On appelle ce second problème un problème d'allocation de ressources. A nouveau, l'étude déjà réalisée permet de quantifier l'efficacité d'une politique de réplication fixée en supposant que la politique d'équilibrage de charge fonctionne de manière optimale. Il reste cependant à optimiser la politique de réplication de contenus utilisée, ce que l'on effectue dans un régime où l'espace de stockage disponible au niveau de chaque serveur est important par rapport à la taille d'un contenu. Finalement, afin de quantifier maintenant les performances minimales atteignables en pratique, on s'intéressera aux mêmes questions lorsque la politique d'équilibrage de charge utilisée est un simple algorithme glouton. Cette étude est réalisée à l'aide d'approximations de champs moyen. On utilisera également les résultats obtenus afin de concevoir des politiques de réplication de contenus adaptatives.

Abstract

The main theme of this thesis is load-balancing in large sparse random graphs. In the computer science context, a load-balancing problem occurs when we have a set of tasks which need to be distributed across multiple resources, and to resolve the load-balancing problem one needs to specify which tasks are going to be handled by which resources. Depending on the context, tasks and resources may have different interpretations. To make things more concrete, we focus in this document on two particular applications:

- a multiple-choice hashing system (often refered to as cuckoo hashing in the literature), where the goal is to efficiently assign buckets to items so that the items or any associated data can be stored to be later retrieved quickly. Tasks are here linked to items, and resources to buckets.

- a content delivery network (CDN) with a multitude of servers to handle storage and service of the contents. In this context, tasks are requests for a particular content and resources are linked with the servers and the particular contents they store, and resolving the load-balancing problem means assigning servers to requests.

The local constraints of which resource is suitable for a particular task as well as the initial amounts of the different available resources and the workload associated with each task can be efficiently represented as a capacitated bipartite graph. Also, in practice and in particular for the two examples mentioned, the systems considered are often of very large size, involving maybe thousands of different tasks and resources, and they tend to be quite random (either by design or due to a lack of coordination capabilities). Therefore, the context of large random graphs is particularly well-suited to the considered evaluations.

As the spectrum of solutions to a particular load-balancing problem is vast, it is primordial to understand the performance of the optimal solution to the load-balancing problem (disregarding its potential complexity) in order to assess the relative efficiency of any given candidate scheme. This optimal load-balancing performance can be derived from the size of maximum capacitated matchings in a large sparse random graph. We analyze this quantity using the cavity method –a powerful tool coming from the study of disordered systems in statistical physics–, showing in the process how to rigorously apply this method to the setups of interest for our work.

Coming back to the cuckoo hashing example, we obtain the load thresholds under which cuckoo hashing succeeds with high probability in building a valid hashtable and further show that the same approach can handle other related schemes. In the distributed CDN context, the performance of load-balancing is not the end of the story, as an associated resource-placement problem naturally arises: in such a system, one can choose how to provision resources and how to pool them, i.e., how to replicate contents over the servers. Our study of capacitated matchings already yields the efficiency of static replications of contents under optimal load-balancing, and we further obtain the limits of the optimal replication when the storage capacity of servers increases. Finally, as optimal load-balancing may be too complex for

many realistic distributed CDN systems, we address the issues of load-balancing performance and resource-placement optimization under a much simpler –random greedy– load-balancing scheme using mean-field large storage approximations. We also design efficient adaptive replication algorithms for this setup.

Contents

Chapter 1

Introduction

The main theme of this thesis is load-balancing in large distributed systems. In the computer science context, a load-balancing problem occurs when we have a set of tasks which need to be distributed across multiple resources, and to resolve the load-balancing problem one needs to specify which tasks are going to be handled by which resources. Depending on the context, tasks and resources may have different interpretations. To make things more concrete, we focus in this document on two particular applications:

- a multiple-choice hashing system (often refered to as cuckoo hashing in the literature), where the goal is to efficiently assign buckets to items so that the items or any associated data can be stored to be later retrieved quickly. Tasks are here linked to items, and resources to buckets.

- a content delivery network (CDN) with a multitude of servers to handle storage and service of the contents. In this context, tasks are requests for a particular content and resources are linked with the servers and the particular contents they store, and resolving the load-balancing problem means assigning servers to requests.

The local constraints of which resource is suitable for a particular task as well as the initial amounts of the different available resources and the workload associated with each task can be efficiently represented as a capacitated bipartite graph. Also, in practice and in particular for the two examples mentioned, the systems considered are often of very large size, involving maybe thousands of different tasks and resources, and they tend to be quite random (either by design or due to a lack of coordination capabilities). Therefore, the context of large random graphs is particularly well-suited to the considered evaluations.

This document is organized as follows: in the remainder of the introduction, we start by recalling a few standard notions of graph theory, including local weak convergence and some classical random graph models. We then explain how the two examples of cuckoo hashing and distributed CDNs can both be abstracted as a more general load-balancing problem, related to generalized matchings (called *capacitated matchings*) in a bipartite graph describing the constraints of a particular instance of the problem. In particular, the performance of large random systems can then be phrased in terms of local weak convergence of these constraints graphs. Then, we

give a short introduction to the cavity method, which underlies most of our work. To conclude the introduction, we provide an overview of the results obtained in this thesis. In Chapter 2, using the framework of local weak convergence, we extend the current range of rigorous applications of the cavity method by computing the asymptotic size of maximum capacitated matchings in a large sparse random graph. Chapter 3 is devoted to computing the cuckoo hashing thresholds, below which cuckoo hashing succeeds in building a valid hashtable, and to demonstrating the potential of this approach to handle related hashing schemes with little extra work. Finally, Chapter 4 is about load-balancing and resource-placement in distributed CDNs.

1.1 Notions of graph theory

In this section, we recall the notions of graph theory that will be useful in this work, including local weak convergence and standard random graph models. Additional concepts and notation will appear in the document as they are needed.

1.1.1 Graph terminology

A (simple, undirected) **graph** G consists of a possibly infinite set of vertices V and a set of edges E, each edge in E linking two distinct vertices of V together; if edge $e \in E$ links vertices u and v together, we write $e = uv$ and also $u \sim v$. The neighborhood of a vertex $v \in V$ in the graph G is the set of other vertices which are linked to v by an edge in E, i.e.,

$$\mathcal{N}(v) = \{u \in V : uv \in E\},$$

and the cardinality of $\mathcal{N}(v)$ is called the degree of v. The set of edges adjacent to a vertex $v \in V$ is denoted by

$$\partial v = \{e \in E : e = uv \text{ for some } u \in V\}.$$

If the degrees of all the vertices in V are finite, then we say G is **locally finite**. If the set V itself is finite, then we say G is finite.

A (simple) path of length k in G between the vertices u and v is a sequence of $k+1$ distinct neighboring vertices of G, the first one being u and the last one v. If there exists a path (of any length) between two vertices, then we say those vertices are connected together. If all pairs of vertices in V are connected together, then the graph G is itself connected. In the same way that $\mathcal{N}(v)$ is the set of vertices connected to v by a path of length 1, we denote by $\mathcal{N}^k(v)$ the set of vertices (excluding v) connected to v by a path of length at most k, for $k \geq 2$. A simple cycle in a graph is a path of length $k > 2$ starting and ending at the same vertex and visiting at most once any other vertex. A connected graph without cycles is called a **tree**. A **bipartite** graph $G = (L \cup R, E)$ is a graph where the vertex set V can be partitioned into two disjoint sets L and R (for "left" and "right") such that all the edges in E link a vertex in L and a vertex in R, i.e., for all $e \in E$ there exists $l \in L$ and $r \in R$ such that $e = lr$.

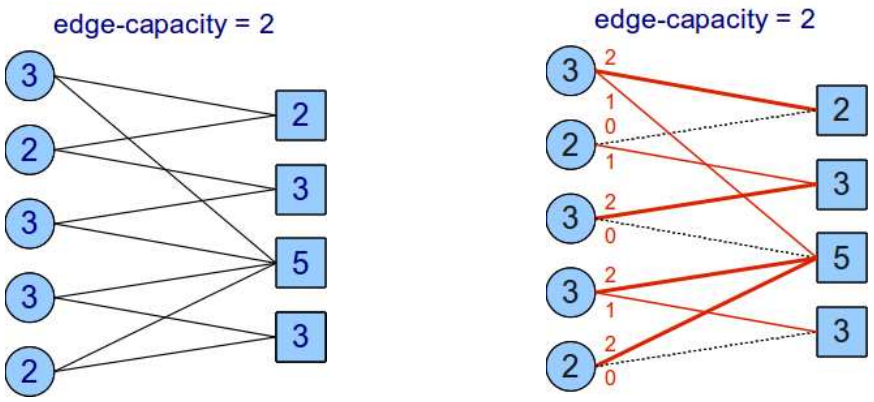

Figure 1.1: Example of capacitated graph and a capacitated matching of it.

Additionally, one can associate a **mark** to each element of a graph (vertex or edge). In our context, marks will most of the time be integers and represent capacity constraints associated with the corresponding element. Therefore, we will often directly call them capacities instead of marks. We denote by $\mathbf{b} = (b_v)_{v \in V}$ the set of capacities of vertices $v \in V$, and by $\mathbf{c} = (c_e)_{e \in E}$ the set capacities of edges $e \in E$. For simplicity, we write $G = (V, E)$, omitting mention of the marks when no confusion is possible. A **capacitated matching** (or c-capacitated b-matching [97]) $\mathbf{x}$ of the graph $G = (V, E)$ is an integer-valued vector with a coordinate associated with each edge in E, i.e., $\mathbf{x} \in \mathbb{N}^E$, such that

- the coordinate of $\mathbf{x}$ associated with an edge does not exceed the capacity (mark) of that edge, i.e., $x_e \leq c_e$ for all $e \in E$;
- the sum of coordinates of $\mathbf{x}$ over the edges adjacent to a vertex does not exceed the capacity of that vertex, i.e., $\sum_{e \in \partial v} x_e \leq b_v$ for all $v \in V$.

If the capacity constraints are all equal to 1, i.e., $\mathbf{b} \equiv 1$ and $\mathbf{c} \equiv 1$, then a capacitated matching of G is simply a matching of G. We let $\mathfrak{M}_G$ be the set of capacitated matchings of G. The size $|\mathbf{x}|$ of a particular capacitated matching $\mathbf{x} \in \mathfrak{M}_G$ is the sum of its coordinates, i.e., $|\mathbf{x}| = \sum_{e \in E} x_e$, and the maximum size of capacitated matchings of G is denoted by $M(G)$. As a notational rule, to denote a sub-collection of a vector, such as $(x_v)_{v \in S}$ where $S \subseteq V$, we use the notation $\mathbf{x}_S$. Hence, for example, the vertex-constraint on the capacitated matching $\mathbf{x}$ could be written $|\mathbf{x}_{\partial v}| \leq b_v$, $\forall v \in V$.

As an example, Figure 1.1 shows a bipartite capacitated graph $G = (L \cup R, E)$ on the left, with the vertex-capacities written inside the vertices and edge-capacities all equal to 2 (to avoid having too many numbers on the figure), and a capacitated matching $\mathbf{x} \in \mathfrak{M}_G$ of it on the right: the value x_e on an edge e is indicated by a red number at the left-endpoint of the edge and it can also be visualized in the thickness and color of the lines (black dotted lines indicate edges e which are not used by the capacitated matching, i.e., $x_e = 0$; red lines are for edges that are used, and they are thicker for larger x_e). The particular capacitated matching represented here has

size $|\mathbf{x}| = 11$ and one can check that the maximum size of capacitated matchings in this graph is $M(G) = 12$, although the sum of vertex-capacities is equal to 13 on both sides L and R.

In this thesis, we will mainly consider large random graphs, with the size of both parts $|L| = n$ and $|R| = m$ tending to infinity together, i.e., with $n \sim \tau m$ for some fixed $\tau > 0$, while the number of edges remains of the order of the number of vertices, i.e., $|E| = \Theta(|V|)$: this is called the **sparse regime** or the **diluted regime**. We will use the expression "with high probability" (or, in short, w.h.p.) to mean with probability tending to 1 as $m, n \to \infty$. In this regime, the graphs of interest converge in a sense explained in Section 1.1.2.

1.1.2 Local weak convergence

The notion of local weak convergence of a sequence of graphs was introduced by Benjamini and Schramm [15] to study random walks on planar graphs. This framework was further formalized by Aldous and Steele in [5] and by Aldous and Lyons in [4]. More recent studies on the subject include [19, 23, 22].

A first step is to introduce the notion of local convergence of rooted graphs. A **rooted graph** (G, v) is a graph G together with a distinguished vertex v of G which is called the root. An **isomorphism** of rooted graphs is a graph isomorphism which sends the root of one to the root of the other, i.e., a one-to-one mapping of the vertices of a graph to those of another graph which preserves the root, the edges (and their marks). We denote by $[G, v]$ the isomorphism class of (G, v) and by $\mathcal{G}_*$ the set of isomorphism classes of rooted locally finite graphs. For any $k \in \mathbb{N}$, we denote by $(G, v)_k$ the rooted subgraph obtained from (G, v) by keeping only those vertices at graph-distance at most k from v, and we write $[G, v]_k = [(G, v)_k]$ for its isomorphism class. We can turn $\mathcal{G}_*$ into a complete separable metric space (i.e., a Polish space) by defining the following metric: let the distance between $[G, v]$ and $[G', v']$ be $1/(1 + \delta)$, where δ is the supremum of those $k \in \mathbb{N}$ such that $[G, v]_k = [G', v']_k$. Then, a sequence of rooted graphs $((G_n, v_n))_{n\in\mathbb{N}}$ **converges locally** to (G, v) in $\mathcal{G}_*$ if the distance between (G_n, v_n) and (G, v) goes to 0 as $n \to \infty$. Equivalently,

$$(G_n, v_n) \underset{n\to\infty}{\longrightarrow} (G, v) \text{ locally} \iff \forall k \in \mathbb{N}, \exists n_k \text{ such that}, \forall n \geq n_k, [G_n, v_n]_k = [G, v]_k.$$

At this point, we recall that a sequence of probability measures $(\rho_n)_{n\in\mathbb{N}}$ on a certain metric space S **converges weakly** to a probability measure ρ, which we denote by $\rho_n \rightsquigarrow \rho$, if for every bounded continuous function $f : S \to \mathbb{R}$, $\int f d\rho_n$ converges to $\int f d\rho$. As $\mathcal{G}_*$ is a Polish space, we can endow it with its Borel σ-algebra (see, e.g., [16]) and consider the complete separable metric space of probability measures over $\mathcal{G}_*$, denoted by $\mathcal{P}(\mathcal{G}_*)$. The definition of weak convergence then applies to probability measures in $\mathcal{P}(\mathcal{G}_*)$. Furthermore, given a finite graph G there is a natural way to form a probability measure on $\mathcal{G}_*$: we let $U(G)$ be the distribution over $\mathcal{G}_*$ obtained by choosing a uniform random vertex of G as a root. Then, we say the sequence of finite graphs $(G_n)_{n\in\mathbb{N}}$ **converges locally weakly** towards the measure ρ on $\mathcal{G}_*$ when the sequence of distributions $(U(G_n))_{n\in\mathbb{N}}$ converges weakly towards ρ.

Any measure ρ which is the limit of a sequence of finite graphs in the sense seen above has the property that it is unimodular [5], which we define next. Similarly to the space $\mathcal{G}_*$, we define the space $\mathcal{G}_{**}$ of isomorphism classes of locally finite connected networks with an ordered pair of distinguished vertices and the natural topology thereon. We call ρ unimodular if it obeys the Mass-Transport Principle (MTP): for Borel functions $f : \mathcal{G}_{**} \to [0, \infty]$, we have

$$\int \sum_{v \in V} f(G, o, v) d\rho([G, o]) = \int \sum_{v \in V} f(G, v, o) d\rho([G, o]).$$

We let $\mathcal{U}$ denote the set of unimodular Borel probability measures on $\mathcal{G}_*$. For $\rho \in \mathcal{U}$, we write $\overline{b}(\rho)$ for the expectation of the capacity constraint of the root with respect to ρ.

1.1.3 Some classical random graph models

In this section, we introduce the Erdős-Rényi random graph, which is possibly the simplest random graph model, as well as some more complicated random graphs which we will use throughout the thesis. We should warn the reader that we will often use the notation for a particular random variable to refer in fact to its distribution when the meaning is clear from the context, so as to avoid introducing one more notation for the distribution itself.

The Erdős-Rényi random graph and the Poisson-Galton-Watson tree

The **Erdős-Rényi random graph** is the simplest and most studied random graph model. Two related versions of the model were introduced by Gilbert [54] and Erdős and Rényi [40]. For a set of n vertices and $p \in [0, 1]$, the Erdős-Rényi random graph $G(n, p)$ is obtained by including independently each edge among the $\binom{n}{2}$ possible edges with probability p. Therefore, taken individually, each vertex has a degree given by a binomial random variable $\mathrm{Bin}(n-1, p)$. As we are interested in the sparse regime, we focus mainly on $p = p(n) \sim \lambda/n$ as $n \to \infty$, for some fixed $\lambda \in \mathbb{R}_+$. Then, a sequence of Erdős-Rényi random graphs $\big(G(n, \lambda/n)\big)_{n \in \mathbb{N}}$ with increasing sizes admits almost surely a local weak limit in $\mathcal{P}(\mathcal{G}_*)$. This local weak limit is the **Poisson-Galton-Watson** (Poisson-GW) distribution $\mathcal{GW}(\mathrm{Poi}(\lambda))$ over trees with expected offspring λ, which was introduced in 1873 by Galton to study genealogical family trees [110]. A sample Poisson-GW tree is obtained through a branching process as follows: the root o has a number of children X given by a Poisson random variable with mean λ, i.e.

$$\mathbb{P}(X = k) = \mathbb{P}(\mathrm{Poi}(\lambda) = k) = \frac{\lambda^k}{k!} e^{-\lambda}.$$

Then, each of these children also independently has a number of children equal to $\mathrm{Poi}(\lambda)$, and so on. The rooted tree obtained via this process is almost surely locally finite. For a detailed study of brancing processes and random graphs, one can refer to [9, 107]. A detailed proof of the almost sure local weak convergence of the Erdős-Rényi random graph can be found in [6, 18, 61].

Random graphs with prescribed degree distribution

The Erdős-Rényi model has only one parameter which can be adjusted, the average degree. In order to better fit real-life graphs, it is useful to be able to capture more of their features and incorporate them into a random graph model. The so-called **configuration model** (introduced by Gallager [48] and refined by Bender and Canfield [14], Bollobás [17] and Wormald [115]) is one step in this direction, where one can specify the degree distribution of the n vertices of the graph. Given a degree distribution $\Phi \in \mathcal{P}(\mathbb{N})$ with finite mean, we first sample independently from Φ the degree d_v of each vertex $v \in V$. Starting from an empty graph, with vertex set V and no edges, we then add edges at random by pairing successively, uniformly and independently the stubs at each vertex in a manner consistent with the degree sequence $\mathbf{d} = (d_1, \dots, d_n)$. This procedure produces a few self-loops and multiple edges: they can be either corrected [75] to obtain a simple random graph $G(n, \mathbf{d})$ uniform among all graphs with the prescribed degree sequence $\mathbf{d}$, or erased / merged [26] to obtain simply a simple random graph with an empirical degree distribution $\widetilde{\mathbf{d}}$ converging to Φ as $n \to \infty$. In any case, a sequence of such random graphs $\big(G(n, \mathbf{d})\big)_{n\in\mathbb{N}}$ (or $\big(G(n, \widetilde{\mathbf{d}})\big)_{n\in\mathbb{N}}$) with increasing sizes admits almost surely a local weak limit $\mathcal{GW}(\Phi) \in \mathcal{P}(\mathcal{G}_*)$, which is the **unimodular Galton-Watson** distribution with degree distribution Φ (see Example 1.1 in [4], and [39] for detailed proofs of the convergence). The procedure to sample a unimodular GW tree is very similar to that explained before in the special case of Poisson-GW trees: the root o has a number of children drawn from Φ; however, all its descendants have numbers of children X independently drawn from the size-biased offspring distribution $\widetilde{\Phi} \in \mathcal{P}(\mathbb{N})$ defined by:

$$\mathbb{P}(X = k) = \widetilde{\Phi}_k = \frac{(k+1)\Phi_{k+1}}{\sum_{i\in\mathbb{N}} i\Phi_i}.$$

For the applications considered in this thesis, we will often encounter a similar model for bipartite random graphs $G = (L \cup R, E)$, where the degree (and constraints) distributions are different for the two parts of the graph: given two distributions Φ^L and Φ^R specifying jointly the degree, the vertex-constraint and the adjacent edge-constraints for vertices in L and R respectively, we can sample two sequences $(\mathbf{d}_l, \mathbf{b}_l, \mathbf{c}_{\partial l})_{l\in L}$ and $(\mathbf{d}_r, \mathbf{b}_r, \mathbf{c}_{\partial r})_{r\in R}$ indicating these quantities for each vertex of the graph. Provided these sequences meet some basic consistency conditions (the sum of degrees should be the same in L and R and the number of edges with a given capacity should be the same when counted from L and from R), we can generate a uniform random bipartite graph accordingly (see [29, 51] for more details on the bipartite case). A jointly defined sequence of such random graphs $(G_n)_{n\in\mathbb{N}}$ rooted at a random left-vertex, with $|L_n| = n$ and $|R_n| \sim \frac{1}{\tau}n$ for some fixed $\tau > 0$, converges almost surely to a local weak limit in $\mathcal{P}(\mathcal{G}_*)$. This time, the local weak limit $\mathcal{GW}(\Phi^L, \Phi^R)$ is a **two-step unimodular Galton-Watson** distribution with laws Φ^L and Φ^R, and a sample from this distribution can be obtained in a very similar way as before: the parameters of the root o are sampled from Φ^L, the parameters of its descendants at even and odd generations are drawn from size-biased offspring distributions $\widetilde{\Phi}^L$ and $\widetilde{\Phi}^R$ respectively, conditionally on the capacity of the

edge linking them to their parent. More details on this model can be found in Chapter 2.

In the next section, we present the two applications considered in this thesis in more detail.

1.2 Load-balancing problems

Load balancing is the process of distributing tasks among resources so as to make the load level of the resources as even as possible. This issue has grown very popular in particular with the emergence of distributed computing, where loading equally all the available processors ensures the quickest completion for a given set of tasks. However, even a basic setup, with two identical processors and a large number n of tasks with known durations, is already an NP-complete problem, as shown by Karp [63]. Furthermore, many internet applications are nowadays supported by server farms comprising thousands of machines. Therefore, in many applications of interest the number of resources itself is also growing. In this situation, with a large number of tasks and resources, the coordination capabilities of the distributed system become a predominant limitation, and it is often more important to find a simple yet efficient scheme –for which randomization is a useful tool– than an optimal deterministic solution, which complexity and communication requirements would render impractical.

In this thesis, we consider two main examples of load-balancing problems which share a common abstraction: the performance in the two setups is characterized by the size of a capacitated matching in a bipartite random graph describing the constraints of the problems. This section introduces more formally the two load-balancing examples mentioned and the main questions we try to answer in each context. Each time, we will point out the connection with capacitated matchings as well as the relevant random graph models.

1.2.1 Cuckoo hashing

A **hashtable** is a data structure that maps items to buckets so as to be able to later retrieve the items or any associated data. The idea is that the item (or an associated descriptor called a key) is hashed into the index of a bucket in an array (also called a value). Among the many possible items, only a small number is expected to be present at any time, therefore it is advantageous to dimension the array to store only that smaller expected number of items rather than the whole collection of items. The purpose of the hashing step is then to map many items to the same bucket index, as hopefully only one of those items is present most of the time; otherwise, a collision occurs and it needs to be resolved in some way. Cuckoo hashing is a particular scheme used to resolve such collisions.

The hashing paradigm is as follows: we are given n items and m buckets. In the most basic setting, we want to have exactly one bucket per item and at most one item per bucket. In more complex versions, multiple copies of each item must be stored, each bucket can hold many items simultaneously and there are restrictions on how many times an (item,bucket) pair can be used. Critical performance metrics

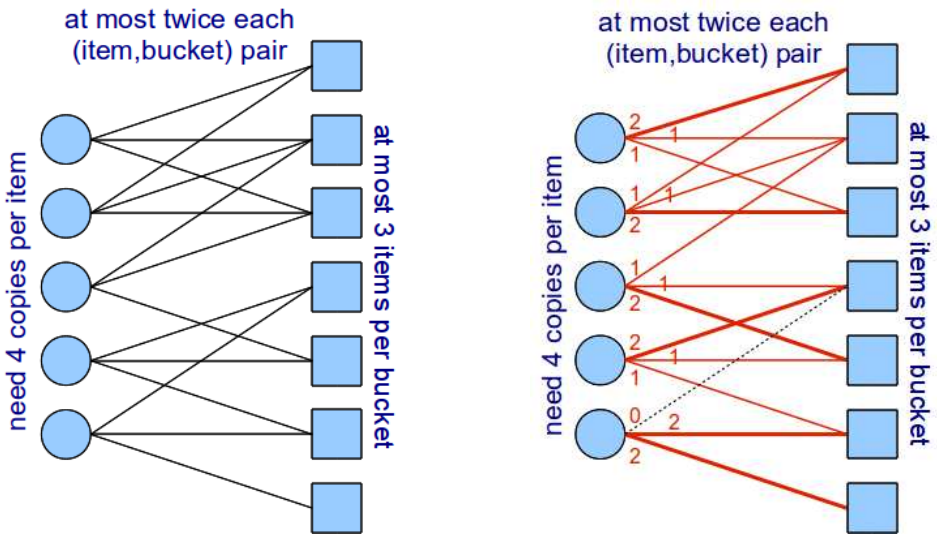

Figure 1.2: Two realizations of the hashing graph, with and without valid hashtable.

for such systems are the size of the hashtable (the number of buckets) needed to hold a given number of items, and the time it takes to either retrieve or insert items. The **multiple-choice hashing** strategy is one that guarantees a constant look-up time, while requiring with high probability a hashtable of size m proportional to n. This strategy consists in pre-defining a set of $d \geq 2$ buckets for each item, which is then only allowed to pick a bucket within that set. Of course, depending on the choices of the pre-defined sets of buckets, it may be impossible to handle simultaneously some sets of items and inserting a new item may not be easy. The first issue becomes very unlikely if the hashtable contains enough buckets compared to the number of items it needs to hold, and a lot of work has actually been devoted to figuring out exactly how large the hashtable needs to be. As for the second issue, **cuckoo hashing** was proposed by Pagh and Rodler [89] as a simple, randomized way to search for a new valid assignement upon arrival of an item: one first checks whether one of the buckets pre-defined for the new item is available, in which case it suffices to pick one of these; otherwise, one of the pre-defined buckets is chosen at random and re-allocated to the new item. The same procedure is then used for the item that has just been evicted (which is thus treated as a new item). An interesting survey [81] reviews the state of research on cuckoo hashing as of 2009 and states some related open questions at that time.

Formally, an instance of the cuckoo hashing scheme can be described by a bipartite graph $G = (L \cup R, E)$ called the **hashing graph** (or **cuckoo graph**), where L is the set of n items and R the set of m buckets. The edges adjacent to an item $l \in L$ indicate the d pre-determined buckets to which l can be assigned. To capture the more complex versions of cuckoo hashing, where one needs to store h copies of each item or where each bucket can hold at most k items at a time, we add respectively capacity constraints $\mathbf{b}_L \equiv h$ to the left-vertices (items) and $\mathbf{b}_R \equiv k$ to the right-vertices (buckets); similarly, if each (item,bucket) pair can be used at most

s times, it is indicated by the edge-capacities $\mathbf{c} \equiv s$. Note that minimum number of copies h of each items is modeled as a capacity-constraint $\mathbf{b}_L$ in the constraint graph; this will be required to establish a link between the feasibility of a cuckoo hashing instance and the size of maximum capacitated matchings in the corresponding hashing graph G. To illustrate the cuckoo hashing setup, Figure 1.2 shows two different hashing graphs, with the items on the left and the buckets on the right side. The two realizations have the same values for the parameters, i.e., the same degrees $d = 3$ for the items, each item must be stored $h = 4$ times, each bucket can store up to $k = 3$ items and each (item,bucket) pair can be used at most twice. Note that the two graphs differ by only one edge (between the central vertices of each part in the right figure). However, one cannot build a valid hashtable from the hashing graph on the left, while we pointed out a valid assignement on the right using the same notations as for capacitated matchings in Figure 1.1.

A given fixed graph G only represents a particular choice for the pre-determined buckets associated with each item. In theory, these pre-determined buckets are determined via d independent hash functions applied to each item, and we can assume that one makes sure each item obtains d *distinct* buckets. This should yield for each item a set of pre-determined buckets which is a random subset of size d choosen uniformly at random among all such subsets of buckets and independently for each item. Then, the hashing graph $G_{n,m}$ is actually a random bipartite graph with a constant degree d for left-vertices and $\mathrm{Poi}(dn/m)$ for right-vertices. As described in Section 1.1.3, the hashing graph $G_{n,m}$ almost surely converges locally weakly to a two-step unimodular Galton-Watson limit, as the size of the system increases with $n \sim \tau m$.

In this context, the main question we try to answer is how large does the hashtable need to be to handle n items chosen at random among the possible ones? We mentioned the number of buckets m required is proportional to n with high probability. The problem is thus to compute precisely the proportionality threshold $\tau^* = \tau^*_{d,h,k,s}$ such that, if $n = \tau m$ with $\tau < \tau^*$, in the limit of $m, n \to \infty$ cuckoo hashing will yield a valid hashtable with high probability.

Before proceeding to the next application, we attempt to give some intuition on why cuckoo hashing works well. Cuckoo hashing relies on a principle called *the power of two choices*: let us forget about the restriction that each bucket has a limited capacity for now, and suppose that we want to throw the n items –just as if they were balls– in the m buckets (and $m = n$ for simplicity) with the goal of minimizing the maximum number of items in any bucket. The simplest *distributed* scheme for approximately balancing the load (i.e., the number of items) of each bucket is to throw each item in a random bucket. This leads to each bucket holding on average one item, and the maximum load is approximately $\frac{\log n}{\log \log n}$ items with high probability [55]. However, a very simple distributed scheme provides an exponential improvement over this basic result, provided one throws the items sequentially: for each item, $d \geq 2$ random buckets are proposed, and one throws the item in the least loaded of the d buckets. This method yields a maximum load of $\frac{\log \log n}{\log d} + \Theta(1)$ with high probability, as shown by Azar, Broder, Karlin and Upfal [10], a result known as the power of (at least) two choices (see [83] for a survey of the techniques used to show this type of result and the applications stemming from it). Broadly

speaking, in the hashing context, the very slowly growing $\log \log n$ term tells us that buckets with a fixed capacity will hardly ever overflow, and even then the cuckoo hashing scheme will come into play and try to re-arrange the items to find a new valid assignement.

The next section is about distributed CDNs. For more explainations on cuckoo hashing, see the corresponding chapter of the thesis (Chapter 3).

1.2.2 Distributed CDNs

According to Cisco [31], the amount of multimedia traffic –the majority of which is video– carried over the Internet nowadays is of the order of exabytes (10^{18}) per month and growing quickly, and most of this traffic is carried by dedicated overlay networks called **content delivery networks** (CDNs). A centralized hosting solution for many of these contents does not seem a scalable solution, as the delivery of a lot of delay-sensitive contents from a large data center in the core of the network to end-users located at the edge of the network would require massive bandwidth provisioning. Therefore, many of the large CDNs have rather adopted a distributed architecture, taking advantage of the low storage prices and deploying a large number of servers and small data centers over many networks (see [37] for a description of the challenges associated with this distributed nature and faced by the CDN *Akamai*). A similar move towards distributed caching is also visible in the **content-centric** approach to networking, initiated by Nelson [88] and which realized that more than 90% of Internet traffic is due to content dissemination and that tremendous bandwidth savings can be achieved via distributed caching. Within this framework, the Internet Service Providers (ISPs) create a **cache network** by disseminating small caches in the network –e.g., by attaching them to the routers–, and contents are cached on their way to the end-user so that closer service points are available should the same content be requested again. However, deploying a CDN in such a way requires the cooperation of ISPs, which certainly involves quite a bit of negotiations, and even then it is not clear at this point how to efficiently organize the system. Alternatively, one can take inspiration from **peer-to-peer** (P2P) systems, which have become incredibly popular for file-sharing since *Napster* was introduced in 1999 and with *BitTorrent* now accounting for more than half of all file-sharing on the Internet. Such systems tend to be highly scalable and cost-efficient, as users who want to participate have to provide resources as well as using them. However, they may have a hard time providing high quality of service guarantees, especially for delay-sensitive content such as for a Video-on-Demand (VoD) service for example, due to the unpredictable behavior of peers (there have been nonetheless a few attempts at deploying such systems, e.g., *PPLive*). For large-scale delivery of delay-sensitive content, it is very tempting to combine some of the ideas above, by having a (smaller) data center assisted by devices already deployed at the users' premises, e.g., set-top boxes or triple-play gateways (e.g., as done in *NaDa* [106] and the *People's CDN*). Indeed, leveraging the resources (storage, upload bandwidth) already present at the edge of the network offers considerable potential for scalability, cost reduction and base performance upgrade, while the data center provides reliability and quality of service guarantees. We call such a system, where a central

data center is assisted by small devices (which we call servers) located at the edge of the network, an **edge-assisted CDN**. We stress the fact that the small servers may have limited capabilities, which makes cheap devices eligible for that role.

In an edge-assisted CDN, as the small servers are mainly there to reduce perational cost, one would want to use them as much as possible and to only direct requests to the data center as a last resort. Therefore, to understand and design such a system, we focus on the operation of the small servers and try to understand the performance of this sub-system. As we focus mainly on delay-sensitive contents (video streaming), when a request arrives we cannot delay it even if it cannot be served immediately by the small servers, therefore it has to be sent to the data center and can be viewed as *lost* for the small servers. We can thus adopt a loss network model (see [64]) for this sub-system, in contrast to queueing (also called *waiting*) models, where similar issues of efficient scheduling and replication have been considered notably in [7]. As a result of the analysis of the servers' loss network, one can then dimension appropriately the data center to meet global performance requirements. We could even imagine that the data center is hosted via a *cloud service*, which allows it to dynamically adapt service capacity to meet instantaneous demand. The efficiency of the system constituted of the small servers alone critically depends on two factors:

- content replication within servers, which mainly amounts to determining *how many times* contents must be replicated,
- and how incoming service requests are matched to servers holding the requested content.

These two issues are called the **replication** and the **matching** problems respectively. The latter can be seen as a load-balancing problem, where tasks are requests for contents and they need to be assigned servers storing the requested content. As for the replication problem, it allows us to play with the constraints of the load-balancing problem by modifying the set of resources which are suitable for a particular task; we refer to this as a resource-placement problem. Ideally, the design of solutions for these two problems should be done jointly. However, it might be a too amibious objective at this point. Therefore, we rather focus on two particular solutions to the matching problem: an *optimal* solution, which might not be practical, and a *random greedy* one, which is on the contrary quite simplistic and should always be feasible. As these two particular load-balancing strategies are in a sense extremal in the set of reasonable algorithms, understanding how they perform and when/why their performance differ should yield fundamental insights as well as quantitative reference points for the design of real systems. Once the performance of the load-balancing policy can be characterized for a given fixed replication, we can turn to the resource-placement problem and attempt to find the best possible replication policy, at least in large system / large storage limits. More details on distributed CDNs can be found in Chapter 4.

To make things more concrete for the reader as well as to make more apparent the connection with capacitated matchings and with the random graph models introduced previously, we now describe succinctly the edge-assisted CDN model. In the edge-assisted CDN model, a data center is assisted by a large number m of

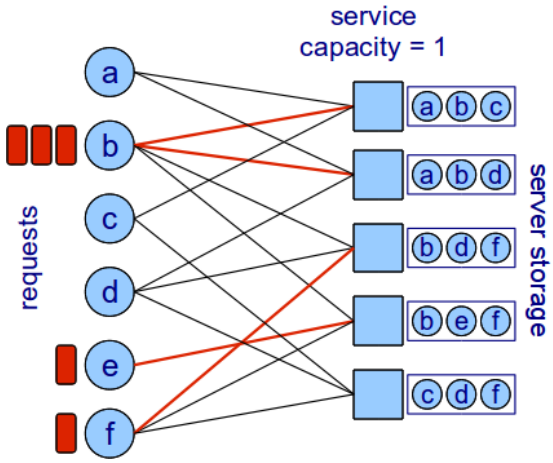

Figure 1.3: Content-server graph with a particular assignement of servers.

small servers. The major feature of this model is its focus on modeling the basic constraints of a realistic system, in particular regarding the limited capacity of servers (in terms of storage, service and coordination) and the way requests are handled, i.e., the matching policy. We work most of the time with a basic version of the model, in which all the contents have the same size (and we do not allow splitting them to parallelize service), and each server stores a fixed number d of contents and can provide service for only one request at a time. The information of which server stores which content is represented as a bipartite graph $G = (S \cup C, E)$, where S is the set of servers, C the set of contents, and there is an edge in E between a content c and a server s if s stores a copy of c (and is thus eligible to serve requests for c). The contents have different popularities, given by their request arrival rates λ_c, and each service takes and exponential time with mean 1. This results in an instantaneous number of requests R_c for each content given by independent $\mathrm{Poi}(\lambda_c)$ random variables. At any time, the current assignment of servers to contents forms a capacitated matching of the graph G (with capacities R_c for the contents and 1 for the servers). As an illustration, Figure 1.3 shows a particular content-server graph, with the contents on the left and servers on the right. Requests are shown as red rectangles beside the corresponding content and the contents stored by the servers are shown in the storage space on the right of the servers. If the servers are serving the contents as indicated by the edges in red, then one of the requests for content b must currently be handled by the data center.

Due to the lack of coordination between the servers, it is not feasible to carefully design the content of all the caches in a *joint* manner. Therefore, at best, one can specify a number of replicas D_c for each content c, and the servers will do random caching choices accordingly, so that the graph G will be a random bipartite graph with a prescribed degree sequence for the contents. At worst, very little information is known a priori on the contents, which are simply grouped into K classes of size $\alpha_1 n, \ldots, \alpha_K n$, and the servers have instructions on the caching preferences in the form of a distribution $\pi \in \mathcal{P}(\{1, \ldots, K\})$ over the different classes. They will then choose independently for each of their d memory slots a random content in a class drawn from π, so that G is a random bipartite graph with a prescribed degree

distribution $\mathrm{Poi}(dm\pi_k/n\alpha_k)$ for the contents of each class k. In this second case, the random graphs admit a two-step unimodular Galton-Watson local weak limit, as described in Section 1.1.3. However, if the precise sequence of degrees of the contents is imposed, then it is clear that we have to impose some conditions on this degree sequence for the graphs to have any chance to converge.

The next section explains the basics of the cavity method, which is used as a tool in the study of load-balancing throughout this thesis.

1.3 Introduction to the cavity method

1.3.1 Context

The cavity method was proposed in 1985 by Mézard, Parisi and Virasoro [79] (see also [77, 78]) to study spin glasses in statistical physics and more generally disordered systems (in our case especially, random graphs in the sparse regime). The goal is generally to compute various algebraic, combinatorial or probabilistic quantities as the size of the system increases. In many scenarios, one can work under the heuristic that the asymptotics of these quantities should only depend on the local geometry of the graphs and that long-range correlations should fade away. When this heuristic argument holds, the interest is twofold:

- firstly, the quantities of interest can be efficiently approximated via distributed, message-passing algorithms;
- secondly, these quantities are continuous with respect to local weak convergence and their limit can be defined on the local weak limit of the graphs, and sometimes computed explicitly.

To cite only a few applications of the cavity method: on the practical side, this approach led to the survey-propagation algorithm for constraint-satisfaction problems (see [80, 25, 74]), to the design of efficient LDPC codes [93] and to polynomial-time approximation algorithms and asymptotic expressions for many counting or decision problems (see, e.g., [112, 13, 50, 11, 49]); and on the theoretic side, to the $\zeta(2)$ limit in the assignement problem [2], the asymptotic enumeration of spanning trees in large graphs [73], the phase transitions for the random graph coloring problem [86, 118]. For a more extensive presentation of the cavity method itself, the interested reader can refer for example to [5, 4] or [34, 76].

1.3.2 The cavity method

We present here a short introduction to the cavity method, which is sufficient for understanding the principles and problems encountered in this thesis.

Suppose we are given a set of elements S and we consider simple configurations over S, i.e., vectors in $\{0,1\}^S$. Suppose also that to each of these configurations $\mathbf{x}$ is associated an energy $U(\mathbf{x})$ (also called Hamiltonian). If $U(\mathbf{x}) = +\infty$, we say that $\mathbf{x}$ is forbidden; on the contrary if $U(\mathbf{x}) < \infty$, then $\mathbf{x}$ is an admissible or valid

configuration. Now, consider a parameter $\beta > 0$ and the Gibbs distribution μ on configurations over S given by

$$\mu(\mathbf{x}) = \frac{e^{-\beta U(\mathbf{x})}}{Z(\beta)}, \tag{1.1}$$

where $Z(\beta) = \sum_{\mathbf{x}} e^{-\beta U(\mathbf{x})}$ is a normalizing factor called the *partition function*. In physics, the parameter β would play the role of an inverse temperature, i.e., $\beta = 1/T$, and the distribution μ would indicate the probability that a system of particles is in a particular state given that they interact according to some specific local, energy-dependent dynamics. The expected energy $\mathbb{E}[U(\mathbf{X})] = \sum_{\mathbf{x}} U(\mathbf{x})\mu(\mathbf{x})$ can be computed from

$$\mathbb{E}[U(\mathbf{X})] = -\frac{\partial}{\partial\beta} \log Z(\beta),$$

and the entropy of the distribution $H(\beta) = \mathbb{E}[-\log U(\mathbf{X})]$ from

$$H(1/T) = -\frac{\partial}{\partial T} \log TZ(1/T).$$

Therefore, these quantities can be computed for example from the partition function $Z(\beta)$, or from the *free energy* function $F(\beta) = -\frac{1}{\beta} \log Z(\beta)$.

To better motivate the computation of the expected energy or the entropy of the Gibbs distribution, let us describe briefly two examples where these quantities are meaningful. The first example concerns matchings and a more complex version of this problem is the topic of Chapter 2. Let S be the set of edges of a finite graph (we will therefore write $S = E$) on a set of vertices V, and let the energy U take the following form:

$$U(\mathbf{x}) = -|\mathbf{x}| - \sum_{v \in V} \log \mathbf{1}(|\mathbf{x}_{\partial v}| \leq 1),$$

so that $U(\mathbf{x}) = -|\mathbf{x}|$ if $\mathbf{x}$ is a matching of G, and $U(\mathbf{x}) = +\infty$ otherwise and $\mathbf{x}$ is a forbidden configuration. Then, μ is a distribution over the matchings of G which weights each matching $\mathbf{x}$ proportionally to $e^{-\beta|\mathbf{x}|}$. As $\beta \to \infty$, the distribution μ concentrates on maximum matchings of G, so that expected energy at temperature $T = 0$ measures the size of maximum matchings while the entropy gives the number of maximum matchings of G:

$$\begin{aligned} \lim_{\beta\to\infty} \mathbb{E}[U(\mathbf{X})] &= \min_{\mathbf{x}} U(\mathbf{x}) = -\max_{\mathbf{x}\in\mathfrak{M}_G} |\mathbf{x}| = M(G), \\ \lim_{\beta\to\infty} H(\beta) &= \log |\arg\min_{\mathbf{x}} U(\mathbf{x})|. \end{aligned}$$

The second example we provide here is the graph-coloring problem. Suppose we want to determine if the vertices of a graph $G = (V, E)$ are colorable using q colors so that no two adjacent vertices have the same color. This time, we let S be the set of vertices V of G and U count the number of edges with the same color at both endpoints:

$$U(\mathbf{x}) = \sum_{uv \in E} \mathbf{1}(x_u = x_v) \text{ for } \mathbf{x} \in \{1, \ldots, q\}^V.$$

If the expected energy at temperature $T = 0$ equals 0, then the graph is q-colorable and the limit $\lim_{\beta\to\infty} H(\beta)$ gives the number of proper q-colorings; otherwise the graph is not q-colorable.

In general, computing the partition function of a system is NP-hard. The cavity method essentially aims at computing this partition function when the energy function U is a sum of local contributions and the problem has a particular tree structure: suppose that $U(\mathbf{x}) = \sum_{a\in A} U_a(\mathbf{x}_a)$, where A is a collection of subsets of S and $\mathbf{x}_a = (x_v)_{v\in a}$, and that the bipartite graph $G = (S \cup A, E)$ is a tree, where $va \in E$ if $v \in a$. This graph is called a factor graph, and the vertices in S (resp. A) are called the *variable* nodes (resp. *factor* nodes). Then, the **Belief-Propagation** (BP) or Bethe-Peierls algorithm [90, 91] (also known as the sum-product algorithm) allows to compute the value of $Z(\beta)$ in an iterative manner: let us write $\psi_a(\mathbf{x}_a) = e^{-\beta U_a(\mathbf{x}_a)}$, so that $\mu(\mathbf{x}) = \frac{1}{Z(\beta)} \prod_{a\in A} \psi_a(\mathbf{x}_a)$, and let o be an arbitrary factor node in A; then

$$Z(\beta) = \sum_{\mathbf{x}} \prod_{a\in A} \psi_a(\mathbf{x}_a) = \sum_{\mathbf{x}_o} \psi_o(\mathbf{x}_o) \prod_{v\in o} \nu_{v\to o}(x_v),$$

where messages ν between neighboring variable nodes v and factor nodes a are defined by

$$\begin{aligned} \nu_{v\to a}(x_v) &= \prod_{b\neq a,\, v\in b} \nu_{b\to v}(x_v), \\ \nu_{a\to v}(x_v) &= \sum_{\mathbf{x}_{a\setminus v}} \psi_a(\mathbf{x}_a) \prod_{u\in a\setminus v} \nu_{u\to a}(x_u). \end{aligned}$$

The messages can be correctly initiated at the leaves of the tree G and then computed iteratively while climbing the tree up to the root o. It turns out that it will be more convenient to use normalized messages $\eta \propto \nu$ so that $\sum_{x_v} \eta_{v\to o}(x_v) = 1 = \sum_{x_v} \eta_{o\to v}(x_v)$. These are defined by the following equations, which we call the BP update equations:

$$\begin{aligned} \eta_{v\to a}(x_v) &= \frac{\prod_{b\neq a,\, v\in b} \eta_{b\to v}(x_v)}{\sum_{y_v} \prod_{b\neq a,\, v\in b} \eta_{b\to v}(y_v)}, \\ \eta_{a\to v}(x_v) &= \frac{\sum_{\mathbf{x}_{a\setminus v}} \psi_a(\mathbf{x}_a) \prod_{u\in a\setminus v} \eta_{u\to a}(x_u)}{\sum_{\mathbf{y}} \psi_a(\mathbf{y}_a) \prod_{u\in a\setminus v} \eta_{u\to a}(y_u)}. \end{aligned} \tag{1.2}$$

Note also that we can obtain the marginal probabilities of μ at a factor node o from the incoming BP messages η:

$$\mu(\mathbf{X}_o = \mathbf{x}_o) = \frac{\psi_o(\mathbf{x}_o)}{Z(\beta)} \prod_{v\in o} \nu_{v\to o}(x_v) = \frac{\psi_o(\mathbf{x}_o) \prod_{v\in o} \eta_{v\to o}(x_v)}{\sum_{\mathbf{y}_o} \psi_o(\mathbf{y}_o) \prod_{v\in o} \eta_{v\to o}(y_v)}$$

We next define the local contributions to the partition function $Z(\beta)$ and to the free energy $F(\beta)$, which are computed only from the incoming messages to the corresponding nodes:

$$\begin{aligned} z_v &= \textstyle\sum_{x_v} \prod_{a\ni v} \eta_{a\to v}(x_v), \\ z_a &= \textstyle\sum_{\mathbf{x}_a} \psi_a(\mathbf{x}_a) \prod_{v\in a} \eta_{v\to a}(x_v), \\ z_{av} = z_{va} &= \textstyle\sum_{x_v} \eta_{a\to v}(x_v)\eta_{v\to a}(x_v) \text{ for } va \in E, \end{aligned} \tag{1.3}$$

and $f_v = -T \log z_v$, $f_a = -T \log z_a$, and $f_{av} = -T \log z_{av}$. Then,

$$\begin{aligned} Z(\beta) &= \frac{\prod_v z_v \prod_a z_a}{\prod_{a,v} z_{av}}, \\ F(\beta) &= \sum_v f_c + \sum_a f_a - \sum_{a,v} f_{av}. \end{aligned}$$

In graphs G which are not trees, the message-passing iteration (1.2) that computes the normalized BP messages η cannot be safely initiated and it is not guaranteed to converge, at least, not to a meaningful value. In general, it is not clear when the BP iteration will converge. For the purpose of the explanations of this section we therefore assume that BP converges; specifically, this issue needs to be addressed when applying the cavity method to a particular problem. Let us consider the case where, as the size of the system increases, the graph $G = (S \cup A, E)$ converges locally weakly to a distribution ρ over unimodular trees. Then, provided correlations between BP messages decay sufficiently fast with the graph distance, which results in continuity properties with respect to local weak convergence, the limiting distribution of the BP messages η incoming to a random root of G is obtained directly on the local weak limit ρ as the distribution of the BP messages incoming to the root of ρ. Furthermore, as ρ is concentrated on unimodular trees, the quantities of interest (e.g., the marginal distribution of μ around o and the local contributions (1.3) of o) can be computed from the BP messages incoming to the root of ρ. This allows to define a limiting object for the Gibbs measure and the partition function directly on the local weak limit via an infinite set of local equations. Finally, when the local weak limit is a Galton-Watson distribution $\rho = \mathcal{GW}$ for some degree law –as for the random graph models discussed before–, then the local equations simplify into recursive distributional equations (RDEs), which may be solved explicitly in some cases to obtain an explicit formula for the quantities of interest (see the survey [3] by Aldous and Bandyopadhyay).

1.4 Overview of the results

In this section, we present a summary of the problems addressed, along with some relevant related work, and of the results obtained in this thesis.

1.4.1 Maximum capacitated matching via the cavity method

The cavity method has been successfully used to obtain results in many problems of computer science, combinatorics and statistical physics. As we explained, this method generally involves the Belief-Propagation message-passing algorithm for computing marginals of probability distributions. Belief-Propagation has been extensively used in various domains where it has been observed to converge quickly to meaningful limits [117, 74]. Although sufficient conditions for its convergence and correctness exist and the theoretical understanding of what may cause BP to fail advances [103, 58, 59, 85], there are only few cases where one can prove rigorously its convergence and the uniqueness of its fixed points for specific problems when the underlying graph is not a tree [111, 52, 12].

The issue of convergence of BP to a unique fixed point is easily resolved when the problem exhibits a property called *correlation decay*, which means that differences in a far-away portion of the system will have negligeable effects locally and generally implies continuity with respect to local weak convergence. However, in some cases, correlation decay fails at low (zero) temperature –a situation described as *ergodicity breaking*– and then multiple fixed-points to the BP equations may exist, which happens in particular for the problem of counting maximum matchings [119]. To resolve this issue, one can apply the cavity method at positive temperature and then take the limit when temperature T goes to 0 (i.e., $\beta \to \infty$). In the case of maximum matchings as well as in extensions [21, 96, 70], monotonicity and unimodularity arguments ensure the convergence of BP to a unique fixed point and also allow us to take the limit as $T \to 0$: in particular, the operator which computes the updated messages outgoing from vertices from the other incoming messages according to Equations (1.2) is non-increasing. Starting from the smallest possible initial messages, this automatically constructs a convergent sequence of BP messages, as it gives rise to two adjacent subsequences of messages (at even and odd numbers of BP iterations). The relevant monotonicity properties were captured precisely by Salez [96] and are related to the theory of negative association of measures initiated by Pemantle [92]. They hold in a context where the system can be encoded with binary values on the edges of the underlying graph and as a result involves BP with scalar messages. As a result, one can compute the size of the largest capacitated matchings, with unit edge capacities, in graphs $G_n = (L_n \cup R_n, E_n)$ converging locally weakly to two-step unimodular Galton-Watson trees $\mathcal{GW}(\Phi^L, \Phi^R)$ with a formula of the following form:

$$\lim_{n\to\infty} \frac{1}{|L_n|} M(G_n) = \mathcal{M}(\Phi^L, \Phi^R) = \inf_{q\in[0,1]} \left\{ \mathcal{F}^L(q),\ q = g^L \circ g^R(q) \right\},$$

where $\mathcal{F}^L$ is a function which computes the local contribution to a maximum capacitated matching around the root of the GW tree, and g^L and g^R are the expected BP update operators at vertices in L and R respectively, so that $q = g^L \circ g^R(q)$ is the two-step BP fixed-point equation. We do not give the expression for the various functions here, as their particular form is not very informative.

Motivated in particular by the two applications of this thesis and also by the general idea that many problems may require similar extensions, we want to go beyond unit edge capacities in the maximum capacitated matching problem, which necessarily involves non-scalar BP messages. The main obstacle here is that BP messages essentially represent distributions over the integers, which introduces a difficulty when trying to compare messages as $\mathcal{P}(\mathbb{N})$ is not totally ordered (contrary to the case of binary values on the edges). Nonetheless, similar monotonicity properties as in the scalar case hold when using the *upshifted likelihood ratio* (lr $\uparrow$) stochastic order for comparing BP messages: for two probability distributions $m, m' \in \mathcal{P}(\mathbb{N})$ on integers, the lr $\uparrow$-order is defined by

$$m \leq_{\mathrm{lr}\uparrow} m' \text{ if } m(i+k+l)m'(i) \leq m(i+l)m'(i+k), \forall i,k,l \in \mathbb{N}.$$

In particular, the BP update operator is non-increasing for the lr $\uparrow$-order, and we can obtain similar convergence results as in the scalar case. Note that the relevant

monotonicity properties do not trivially hold for all the natural stochastic orders: for example they do not hold for the strong stochastic order. The important feature of the lr $\uparrow$-order which makes it appropriate for our purpose is its preservation with respect to conditioning on intervals, a property characteristic of the *uniform conditional* stochastic order (ucso, see Whitt [113, 114]). Intuitively, such a preservation property is necessary for a local property (the BP update is non-increasing) to translate into a global effect (the even and odd iterations of BP yield adjacent sequences of messages), because the "important" part of a message incoming to a vertex v to determine the marginal of the Gibbs distribution (1.1) around v may end up being any interval due to the effect of the other messages incoming to v.

As an outcome of the study of the BP update operators, we can prove convergence of BP to a unique fixed point on any locally finite unimodular graph. Then, following the cavity method, the asymptotic behavior of $M(G)$ as the size of G increases can be obtained in the 0 temperature limit on the local weak limit $\mathcal{GW}(\Phi^L, \Phi^R)$ (defined in Section 1.1.3). Due to ergodicity breaking at temperature 0, there are however many BP fixed-points, and to identify the correct one we reason as in in [96, 70]. Furthermore, we do not need to solve recursive distributional equations on the BP messages themselves (which would be recursive equations on distributions over distributions) but rather only RDEs over the boundaries of the support of the BP messages, which are here intervals due to log-concavity of the BP fixed-point messages. The expression for the asymptotic of $M(G)$ is finally given by Theorem 2.1, which generalizes the formulas in [96, 70]. Furthermore, it is interesting to note that a similar expression is also valid for finite bipartite graphs (Theorem 2.11), which extends a result of Chertkov [30].

1.4.2 Cuckoo hashing

In the simplest settings, a popular approach to the problem of cuckoo hashing has been to study a certain core of the hashing graph, whose density of right-vertices exceeds a certain value at the same time as cuckoo hashing starts failing in random graphs of the type described above. The study of this core of the hashing graph can be done using various techniques (combinatorics and differential equation approximation of a pealing algorithm), each of them being more or less likely to apply to extensions of the original problem. Therefore, the thresholds for various generalization of the basic scheme ($d = 2, h = k = s = 1$) have been computed step by step: the extension to $d \geq 2$ random choices was done in [35, 42, 46]; allowing each bucket to store at most k items was done in [36, 27, 41]; the case where each item needs to be replicated h times is the most recent, with [53] solving a restricted range of parameters and [70] the general case. The extension where each (item,bucket) pair can be used a maximum of s times was not covered previously. In all the cases, the expression of the load threshold τ^* involves a solution of a fixed-point equation which can be traced back to the cavity method and fixed-point messages of BP.

An alternative approach to the cuckoo hashing problem is to establish a link with a (capacitated) matching problem, as remarked in [46, 70] and also in [35] with a detour by XORSAT. Indeed, a partial assignement of items to buckets can be mapped to a capacitated matching of the hashing graph G, and conversely any

capacitated matching of G corresponds to a partial assignement of items to buckets, because none of the constraints imposed by the hashing scheme can be violated due to the capacity constraints imposed on the matching. However, for a capacitated matching $\mathbf{x} \in \mathfrak{M}_G$ to map to a *valid* hashtable, the corresponding assignment needs to keep h copies of each item, i.e., $|\mathbf{x}_{\partial l}| = h$ for every item $l \in L$. Therefore, it is possible to build a valid hashtable whenever there exists a capacitated matching $\mathbf{x} \in \mathfrak{M}_G$ of size $|\mathbf{x}| = h|L|$. At this point, we can leverage results from the study of capacitated matchings for the random graphs considered here. This allows us to compute the limit $\lim_{m,n\to\infty} \frac{1}{n} M(G_{n,m}) = \mathcal{M}(\Phi^L, \Phi^R_\tau)$ with $n \sim \tau m$ and where Φ^L and Φ^R_τ are the joint laws for the vertex-degrees, vertex-constraints and adjacent edge-constraints on each part (here, the degrees are fixed equal to d on L_n and randomly drawn from $\mathrm{Poi}(d\tau)$ on R_n, and the capacity constraints are fixed equal to h on L_n, to k on R_n and to s on E_n). The threshold τ^* is then computed (Theorem 3.1) as

$$\tau^* = \sup\left\{\tau :\ \mathcal{M}(\Phi^L, \Phi^R_\tau) = h\right\} = \inf\left\{\tau :\ \mathcal{M}(\Phi^L, \Phi^R_\tau) < h\right\}.$$

Note that an additional step is needed to obtain the result above (as in [70]), as $\lim_{n\to\infty} \frac{1}{n} M(G_{n,m}) = h$ does not prevent failure on a vanishing fraction of items. In fact, this property that only a vanishing fraction of items cannot be stored in the hashtable defines a second threshold $\widetilde{\tau}^* \geq \tau^*$. Fortunately, one can show that the two thresholds are equal for the random graph models considered here.

We also explore another issue related to cuckoo hashing, which concerns schemes with limited randomness. Indeed, a major drawback of the fully random (multiple-choice) hashing scheme described previously is the amount of randomness involved: the standard approach requires n independent, uniform choices of sets of d buckets among m. Phrased differently, this means roughly $d \log m$ independent random bits per item. Generating unbiased, perfectly independent random bits does not come for free [122, 109]. Therefore, a lot of effort has been put into reducing the amount of randomness needed (see [82] and references therein for an account of some of the directions investigated so far). A candidate alternative is **double hashing** [56, 72], which seems to have similar performance to the fully random scheme while requiring only roughly $2 \log m$ independent random bits per item: assume m is a prime number and label the m buckets with the integers from 1 to m; for each item, independently draw two random numbers $f \in \{1, \ldots, m\}$ and $g \in \{1, \ldots, \frac{m-1}{2}\}$; the pre-defined set of buckets associated with a couple (f, g) are the buckets labeled $f + ig \pmod m$, for $i \in \{0, \ldots d-1\}$. Although the reduced randomness of the double hashing scheme is the very reason why this method could be preferred over fully random hashing, it also makes the theoretical analyzis of its performance more difficult. Mitzenmacher and Thaler [82, 84] managed to prove that the performance of double hashing is essentially the same as that of fully random hashing in the balls and bins setup of Azar, Broder, Karlin and Upfal [10] already mentioned in Section 1.2.1: they show that the maximum load is $\frac{\log\log n}{\log d} + O(d)$ w.h.p. and only conjecture that the double hashing threshold is equal to the (fully random) cuckoo hashing threshold τ^*. It turns out that the cavity method is very suitable to study such an extension of the fully random cuckoo hashing scheme. Indeed, only two steps of the proof need to be done again for the double hashing random graph model: one needs to

show that such random graphs converge a.s. locally weakly to the same two-step unimodular Galton-Watson distribution $\mathcal{GW}(\Phi^L, \Phi^R_\tau)$ as the original cuckoo hashing scheme, and then one would need to perform the additional step we mentioned to get from the threshold $\widetilde{\tau}^*$ to the actual threshold τ^*. The local weak convergence of the double hashing graphs can be proved by studying a two-step Breadth-First Search Exploration (BFS) of the hashing graph, which is a variation of the single-step BFS used to prove local weak convergence of the classical (unipartite) random graphs (see e.g., [20]). We stop after showing that the second threshold $\widetilde{\tau}^*$ of double hashing is equal to the load threshold τ^* of fully random hashing, as we consider it is sufficient to demonstrate the potential of the cavity method for this problem.

1.4.3 Load balancing and resource placement in distributed CDNs

The content replication policy within servers of a CDN strongly influences the performance of the system, therefore the problem of optimizing this replication has been considered under many viewpoints, with different modeling assumptions and performance metrics leading naturally to different content replication proposals. However, there is one replication strategy which arises frequently under many different models: it is **proportional replication**, which keeps for each content a number of copies proportional to its popularity. Using large deviation inequalities, [101] investigated this replication under the edge-CDN model with an optimal matching policy, and proved that it is efficient for a large system and large storage.

In order to characterize more precisely the performance of the proportional replication, we can use the connection with capacitated matchings. Indeed, if we allow requests to be re-allocated at any time between servers or even with the data center, then the performance of the system under optimal matching is given by the expected size of a maximum capacitated matching of the graph G (computed in Theorem 2.1), with the capacities mentioned before. This result can in fact be obtained under a richer model than the one explained above: the addition of arbitrary integer edge-capacities allows to deal with the case where the servers have non-unitary service capacities, and the contents may have different sizes as long as they are segmented into constant size coded segments. However, this more complex model is not very easy to manipulate, so we prefer to stick to the basic model to analyse the performance asymptotics. For the model where the contents are grouped into popularity classes, in the limit of large storage size we can discriminate between the multiple fixed-points of BP at 0 temperature depending on the value of the load ρ. The asymptotic behavior of the correct fixed-point of BP then yields a large deviation principle for the inefficiency ι of the system (Proposition 4.2):

$$\frac{\log(\iota)}{d} \overset{d\to\infty}{\to} \begin{cases} -\inf_k \frac{\pi_k}{\tau\alpha_k} & \text{if } \rho < 1, \\ \log\left(\sum_{k=1}^K \pi_k e^{-\lambda_k}\right) & \text{if } \rho > 1, \\ \min\left(-\inf_k \frac{\pi_k}{\tau\alpha_k}, \log\left(\sum_{i=k}^K \pi_k e^{-\lambda_k}\right)\right) & \text{if } \rho = 1, \end{cases}$$

where we recall that π is the distribution servers use to choose which class of content to cache, α_k is the fraction of contents in class k, λ_k is the request arrival rate for

class k, and $\tau \sim n/m$. It turns out that inefficiency decays exponentially fast to 0 as storage increases for any reasonable replication policy (therefore the asymptotic efficiency property shown before for proportional replication is actually a quite common property) and the decay exponents have simple expressions pointing out the very local nature of the events that typically cause inefficiency. Furthermore, from the expressions obtained, it is easy to compute the asymptotics of the optimal replication of contents for the various load regimes. Surprisingly, proportional replication is never asymptotically optimal: it favors too much the popular contents in the underloaded regime, and not enough in the overloaded regime.

The performance under optimal matching being better understood, we turn to the study of the random greedy matching algorithm. The first question there is whether the inefficiency of the system will also decay exponentially fast to 0 under greedy matching. Secondly, we seek understanding of how to efficiently replicate contents in this setup. To give an answer to the first question we consider a static setting where all the contents have the same popularity and the requests for the contents arrive in random order. Then, it is a classical approach to approximate by differential equations [33] the evolution of this large system under greedy matching. Unfortunately, we cannot solve these differential equations. However, they are sufficient to obtain the asymptotic behavior of the inefficiency for large storage. The inefficiency under greedy matching still decays exponentially fast to 0 (although slower than for optimal matching) for non-critical loads, but it undergoes a phase transition at critical load $\rho = 1$:

$$\begin{array}{ll} \iota \leq e^{-\frac{d}{\tau}|\rho-1|(1+o(1))} & \text{if } \rho \neq 1, \\ \iota = \frac{\tau \log 2}{d} + o(1/d) & \text{if } \rho = 1. \end{array}$$

This hints that one may need to go beyond greedy matching for certain applications, where for example regions of the system may naturally be maintained in a critical regime due to load-balancing happening at higher levels. Nevertheless, as long as the system stays in the underloaded regime $\rho < 1$, it makes sense to study the performance of replication strategies under greedy matching. In order to understand the influence of the replication of a particular content c, we approximate the evolution of the number Z_c of available replicas of c by a simple Markov chain, based on asymptotic behaviors and a mean-field heuristic. This allows us to obtain an expression for the stationary distribution of Z_c as well as for the stationary loss rate γ_c (Equation (4.21)), with the following large deviation principle:

$$\gamma_c \approx e^{-D_c|\ln \rho|(1+o(1))},$$

where we recall that D_c is the number of replicas for content c. Note that this expression again confirms the exponential decay of the inefficiency as storage increases and the phase transition at critical load, although the expressions for the decay exponents cannot be compared as the models are slightly different ("batch arrival" against stationary distribution). The accuracy of these approximations is confirmed by simulations. A simple optimization problem then gives the asymptotic expression for the optimal replication of contents (which is uniform with corrections that are logarithmic in the storage size) and the general principle that one should aim to equalize the loss rates of the contents. Guided by this principle and inspired by

replication strategies based on eviction rules from cache networks, we design adaptive replication schemes reacting to losses in the system. Finally, we can leverage the precise expression obtained for the loss rate to propose a *virtual loss* mechanism which anticipate losses in order to react faster, and still manage to equalize the loss rates. The static optimized replication as well as the dynamic algorithms are compared in simulations to the proportional replication policy and show significant improvements. Generally, the principle of equalizing the loss rates is supported by the simulations, and the speed enhancement provided by the virtual loss method is quite significant.

Chapter 2

Maximum capacitated matching via the cavity method

Towards answering questions related to load-balancing in random environments, in terms of average performance as well as guaranteed maximum load, we study capacitated matchings in random graphs via the cavity method. Our main result is a law of large numbers characterizing the asymptotic maximum size capacitated matching in the limit of large bipartite random graphs, when the graphs admit a *local weak limit* that is a tree. An analysis of belief propagation algorithms (BP) with multivariate belief vectors underlies the proof. In particular, we show convergence of the corresponding BP by exploiting monotonicity of the belief vectors with respect to the so-called *upshifted likelihood ratio* stochastic order, providing a new set of structural conditions which ensure convergence of BP.

2.1 Introduction

Belief Propagation (BP) is a popular message-passing algorithm for determining approximate marginal distributions in Bayesian networks [91] and statistical physics [76] or for decoding LDPC codes [93]. The popularity of BP stems from its successful application to very diverse contexts where it has been observed to converge quickly to meaningful limits [117, 74]. In contrast, relatively few theoretical results are available to prove rigorously its convergence and uniqueness of its fixed points when the underlying graph is not a tree [12].

In conjunction with the local weak convergence [5], BP has also been used as an analytical tool to study combinatorial optimization problems on random graphs: through a study of its fixed points, one can determine so-called Recursive Distributional Equations (RDE) associated with specific combinatorial problems. In turn, these RDEs determine the asymptotic behaviour of solutions to the associated combinatorial problems in the limit of large instances. Representative results in this vein concern matchings [21], spanning subgraphs with degree constraints [96] and orientability of random hypergraphs [70].

All these problems can be encoded with binary values on the edges of the underlying graph and these contexts involve BP with scalar messages. A key step in these results consists in showing monotonicity of the BP message-passing routine with respect to the input messages. As an auxiliary result, the analyses of [96] and [70] provide structural monotonicity properties under which BP is guaranteed to converge (when messages are scalar).

The present work is in line with [96, 70] and contributes to a rigorous formalization of the cavity method, originating from statistical physics [79, 67], and applied here to a generalized matching problem [71]. The initial motivation is the analysis of generalized matching problems in bipartite graphs with both edge and node capacities. This generic problem has several applications. In particular, it accurately models the service capacity of distributed content delivery networks under various content encoding scenarios, by letting nodes of the bipartite graph represent either contents or servers. It also models problem instances of cuckoo hashing, where in that context nodes represent either objects or keys to be matched.

Previous studies of these two problems [70] essentially required unit edge capacities, which in turn ensured that the underlying BP involved only scalar messages. It is however necessary to go beyond such unit edge capacities to accurately model general server capacities and various content coding schemes in the distributed content delivery network case (see Chapter 4). The extension to general edge capacities is also interesting in the context of cuckoo hashing when keys can represent sets of addresses to be matched to objects (see Chapter 3).

Our main contribution is Theorem 2.1, a law of large numbers characterizing the asymptotic size of maximum size generalized matchings in random bipartite graphs in terms of RDEs. It is stated in Section 2.2.

Besides obtaining these new laws of large numbers, our results also have algorithmic implications. Indeed to prove Theorem 2.1, in Section 2.3 we state Proposition 2.2, giving simple continuity and monotonicity conditions on the message-passing routine of BP which guarantee its convergence to a unique fixed-point. This

result is shown to apply in the present context for the so-called upshifted likelihood ratio stochastic order. Beyond its application to the present matching problem, this structural result might hold under other contexts, and with stochastic orders possibly distinct from the upshifted likelihood ratio order to establish convergence of BP in the case of multivariate messages.

The overall proof strategy is exposed in Section 2.3. The local properties of the BP update and estimate routines that will be useful are explained in Section 2.4. We then exploit these properties to demonstrate convergence of BP in finite graphs in Section 2.5 and show that the BP estimates are asymptotically exact for asymptotically tree-like graphs in Section 2.6, which proves Theorem 2.1. Finally, in Section 2.7 we present exactness results for the case of finite bipartite graphs.

2.2 Asymptotic size of maximum capacitated matchings

Let $G = (V, E)$ be a finite graph, with additionally integer constraints b_v attached to vertices $v \in V$ and integer constraints c_e attached to edges $e \in E$. Recall that a vector $\mathbf{x} = (x_e)_{e\in E} \in \mathbb{N}^E$ is called a *capacitated matching* of G if

$$\forall e \in E,\ 0 \leq x_e \leq c_e \text{ and } \forall v \in V,\ \sum_{e\in\partial v} x_e \leq b_v,$$

and we denote by $M(G)$ the maximum size of a capacitated matching of G. Our aim is to characterize the behaviour of $M(G)/|V|$ for large graphs G in the form of a law of large numbers as $|V|$ goes to infinity.

We focus mainly on sequences of graphs $(G_n)_{n\in\mathbb{N}}$ which converge locally weakly towards a Galton-Watson distribution. In particular, for the applications considered in this thesis, the graphs are bipartite graphs $G = (L \cup R, E)$ and their local weak limit is a bipartite GW distribution $\mathcal{GW}$ with joint laws Φ^L and Φ^R for the degree distributions, the vertex-constraints and the adjacent edge-constraints $(D_v, B_v, C_{\partial v})$ of each vertex v depending on whether it is in L or R. To sample a tree from $\mathcal{GW}$, assuming the root is in L (the other case being totally symmetrical), we first draw a sample from Φ^L for the root o. Then, the parameters of its descendants at even and odd generations are drawn from size-biased offspring distributions $\widetilde{\Phi}^L$ and $\widetilde{\Phi}^R$ respectively, conditionally on the capacity of the edge linking them to their parent. The size-biased offspring distributions $\widetilde{\Phi}^L$ and $\widetilde{\Phi}^R$ are defined by

$$\widetilde{\Phi}^L_{k,b,\{c_1,\dots,c_k\}|c} = \frac{\Phi^L_{k+1,b,\{c,c_1,\dots,c_k\}}(1+\sum_{i=1}^k \mathbf{1}(c_i = c))}{\sum_{d',b',\{c_1,\dots,c'_{d'-1}\}} \Phi^L_{d',b',\{c,c'_1,\dots,c'_{d-1}\}}(1+\sum_{i=1}^{d'-1}\mathbf{1}(c'_i = c))},$$

and similarly for R. We denote the offspring, vertex-constraints and edge-constraints sampled from the sized biased distributions, say from $\widetilde{\Phi}^L$, as $\widetilde{D}^L$, $\widetilde{B}^L$ and $\widetilde{C}^L$. Note that the distributions Φ^L and Φ^R have to satisfy a *consistency constraint*, which imposes that the probability of edges with a given capacity c is the same seen from

vertices in L and in R:

$$\frac{1}{\mathbb{E}[D^L]}\mathbb{E}\sum_{i=1}^{D^L}\mathbf{1}(C_i^L=c)=\frac{1}{\mathbb{E}[D^R]}\mathbb{E}\sum_{i=1}^{D^R}\mathbf{1}(C_i^R=c).$$

Before stating the main theorem of this chapter, we define the following notations: we let $[z]_x^y=\max\{x,\min\{y,z\}\}$ and $(z)^+=\max\{z,0\}$.

Theorem 2.1 (Maximum capacitated matching for Galton-Watson limits)**.** *Let $(G_n)_{n\in\mathbb{N}}$ be a sequence of bipartite graphs, with $G_n=(L_n\cup R_n,E_n)$, converging locally weakly towards a two-step unimodular Galton-Watson distribution with laws Φ^L and Φ^R for the degree distributions, the vertex-constraints and the adjacent edge-constraints $(D_v,B_v,C_{\partial v})$ for vertex v. Provided $\mathbb{E}[B^L]$ and $\mathbb{E}[B^R]$ are finite, the limit $\mathcal{M}(\Phi^L,\Phi^R)=\lim_{n\to\infty}M(G_n)/|L_n|$ exists and equals*

$$\begin{aligned}\mathcal{M}(\Phi^L,\Phi^R)=\inf\Bigg\{&\mathbb{E}\left[\min\left\{B^L,\textstyle\sum_{i=1}^{D^L}X_i(C_i^L)\right\}\right]\\&+\tfrac{\mathbb{E}[D^L]}{\mathbb{E}[D^R]}\mathbb{E}\left[\left(B^R-\textstyle\sum_{i=1}^{D^R}\left[B^R-\sum_{j\neq i}Y_j(C_j^R)\right]_0^{C_i^R}\right)^+\mathbf{1}\left(B^R<\textstyle\sum_{i=1}^{D^R}C_i^R\right)\right]\Bigg\}\end{aligned}$$

where for all i, $(X_i(c),Y_i(c))_{c\in\mathbb{N}}$ is an independent copy of $(X(c),Y(c))_{c\in\mathbb{N}}$, and the infimum is taken over distributions for $(X(c),Y(c))_{c\in\mathbb{N}}$ satisfying the RDE

$$\begin{aligned}Y(c)&=\left\{\left[\widetilde{B}^L-\sum_{i=1}^{\widetilde{D}^L}X_i(\widetilde{C}_i^L)\right]_0^c\middle|C_0^L=c\right\};\\X(c)&=\left\{\left[\widetilde{B}^R-\sum_{i=1}^{\widetilde{D}^R}Y_i(\widetilde{C}_i^R)\right]_0^c\middle|C_0^R=c\right\}.\end{aligned}$$

Remark 2.1. *A similar result holds when the graphs are not bipartite; the limiting tree is then simply a Galton-Watson tree described by a joint distribution Φ. We set $\Phi^L=\Phi^R=\Phi$, and the formula in Theorem 2.1 then yields $\lim_{n\to\infty}\frac{2M(G_n)}{|V_n|}=\mathcal{M}(\Phi,\Phi)$.*

2.3 Main proof elements

We start with a high level description of the path followed towards Theorem 2.1. The proof strategy uses a detour, by introducing a finite activity parameter $\lambda>0$, which plays the role of $e^{-\beta}$ in Section 1.3.2. For a given finite graph G, a Gibbs distribution μ_G^λ is defined on edge occupancy parameters $\mathbf{x}$ (Section 2.3.1) such that an average under μ_G^λ approaches the quantity of interest $M(G)/|V|$ as λ tends to infinity. Instead of considering directly the limit of this parameter over a series of converging graphs G_n, we take an indirect route, changing the order of limits over λ and n.

We thus first determine for fixed λ the asymptotics in n of averages under $\mu^\lambda_{G_n}$. This is where BP comes into play. We characterize the behaviour of BP associated with μ^λ_G on finite G (Section 2.3.2), establishing its convergence to a unique fixed point thanks to structural properties of monotonicity for the upshifted likelihood ratio order, and of log-concavity of messages (Sections 2.4 and 2.5). This allows us to show that limits over n of averages under $\mu^\lambda_{G_n}$ are characterized by fixed point relations à la BP. Taking limits over $\lambda \to \infty$, one derives from these fixed points the RDEs appearing in the statement of Theorem 2.1. It then remains to justify interchange of limits in λ and n. These last three steps are handled similarly to [70].

Before we proceed, we introduce some necessary notation. Letters or symbols in bold such as $\mathbf{x}$ denote collections of objects $(x_i)_{i\in I}$ for some set I. For a subset S of I, $\mathbf{x}_S$ is the sub-collection $(x_i)_{i\in S}$ and $|\mathbf{x}_S| := \sum_{i\in S} x_i$ is the L_1-norm of $\mathbf{x}_S$. Inequalities between collections of items should be understood componentwise, thus $\mathbf{x} \le \mathbf{c}$ means $x_i \le c_i$ for all $i \in I$. For distributions m_i, we let $\mathbf{m}_S(\mathbf{x}) := \prod_{i\in S} m_i(x_i)$. When summing such terms as in $\sum_{\mathbf{x}\in\mathbb{N}^S:|\mathbf{x}|\le b,\, \mathbf{x}\le\mathbf{c}} \mathbf{m}_S(\mathbf{x})$, we shall omit the constraint $\mathbf{x} \in \mathbb{N}^S$. Similarly, we let $*_S\mathbf{m} = *_{i\in S} m_i$, where $*$ is the convolution of two vectors (will be defined in Section 2.4).

Finally, we will work most of the time as if $b_v \le |\mathbf{c}_{\partial v}|$ for all $v \in V$ and $c_e \le b_v$ for all $e \in \partial v$. If one of these assumptions were not verified, we would have to replace b_v by $\min\{b_v, |\mathbf{c}_{\partial v}|\}$ and c_{uv} by $\min\{c_{uv}, b_u, b_v\}$ everywhere, which would quickly become cumbersome. However, note that these considerations have been taken care of in the expression of Theorem 2.1 with the otherwise useless term $\mathbf{1}\left(B^R < \sum_{i=1}^{D^R} C_i^R\right)$.

2.3.1 Gibbs measure

Let $G = (V, E)$ be a finite graph, with collections of vertex- and edge-constraints $\mathbf{b} = (b_v)_{v\in V}$ and $\mathbf{c} = (c_e)_{e\in E}$. The Gibbs measure at activity parameter $\lambda \in \mathbb{R}_+$ on the set of all vectors in $\mathbb{N}^E$ is then defined, for $\mathbf{x} \in \mathbb{N}^E$, as

$$\begin{aligned}\mu^\lambda_G(\mathbf{x}) &= \frac{\lambda^{|\mathbf{x}|}}{Z_G(\lambda)}\mathbf{1}(\mathbf{x} \in \mathfrak{M}(G)) \\ &= \frac{\lambda^{|\mathbf{x}|}}{Z_G(\lambda)} \prod_{v\in V} \mathbf{1}\Big(\sum_{e\in\partial v} x_e \le b_v\Big) \prod_{e\in E} \mathbf{1}(x_e \le c_e),\end{aligned}$$

where $Z_G(\lambda)$ is a normalization factor called partition function.

When $\lambda \to \infty$, μ^λ_G tends to the uniform probability measure on the set of all allocations of G of maximum size. Thus, $\lim_{\lambda\to\infty} \mu_G(|\mathbf{X}|) = M(G)$, where $\mu^\lambda_G(|\mathbf{X}|)$ is the expected size of a random allocation $\mathbf{X}$ drawn according to μ^λ_G. Hence, we can compute $M(G)/|V|$ as follows:

$$\begin{aligned}\frac{M(G)}{|V|} &= \lim_{\lambda\to\infty} \mu^\lambda_G\left(\sum_{v\in V} \frac{1}{|V|}\frac{\sum_{e\in\partial v} X_e}{2}\right) \\ &= \frac{1}{2}\lim_{\lambda\to\infty} \mathbb{E}\left[\mu^\lambda_G\left(\sum_{e\in\partial o} X_e\right)\right], \qquad (2.1)\end{aligned}$$

where o is a root-vertex chosen uniformly at random among all vertices in V, and the first expectation is with respect to the choice of o.

2.3.2 Associated BP message passing

We introduce the set $\overrightarrow{E}$ of directed edges of G comprising two directed edges $\overrightarrow{uv}$ and $\overrightarrow{vu}$ for each undirected edge $uv \in E$. We also define $\overrightarrow{\partial v}$ as the set of edges directed towards vertex $v \in V$, $\overrightarrow{\partial v}$ as the set of edges directed outwards from v, and $\overrightarrow{\partial e} := (\overrightarrow{wv})_{w \in \partial v \setminus u}$ if $\overrightarrow{e}$ is the directed edge $\overrightarrow{vu}$.

An allocation puts an integer weight on each edge of the graph. Accordingly the messages to be sent along each edge are distributions over the integers. We let $\mathcal{P}$ be the set of all probability distributions on integers with bounded support, i.e.

$$\mathcal{P} \;=\; \Big\{ p \in [0,1]^{\mathbb{N}}; \sum_{i \in \mathbb{N}} p(i) = 1 \text{ and } \exists k \in \mathbb{N} \text{ such that } p(i) = 0, \forall i > k \Big\},$$

and $\widetilde{\mathcal{P}}$ the set of distributions in $\mathcal{P}$ whose support is an interval containing 0.

A message on directed edge $\overrightarrow{e}$ with capacity c_e is a distribution in $\mathcal{P}$ with support in $\{0, \dots, c_e\}$. The message to send on edge $\overrightarrow{e}$ outgoing from vertex v is computed from the messages incoming to v on the other edges via

$$\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}](x) = \frac{\lambda^x \mathbf{1}(x \le c_{vu}) \sum_{|\mathbf{y}| \le b_v - x} \mathbf{m}_{\partial \overrightarrow{vu}}(\mathbf{y})}{\sum_{t \le c_{vu}} \lambda^t \sum_{|\mathbf{y}| \le b_v - t} \mathbf{m}_{\partial \overrightarrow{vu}}(\mathbf{y})},$$

where we introduced the operator $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}} : \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}} \to \widetilde{\mathcal{P}}$. For notational convenience, we write $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}]$ instead of $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}_{\overrightarrow{\partial e}}]$. We also write $\mathcal{R}_{\overrightarrow{e}}$ for $\mathcal{R}^{(1)}_{\overrightarrow{e}}$. The two operators are linked via the relationship

$$\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}](x) = \frac{\lambda^x \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}](x)}{\sum_{t \ge 0} \lambda^t \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}](t)}.$$

We also define an operator $\mathcal{D}_v : \widetilde{\mathcal{P}}^{\overrightarrow{\partial v}} \to \mathbb{R}^+$ meant to approximate the average occupancy at a vertex v under μ^λ_G from the messages incoming to v:

$$\mathcal{D}_v[\mathbf{m}] = \frac{\sum_{|\mathbf{x}| \le b_v} |\mathbf{x}| \mathbf{m}_{\overrightarrow{\partial v}}(\mathbf{x})}{\sum_{|\mathbf{x}| \le b_v} \mathbf{m}_{\overrightarrow{\partial v}}(\mathbf{x})}.$$

Finally we denote by $\mathcal{R}^{(\lambda)}_G$ the operator that performs the action of all the $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}$ for all $\overrightarrow{e}$ simultaneously, i.e., $\mathcal{R}^{(\lambda)}_G[\mathbf{m}] = \Big(\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}]\Big)_{\overrightarrow{e} \in \overrightarrow{E}}$ (the same type of notation will be used for other operators). It is well known that belief propagation converges and is exact on finite trees [76]:

Proposition 2.1. *If the graph G is a finite tree, the fixed point equation $\mathbf{m} = \mathcal{R}^{(\lambda)}_G[\mathbf{m}]$ admits a unique solution $\mathbf{m}^{(\lambda)} \in \widetilde{\mathcal{P}}^{\overrightarrow{E}}$, and it satisfies for every vertex v:*

$$\mu^\lambda_G \left(\sum_{e \in \partial v} X_e \right) = \mathcal{D}_v[\mathbf{m}^{(\lambda)}].$$

However, to be able to take the limit as the activity parameter λ goes to infinity as well as to deal with cases when G is not a tree anymore, we need to study further the operators $\mathcal{R}_{\overrightarrow{e}}$ and $\mathcal{D}_v$, which we term the *local* operators.

2.4 Structural properties of local operators

In this section, we obtain monotonicity properties for the operators $\mathcal{R}_{\overrightarrow{e}}$ and $\mathcal{D}_v$ that will ensure convergence of belief propagation, but for that we need to decide which order to use for distributions in $\mathcal{P}$. One would expect the operator $\mathcal{R}_{\overrightarrow{e}}$ to be non-increasing in some sense (as it is the case in [96, 70], where the inputs to the operators are real values instead of vectors), however this does not hold if we use the strong stochastic order on distributions. We will show that another order, namely the upshifted likelihood-ratio order, is more adapted. We focus on the one-hop neighborhood of a vertex v of a graph G, i.e., on vertex v and its set ∂v of incident edges. We thus only consider the directed edges in $\overrightarrow{\partial v} \cup \overleftarrow{\partial v}$. We let b_v be the vertex-constraint at v and $\mathbf{c} = (c_e)_{e\in\partial v}$ be the collection of the edge-constraints on the edges in ∂v.

As $\lambda \to \infty$, we will have to deal with limiting messages that may not have 0 in their support. We thus define $\alpha_{\overrightarrow{e}}$ as the infimum of the support of $m_{\overrightarrow{e}} \in \mathcal{P}$, i.e., $\alpha_{\overrightarrow{e}} = \min\{x \in \mathbb{N} :\ m_{\overrightarrow{e}}(x) > 0\}$. When there may be confusion, we will write $\alpha(m_{\overrightarrow{e}})$ for the infimum of the support of $m_{\overrightarrow{e}}$. We also extend the definition of the local operators given previously so that they allow inputs with arbitrary supports in $\mathbb{N}$: for an edge $\overrightarrow{e}$ outgoing from vertex v, we define $\mathcal{R}_{\overrightarrow{e}} : \mathcal{P}^{\overrightarrow{\partial e}} \to \widetilde{\mathcal{P}}$, $\mathcal{D}_v : \mathcal{P}^{\overrightarrow{\partial v}} \to \mathbb{R}^+$ and $\mathcal{S}_{\overrightarrow{e}} : \mathbb{N}^{\overrightarrow{\partial e}} \to \mathbb{N}$ as

$$\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}](x) = \begin{cases} \frac{\mathbf{1}(x\le c_e)\sum_{|\mathbf{y}|\le b_v - x}\mathbf{m}_{\overrightarrow{\partial e}}(\mathbf{y})}{\sum_{t\le c_e}\sum_{|\mathbf{y}|\le b_v - t}\mathbf{m}_{\overrightarrow{\partial e}}(\mathbf{y})} & \text{if } |\boldsymbol{\alpha}_{\overrightarrow{\partial e}}| \le b_v \\ \chi_0(x) := \mathbf{1}(x=0) & \text{otherwise} \end{cases} \tag{2.2}$$

$$\mathcal{D}_v[\mathbf{m}] = \begin{cases} \frac{\sum_{|\mathbf{x}|\le b_v}|\mathbf{x}|\mathbf{m}_{\overrightarrow{\partial v}}(\mathbf{x})}{\sum_{|\mathbf{x}|\le b_v}\mathbf{m}_{\overrightarrow{\partial v}}(\mathbf{x})} & \text{if } |\boldsymbol{\alpha}_{\overrightarrow{\partial v}}| \le b_v \\ b_v & \text{otherwise} \end{cases} \tag{2.3}$$

$$\mathcal{S}_{\overrightarrow{e}}(\mathbf{x}) = \left[b_v - |\mathbf{x}_{\overrightarrow{\partial e}}|\right]_0^{c_e}. \tag{2.4}$$

Note that the support of $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}]$ is $\{0,\dots,\mathcal{S}_{\overrightarrow{e}}(\boldsymbol{\alpha})\}$.

Among the many stochastic orders for comparing distributions (see e.g., [87]), the one best adapted to the structure of operators $\mathcal{R}_{\overrightarrow{e}}$ and $\mathcal{D}_v$ is the so-called *upshifted likelihood-ratio* stochastic order (abbreviated lr $\uparrow$). For two vectors m and m' in $\mathbb{R}^{\mathbb{N}}$, we say that m is smaller than m' (for the lr $\uparrow$ stochastic order) and we write $m \le_{\mathrm{lr}\uparrow} m'$ if

$$m(i+k+l)m'(i) \le m(i+l)m'(i+k), \forall i,k,l \in \mathbb{N}.$$

If $m \le_{\mathrm{lr}\uparrow} m'$, then $\alpha(m) \le \alpha(m')$ and a similar statement holds for the supremum of the support of the two vectors. We will almost always use the lr $\uparrow$-order when comparing distributions.

We shall also need the following definition. A distribution $(p_j)_{j\ge 0}$ is *log-concave* if its support is an interval and $p_i p_{i+2} \le p_{i+1}^2$, for all $i \in \mathbb{N}$. This property has strong

ties with the lr ↑-order. In particular one can note that p is log-concave if and only if $p \leq_{\mathrm{lr}\uparrow} p$. We let $\mathcal{P}_{\mathrm{lc}} \subset \mathcal{P}$ be the set of all log-concave distributions over integers with finite support, and $\widetilde{\mathcal{P}}_{\mathrm{lc}} = \widetilde{\mathcal{P}} \cap \mathcal{P}_{\mathrm{lc}}$:

$$\begin{aligned}\mathcal{P}_{\mathrm{lc}} \;=\; \Big\{ & p \in [0,1]^{\mathbb{N}}; \sum_{i\in\mathbb{N}} p(i) = 1,\ p \text{ is log-concave,} \\ & \text{and } \exists k \in \mathbb{N} \text{ such that } p(i) = 0, \forall i > k \Big\}.\end{aligned}$$

The key result of this Section is then the following:

Proposition 2.2 (Monotonicity of the local operators for the lr ↑-order)**.** *The operator $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}$ is non-increasing; furthermore, if the inputs of $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}$ are log-concave, then the output is also log-concave. The operator $\mathcal{D}_v$ is non-decreasing, and strictly increasing if all its inputs are log-concave with* 0 *in their support.*

The proof will rely on the following lemma from [98] establishing stability of lr ↑-order w.r.t. convolution $*$, where $m * m'(x) = \sum_y m(y)m'(x-y)$:

Lemma 2.1. *For a set $\overrightarrow{S}$ of directed edges, if $\mathbf{m}^1_{\overrightarrow{S}} \leq_{lr\uparrow} \mathbf{m}^2_{\overrightarrow{S}}$ in $\mathcal{P}^{\overrightarrow{S}}$, then $*_{\overrightarrow{S}}\mathbf{m}^1 \leq_{lr\uparrow} *_{\overrightarrow{S}}\mathbf{m}^2$.*

We shall also need the following notions:

- the *reweighting* of a vector m by a vector p is defined by $m\centerdot p(x) := \frac{m(x)p(x)}{\sum_{y\in\mathbb{N}} m(y)p(y)}$ for $x \in \mathbb{N}$, for p and m with non-disjoint supports and $|p| < \infty$ or $|m| < \infty$. If p or m is in $\mathcal{P}$, then $m \centerdot p \in \mathcal{P}$. Note that $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{m}] = \lambda^{\mathbb{N}} \centerdot \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}]$, where $\lambda^{\mathbb{N}} = (\lambda^x)_{x\in\mathbb{N}}$.
- the *shifted reversal* of a vector p is defined by $p^\diamond(x) = p(b-x)\mathbf{1}(x \leq b)$ for $b, x \in \mathbb{N}$; if $p \in \mathcal{P}$ and its support is included in $[0, b]$, then $p^\diamond \in \mathcal{P}$ as well. We did not mention what b is in the previous definition, because we are going to use different values of b (and hence in fact different operators) when applying the shifted reversal operator to different vectors. The rule will always be that the value of b that must be used when applying $\diamond$ to a vector $p_{\overrightarrow{e}}$ associated to the directed edge $\overrightarrow{e}$ is the vertex-constraint b_v of the tail v of $\overrightarrow{e}$.

It is straightforward to check that

Lemma 2.2. *Reweighting preserves the lr ↑-order; shifted reversal reverses the lr ↑-order.*

Note that by the previous lemma it suffices to prove the results of Proposition 2.2 for $\mathcal{R}_{\overrightarrow{e}}$ and they will then extend to $\mathcal{R}^{(\lambda)}_{\overrightarrow{e}}$.

Proof of Proposition 2.2. Let $\overrightarrow{e}$ be an edge outgoing from vertex v, and $\mathbf{m}^1_{\overrightarrow{\partial v}}, \mathbf{m}^2_{\overrightarrow{\partial v}} \in \mathcal{P}^{\overrightarrow{\partial v}}$ such that $\mathbf{m}^1_{\overrightarrow{\partial v}} \leq_{\mathrm{lr}\uparrow} \mathbf{m}^2_{\overrightarrow{\partial v}}$. Firstly, if $|\boldsymbol{\alpha}^2_{\overrightarrow{\partial e}}| \geq b_v$, then $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^2] = \chi_0$ and automatically $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^1] \geq_{\mathrm{lr}\uparrow} \chi_0 = \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^2]$. Then, if $|\boldsymbol{\alpha}^2_{\overrightarrow{\partial e}}| \leq b_v$, we also have $|\boldsymbol{\alpha}^1_{\overrightarrow{\partial e}}| \leq |\boldsymbol{\alpha}^2_{\overrightarrow{\partial e}}| \leq b_v$. Let $\chi_{[0,b_v]}(x) = \mathbf{1}(0 \leq x \leq b_v)$ and $\theta^i_{\overrightarrow{e}} = *_{\overrightarrow{\partial e}}\mathbf{m}^i$; we have

$\chi_{[0,b_v]} * \theta^i_{\overrightarrow{e}}(x) = \sum_{x-b_v\leq|\mathbf{y}|\leq x} \mathbf{m}^i_{\overrightarrow{\partial e}}(\mathbf{y})$. $\chi_{[0,b_v]}$ is log-concave, so $\chi_{[0,b_v]} \leq_{\mathrm{lr}\uparrow} \chi_{[0,b_v]}$ and Lemma 2.1 then implies $\chi_{[0,b_v]} * \theta^1_{\overrightarrow{e}} \leq_{\mathrm{lr}\uparrow} \chi_{[0,b_v]} * \theta^2_{\overrightarrow{e}}$. Lemma 2.2 then says $\left(\theta^1_{\overrightarrow{e}}\right)^\diamond \geq_{\mathrm{lr}\uparrow} \left(\theta^2_{\overrightarrow{e}}\right)^\diamond$. It is easy to check that

$$\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^i] = \chi_{[0,c_e]} \centerdot \left(\chi_{[0,b_v]} *_{\overrightarrow{\partial e}} \mathbf{m}^i\right)^\diamond; \tag{2.5}$$

and furthermore, as $\left(\chi_{[0,b_v]} *_{\overrightarrow{\partial e}} \mathbf{m}^i\right)^\diamond(0) > 0$, Lemma 2.2 again implies $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^1] \geq_{\mathrm{lr}\uparrow} \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^2]$.

If now $\mathbf{m}_{\overrightarrow{\partial e}} \in \mathcal{P}^{\overrightarrow{\partial e}}_{\mathrm{lc}}$, then $\mathbf{m}_{\overrightarrow{\partial e}} \leq_{\mathrm{lr}\uparrow} \mathbf{m}_{\overrightarrow{\partial e}}$ and $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}] \geq_{\mathrm{lr}\uparrow} \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}]$, which shows $\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}] \in \mathcal{P}_{\mathrm{lc}}$.

Similarly, if $|\boldsymbol{\alpha}^2_{\overrightarrow{\partial v}}| \geq b_v$ then $\mathcal{D}_v[\mathbf{m}^2] = b_v$ and automatically $\mathcal{D}_v[\mathbf{m}^1] \leq b_v = \mathcal{D}_v[\mathbf{m}^2]$. If now $|\boldsymbol{\alpha}^2_{\overrightarrow{\partial v}}| < b_v$, we also have $|\boldsymbol{\alpha}^1_{\overrightarrow{\partial v}}| < b_v$. Lemma 2.1 shows $\theta^1_v = *_{\overrightarrow{\partial v}}\mathbf{m}^1 \leq_{\mathrm{lr}\uparrow} \theta^2_v = *_{\overrightarrow{\partial v}}\mathbf{m}^2$. As $|\boldsymbol{\alpha}^i| \leq b_v$, Lemma 2.2 says $\chi_{[0,b_v]} \centerdot \theta^1_v \leq_{\mathrm{lr}\uparrow} \chi_{[0,b_v]} \centerdot \theta^2_v$. This implies that the mean of $\chi_{[0,b_v]} \centerdot \theta^1_v$ is no larger than that of $\chi_{[0,b_v]} \centerdot \theta^2_v$, which is exactly $\mathcal{D}_v[\mathbf{m}^1] \leq \mathcal{D}_v[\mathbf{m}^2]$.

Furthermore, if $\mathbf{m}^1_{\overrightarrow{\partial v}} <_{\mathrm{lr}\uparrow} \mathbf{m}^2_{\overrightarrow{\partial v}}$ in $\mathcal{P}^{\overrightarrow{\partial v}}_{\mathrm{lc}}$ and $|\boldsymbol{\alpha}^1_{\overrightarrow{\partial v}}| = |\boldsymbol{\alpha}^2_{\overrightarrow{\partial v}}| = 0$, then a direct calculation will shows that $\chi_{[0,b_v]} \centerdot \theta^1_v <_{\mathrm{lr}\uparrow} \chi_{[0,b_v]} \centerdot \theta^2_v$, which implies $\mathcal{D}_v[\mathbf{m}^1] < \mathcal{D}_v[\mathbf{m}^2]$. More precisely, fix $\overrightarrow{e} \in \overrightarrow{\partial v}$; it is sufficient to work with $m^1_{\overrightarrow{e}'} = m^2_{\overrightarrow{e}'}$ for all $\overrightarrow{e}' \neq \overrightarrow{e}$, as then the loose inequality obtained before allows us to conclude. Then, there exists a minimum $i \in \mathbb{N}$ such that $m^1_{\overrightarrow{e}}(i+1)m^2_{\overrightarrow{e}}(i) < m^1_{\overrightarrow{e}}(i)m^2_{\overrightarrow{e}}(i+1)$. It is then immediate that $\theta^1_v(i+1)\theta^2_v(i) < \theta^1_v(i)\theta^2_v(i+1)$ as the only term differing between the two sides is the one for which we have strict inequality. This implies $(\theta^1_v(x))_{x\leq b_v} <_{\mathrm{lr}\uparrow} (\theta^2_v(x))_{x\leq b_v}$, as we already obtained the loose inequality. For these vectors, reweighting by $\chi_{[0,b_v]}$ preserves the strict lr $\uparrow$-ordering, thus we have $\chi_{[0,b_v]} \centerdot \theta^1_v <_{\mathrm{lr}\uparrow} \chi_{[0,b_v]} \centerdot \theta^2_v$ as claimed. □

To pave the way for the analysis of the limit $\lambda \to \infty$, we distinguish between two collections of messages $\mathbf{m}_{\overrightarrow{\partial v}} \in \mathcal{P}^{\overrightarrow{\partial v}}$ and $\mathbf{n}_{\overrightarrow{\partial v}} \in \widetilde{\mathcal{P}}^{\overrightarrow{\partial v}}$, and we define $\beta_{\overrightarrow{e}}$ as the supremum of the support of $n_{\overrightarrow{e}} \in \widetilde{\mathcal{P}}$, i.e., $\beta_{\overrightarrow{e}} = \max\{x \in \mathbb{N} : n_{\overrightarrow{e}}(x) > 0\}$. Again, when there may be confusion, we will write $\beta(n_{\overrightarrow{e}})$ for the supremum of the support of $n_{\overrightarrow{e}}$.

We also introduce an additional operator: for an edge $\overrightarrow{e}$ outgoing from v we define $\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}} : \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}} \to \widetilde{\mathcal{P}}$ by $\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}] = \mathcal{R}^{(\lambda)}_{\overrightarrow{e}}[\lambda^{\mathbb{N}} \centerdot \mathbf{n}]$, where $\lambda^{\mathbb{N}} \centerdot \mathbf{n} = \left(\lambda^{\mathbb{N}} \centerdot n_{\overrightarrow{e}}\right)_{\overrightarrow{e}\in\overrightarrow{E}}$. As reweighting preserves the lr $\uparrow$-order, the operator $\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}$ is non-increasing. It also verifies the following useful monotonicity property with respect to λ:

Proposition 2.3 (Monotonicity in λ)**.** *For $\mathbf{n}_{\overrightarrow{\partial e}} \in \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}}$, the mapping $\lambda \mapsto \mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}]$ is non-decreasing.*

Proof. Let $\overrightarrow{e}$ be an edge outgoing from v, and $\mathbf{n}_{\overrightarrow{\partial e}} \in \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}}$. First-of-all, the support of the vectors considered is independent of $\lambda \in \mathbb{R}^+$. We will use the expression of $\mathcal{R}_e$ from equation (2.5). It is easy to check that

$$\begin{aligned}\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}] &= \lambda^{\mathbb{N}} \centerdot \chi_{[0,c_e]} \centerdot \left(\chi_{[0,b_v]} *_{\overrightarrow{\partial e}} \left(\lambda^{\mathbb{N}} \centerdot \mathbf{n}\right)\right)^\diamond \\ &= \chi_{[0,c_e]} \centerdot \left(\left(\lambda^{\mathbb{N}}\right)^\diamond *_{\overrightarrow{\partial e}} \mathbf{n}\right)^\diamond,\end{aligned}$$

which shows that $\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}]$ is non-decreasing in λ (as shifted reversal is the only operator used which reverses the lr $\uparrow$-order instead of preserving it, and it is applied twice to $\lambda^{\mathbb{N}}$). $\square$

Proposition 2.3 allows us to look at the limit of $\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}$ as $\lambda \to \infty$. We thus define yet another operator, allowing inputs with arbitrary supports in $\mathbb{N}$: for an edge $\overrightarrow{e}$ outgoing from vertex v, we define $\mathcal{Q}_{\overrightarrow{e}} : \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}} \to \mathcal{P}$ as

$$\mathcal{Q}_{\overrightarrow{e}}[\mathbf{n}](x) \;=\; \begin{cases} \frac{\mathbf{1}(x \le c_e) \sum_{|\mathbf{y}|=b_v-x} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})}{\sum_{t \le c_e} \sum_{|\mathbf{y}|=b_v-t} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})} & \text{if } |\boldsymbol{\beta}_{\overrightarrow{\partial e}}| \ge b_v - c_e \\ \chi_{c_e}(x) & \text{otherwise} \end{cases} \tag{2.6}$$

Note that the support of $\mathcal{Q}_{\overrightarrow{e}}[\mathbf{n}]$ is $\{\mathcal{S}_{\overrightarrow{e}}(\boldsymbol{\beta}), \dots, c_e\}$. We now establish the following result:

Proposition 2.4 (Continuity for log-concave inputs and limiting operators)**.** *The operators $\mathcal{R}_{\overrightarrow{e}}$ and $\mathcal{D}_v$ given by equations (2.2),(2.3) are continuous for the L_1 norm for inputs in $\widetilde{\mathcal{P}}_{lc}$. Also, $\mathcal{Q}_{\overrightarrow{e}}$ defined in equation (2.6) satisfies $\mathcal{Q}_{\overrightarrow{e}}[\mathbf{n}] = \lim \uparrow_{\lambda\to\infty} \mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}]$ for any $\mathbf{n}_{\overrightarrow{\partial e}} \in \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}}$.*

Proof. Consider a sequence $(\mathbf{m}^{(k)}_{\overrightarrow{\partial v}})_{k\in\mathbb{N}}$ in $\widetilde{\mathcal{P}}^{\overrightarrow{\partial v}}_{\mathrm{lc}}$ converging towards $\mathbf{m}_{\overrightarrow{\partial v}}$, which thus belongs to $\mathcal{P}^{\overrightarrow{\partial v}}_{\mathrm{lc}}$. Let $\overrightarrow{e}$ be an edge outgoing from v:

$$\mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^{(k)}](x) = \frac{\mathbf{1}(x \le c_e) \sum_{|\mathbf{y}|\le b_v-x} \mathbf{m}^{(k)}_{\overrightarrow{\partial e}}(\mathbf{y})}{\sum_{t\le c_e} \sum_{|\mathbf{y}|\le b_v-t} \mathbf{m}^{(k)}_{\overrightarrow{\partial e}}(\mathbf{y})}$$

If $|\boldsymbol{\alpha}_{\overrightarrow{\partial e}}| \le b_v$, then the expression above is clearly continous. If $|\boldsymbol{\alpha}_{\overrightarrow{\partial e}}| \ge b_v$, then the numerator is equivalent as $k \to \infty$ to $\sum_{|\mathbf{y}|=b_v} \mathbf{m}^{(k)}_{\overrightarrow{\partial e}}(\mathbf{y})$ and the numerator to $\sum_{|\mathbf{y}|=b_v-x} \mathbf{m}^{(k)}_{\overrightarrow{\partial e}}(\mathbf{y})$, for $x \le c_e$. Then,

$$\lim_{k\to\infty} \mathcal{R}_{\overrightarrow{e}}[\mathbf{m}^{(k)}](x) = \chi_0(x).$$

A similar reasoning applies to $\mathcal{D}_v$:

$$\mathcal{D}_v[\mathbf{m}^{(k)}] = \frac{\sum_{|\mathbf{x}|\le b_v} |\mathbf{x}| \mathbf{m}^{(k)}_{\overrightarrow{\partial v}}(\mathbf{x})}{\sum_{|\mathbf{x}|\le b_v} \mathbf{m}^{(k)}_{\overrightarrow{\partial v}}(\mathbf{x})}$$

The expression above is clearly continous when $|\boldsymbol{\alpha}_{\overrightarrow{\partial v}}| \le b_v$. If $|\boldsymbol{\alpha}_{\overrightarrow{\partial v}}| \ge b_v$, then the denominator is equivalent to $\sum_{|\mathbf{x}|=b_v} b_v \mathbf{m}^{(k)}_{\overrightarrow{\partial v}}(\mathbf{x})$ and the numerator to $\sum_{|\mathbf{x}|=b_v} \mathbf{m}^{(k)}_{\overrightarrow{\partial v}}(\mathbf{x})$. So,

$$\lim_{k\to\infty} \mathcal{D}_v[\mathbf{m}^{(k)}] = b_v.$$

We now turn to the operator $\mathcal{Q}_{\overrightarrow{e}}$. Let $\mathbf{n}_{\overrightarrow{\partial e}} \in \widetilde{\mathcal{P}}^{\overrightarrow{\partial e}}$, we have

$$\mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}](x) = \frac{\mathbf{1}(x \le c_e) \lambda^x \sum_{|\mathbf{y}|\le b_v-x_e} \lambda^{|\mathbf{y}|} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})}{\sum_{t\le c_e} \lambda^t \sum_{|\mathbf{y}|\le b_v-t} \lambda^{|\mathbf{y}|} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})}$$

Suppose that $|\boldsymbol{\beta}_{\overrightarrow{\partial e}}| \geq b_v - c_e$, then the denominator of the expression above is equivalent as $\lambda \to \infty$ to $\lambda^{b_v} \sum_{b_v - c_e \leq |\mathbf{y}| \leq b_v} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y}) > 0$ and the numerator to $\mathbf{1}(x \leq c_e)\lambda^{b_v} \sum_{|\mathbf{y}|=b_v-x_e} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y}) + \mathrm{o}(\lambda^{b_v})$. Hence,

$$\lim_{\lambda\to\infty} \mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}](x) = \frac{\mathbf{1}(x \leq c_e) \sum_{|\mathbf{y}|=b_v-x_e} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})}{\sum_{b_v-c_e\leq|\mathbf{y}|\leq b_v} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y})}.$$

Suppose now that $|\boldsymbol{\beta}_{\overrightarrow{\partial e}}| < b_v - c_e$. The denominator is equivalent to

$$\lambda^{c_e+|\boldsymbol{\beta}_{\overrightarrow{\partial e}}|} \sum_{|\mathbf{y}|=|\boldsymbol{\beta}_{\overrightarrow{\partial e}}|} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y}) > 0$$

and the numerator to

$$\mathbf{1}(x \leq c_e)\lambda^{x+|\boldsymbol{\beta}_{\overrightarrow{\partial e}}|} \sum_{|\mathbf{y}|=|\boldsymbol{\beta}_{\overrightarrow{\partial e}}|} \mathbf{n}_{\overrightarrow{\partial e}}(\mathbf{y}) + \mathrm{o}(\lambda^{c_e+|\boldsymbol{\beta}_{\overrightarrow{\partial e}}|}).$$

Thus,

$$\lim_{\lambda\to\infty} \mathcal{Q}^{(\lambda)}_{\overrightarrow{e}}[\mathbf{n}](x) = \chi_{c_e}(x).$$

□

It follows naturally that $\mathcal{Q}_{\overrightarrow{e}}$ is non-increasing. Moreover, we can extend the results of Proposition 2.2 to the extended operators, i.e., $\mathcal{R}_{\overrightarrow{e}}$ is still non-increasing and $\mathcal{D}_v$ non-decreasing.

2.5 Finite graphs

Although we will work later (in Section 2.6) in the more general framework of unimodular graphs, we begin with the more intuitive case of finite graphs, so as to better explain the mechanisms of the proof. We exploit the properties of the local operators shown in the previous section to show convergence of BP to fixed-point messages $\mathbf{m}^{(\lambda)}$ in finite graphs at finite activity parameter λ, and compute the limit as $\lambda \to \infty$ of $\mathcal{D}_v[\mathbf{m}^{(\lambda)}]$ for any vertex v. However, this limit will not always be equal to the limit of the marginal of μ_G^λ at v of equation (2.1), i.e., BP is not exact in all graph. As for finite graphs, we have already mentionned that BP is exact in finite trees, and it will further be shown in the appendix that it is also exact in finite bipartite graphs (as an extension of [30]).

The main result of this section is the following:

Proposition 2.5 (Convergence of BP to a unique fixed point)**.** *Synchronous BP message updates according to* $\mathbf{m}^{t+1} = \mathcal{R}_G^{(\lambda)}[\mathbf{m}^t]$ *for* $t \geq 0$ *converge to the unique solution* $\mathbf{m}^{(\lambda)}$ *of the fixed point equation* $\mathbf{m} = \mathcal{R}_G^{(\lambda)}[\mathbf{m}]$.

Proof. For all $\overrightarrow{e} \in \overrightarrow{E}$ initialize the message on $\overrightarrow{e}$ at $m^0_{\overrightarrow{e}} = \chi_0 \in \widetilde{\mathcal{P}}_{\mathrm{lc}}$. As $\mathcal{R}_G^{(\lambda)}$ is non-increasing and χ_0 is a smallest element for the $\mathrm{lr}\uparrow$ order, it can readily be shown that the following inequalities hold for all $t \geq 0$:

$$\mathbf{m}^{2t} \leq_{\mathrm{lr}\uparrow} \mathbf{m}^{2t+2} \leq_{\mathrm{lr}\uparrow} \mathbf{m}^{2t+3} \leq_{\mathrm{lr}\uparrow} \mathbf{m}^{2t+1}.$$

In other words the two series $(\mathbf{m}^{2t})_{t\geq 0}$ and $(\mathbf{m}^{2t+1})_{t\geq 0}$ are *adjacent* and hence converge to respective limits $\mathbf{m}^-$, $\mathbf{m}^+$ in $\widetilde{\mathcal{P}}_{\mathrm{lc}}^{\overrightarrow{E}}$ such that $\mathbf{m}^- \leq_{\mathrm{lr}\uparrow} \mathbf{m}^+$. Continuity of $\mathcal{R}_G^{(\lambda)}$ further guarantees that $\mathbf{m}^+ = \mathcal{R}_G^{(\lambda)}(\mathbf{m}^-)$ and $\mathbf{m}^- = \mathcal{R}_G^{(\lambda)}(\mathbf{m}^+)$. Moreover, considering any other sequence of vectors of messages $(\mathbf{m}'^t)_{t\geq 0}$ with an arbitrary initialization, since $\mathbf{m}^0 \leq_{\mathrm{lr}\uparrow} \mathbf{m}'^0$, monotonicity of $\mathcal{R}_G^{(\lambda)}$ ensures that for all $t \geq 0$, one has

$$\mathbf{m}^{2t} \leq_{\mathrm{lr}\uparrow} \mathbf{m}'^{2t}, \mathbf{m}'^{2t+1} \leq_{\mathrm{lr}\uparrow} \mathbf{m}^{2t+1}.$$

The result will then follow if we can show that $\mathbf{m}^+ = \mathbf{m}^-$, which would then be the unique fixed point of $\mathcal{R}_G^{(\lambda)}$. We establish this by exploiting the fact that $\mathcal{D}_v$ is strictly increasing for inputs in $\widetilde{\mathcal{P}}_{\mathrm{lc}}$. As $\mathbf{m}^- \leq_{\mathrm{lr}\uparrow} \mathbf{m}^+$ and $\mathcal{D}_v$ is non-decreasing for the lr $\uparrow$-order for all $v \in V$, it follows $\mathcal{D}_v[\mathbf{m}^-] \leq \mathcal{D}_v[\mathbf{m}^+]$ for all $v \in V$. Then, summing over all vertices of G, we get

$$\begin{aligned}
\sum_{v\in V} \mathcal{D}_v[\mathbf{m}^-] &= \sum_{v\in V}\sum_{u\sim v} \frac{\sum_{x\in\mathbb{N}} x m^-_{\overrightarrow{uv}}(x)\mathcal{R}_{\overrightarrow{vu}}[\mathbf{m}^-](x)}{\sum_{x\in\mathbb{N}} m^-_{\overrightarrow{uv}}(x)\mathcal{R}_{\overrightarrow{vu}}[\mathbf{m}^-](x)} \\
&= \sum_{v\in V}\sum_{u\sim v} \frac{\sum_{x\in\mathbb{N}} x\mathcal{R}_{\overrightarrow{uv}}[\mathbf{m}^+](x) m^+_{\overrightarrow{vu}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{R}_{\overrightarrow{uv}}[\mathbf{m}^+](x) m^+_{\overrightarrow{vu}}(x)} \\
&= \sum_{u\in V}\sum_{v\sim u} \frac{\sum_{x\in\mathbb{N}} x\mathcal{R}_{\overrightarrow{uv}}[\mathbf{m}^+](x) m^+_{\overrightarrow{vu}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{R}_{\overrightarrow{uv}}[\mathbf{m}^+](x) m^+_{\overrightarrow{vu}}(x)} \\
&= \sum_{u\in V} \mathcal{D}_u[\mathbf{m}^+].
\end{aligned}$$

Hence, in fact, $\mathcal{D}_v[\mathbf{m}^-] = \mathcal{D}_v[\mathbf{m}^+]$ for all $v \in V$. As $\mathcal{D}_v$ is strictly increasing for these inputs, $\mathbf{m}^- = \mathbf{m}^+ = \mathbf{m}^{(\lambda)}$ follows. □

We now state results on the limiting behaviour of the fixed point of BP on a fixed finite graph G as $\lambda \to \infty$. The fixed-point $\mathbf{m}^{(\lambda)}$ at finite λ admits a limit $\mathbf{m}^{(\infty)}$, and the value of $\sum_v \mathcal{D}_v[\mathbf{m}^{(\infty)}]$ is equal to $\sum_v F_v(\boldsymbol{\alpha}^{(\infty)})$, where F_v is defined in the propositions below. This sum is computed from the infimum $\boldsymbol{\alpha}^{(\infty)}$ of the support of $\mathbf{m}^{(\infty)}$. Furthermore, it also happens that the value of $\sum_v F_v(\boldsymbol{\alpha}^{(\infty)})$ can be obtained directly as the smallest value of $\sum_v F_v(\boldsymbol{\alpha})$ over all two-step fixed points $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$.

It seems quite interesting that the value of $\sum_v \mathcal{D}_v[\mathbf{m}^{(\infty)}]$ can be computed from the support of the messages $\mathbf{m}^{(\infty)}$ only and through functions F_v which form was not a priori obvious. It is also intriguing that we can bypass the computation of $\boldsymbol{\alpha}^{(\infty)}$ and perform instead a search over the fixed points $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$. However, the fact $\boldsymbol{\alpha}^{(\infty)}$ is itself solution of the fixed point equation is not surprising: indeed, we mentioned when introducing the limiting operators $\mathcal{R}_G$ and $\mathcal{Q}_G$ (in equations (2.2) and (2.6)) that the supremum of the support of messages $\mathbf{n} = \mathcal{R}_G[\mathbf{m}]$ is computed from the infimum of the support of the messages $\mathbf{m}$ using $\boldsymbol{\beta}(\mathbf{n}) = \mathcal{S}_G(\boldsymbol{\alpha}(\mathbf{m}))$, and similarly the infimum of the support of messages $\mathbf{m} = \mathcal{Q}_G[\mathbf{m}]$ is computed from the supremum of the support of the messages $\mathbf{n}$ using $\boldsymbol{\alpha}(\mathbf{m}) = \mathcal{S}_G(\boldsymbol{\beta}(\mathbf{n}))$. Hence, for the fixed point $\mathbf{m}^{(\infty)} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}^{(\infty)}]$, it is immediate that we have $\boldsymbol{\alpha}^{(\infty)} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha}^{(\infty)})$.

Proposition 2.6 (Limit of $\lambda \to \infty$)**.** *$\mathbf{m}^{(\lambda)}$ is non-decreasing in λ for the lr$\uparrow$-order, and $\mathbf{m}^{(\infty)} = \lim \uparrow_{\lambda\to\infty} \mathbf{m}^{(\lambda)} \in \mathcal{P}_{lc}^{\overrightarrow{E}}$ is the minimal solution (for the lr$\uparrow$-order) of $\mathbf{m}^{(\infty)} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}^{(\infty)}]$.*

Proof. We first show that $\lambda \mapsto \mathbf{m}^{(\lambda)}$ and $\lambda \mapsto \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{(\lambda)}$ are respectively non-decreasing and non-increasing, where $\lambda^{-\mathbb{N}}(x) = \lambda^{-x}$ for $x \in \mathbb{N}$. We proceed by induction: $m^0_{\overrightarrow{e}} = \chi_0$ for all $\overrightarrow{e} \in \overrightarrow{E}$, hence $\lambda \mapsto \mathbf{m}^0$ and $\lambda \mapsto \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^0$ are constant functions. Suppose now that $\lambda \mapsto \mathbf{m}^k$ and $\lambda \mapsto \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^k$ are respectively non-decreasing and non-increasing, for some $k \in \mathbb{N}$. $\lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{k+1} = \mathcal{R}_G[\mathbf{m}^k]$. As $\lambda \mapsto \mathbf{m}^k$ is non-increasing and $\mathcal{R}_G$ is non-increasing, it follows $\lambda \mapsto \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{k+1}$ is non-increasing. Similarly, let $\lambda' \geq \lambda$ and call $\mathbf{m}'$ the messages associated with λ'. We have $\mathbf{m}^{k+1} = \mathcal{Q}_G^{(\lambda)}[\lambda^{-\mathbb{N}} \centerdot \mathbf{m}^k] \leq_{\mathrm{lr}\uparrow} \mathcal{Q}_G^{(\lambda)}[\lambda'^{-\mathbb{N}} \centerdot \mathbf{m}'^k]$ because $\mathbf{m}^k$ is non-increasing in λ and $\mathcal{Q}_G^{(\lambda)}$ is non-decreasing. Also, $\mathcal{Q}^{(\lambda)}$ is non-decreasing in λ, so $\mathcal{Q}_G^{(\lambda)}[\lambda'^{-\mathbb{N}} \centerdot \mathbf{m}'^k] \leq_{\mathrm{lr}\uparrow} \mathcal{Q}_G^{(\lambda')}[\lambda'^{-\mathbb{N}} \centerdot \mathbf{m}'^k] = \mathbf{m}'^{k+1}$.

Taking the limit $k \to \infty$, we obtain that $\lambda \mapsto \mathbf{m}^{(\lambda)}$ and $\lambda \mapsto \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{(\lambda)}$ are respectively non-decreasing and non-increasing. This allows us to define $\mathbf{m}^{(\infty)} = \lim \uparrow_{\lambda\to\infty} \mathbf{m}^{(\lambda)} \in \mathcal{P}_{\mathrm{lc}}^{\overrightarrow{E}}$ and $\mathbf{n}^{(\infty)} = \lim \downarrow_{\lambda\to\infty} \lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{(\lambda)} \in \widetilde{\mathcal{P}}_{\mathrm{lc}}^{\overrightarrow{E}}$. Passing to the limit $\lambda \to \infty$ in $\lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{(\lambda)} = \mathcal{R}_G[\mathbf{m}^{(\lambda)}]$ and in $\mathbf{m}^{(\lambda)} = \mathcal{Q}_G^{(\lambda)}[\lambda^{-\mathbb{N}} \centerdot \mathbf{m}^{(\lambda)}]$, we obtain

$$\mathbf{n}^{(\infty)} = \mathcal{R}_G[\mathbf{m}^{(\infty)}] \text{ and } \mathbf{m}^{(\infty)} = \mathcal{Q}_G[\mathbf{n}^{(\infty)}]. \tag{2.7}$$

Let $\mathbf{m} \in \mathcal{P}^{\overrightarrow{E}}$ be another solution of the two-step equation given by (2.7). For all $\overrightarrow{e} \in \overrightarrow{E}$ we have $m_{\overrightarrow{e}} \geq_{\mathrm{lr}\uparrow} \chi_0$, hence $\mathbf{m}^{2k} \leq_{\mathrm{lr}\uparrow} \left(\mathcal{Q}_G^{(\lambda)} \circ \mathcal{R}_G\right)^k [\mathbf{m}] \leq_{\mathrm{lr}\uparrow} \mathbf{m}$, where the first inequality is obtained by applying the non-decreasing operator $\mathcal{Q}_G^{(\lambda)} \circ \mathcal{R}_G$ k times and the second one follows from the fact $\mathcal{Q}_G^{(\lambda)} \circ \mathcal{R}_G[\mathbf{m}]$ is non-decreasing in λ, thus $\left(\mathcal{Q}_G^{(\lambda)} \circ \mathcal{R}_G\right)^k [\mathbf{m}] \leq_{\mathrm{lr}\uparrow} (\mathcal{Q}_G \circ \mathcal{R}_G)^k [\mathbf{m}] = \mathbf{m}$. Taking the limit $k \to \infty$ yields $\mathbf{m}^{(\lambda)} \leq_{\mathrm{lr}\uparrow} \mathbf{m}$, and then $\lambda \to \infty$ gives $\mathbf{m}^{(\infty)} \leq_{\mathrm{lr}\uparrow} \mathbf{m}$. □

Then, it is possible to compute the limiting value of the BP estimate in any finite graph G, i.e., the limit as $\lambda \to \infty$ of the sum over the vertices v of G of $\mathcal{D}_v[\mathbf{m}^{(\lambda)}]$. The case of unimodular random graphs being more general, the proof of the following proposition is included only in this context.

Proposition 2.7 (BP estimate in finite graphs)**.** *In a finite graph G, we have*

$$\begin{aligned} \lim \uparrow_{\lambda\to\infty} \sum_{v\in V} \mathcal{D}_v[\mathbf{m}^{(\lambda)}] &= \sum_{v\in V} \mathcal{D}_v[\mathbf{m}^{(\infty)}] \\ &= \sum_{v\in V} F_v(\boldsymbol{\alpha}^{(\infty)}) \\ &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \sum_{v\in V} F_v(\boldsymbol{\alpha}), \end{aligned}$$

where $F_v(\boldsymbol{\alpha}) = \min(b_v, |\boldsymbol{\alpha}_{\overrightarrow{\partial v}}|) + (b_v - |\boldsymbol{\alpha}_{\overleftarrow{\partial v}}|)^+$.

Moreover, if G is bipartite with $V = L \cup R$, we have

$$\frac{1}{2} \lim \uparrow_{\lambda\to\infty} \sum_{v\in V} \mathcal{D}_v[\mathbf{m}^{(\lambda)}] = \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} F^L(\boldsymbol{\alpha}),$$

where $F^L(\boldsymbol{\alpha}) = \sum_{l\in L} \min\left\{b_l, |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}|\right\} + \sum_{r\in R,\, |\mathbf{c}_{\partial r}|>b_r} \left(b_r - |\mathcal{S}_{\overleftarrow{\partial r}}(\boldsymbol{\alpha})|\right)^+$.

We can define $F^R(\boldsymbol{\alpha})$ with a similar expression with the roles of L and R inverted. It is clear that $F^R(\boldsymbol{\alpha}) = F^L(\boldsymbol{\alpha})$.

Remark 2.2. *In a finite tree, there is only one possible value for* $\alpha_{\overrightarrow{e}} = \mathcal{S}_{\overrightarrow{e}} \circ \mathcal{S}_{\overrightarrow{\partial e}}[\boldsymbol{\alpha}]$, *where* $\circ$ *is the composition operation, when* $\overrightarrow{e}$ *is an edge outgoing from a leaf* v*: it is* $\alpha_{\overrightarrow{e}} = \min\{b_v, c_e\}$. *It is then possible to compute the whole, unique fixed-point vector* $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$ *in an iterative manner, starting from the leaves of the tree and climbing up. This gives a simple, iterative way to compute the maximum size of allocations in finite trees, which is the natural extension of the leaf-removal algorithm for matchings.*

2.6 Limit of large graphs

This section extends the results obtained so far for finite graphs to infinite graphs. We show that most calculations done in the case of finite graphs extend naturally to the context of (potentially infinite) unimodular graphs. The transition from finite graphs to their local weak limit has already been dealt with in the literature. Furthermore, when the infinite limiting graph is a Galton-Watson tree, the branching property of this limiting tree allows us to transform the fixed-point equation of BP into recursive distributional equations, which can be solved in practice.

2.6.1 Convergence of BP in unimodular graphs and computation of BP estimate

Our first result is that the BP updates admit a unique fixed-point at finite activity parameter λ:

Proposition 2.8. *Let* $\rho \in \mathcal{U}$ *with* $\bar{b}(\rho) < \infty$. *Then, the fixed point equation* $\mathbf{m} = \mathcal{R}^{(\lambda)}[\mathbf{m}]$ *admits a unique solution* $\mathbf{m}^{(\lambda)}$ *for any* $\lambda \in \mathbb{R}^+$ *for* ρ*-almost every marked graph* G.

Proof. As in the finite graph case and because $\mathcal{R}_G^{(\lambda)}$ is a non-increasing operator, we can define two adjacent sequences of messages over the directed edges of G by iterating $\mathcal{R}_G^{(\lambda)}$: $\mathbf{m}^0_{\overrightarrow{e}} = \chi_0$ for every $\overrightarrow{e}$ and $\mathbf{m}^{t+1} = \mathcal{R}_G^{(\lambda)}[\mathbf{m}^t]$. Then we can define

$$\begin{aligned} \mathbf{m}^- &= \lim \uparrow_{t\to\infty} \mathbf{m}^{2t}, \\ \mathbf{m}^+ &= \lim \downarrow_{t\to\infty} \mathbf{m}^{2t+1}. \end{aligned}$$

Again, any solution of the fixed point equation $\mathbf{m} = \mathcal{R}_G^{(\lambda)}[\mathbf{m}]$ must be between $\mathbf{m}^-$ and $\mathbf{m}^+$. However, the proof differs from the finite graph case when we want to show that two limit points $\mathbf{m}^-$ and $\mathbf{m}^+$ are identical. Indeed, we cannot sum $\mathcal{D}_v$ over all the vertices of v anymore. Instead, we use the MTP for

$$f(G, o, v) = \frac{\sum_{x\in\mathbb{N}} x m^-_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)}{\sum_{x\in\mathbb{N}} m^-_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)},$$

which yields

$$\begin{aligned}\int \mathcal{D}_o[\mathbf{m}^-]d\rho([G,o]) &= \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}} x m^-_{\overrightarrow{ov}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)}{\sum_{x\in\mathbb{N}} m^-_{\overrightarrow{ov}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)} d\rho([G,o]) \\ &= \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}} x\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^+](x)m^+_{\overrightarrow{ov}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)m^-_{\overrightarrow{ov}}(x)} d\rho([G,o]) \\ &= \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}} x\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^+](x)m^+_{\overrightarrow{ov}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^-](x)m^-_{\overrightarrow{ov}}(x)} d\rho([G,o]) \\ &= \int \mathcal{D}_o[\mathbf{m}^+]d\rho([G,o])\end{aligned}$$

Because $\overline{b}(\rho) < \infty$, these expectations are finite, and as $\mathcal{D}_v$ is strictly increasing for all $v \in V$ it yields that $\mathbf{m}^-_{\overrightarrow{\partial o}} = \mathbf{m}^+_{\overrightarrow{\partial o}}$, ρ-almost surely. By Lemma 2.3 [4], this result extends to the edges incoming to ρ-almost every vertex in G, hence $\mathbf{m}^- = \mathbf{m}^+ = \mathbf{m}^{(\lambda)}$ ρ-a.s. □

The rest of the reasoning goes as in the finite graph case (using the MTP again, instead of summing over all directed edges). Proposition 2.6 is still valid. If $\rho \in \mathcal{U}$ is concentrated on bipartite unimodular graphs, with parts L and R, we introduce the probability measures ρ^L and ρ^R on $\mathcal{U}$ by conditioning on the root being in L or R:

$$\rho^L([G,v]) = \rho([G,v])\frac{\mathbf{1}(v \in L)}{\int \mathbf{1}(o \in L)d\rho([G,o])},$$

and similarly for ρ^R. To ease the notation, we also write $\tilde{\rho}^L = \int \mathbf{1}(o \in L)d\rho([G,o])\rho^L$ and similarly for ρ^R. Of course, $\tilde{\rho}^L$ and $\tilde{\rho}^R$ are not probability measures. The following proposition is analogous to Proposition 2.7:

Proposition 2.9 (BP estimate in unimodular random graphs)**.** *Let $\rho \in \mathcal{U}$ with $\overline{b}(\rho) < \infty$,*

$$\begin{aligned}\lim\uparrow_{\lambda\to\infty} \int \mathcal{D}_o[\mathbf{m}^{(\lambda)}]d\rho([G,o]) &= \int \mathcal{D}_o[\mathbf{m}^{(\infty)}]d\rho([G,o]) \\ &= \int F_o(\boldsymbol{\alpha}^{(\infty)})d\rho([G,o]) \\ &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \int F_o(\boldsymbol{\alpha})d\rho([G,o]),\end{aligned}$$

where $F_v(\boldsymbol{\alpha}) = \min(b_v, |\boldsymbol{\alpha}_{\overrightarrow{\partial v}}|) + (b_v - |\boldsymbol{\alpha}_{\overleftarrow{\partial v}}|)^+$ as before.

Moreover, if in addition the measure ρ is concentrated on bipartite graphs, with parts L and R, we have

$$\begin{aligned}&\tfrac{1}{2}\lim\uparrow_{\lambda\to\infty} \int \mathcal{D}_o[\mathbf{m}^{(\lambda)}]d\rho([G,o]) \\ &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \Bigg\{ \int \min\left\{b_o, |\boldsymbol{\alpha}_{\overrightarrow{\partial o}}|\right\} d\tilde{\rho}^L([G,o]) \\ &+ \int \left(b_o - |\mathcal{S}_{\overleftarrow{\partial o}}(\boldsymbol{\alpha})|\right)^+ \mathbf{1}(|\mathbf{c}_{\partial o}| > b_o) d\tilde{\rho}^R([G,o]) \Bigg\}.\end{aligned}$$

We will need the following lemmas:

Lemma 2.3. *Let* $\mathbf{m}_{\overrightarrow{\partial v}} \in \mathcal{P}^{\overrightarrow{\partial v}}$ *and* $\mathbf{n}_{\overleftarrow{\partial v}} = \mathcal{Q}_{\overleftarrow{\partial v}}[\mathbf{m}]$. *We have*

$$\mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v) \sum_{\overrightarrow{e} \in \overrightarrow{\partial v}} \frac{\sum_x (x - \mathcal{S}_{\overleftarrow{e}}(\boldsymbol{\beta}))\mathcal{Q}_{\overleftarrow{e}}[\mathbf{n}](x) n_{\overrightarrow{e}}(x)}{\sum_x \mathcal{Q}_{\overleftarrow{e}}[\mathbf{n}](x) n_{\overrightarrow{e}}(x)} = \left(b_v - |\mathcal{S}_{\overleftarrow{\partial v}}(\boldsymbol{\beta})|\right)^+$$

Proof. Both sides are equal to 0 unless $|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v$. We have

$$\begin{aligned}
& \mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v) \sum_{\overrightarrow{e} \in \overrightarrow{\partial v}} \frac{\sum_x (x - \mathcal{S}_{\overleftarrow{e}}(\boldsymbol{\beta}))\mathcal{Q}_{\overleftarrow{e}}[\mathbf{n}](x) n_{\overrightarrow{e}}(x)}{\sum_x \mathcal{Q}_{\overleftarrow{e}}[\mathbf{n}](x) n_{\overrightarrow{e}}(x)} \\
= \; & \mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v) \frac{\sum_{|\mathbf{x}|=b_v} (|\mathbf{x}| - |\mathcal{S}_{\overleftarrow{\partial v}}(\boldsymbol{\beta})|)\mathbf{n}_{\overrightarrow{\partial v}}(\mathbf{x})}{\sum_{|\mathbf{x}|=b_v} \mathbf{n}_{\overrightarrow{\partial v}}(\mathbf{x})} \\
= \; & (b_v - |\mathcal{S}_{\overleftarrow{\partial v}}(\boldsymbol{\beta})|)\mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v) \\
= \; & (b_v - |\mathcal{S}_{\overleftarrow{\partial v}}(\boldsymbol{\beta})|)^+
\end{aligned}$$

where the last line follows from the fact $\mathcal{S}_{\overleftarrow{e}}(\boldsymbol{\beta}) = [b_v - |\boldsymbol{\beta}_{\overrightarrow{\partial v}}| + \beta_{\overrightarrow{e}}]_0^{c_e} \leq \beta_{\overrightarrow{e}}$ if $|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| > b_v$, the inequality being strict whenever $\beta_{\overrightarrow{e}} > 0$, and it is then easy to check that $\sum_{\overleftarrow{e} \in \overleftarrow{\partial v}} \mathcal{S}_{\overleftarrow{e}}(\boldsymbol{\beta}) \leq \sum_{\overrightarrow{e} \in \overrightarrow{\partial v}} \beta_{\overrightarrow{e}} - (|\boldsymbol{\beta}_{\overrightarrow{\partial v}}| - b_v) \leq b_v$. □

Lemma 2.4. *Consider* $\mathbf{m}' = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}]$ *such that* $\boldsymbol{\alpha}(\mathbf{m}) = \boldsymbol{\alpha}(\mathbf{m}')$.

- *If* $\mathbf{m}' \geq_{lr\uparrow} \mathbf{m}$, *then* $\int \mathcal{D}_o[\mathbf{m}] d\rho([G,o]) \leq \int F_o(\boldsymbol{\alpha}(\mathbf{m})) d\rho([G,o])$.
- *If* $\mathbf{m}' \leq_{lr\uparrow} \mathbf{m}$, *then* $\int \mathcal{D}_o[\mathbf{m}] d\rho([G,o]) \geq \int F_o(\boldsymbol{\alpha}(\mathbf{m})) d\rho([G,o])$.

Proof. Let $\mathbf{n} = \mathcal{R}_G[\mathbf{m}]$. For any $v \in V$, $|\boldsymbol{\alpha}_{\overrightarrow{\partial v}}(\mathbf{m})| \leq b_v$ is equivalent to $|\boldsymbol{\beta}_{\overleftarrow{\partial v}}(\mathbf{n})| \geq b_v$. We have

$$\begin{aligned}
& \int \mathcal{D}_o[\mathbf{m}] d\rho([G,o]) \\
& = \int \Big(\mathbf{1}(|\boldsymbol{\alpha}_{\overrightarrow{\partial o}}(\mathbf{m})| < b_o) \textstyle\sum_{v \in \partial o} \frac{\sum_{x \in \mathbb{N}} (x - \alpha(m_{\overrightarrow{vo}})) m_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x \in \mathbb{N}} m_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)} \\
& + \min(b_o, |\boldsymbol{\alpha}_{\overrightarrow{\partial o}}(\mathbf{m})|) \Big) d\rho([G,o])
\end{aligned}$$

Furthermore, for any $v \sim o$, $\sum_{x \in \mathbb{N}} (x - \alpha(m_{\overrightarrow{vo}})) m_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)$ is non-zero only if $\alpha(m_{\overrightarrow{vo}}) < \mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\alpha}(\mathbf{n}))$, which also implies $|\boldsymbol{\alpha}_{\overrightarrow{\partial o}}(\mathbf{m})| < b_o$. Thus, the second term in the expression of $\int \mathcal{D}_o[\mathbf{m}] d\rho([G,o])$ above is equal to

$$\int \sum_{v \sim o} \frac{\sum_{x \in \mathbb{N}} (x - \alpha(m_{\overrightarrow{vo}})) m'_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x \in \mathbb{N}} m'_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)} \mathbf{1}(\alpha_{\overrightarrow{vo}} < \mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\alpha}(\mathbf{n}))) d\rho([G,o])$$

Suppose first that $\mathbf{m}' \geq_{\mathrm{lr}\uparrow} \mathbf{m}$. Then, for any $v \sim o$, we have $m'_{\overrightarrow{vo}} \centerdot \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}] \geq_{\mathrm{lr}\uparrow} m_{\overrightarrow{vo}} \centerdot \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}]$, as also $\alpha(m'_{\overrightarrow{vo}}) = \alpha(m_{\overrightarrow{vo}})$. Hence, we have

$$\frac{\sum_{x \in \mathbb{N}} (x - \alpha(m_{\overrightarrow{vo}})) m'_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x \in \mathbb{N}} m'_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)} \geq \frac{\sum_{x \in \mathbb{N}} (x - \alpha(m_{\overrightarrow{vo}})) m_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x \in \mathbb{N}} m_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}$$

It follows

$$\begin{aligned}
&\int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\alpha(m_{\overrightarrow{vo}}))m'_{\overrightarrow{vo}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x\in\mathbb{N}} m'_{\overrightarrow{vo}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}\mathbf{1}(\alpha(m_{\overrightarrow{vo}}) < \mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\alpha}(\mathbf{n})))d\rho([G,o]) \\
&\geq \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\alpha(m_{\overrightarrow{vo}}))m'_{\overrightarrow{vo}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}{\sum_{x\in\mathbb{N}} m'_{\overrightarrow{vo}}(x)\mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}](x)}\mathbf{1}(\alpha(m'_{\overrightarrow{vo}}) < \mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\alpha}(\mathbf{n})))d\rho([G,o]) \\
&= \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\mathcal{S}_{\overrightarrow{vo}}(\boldsymbol{\beta}(\mathbf{n})))\mathcal{Q}_{\overrightarrow{vo}}[\mathbf{n}](x)n_{\overrightarrow{ov}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{Q}_{\overrightarrow{vo}}[\mathbf{n}](x)n_{\overrightarrow{ov}}(x)}\mathbf{1}(\mathcal{S}_{\overrightarrow{vo}}(\boldsymbol{\beta}(\mathbf{n})) < \beta(n_{\overrightarrow{ov}}))d\rho([G,o]) \\
&= \int \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})))\mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}\mathbf{1}(\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})) < \beta(n_{\overrightarrow{vo}}))d\rho([G,o]),
\end{aligned}$$

where the last equality follows from by the Mass-Transport Principle. Given the facts that $\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})) < \beta(n_{\overrightarrow{vo}})$ implies $|\boldsymbol{\beta}_{\overrightarrow{\partial o}}(\mathbf{n})| > b_o$, that the quantity $\mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)$ is non-zero only if $\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})) \leq \beta(n_{\overrightarrow{vo}})$, and that $|\boldsymbol{\beta}_{\overrightarrow{\partial o}}(\mathbf{n})| > b_o$ implies $\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})) \leq \beta(n_{\overrightarrow{vo}})$, we have in fact

$$\begin{aligned}
&\sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})))\mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}\mathbf{1}(\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})) < \beta(n_{\overrightarrow{vo}})) \\
&= \sum_{v\sim o} \frac{\sum_{x\in\mathbb{N}}(x-\mathcal{S}_{\overrightarrow{ov}}(\boldsymbol{\beta}(\mathbf{n})))\mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}{\sum_{x\in\mathbb{N}} \mathcal{Q}_{\overrightarrow{ov}}[\mathbf{n}](x)n_{\overrightarrow{vo}}(x)}\mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial o}}(\mathbf{n})| > b_o|) \\
&= \mathbf{1}(|\boldsymbol{\beta}_{\overrightarrow{\partial o}}(\mathbf{n})| > b_o|)(b_o - |\mathcal{S}_{\overleftarrow{\partial o}}(\boldsymbol{\beta}(\mathbf{n}))|)^+ \\
&= (b_o - |\boldsymbol{\alpha}_{\overleftarrow{\partial o}}(\mathbf{m}')|)^+,
\end{aligned}$$

according to Lemma 2.3. We thus obtained that

$$\int \mathcal{D}_o[\mathbf{m}]d\rho([G,o]) \leq \int F_o(\boldsymbol{\alpha}(\mathbf{m}))d\rho([G,o]) \text{ if } \mathbf{m}' \geq_{\mathrm{lr}\uparrow} \mathbf{m}.$$

The proof for the case $\mathbf{m}' \leq_{\mathrm{lr}\uparrow} \mathbf{m}$ is identical. □

proof of Proposition 2.9. For any $v \in V$, as $\mathcal{D}_v$ is continuous for inputs in $\widetilde{\mathcal{P}}_{\mathrm{lc}}$ such as $\mathbf{m}^{(\lambda)}$, we have $\lim\uparrow_{\lambda\to\infty} \mathcal{D}_v[\mathbf{m}^{(\lambda)}] = \mathcal{D}_v[\mathbf{m}^{(\infty)}]$ and the first equality follows by monotone convergence. Lemma 2.4 shows

$$\int \mathcal{D}_o[\mathbf{m}^{(\infty)}]d\rho([G,o]) = \int F_o(\boldsymbol{\alpha}(\mathbf{m}^{(\infty)}))d\rho([G,o]).$$

It is clear that $\boldsymbol{\alpha}(\mathbf{m}^{(\infty)}) = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha}(\mathbf{m}^{(\infty)}))$, as $\mathbf{m}^{(\infty)} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}^{(\infty)}]$. Consider now $\boldsymbol{\alpha} \in \mathbb{N}^{\overrightarrow{E}}$ such that $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$ and the measure ρ with marks $\boldsymbol{\alpha}$ is unimodular. We define $\mathbf{m}^{(0)}_{\overrightarrow{e}} = \chi_{\alpha_{\overrightarrow{e}}}$ and $\mathbf{m}^{(k+1)} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}^{(k)}]$. We have $\boldsymbol{\alpha}(\mathbf{m}^{(k)}) = \boldsymbol{\alpha}$ for any $k \in \mathbb{N}$, and then necessarily $\mathbf{m}^{(k)} \geq_{\mathrm{lr}\uparrow} \mathbf{m}^{(0)}$ (it suffices to look at the support of $m^{(k)}_{\overrightarrow{e}}$ and $m^{(0)}_{\overrightarrow{e}}$ to check this). It follows, because $\mathcal{Q}_G \circ \mathcal{R}_G$ is non-decreasing, that $\left(\mathbf{m}^{(k)}\right)_{k\in\mathbb{N}}$ is a non-decreasing sequence, and thus $\int \mathcal{D}_o[\mathbf{m}^{(k)}]d\rho([G,o])$ is non-decreasing in k (as $\mathcal{D}_o$ is non-decreasing).

We define $\mathbf{m} = \lim\uparrow_{k\in\mathbb{N}} \mathbf{m}^{(k)}$. Clearly, $\boldsymbol{\alpha}(\mathbf{m}) \geq \boldsymbol{\alpha}$ and $\mathbf{m} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}]$. Moreover, by monotone convergence, we have

$$\begin{aligned}
\lim\uparrow_{k\to\infty} \int \mathcal{D}_o[\mathbf{m}^{(k)}]d\rho([G,o]) &= \int \mathcal{D}_o[\mathbf{m}]d\rho([G,o]) \\
&= \int F_o(\boldsymbol{\alpha}(\mathbf{m}))d\rho([G,o]) \\
&\geq \int \mathcal{D}_o[\mathbf{m}^{(\infty)}]d\rho([G,o]) \\
&= \int F_o(\boldsymbol{\alpha}(\mathbf{m}^{(\infty)}))d\rho([G,o]),
\end{aligned}$$

where we used Lemma 2.4 for $\mathbf{m}$ and for $\mathbf{m}^{(\infty)}$ and Proposition 2.6 together with the fact $\mathcal{D}_o$ is non-decreasing to get the inequality. For any $k \in \mathbb{N}$ we have $\mathbf{m}^{(k+1)} = \mathcal{Q}_G \circ \mathcal{R}_G[\mathbf{m}^{(k)}] \geq_{\text{lr}\uparrow} \mathbf{m}^{(k)}$, thus applying Lemma 2.4 we obtain that

$$\begin{aligned}
\int F_o(\boldsymbol{\alpha}) d\rho([G,o]) &= \int F_o(\boldsymbol{\alpha}(\mathbf{m}^{(k)})) d\rho([G,o]) \\
&\geq \int \mathcal{D}_o[\mathbf{m}^{(k)}] d\rho([G,o]) \\
&\underset{k\to\infty}{\nearrow} \int \mathcal{D}_o[\mathbf{m}] d\rho([G,o]) \\
&\geq \int F_o(\boldsymbol{\alpha}(\mathbf{m}^{(\infty)})) d\rho([G,o]),
\end{aligned}$$

which completes the proof of the first part of the proposition.

Assume now that in addition the measure ρ is concentrated on bipartite graphs, with parts L and R. For $\lambda \in \mathbb{R}^+$, recall $\mathbf{m}^{(\lambda)} = \mathcal{R}_G^{(\lambda)}[\mathbf{m}^{(\lambda)}]$. Applying the MTP to ρ with

$$f^L(G,o,v) = \frac{\sum_{x\in\mathbb{N}} x m^{(\lambda)}_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^{(\lambda)}](x)}{\sum_{x\in\mathbb{N}} m^{(\lambda)}_{\overrightarrow{vo}}(x) \mathcal{R}_{\overrightarrow{ov}}[\mathbf{m}^{(\lambda)}](x)} \mathbf{1}(o \in L)$$

and f^R defined similarly, we obtain

$$\begin{aligned}
\int \mathcal{D}_o[\mathbf{m}^{(\lambda)}] d\tilde{\rho}^L([G,o]) &= \int \sum_v f^L(G,o,v) d\rho([G,o]) \\
&= \int \sum_v f^L(G,v,o) d\rho([G,o]) \\
&= \int \sum_v f^R(G,o,v) d\rho([G,o]) \\
&= \int \mathcal{D}_o[\mathbf{m}^{(\lambda)}] d\tilde{\rho}^R([G,o])
\end{aligned}$$

Letting $\lambda \to \infty$ yields $\int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^L([G,o]) = \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^R([G,o])$. As also

$$\int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\rho([G,o]) = \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^L([G,o]) + \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^R([G,o]),$$

we get

$$\frac{1}{2} \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\rho([G,o]) = \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^L([G,o]).$$

We then follow exactly the steps in the proof of the first part of the proposition, for $\tilde{\rho}^L$ instead of ρ. This gives

$$\begin{aligned}
\int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\tilde{\rho}^L([G,o]) &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \Bigg\{ \int \min(b_o, |\boldsymbol{\alpha}_{\overrightarrow{\partial o}}|) d\tilde{\rho}^L([G,o]) \\
&\quad + \int (b_o - |\boldsymbol{\alpha}_{\overleftarrow{\partial o}}|)^+ d\tilde{\rho}^R([G,o]) \Bigg\},
\end{aligned}$$

which completes the proof of Proposition 2.9. □

2.6.2 From finite graphs to unimodular trees

Showing that the BP estimate is asymptotically exact for sequences of (sparse) random graphs is quite systematic and follows the same steps as in [21], [96] and [70]. We simply state here that we can invert the limits in n and λ (see Proposition 6 in [70]).

Proposition 2.10 (Asymptotic correctness for large, sparse random graphs)**.** *Let $G_n = (V_n, E_n)_{n\in\mathbb{N}}$ be a sequence of finite marked graphs with random weak limit ρ concentrated on unimodular trees, with $\overline{b}(\rho) < \infty$. Then,*

$$\begin{aligned}\lim_{n\to\infty} \frac{2M_n}{|V_n|} &= \int \mathcal{D}_o[\mathbf{m}^{(\infty)}] d\rho([G,o]) \\ &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \int F_o(\boldsymbol{\alpha}) d\rho([G,o]).\end{aligned}$$

2.6.3 Galton-Watson trees

The main theorem follows quite straightforwardly from Propositions 2.9 and 2.10. The missing steps are standard and can be found in [70]; they resemble much the computation done in the proof of Proposition 2.9.

proof of Theorem 2.1. First-of-all, the fact that the sequence of graphs considered converges locally weakly to unimodular Galton-Watson trees follows from standard results in the random graphs literature (see [66] for random hypergraphs or [34] for graphs with fixed degree sequence). Propositions 2.9 and 2.10 together then give:

$$\begin{aligned}\lim_{n\to\infty} \frac{2M(G_n)}{|L_n|+|R_n|} &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \int F_o(\boldsymbol{\alpha}) d\rho([G,o]), \\ \lim_{n\to\infty} \frac{2M(G_n)}{|L_n|} &= \inf_{\boldsymbol{\alpha}=\mathcal{S}_G\circ\mathcal{S}_G(\boldsymbol{\alpha})} \left\{ \int \min\left\{b_o, |\boldsymbol{\alpha}_{\overrightarrow{\partial o}}|\right\} d\rho^L([G,o]) \right. \\ &+ \left. \frac{\mathbb{E}[D^L]}{\mathbb{E}[D^R]} \int \left(b_o - |\mathcal{S}_{\overleftarrow{\partial o}}(\boldsymbol{\alpha})|\right)^+ \mathbf{1}(|\mathbf{c}_{\partial o}| > b_o) d\rho^R([G,o]) \right\}.\end{aligned}$$

The second step uses the Markovian nature of the limiting Galton-Watson tree to simplify the infinite recursions $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$ into recursive distributional equations as the ones described in Theorem 2.1: as G is a unimodular tree, for any vertex $v \in V$, all the components of $\boldsymbol{\alpha}_{\overrightarrow{\partial v}}$ can be chosen independently (as they are independent in $\boldsymbol{\alpha}^{(\infty)}_{\overrightarrow{\partial v}}$, which achieves the infimum). Then, for $\overrightarrow{e}$ incoming to v, $\alpha_{\overrightarrow{e}}$ is determined only from the subtree stemming from the tail of $\overrightarrow{e}$; furthermore it satisfies $\alpha_{\overrightarrow{e}} = \mathcal{S}_{\overrightarrow{e}} \circ \mathcal{S}_{\overrightarrow{\partial e}}[\boldsymbol{\alpha}]$. However, the distribution of the subtree at the tail of an $\overrightarrow{e}'$ which is an input to $\mathcal{S}_{\overrightarrow{\partial e}}$ is the same as that of the subtree at the tail of $\overrightarrow{e}$, by the two-step branching property of the two-step unimodular Galton-Watson tree G. This implies that, for $\overrightarrow{e}$ incoming to a root $o \in L$, $\alpha_{\overrightarrow{e}}$ is solution of the two-step RDE given in the statement of the theorem. As detailed in Lemma 6 of [5], there is actually a one-to-one mapping between the solutions of $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G[\boldsymbol{\alpha}]$ on a Galton-Watson tree G and the solutions of the RDE considered here. This completes the proof. □

2.7 Exactness in finite bipartite graphs

We finish by showing that the limiting BP estimate will actually be exact in the case of finite bipartite graphs. In [30] Chertkov builds upon the fact an associated linear programming relaxation is gapless due to the total unimodularity of the incidence matrix of bipartite graphs to show that the limiting BP estimate is correct for bipartite graphs, i.e., that

$$\frac{1}{2}\lim \uparrow_{\lambda\to\infty} \sum_{v\in V} \mathcal{D}_v[\mathbf{m}^{(\lambda)}] = M(G). \tag{2.8}$$

Even though [30] only deals with unitary capacities, it is possible to apply similar ideas to our setup; however our proof will follow a slightly different path as we believe it gives more insights on the interpretation of the BP messages.

Proposition 2.11 (Correctness for finite bipartite graphs)**.** *In a finite bipartite graph $G = (V, E)$ with $V = L \cup R$,*

$$M(G) = \inf_{\boldsymbol{\alpha}=\mathcal{S}_G(\boldsymbol{\alpha})} F^L(\boldsymbol{\alpha}).$$

Remark 2.3. *Note first that Proposition 2.11 displays an infimum over one-step fixed points of $\mathcal{S}_G$, while Proposition 2.7 is concerned with two-step fixed points instead. However, we are dealing here with bipartite graphs, hence from a two-step fixed point $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$ we can easily construct a one-step fixed point $\boldsymbol{\alpha}'$ in the following manner: let $l \sim r$ be two neighbors, with $l \in L$ and $r \in R$; let $\alpha'_{l\to r} = \alpha_{l\to r}$ and $\alpha'_{r\to l} = \mathcal{S}_{r\to l}(\boldsymbol{\alpha})$. It is straightforward to check that $\boldsymbol{\alpha}' = \mathcal{S}_G(\boldsymbol{\alpha}')$. Furthermore, the expression for bipartite graphs in Proposition 2.7 can be computed from only the set of messages going from L to R, denoted $\boldsymbol{\alpha}_{L\to R}$, and the messages $\mathcal{S}_G(\boldsymbol{\alpha})$ going from R to L, denoted by $\mathcal{S}_{R\to L}(\boldsymbol{\alpha})$. Hence in fact it can be computed from the one-step fixed point $\boldsymbol{\alpha}'$ associated with a two-step fixed point $\boldsymbol{\alpha}$ in the manner described just before.*

Conversely, from any two one-step fixed points $\boldsymbol{\alpha}^1 = \mathcal{S}_G(\boldsymbol{\alpha}^1)$ and $\boldsymbol{\alpha}^2 = \mathcal{S}_G(\boldsymbol{\alpha}^2)$ we can construct a two-step fixed point $\boldsymbol{\alpha}$ as follows: let $l \sim r$ be two neighbors, with $l \in L$ and $r \in R$; let $\boldsymbol{\alpha}_{l\to r} = \boldsymbol{\alpha}^1_{l\to r}$ and $\boldsymbol{\alpha}_{r\to l} = \boldsymbol{\alpha}^2_{r\to l}$. It is easy to check that $\boldsymbol{\alpha} = \mathcal{S}_G \circ \mathcal{S}_G(\boldsymbol{\alpha})$. Furthermore, it is obvious from the expressions in Proposition 2.7 that $\sum_{v\in V} F_v(\boldsymbol{\alpha}) = F^L(\boldsymbol{\alpha}^1) + F^R(\boldsymbol{\alpha}^2)$. Thus, we conclude that for bipartite graphs the values are the same in Propositions 2.7 and 2.11 whether we consider one-step or two-step fixed points in the infimums. It follows also that Propositions 2.7 and 2.11 together recover the result annouced in equation (2.8).

proof of Proposition 2.11. The maximum size $M(G)$ of a c-capacitated b-matching of G is given by the value of the following integer program (IP):

$$\begin{aligned} \max \quad & |\mathbf{x}| \\ \text{s.t.} \quad & \begin{pmatrix} A \\ Id \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \\ & \mathbf{x} \in \mathbb{N}^{|E|} \end{aligned}$$

where Id is the identity matrix of size $|E|$ and A is the incidence matrix of G, i.e., $A_{ve} = \mathbf{1}(e \in \partial v)$.

Because the matrix $\begin{pmatrix} A \\ Id \end{pmatrix}$ is totally unimodular (see [57]), the value of the above program is equal to that of its linear programming relaxation (P):

$$\begin{array}{ll} \max & |\mathbf{x}| \\ \text{s.t.} & \begin{pmatrix} A \\ Id \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} \mathbf{b} \\ \mathbf{c} \end{pmatrix} \\ & \mathbf{x} \geq 0 \end{array}$$

Due to strong duality of linear programming, the value of (P) is equal to that of its dual (D):

$$\begin{array}{ll} \min & \mathbf{b}^T\mathbf{y} + \mathbf{c}^T\mathbf{z} \\ \text{s.t.} & A^T\mathbf{y} + \mathbf{z} \geq 1 \\ & \mathbf{y}, \mathbf{z} \geq 0 \end{array}$$

And again, total unimodularity gives that the value of this program is also equal to that of its integer version (ID):

$$\begin{array}{ll} \min & \mathbf{b}^T\mathbf{y} + \mathbf{c}^T\mathbf{z} \\ \text{s.t.} & A^T\mathbf{y} + \mathbf{z} \geq 1 \\ & \mathbf{y} \in \{0,1\}^{|V|}, \mathbf{z} \in \{0,1\}^{|E|} \end{array}$$

If $\mathbf{z} = 0$, then $\mathbf{y}$ satisfying the constraints of (ID) would be called a vertex cover of G, hence we will call a vector $(\mathbf{y}, \mathbf{z})$ satisfying the constraints of (ID) a generalized cover of the edges of G.

Lemma 2.5 (To each fixed point a generalized cover of the edges of G)**.** *Let $\boldsymbol{\alpha} = \mathcal{S}_G(\boldsymbol{\alpha})$. We can associate to $\boldsymbol{\alpha}$ a generalized cover $(\mathbf{y}^{\boldsymbol{\alpha}}, \mathbf{z}^{\boldsymbol{\alpha}})$ of the edges of G such that*

$$\mathbf{b}^T\mathbf{y}^{\boldsymbol{\alpha}} + \mathbf{c}^T\mathbf{z}^{\boldsymbol{\alpha}} = F^L(\boldsymbol{\alpha}).$$

Proof. Let $\boldsymbol{\alpha} = \mathcal{S}_G(\boldsymbol{\alpha})$. We define the generalized cover $(\mathbf{y}^{\boldsymbol{\alpha}}, \mathbf{z}^{\boldsymbol{\alpha}})$ of the edges of G as follows:

$$\begin{array}{rll} \forall l \in L,\ y_l^{\boldsymbol{\alpha}} = 1 & \text{iff} & |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}| \geq b_l, \\ \forall r \in R,\ y_r^{\boldsymbol{\alpha}} = 1 & \text{iff} & |\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| > b_r, \\ \forall lr \in E,\ z_{lr}^{\boldsymbol{\alpha}} = 1 & \text{iff} & \alpha_{l\to r} = \alpha_{r\to l} = c_{lr} \text{ and } y_l^{\boldsymbol{\alpha}} = 0 \\ & & \text{(and automatically } y_r^{\boldsymbol{\alpha}} = 0 \text{ as well).} \end{array}$$

First-of-all, we should check that $(\mathbf{y}^{\boldsymbol{\alpha}}, \mathbf{z}^{\boldsymbol{\alpha}})$ is indeed a generalized cover of the edges of G. Suppose $l \sim r$ are two neighboring vertices in L and R respectively. If $y_l^{\boldsymbol{\alpha}} = 0$, then $|\boldsymbol{\alpha}_{\overrightarrow{\partial l}}| < b_l$ and thus $\alpha_{l\to r} = \left[b_l - |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}| + \alpha_{r\to l}\right]_0^{c_{lr}}$ satisfies either $\alpha_{l\to r} > \alpha_{r\to l}$ or $\alpha_{l\to r} = \alpha_{r\to l} = c_{lr}$. The former implies $y_r^{\boldsymbol{\alpha}} = 1$ (because $\alpha_{r\to l} = \left[b_r - |\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| + \alpha_{l\to r}\right]_0^{c_{lr}} < \alpha_{l\to r}$ is only possible if $|\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| > b_r$) and the latter means $y_r^{\boldsymbol{\alpha}} = 0$ and $z_{lr}^{\boldsymbol{\alpha}} = 1$. Next, and similarly, if $y_r^{\boldsymbol{\alpha}} = 0$, then $|\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| \leq b_r$ and $\alpha_{r\to l} =$

$\left[b_r - |\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| + \alpha_{l\to r}\right]_0^{c_{lr}} \geq \alpha_{l\to r}$. Again, either $\alpha_{l\to r} < c_{lr}$, in which case $y_l^{\boldsymbol{\alpha}} = 1$ necessarily, or $\alpha_{l\to r} = c_{lr} = \alpha_{r\to l}$ and then $y_l^{\boldsymbol{\alpha}} = 1$ or $z_{lr}^{\boldsymbol{\alpha}} = 1$. It shows that $(\mathbf{y}^{\boldsymbol{\alpha}}, \mathbf{z}^{\boldsymbol{\alpha}})$ is a generalized cover of the edges of G.

Now, we want to compute the weight $\mathbf{b}^T\mathbf{y}^{\boldsymbol{\alpha}} + \mathbf{c}^T\mathbf{z}^{\boldsymbol{\alpha}}$ of the generalized cover. We have

$$\mathbf{b}^T\mathbf{y}^{\boldsymbol{\alpha}} + \mathbf{c}^T\mathbf{z}^{\boldsymbol{\alpha}} = \sum_{l\in L} b_l \mathbf{1}(y_l^{\boldsymbol{\alpha}} = 1) + \sum_{r\in R} b_r \mathbf{1}(y_r^{\boldsymbol{\alpha}} = 1) + \sum_{lr\in E} c_{lr}\mathbf{1}(z_{lr}^{\boldsymbol{\alpha}} = 1).$$

On the other side, we have

$$F^L(\boldsymbol{\alpha}) = \sum_{l\in L} b_l \mathbf{1}(y_l^{\boldsymbol{\alpha}} = 1) + \sum_{l\in L} |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}|\mathbf{1}(y_l^{\boldsymbol{\alpha}} = 0) + \sum_{r\in R,\, |\mathbf{c}_{\partial r}|>b_r} \left(b_r - |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\right)^+.$$

For $r \in R$ such that $|\mathbf{c}_{\partial r}| > b_r$, we can notice that $\left(b_r - |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\right)^+ > 0 \Leftrightarrow |\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| > b_r$. Hence, it follows

$$\sum_{r\in R,\, |\mathbf{c}_{\partial r}|>b_r} \left(b_r - |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\right)^+ = \sum_{r\in R} \left(b_r - |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\right) \mathbf{1}(y_r^{\boldsymbol{\alpha}} = 1).$$

It remains only to show that

$$\sum_{lr\in E} c_{lr}\mathbf{1}(z_{lr}^{\boldsymbol{\alpha}} = 1) = \sum_{l\in L} |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}|\mathbf{1}(y_l^{\boldsymbol{\alpha}} = 0) - \sum_{r\in R} |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\mathbf{1}(y_r^{\boldsymbol{\alpha}} = 1).$$

If $y_r^{\boldsymbol{\alpha}} = 1$, then $|\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| > b_r$ and we have $\alpha_{r\to l} = \left[b_r - |\boldsymbol{\alpha}_{\overrightarrow{\partial r}}| + \alpha_{l\to r}\right]_0^{c_{lr}}$ equal to 0 or strictly smaller than $\alpha_{l\to r}$. Hence,

$$\sum_{r\in R} |\boldsymbol{\alpha}_{\overleftarrow{\partial r}}|\mathbf{1}(y_r^{\boldsymbol{\alpha}} = 1) = \sum_{l\sim r} \alpha_{r\to l}\mathbf{1}(\alpha_{l\to r} > \alpha_{r\to l}).$$

Similarly, if $y_l^{\boldsymbol{\alpha}} = 0$, then $|\boldsymbol{\alpha}_{\overrightarrow{\partial l}}| < b_l$ and we have $\alpha_{l\to r} = \left[b_l - |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}| + \alpha_{r\to l}\right]_0^{c_{lr}}$ equal to c_{lr} or stricly larger than $\alpha_{r\to l}$. Hence,

$$\begin{aligned}&\textstyle\sum_{l\in L} |\boldsymbol{\alpha}_{\overrightarrow{\partial l}}|\mathbf{1}(y_l^{\boldsymbol{\alpha}} = 0)\\ &= \textstyle\sum_{l\sim r} \alpha_{r\to l}\mathbf{1}(\alpha_{l\to r} > \alpha_{r\to l}) + \sum_{l\sim r} c_{lr}\mathbf{1}(\alpha_{l\to r} = \alpha_{r\to l} = c_{lr} \text{ and } y_l^{\boldsymbol{\alpha}} = 0).\end{aligned}$$

The condition in the last indicator function is equivalent to $z_{lr}^{\boldsymbol{\alpha}} = 1$, which completes the proof of Lemma 2.5. □

Lemma 2.6 (To a maximum c-capacitated b-matching a fixed-point)**.** *Let* $\mathbf{x}$ *be a maximum size c-capacitated b-matching of G. We can obtain from* $\mathbf{x}$ *a fixed point* $\boldsymbol{\alpha}^{\mathbf{x}} = \mathcal{S}_G(\boldsymbol{\alpha}^{\mathbf{x}})$ *such that*

$$F^L(\boldsymbol{\alpha}^{\mathbf{x}}) = |\mathbf{x}|.$$

Proof. Let $\mathbf{x}$ be a maximum size c-capacitated b-matching of G. The proof will proceed in many steps. First-of-all, we define from $\mathbf{x}$ a sequence of integer messages $\boldsymbol{\alpha}^k \in \mathbb{N}^{\overrightarrow{E}}$, for $k \in \mathbb{N}$, as follows:

$$\begin{aligned}\forall \overrightarrow{e} \in \overrightarrow{E}, \qquad & \alpha^0_{\overrightarrow{e}} = x_e;\\ \forall k \in \mathbb{N}, \qquad & \boldsymbol{\alpha}^{k+1} = \mathcal{S}_G(\boldsymbol{\alpha}^k).\end{aligned}$$

We will first show that this sequence converges. We will use the notion of alternating and augmenting paths for $\mathbf{x}$: for $t \in \mathbb{N}^*$, an alternating path $(\overrightarrow{e}_1, \dots, \overrightarrow{e}_t)$ for $\mathbf{x}$, with $\overrightarrow{e}_1 = \overrightarrow{v_1 v_2}$, is an oriented path of G (not necessarily edge-disjoint) such that

$$\begin{aligned} |x_{\partial v_1}| &< b_{v_1}, \\ x_{e_k} &< c_{e_k}, \text{ for all odd } k, \\ x_{e_k} &> 0, \text{ for all even } k. \end{aligned}$$

An augmenting path for $\mathbf{x}$ is an alternating path $(\overrightarrow{e}_1, \dots, \overrightarrow{e}_t)$ for $\mathbf{x}$, with $\overrightarrow{e}_t = \overrightarrow{v_t v_{t+1}}$, such that t is odd and $|x_{\partial v_{t+1}}| < b_{v_{t+1}}$. If there exists an augmenting path for $\mathbf{x}$, then $\mathbf{x}$ can be augmented in the sense that there exists a c-capacitated b-matching $\mathbf{x}'$ of G with $|\mathbf{x}'| > |\mathbf{x}|$. Hence, as here $\mathbf{x}$ is of maximum size, we know that there exists no augmenting path for $\mathbf{x}$.

Lemma 2.7. *For any $\overrightarrow{e} \in \overrightarrow{E}$ and $k \in \mathbb{N}$, $\alpha^k_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}}$ iff there exist an alternating path for $\mathbf{x}$ of odd length at most k that terminates with $\overrightarrow{e}$; $\alpha^k_{\overrightarrow{e}} < \alpha^0_{\overrightarrow{e}}$ iff there exist an alternating path for $\mathbf{x}$ of even length at most k that terminates with $\overrightarrow{e}$.*

As a consequence, for all $v \in V$, we have that $\boldsymbol{\alpha}^k_{\overrightarrow{\partial v}}$ has its components either all non-increasing or all non-decreasing in k.

Proof. The proof is by induction. The statement is void for $k = 0$. For $k = 1$ and $\overrightarrow{e} = \overrightarrow{vu}$, we need only consider the case $\alpha^1_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}} = \alpha^0_{\overleftarrow{e}} = x_e$, which implies immediately $x_e < c_e$ and $|x_{\overrightarrow{\partial v}}| < b_u$, thus the statement holds also for $k = 1$. Suppose that the statement is true for all $k \leq K > 0$.

First suppose also that there exists $k, k' \leq K$, $v \in V$ and two (potentially identical) neighbors u and w of u such that $\alpha^k_{u \to v} > \alpha^0_{u \to v}$ and $\alpha^{k'}_{w \to v} < \alpha^0_{w \to v}$, for $k, k' \leq K$. Then, we can connect the two alternating paths, $(\overrightarrow{e}^u_1, \dots, \overrightarrow{e}^u_t)$ of odd length with $\overrightarrow{e}^u_t = \overrightarrow{uv}$ and $(\overrightarrow{e}^w_1, \dots, \overrightarrow{e}^w_{t'})$ of even length with $\overrightarrow{e}^w_{t'} = \overrightarrow{wv}$, to obtain an augmenting path $(\overrightarrow{e}^u_1, \dots, \overrightarrow{e}^u_t, \overleftarrow{e}^w_{t'}, \dots, \overleftarrow{e}^w_1)$ for $\mathbf{x}$, impossible.

Hence, for all $v \in V$, we have either $\boldsymbol{\alpha}^k_{\overrightarrow{\partial v}} \geq \boldsymbol{\alpha}^0_{\overrightarrow{\partial v}}$ for all $k \leq K$ or $\boldsymbol{\alpha}^k_{\overrightarrow{\partial v}} \leq \boldsymbol{\alpha}^0_{\overrightarrow{\partial v}}$ for all $k \leq K$. A very quick induction further shows that, for all $\overrightarrow{e} \in \overrightarrow{E}$, the sequence $(\alpha^k_{\overrightarrow{e}})_{k \leq K}$ is monotone. Indeed, for $\overrightarrow{e}$ outgoing from v and assuming the sequences $(\boldsymbol{\alpha}^k_{\overrightarrow{\partial v}})_{k \leq t}$ are non-increasing (resp. non-decreasing) for some $t \leq K$, $\mathcal{S}_{\overrightarrow{e}}$ is non-increasing so the sequence $(\boldsymbol{\alpha}^k_{\overrightarrow{e}})_{1 \leq k \leq t+1}$ is non-decreasing (resp. non-increasing). In the case $(\boldsymbol{\alpha}^k_{\overrightarrow{e}})_{1 \leq k \leq t+1}$ non-increasing, as $\alpha^1_{\overrightarrow{e}} \geq \alpha^0_{\overrightarrow{e}}$ (because $|\boldsymbol{\alpha}^0_{\overrightarrow{\partial v}}| = |\mathbf{x}_{\partial v}| \leq b_v$), we immediately get that $(\boldsymbol{\alpha}^k_{\overrightarrow{e}})_{k \leq t+1}$ is non-decreasing. In the case $(\boldsymbol{\alpha}^k_{\overrightarrow{e}})_{1 \leq k \leq t+1}$ strictly decreasing, we only need to show that $\alpha^1_{\overrightarrow{e}} = \alpha^0_{\overrightarrow{e}}$. We have that $(\boldsymbol{\alpha}^k_{\overrightarrow{\partial e}})_{k \leq t}$ is non-decreasing in every component and strictly increasing in some. Therefore, there exists $\overrightarrow{e}' \in \overrightarrow{\partial e}$ such that there is an odd-length alternating path for $\mathbf{x}$ terminating with $\overrightarrow{e}'$. If $\alpha^1_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}}$, then $|\boldsymbol{\alpha}^0_{\overrightarrow{\partial v}}| = |\mathbf{x}_{\partial v}| < b_v$, and the odd-length alternating path we found is thus an augmenting path for $\mathbf{x}$, impossible.

Then, consider some $\overrightarrow{e} = \overrightarrow{uv} \in \overrightarrow{E}$ such that $\alpha^{K+1}_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}}$. If $\alpha^K_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}}$, then it is already known that there is some odd-length alternating path for $\mathbf{x}$ terminating with $\overrightarrow{e}$. If else $\alpha^K_{\overrightarrow{e}} = \alpha^0_{\overrightarrow{e}}$, and thus in particular $\alpha^1_{\overrightarrow{e}} = x_e$. We have $x_e = \alpha^1_{\overrightarrow{e}} = \left[b_u - |x_{\overrightarrow{\partial u}}| + x_e\right]_0^{c_e} < c_e$. Now, $\alpha^{K+1}_{\overrightarrow{e}} > \alpha^K_{\overrightarrow{e}}$, so there must be some $w \sim u$, $w \neq v$

such that $\alpha^K_{w\to u} < \alpha^{K-1}_{w\to u}$. Thus, we can find an alternating path for $\mathbf{x}$ of even length at most K terminating with $\overrightarrow{wu}$, to which we can add $\overrightarrow{e} = \overrightarrow{uv}$ to obtain an alternating path for $\mathbf{x}$ of odd length at most $K+1$ terminating with $\overrightarrow{e}$ (because also $x_e < c_e$).

Conversely, suppose there exists an alternating path $(\overrightarrow{e}_1, \dots, \overrightarrow{e}_t)$ for $\mathbf{x}$ of odd length $t \leq K+1$ terminating with $\overrightarrow{e} = \overrightarrow{e}_t = \overrightarrow{uv}$, hence $x_e < c_e$. If $t \leq K$, then we are done. Else, $t = K+1$ and necessarily $t \geq 3$. Thus, $(\overrightarrow{e}_1, \dots, \overrightarrow{e}_K)$ is an alternating path for $\mathbf{x}$ of even length terminating with some edge $\overrightarrow{e}_K = \overrightarrow{wu}$, with $w \neq v$. We must have $\alpha^K_{w\to u} < \alpha^0_{w\to u}$, and thus also $\boldsymbol{\alpha}^K_{\overrightarrow{\partial u}} \leq \boldsymbol{\alpha}^0_{\overrightarrow{\partial u}}$. Then, we have $\alpha^{K+1}_{\overrightarrow{e}} = \left[b_u - \sum_{w\sim u,\, w\neq v} \alpha^K_{w\to u}\right]_0^{c_e} \geq \alpha^1_{\overrightarrow{e}}$ with strict inequality if $\alpha^1_{\overrightarrow{e}} < c_e$. If $\alpha^1_{\overrightarrow{e}} > \alpha^0_{\overrightarrow{e}} = x_e$, we are done; else, $\alpha^1_{\overrightarrow{e}} = x_e < c_e$ and we are done as well.

The proof that $\alpha^{K+1}_{\overrightarrow{e}} < \alpha^0_{\overrightarrow{e}}$ is equivalent to the existence of an even-length alternating path for $\mathbf{x}$ of length at most $K+1$ is completely similar and completes the proof of Lemma 2.7. □

As every component of $\boldsymbol{\alpha}^k$ is either non-increasing in k or non-decreasing in k, it is clear that $\left(\boldsymbol{\alpha}^k\right)_{k\in\mathbb{N}}$ converges to some $\boldsymbol{\alpha}^{\mathbf{x}} \in \mathbb{N}^{\overrightarrow{E}}$ which satisfies $\boldsymbol{\alpha}^{\mathbf{x}} = \mathcal{S}_G(\boldsymbol{\alpha}^{\mathbf{x}})$. Furthermore, the interpretation provided by Lemma 2.7 still holds for $\boldsymbol{\alpha}^{\mathbf{x}}$, and thus also, for all $v \in V$, either $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial v}} \geq \mathbf{x}_{\partial v}$ or $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial v}} \leq \mathbf{x}_{\partial v}$. We will now show that $F^L(\boldsymbol{\alpha}^{\mathbf{x}}) = |\mathbf{x}| = \sum_{l\in L} |\mathbf{x}_{\partial l}|$, which will complete the proof of the Lemma 2.6.

Consider $l \in L$ such that $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| > |\mathbf{x}_{\partial l}|$. This means $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}} \geq \mathbf{x}_{\partial l}$ and there exists $r \sim l$ such that $\alpha^{\mathbf{x}}_{r\to l} > x_{lr}$. Then, we must have $\alpha^{\mathbf{x}}_{l\to r} \leq x_{lr}$, otherwise we would have two alternating paths for $\mathbf{x}$ of odd length and terminating with $\overrightarrow{lr}$ and $\overrightarrow{rl}$, from which we would obtain an augmenting path for $\mathbf{x}$. We have $\alpha^{\mathbf{x}}_{r\to l} > \alpha^{\mathbf{x}}_{l\to r} = \left[b_l - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| + \alpha^{\mathbf{x}}_{r\to l}\right]_0^{c_{lr}}$, which implies $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| > b_l$. Suppose towards a contradiction that $|\mathbf{x}_{\partial l}| < b_l$, then $\boldsymbol{\alpha}^1_{\overrightarrow{\partial l}} \geq \mathbf{x}_{\partial l}$ and there exists $r' \sim l$ such that $\alpha^1_{l\to r'} > x_{lr'}$, so that $\alpha^{\mathbf{x}}_{l\to r'} > x_{lr'}$. Again, we must have $\alpha^{\mathbf{x}}_{r'\to l} \leq x_{lr'}$ otherwise we could find an augmenting path for $\mathbf{x}$, and so $\alpha^{\mathbf{x}}_{r'\to l} < \alpha^{\mathbf{x}}_{l\to r'}$, which implies $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| < b_l$, a contradiction. So, $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| > |\mathbf{x}_{\partial l}|$ implies $|\mathbf{x}_{\partial l}| = b_l$.

It remains only to check that

$$\sum_{l\in L,\, |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}|<|\mathbf{x}_{\partial l}|} |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| + \sum_{r\in R,\, |\mathbf{c}_{\partial r}|>b_r} \left(b_r - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overleftarrow{\partial r}}|\right)^+ = \sum_{l\in L} |\mathbf{x}_{\partial l}| \mathbf{1}(|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| < |\mathbf{x}_{\partial l}|).$$

Consider now $r \in R$. We can easily check that the event that both $|\mathbf{c}_{\partial r}| > b_r$ and $\left(b_r - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overleftarrow{\partial r}}|\right)^+ > 0$ is equivalent to the event $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}}| > b_r$. For such an r, we have of course $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}}| > |\mathbf{x}_{\partial r}|$, and thus $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}} \geq \mathbf{x}_{\partial r}$. Furthermore, as seen above, $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}}| > |\mathbf{x}_{\partial r}|$ implies $|\mathbf{x}_{\partial r}| = b_r$. Then, we can write $\left(b_r - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overleftarrow{\partial r}}|\right)^+ = \sum_{l\in L} (x_{lr} - \alpha^{\mathbf{x}}_{r\to l})^+$.

Conversely, for any $r \in R$, if there exists $l \sim r$ such that $(x_{lr} - \alpha^{\mathbf{x}}_{r\to l})^+ > 0$, then, as before, we must have $\alpha^{\mathbf{x}}_{l\to r} \geq x_{lr}$, else we can find an augmenting path for $\mathbf{x}$. Now, $\alpha^{\mathbf{x}}_{l\to r} > \alpha^{\mathbf{x}}_{r\to l} = \left[b_r - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}}| + \alpha^{\mathbf{x}}_{l\to r}\right]_0^{c_{lr}}$ implies $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial r}}| > b_r$. Finally,

$$\sum_{r\in R,\, |\mathbf{c}_{\partial r}|>b_r} \left(b_r - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overleftarrow{\partial r}}|\right)^+ = \sum_{l\in L,\, r\in R} (x_{lr} - \alpha^{\mathbf{x}}_{r\to l})^+ .$$

Consider now $l \in L$ such that $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| < |\mathbf{x}_{\partial l}|$. It means $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}} \leq \mathbf{x}_{\partial l}$ and thus $|\mathbf{x}_{\partial l}| - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| = \sum_{r \in R} (x_{lr} - \alpha^{\mathbf{x}}_{r \to l})^+$.

Conversely, for any $l \in L$, if there exists $r \sim l$ such that $(x_{lr} - \alpha^{\mathbf{x}}_{r \to l})^+ > 0$, then $\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}} \leq \mathbf{x}_{\partial l}$ and thus $|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| < |\mathbf{x}_{\partial l}|$. Finally,

$$\sum_{l \in L} \left(|\mathbf{x}_{\partial l}| - |\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| \right) \mathbf{1}(|\boldsymbol{\alpha}^{\mathbf{x}}_{\overrightarrow{\partial l}}| < |\mathbf{x}_{\partial l}|) = \sum_{l \in L,\, r \in R} (x_{lr} - \alpha^{\mathbf{x}}_{r \to l})^+ ,$$

which completes the proof of Lemma 2.6. □

Lemmas 2.5 and 2.6 together imply Proposition 2.11. □

Chapter 3

Cuckoo hashing thresholds

A lot of interest has recently arisen in the analysis of multiple-choice *cuckoo hashing* schemes. In this context, a main performance criterion is the load threshold under which the hashing scheme is able to build a valid hashtable with high probability in the limit of large systems. Various techniques have successfully been used to answer this question (differential equations, combinatorics, cavity method) for increasing levels of generality of the model. However, the hashing scheme analysed so far is quite utopic in that it requires to generate a lot of independent, fully random choices. Schemes with reduced randomness exists, such as *double hashing* –which is expected to provide similar asymptotic results as the ideal scheme–, yet they have been more resistant to analysis so far. As a first contribution, we extend even more the range of setups in which the load thresholds are known, relying on a link with *capacitated matchings* which can be studied via the cavity method. In addition, we point out that this approach quite naturally extends to the analysis of double hashing and allows us to compute the corresponding threshold. The path followed here is to show that the graph induced by the double hashing scheme has the same local weak limit as the one obtained with full randomness.

3.1 Introduction

A *hashtable* is a data structure that maps items to buckets so as to be able to later retrieve the items or any associated data. The idea is that the item (or an associated descriptor called a key) is hashed into the index of a bucket in an array. The array contains much fewer buckets than there are potential items in the universe and the purpose of the hashing step is that it maps many items to the same bucket index. Hopefully, most of the time, only one of the items is present among those mapped to the same bucket; otherwise, a collision occurs and it needs to be resolved in some way. Cuckoo hashing is a particular scheme used to resolve such collisions.

The hashing paradigm is as follows: we are given n items and m buckets. In the most basic setting, we want to have exactly one bucket per item and at most one item per bucket. In more complex versions, multiple copies of each item must be stored, each bucket can hold many items simultaneously and there are restrictions on how many times an (item,bucket) pair can be used. Critical performance metrics for such systems are the size of the hashtable (the number of buckets) needed to hold a given number of items, and the time it takes to either retrieve or insert items. The *multiple-choice hashing* strategy is one that guaranties a constant look-up time, while requiring with high probability a hashtable of size m proportional to n. This strategy consists in pre-defining a set of $d \geq 2$ buckets for each item, which is then only allowed to pick a bucket within that set. Of course, depending on the choices of the pre-defined sets of buckets, it may be impossible to handle simultaneously some sets of items and inserting a new item may not be easy. The first issue becomes very unlikely if the hashtable contains enough buckets compared to the number of items it needs to hold, and a lot of work has actually been devoted to figuring out exactly how large the hashtable needs to be. As for the second issue, *cuckoo hashing* was proposed by Pagh and Rodler [89] as a simple, randomized way to search for a new valid assignement upon arrival of an item: one first checks whether one of the buckets pre-defined for the new item is available, in which case it suffices to pick one of these; otherwise, one of the pre-defined buckets is chosen at random and re-allocated to the new item. The same procedure is then used for the item that has just been evicted (which is thus treated as a new item).

In a fully random context, multiple-choice hashing has been shown to have good performance: if the set of d buckets pre-defined for each item are chosen independently and uniformly at random among all sets of d buckets, then there exists a threshold τ^* such that, if $n = \tau m$ with $\tau < \tau^*$, in the limit of $m, n \to \infty$ cukoo hashing will yield a valid hashtable with high probability. Furthermore, the cuckoo hashing update procedure is also quite efficient [47, 43], in that the insertion time is polylogarithmic in the system size w.h.p. A recent refinement [65] of the update algorithm also achieves constant average insertion time with high probability when items are only inserted. A major drawback of the fully random (multiple-choice) hashing scheme described above is the amount of randomness involved: the standard approach requires n independent, uniform choices of sets of d buckets among m. Phrased differently, this means roughly $d \log m$ independent random bits per item. Generating unbiased, perfectly independent random bits does not come for free [122, 109], therefore a lot of effort has been put into reducing this amount of

randomness needed (see [82] and references therein for an account of some of the directions investigated so far). A candidate alternative is *double hashing* [56, 72], which seems to have similar performances as the fully random scheme while requiring only roughly $2 \log m$ independent random bits per item: assume m is a prime number and label the m buckets with the integers from 1 to m; for each item, independently draw two random numbers $f \in \{1, \dots, m\}$ and $g \in \{1, \dots, \frac{m-1}{2}\}$; the pre-defined set of buckets associated with a couple (f, g) are the buckets labeled $f + ig \pmod m$, for $i \in \{0, \dots d-1\}$. Although the reduced randomness of the double hashing scheme is the very reason why this method could be prefered over fully random hashing, it also makes the theoretical analysis of its performance more difficult. An interesting survey [81] reviews the state of research on cuckoo hashing as of 2009 and state some related open questions at that time.

In this chapter, we consider two problems related to cuckoo hashing. The first one is to extend the current setups for which the cuckoo hashing threshold is known: the schemes for which the threshold is currently known require each item to be stored multiple times and allow each bucket to contain many items simultaneously, yet each item cannot be stored multiple times in the same bucket; we relax this last constraint and compute the threshold in the case where each (item,bucket) pair be used multiple times. The method employed relies on a characterization of maximum generalized matchings using the cavity method. Then, we turn to the analysis of double-hashing, and we show that the approach using the cavity method naturally extends to the case of double hashing: the core of the work, which is computing the asymptotic size of generalized matchings in large random graphs, need not be done again; however, one needs to check that the hashing system using double hashing has the same local characteristics as with fully-random hashing in the sense that the graphs obtained need to converge in the *local weak* sense to the same limit (see Section 1.1.2 for the definition). The remainder of this chapter is divided into two parts: in Section 3.2, we compute the cuckoo hashing threshold for the extension where (item,bucket) pair can be used multiple times, and Section 3.3 deals with the double-hashing scheme and shows this scheme with limited randomness has essentially the same load threshold as the fully-random scheme.

3.2 Cuckoo hashing threshold and hypergraph orientability

In this section, we compute the cuckoo hashing threshold for the most general model, where many copies of each item need to be stored, each bucket can hold several items, and each (item,bucket) pair can be re-used up to a certain number of times. The precise formulation of the problem is described next (as well as an alternative formulation in terms of hypergraph orientation), before proceeding to the statement of the result and its proof.

3.2.1 Hashing graph and hypergraph orientation formulation

A multiple-choice hashing system can be represented as a bipartite graph $G = (L \cup R, E)$ that we call the *hashing graph* (or *cuckoo graph*), where the left part L represents the items and the right part R the buckets; the edges of the graph indicate which buckets can be assigned to which items. To capture the more complex versions of cuckoo hashing, where one needs to store h copies of each item or where each bucket can hold at most k items at a time, we add respectively capacity constraints $\mathbf{b}_L \equiv h$ to the left-vertices (items) and $\mathbf{b}_R \equiv k$ to the right-vertices (buckets); similarly, if each (item,bucket) pair can be used at most s times, it is indicated by the edge-capacities $\mathbf{c} \equiv s$. In the case of fully random hashing, the d edges adjacent to a left-vertex can be anything, while for double hashing they are restricted to having a certain structure. Given such a hashing graph G, it is possible to build a valid hashtable if we can associate each items to h buckets (with re-use up to s times of the same (item,bucket) pair) without exceeding the capacity of any bucket, i.e., if there exists a left-perfect *capacitated matching* of G.

In addition, the items with which the hashing system is presented are random and thus the hashing graph is random, following a different distribution whether we consider fully random or double hashing. In the case of fully-random hashing, the set of pre-determined buckets for each item is a random subset of size d choosen uniformly at random among all such subsets of buckets and independently for each item. Then, the hashing graph $G_{n,m}$ is a random bipartite graph with a constant degree d for left-vertices and $\mathrm{Poi}(dn/m)$ for right-vertices, and it almost surely converges locally weakly to a bipartite unimodular Galton-Watson limit, as the size of the system increases.

An alternative description of the above setup consists in the following hypergraph orientation problem, a terminology used in [53, 70]. For $d \in \mathbb{N}^*$, a d-uniform hypergraph is a hypergraph whose hyperedges all have size d. Hyperedges represent items, and vertices buckets. We assign marks in $\{0, \dots, s\}$ to each of the endpoints of a hyperedge. For $h < d$ in $\mathbb{N}^*$, a hyperedge is said to be (h, s)-oriented if the sum of the marks at its endpoints is equal to h. The in-degree of a vertex of the hypergraph is the sum of the marks assigned to it in all its adjacent hyperedges. For a positive integer k, a (k, h, s)-orientation of a d-uniform hypergraph is an assignement of marks to all endpoints of all hyperedges such that every hyperedge is (h, s)-oriented and every vertex has in-degree at most k. If such a (k, h, s)-orientation exists, we say that the hypergraph is (k, h, s)-orientable, which means that one can build a valid hashtable. We now consider the probability space $\mathcal{H}_{m,n,d}$ of the set of all d-uniform hypergraphs with m vertices and n hyperedges, and we denote by $H_{m,n,d}$ a random sample from $\mathcal{H}_{m,n,s}$. The load threshold τ^* under which cuckoo hashing builds a valid hashtable w.h.p. is also the edge-density threshold under which hypergraphs drawn from $\mathcal{H}_{m,n,s}$ are (k, h, s)-orientable w.h.p.

3.2.2 Cuckoo hashing threshold via maximum capacitated matchings

Given a collection of m buckets and n items, the cuckoo hashing threshold τ^* is the largest value such that, for all $\tau < \tau^*$, if $n \sim \tau m$ then it is possible to build a valid

hashtable w.h.p. as $m, n \to \infty$; on the contrary, if $\tau > \tau^*$ then the probability of a valid assignement of items to buckets will tend a.s. to 0 with the system size. The threshold in the most basic schenario $d = 2, h = k = s = 1$ is equal to $1/2$, and the following extensions can be naturally considered:

- each item can choose among $d \geq 2$ random buckets [35, 42, 46];
- each bucket can hold a maximum of k items [36, 27, 41];
- each item must be replicated at least h times [53, 70];
- each (item,bucket) pair can be used a maximum of s times (not covered previously).

When confusion may be possible and we want to insist on the value of the parameters d, k, h, s, we may write $\tau^*_{d,k,h,s}$ for the cuckoo hashing threshold. In this context, we can interpret Theorem 2.1 as follows:

Theorem 3.1 (Threshold for (k, h, s)-orientability of d-uniform hypergraphs)**.** *Let d, k, h, s be positive integers such that $k, h \geq s$, $(d-1)s \geq h$ and $k + (d-2)s > h$ (i.e., at least one of the two inequalities $k \geq s$ and $(d-1)s \geq h$ is strict). We define Φ^L and Φ^R_τ by $(d, h, \{s\}) \sim \Phi^L$ and $(\mathrm{Poi}(\tau d), k, \{s\}) \sim \Phi^R_\tau$, and*

$$\tau^*_{d,k,h,s} = \sup\left\{\tau : \mathcal{M}(\Phi^L, \Phi^R_\tau) = h\right\} = \inf\left\{\tau : \mathcal{M}(\Phi^L, \Phi^R_\tau) < h\right\},$$

where $\mathcal{M}(\Phi^L, \Phi^R_\tau)$ is defined in Theorem 2.1. Then,

$$\lim_{m\to\infty} \mathbb{P}\left(H_{m,\lfloor \tau m\rfloor,d} \text{ is } (k,h,s)\text{-orientable}\right) = \begin{cases} 1 \text{ if } \tau < \tau^*_{d,k,h,s}, \\ 0 \text{ if } \tau > \tau^*_{d,k,h,s}. \end{cases}$$

This result extends those from [70], where the value of the threshold $\tau^*_{d,k,h,1}$ was computed. We now turn to the proof. The idea is that Theorem 2.1 gives that only a vanishing fraction of the hyper-edges of $H_{m,\lfloor \tau m\rfloor,d}$ may not be (h, s) oriented in the limit $m \to \infty$; this property defines a second load threshold $\widetilde{\tau}^*_{d,k,h,s} \geq \tau^*_{d,k,h,s}$. It only remains to show that the two thresholds are actually equal for $\mathcal{H}_{m,\lfloor \tau m\rfloor,s}$.

Proof. For any d-uniform hypergraph H, recall that $G = (L \cup R, E)$ is the associated bipartite hashing graph, where R contains the vertices of H and L the hyperedges of H, and that G is then a random bipartite graph with constant degree d on L and degree $\mathrm{Poi}(d|L|/|R|)$ on R. Then, for $H_{m,n,d}$ with $m \sim \tau n$, the asymptotic value of $M(G_{n,m})$ is described by Theorem 2.1. First-of-all, it is clear by coupling that $\tau \mapsto \mathcal{M}(\Phi^L, \Phi^R_\tau)$ is a non-increasing function. Theorem 2.1 immediately shows that

$$\lim_{m,n\to\infty} \frac{1}{n} M(G_{n,m}) \begin{cases} = h \text{ if } \tau < \tau^*_{d,k,h,s}, \\ < h \text{ if } \tau > \tau^*_{d,k,h,s}, \end{cases}$$

which immediately implies that $H_{m,n,d}$ is asymptotically almost surely not (k, h, s)-orientable if $\tau > \tau^*_{d,k,h,s}$.

Let now $\tau < \tau^*_{d,k,h,s}$. There may still exist $o(n)$ hyperedges which are not (h, s)-oriented; in a sense. We will then rely on specific properties of $H_{m,n,d}$ to show that

asymptotically a.s. all hyperedges are (h, s)-oriented. We follow here a similar path as in [53, 70]. It is easier to work with a different model of hypergraphs, that we call $H_{m,p,d}$, and that is essentially equivalent to the $H_{m,\lfloor \tau m \rfloor,d}$ model [66]: each possible d-hyperedge is included independently with probability p, with $p = \tau d / \binom{m-1}{d-1}$. We denote by $G_{m,p}$ the corresponding bipartite random graph.

We let $\widetilde{\tau}$ be such that $\tau < \widetilde{\tau} < \tau^*_{d,k,h,s}$, and consider the bipartite graph $G_{m,\widetilde{p}} = (L_{m,\widetilde{p}} \cup R_m, E_{m,\widetilde{p}})$ obtained from $H_{m,\widetilde{p},d}$ with $\widetilde{p} = \widetilde{\tau} d / \binom{m-1}{d-1}$. Consider a maximum capacitated matching $\widetilde{\mathbf{x}} \in \mathbb{N}^{E_{m,\widetilde{p}}}$ of $G_{m,\widetilde{p}}$. We say that a vertex of $l \in L_{m,\widetilde{p}}$ (resp. a vertex $r \in R_m$) is covered if $\sum_{e \in \partial l} \widetilde{x}_e = h$ (resp. $\sum_{e \in \partial r} \widetilde{x}_e = k$); we also say that an edge $e \in E_{m,\widetilde{p}}$ is saturated if $\widetilde{x}_e = s$.

Let l be a vertex in $L_{m,\widetilde{p}}$ that is not covered. We define $K(l)$ as the minimum subgraph of $G_{m,\widetilde{p}}$ such that:

- l belongs to $K(l)$;
- all the unsaturated edges adjacent to a vertex in $L_{m,\widetilde{p}} \cap K(l)$ belong to $K(l)$ (and thus their endpoints in R_m also belongs to $K(l)$);
- all the edges e for which $\widetilde{x}_e > 0$ and that are adjacent to a vertex in $R_m \cap K(l)$ belong to $K(l)$ (and so do their endpoints in $L_{m,\widetilde{p}}$).

The subgraph $K(l)$ defined in this way is in fact constitued of l and all the paths starting from l and alternating between unsaturated edges and edges e with $\widetilde{x}_e > 0$ (we call such a path an alternating path for $\widetilde{\mathbf{x}}$). It is then easy to see that all the vertices in $R_m \cap K(l)$ must be covered, otherwise we could obtain a strictly larger capacitated matching by applying the following change: take the path $(e_1, \dots, e_{2t+1})$ between l and an unsaturated vertex in $R_m \cap K(l)$; add 1 to each $\widetilde{x}_{e_i}$ for i odd, and remove 1 from each $\widetilde{x}_{e_i}$ for i even; all these changes are possible due to the way the edges in $K(l)$ have been chosen, and the resulting capacitated matching has size larger by 1 than $|\widetilde{\mathbf{x}}|$.

We will now show that the subgraph $K(l)$ is dense, in the sense that the average induced degree of its vertices is strictly larger than 2. We first show that all the vertices in $K(l)$ have degree at least 2. We have $(d-1)s \geq h$ and l is not covered, hence l has at least two adjacent edges in $G_{m,\widetilde{p}}$ which are not saturated, thus the degree of l in $K(l)$, written $\deg_{K(l)} l$, is at least 2. Let r be a vertex in $R_m \cap K(l)$. By definition, there exists an edge $e \in \partial r \cap K(l)$ through which r is reached from l in an alternating path, and $\widetilde{x}_e < s$. Then, because $\sum_{e \in \partial r} \widetilde{x}_e = k$ and $k \geq s$ there must be another edge e' adjacent to r such that $\widetilde{x}_{e'} > 0$; such an edge belongs to $K(l)$ and thus r is at least of degree 2 in $K(l)$. Let now l' be a vertex in $L_{m,\widetilde{p}} \cap K(l)$, $l' \neq l$. By definition, there must exist an edge $e \in \partial l' \cap K(l)$ such that $\widetilde{x}_e > 0$. Because $(d-1)s \geq h$ and $\widetilde{x}_e > 0$ there must be another edge e' adjacent to l' such that $\widetilde{x}_{e'} < s$; e' belongs to $K(l)$ and thus $\deg_{K(l)} l' \geq 2$.

Consider a path $(e_1 = (v_1 v_2), \dots, e_t = (v_t v_{t+1}))$ in $K(l)$ such that $v_1 \in L_{m,\widetilde{p}} \cap K(l)$ and any two consecutive edges in the path are distinct. We will show that at least one vertex out of $2s$ consecutive vertices along this path must have degree at least 3 in $K(l)$, by showing that $\widetilde{x}_{e_{2(i+1)+1}} < \widetilde{x}_{e_{2i+1}}$ provided $v_{2(i+1)}$ and $v_{2(i+1)+1}$ have degree 2 in $K(l)$ for all i. $v_{2(i+1)} \in R_m \cap K(l)$ must be covered, so if $\deg_{K(l)} v_{2(i+1)} = 2$ we must have $\widetilde{x}_{e_{2(i+1)}} = k - \widetilde{x}_{e_{2i+1}}$. Then, if $\deg_{K(l)} v_{2(i+1)+1} = 2$, all the edges

adjacent to $v_{2(i+1)+1}$ except $e_{2(i+1)}$ and $e_{2(i+1)+1}$ must be saturated, thus we must also have $(d-2)s + \widetilde{x}_{e_{2(i+1)}} + \widetilde{x}_{e_{2(i+1)+1}} \leq h$. This immediately yield $\widetilde{x}_{e_{2(i+1)+1}} + \{k + (d-2)s - h\} \leq \widetilde{x}_{e_{2i+1}}$, and thus $\widetilde{x}_{e_{2(i+1)+1}} < \widetilde{x}_{e_{2i+1}}$ as claimed. But $\widetilde{x}_{e_{2i+1}} < s$ and so $\widetilde{x}_{e_{2i+2s+1}} \leq -1$ if the hypothesis that all the vertices encountered meanwhile have degree 2 in $K(l)$ were correct, which is thus not possible. Note that we did not need to assume that the path considered is vertex-disjoint, hence in particular it is not possible that $K(l)$ is reduced to a single cycle.

We will now count vertices and edges of $K(l)$ in a way that clearly shows that the number of edges in $K(l)$ is at least η times its number of vertices, with $\eta > 1$. We can always see $K(l)$ as a collection P of edge-disjoint paths, with all vertices interior to a path of degree 2 in $K(l)$ and the extremal vertices of a path having degree at least 3 in $K(l)$. To form $K(l)$ we would simply need to merge the extremal vertices of some of these paths. We have shown before that each path in P has at most $2s$ vertices. Let $\rho = (e_1 = (v_1 v_2), \ldots, e_t = (v_t v_{t+1}))$ be a path in P, we let $\theta_E(\rho) = t$ be the number of edges in ρ and $\theta_V(\rho) = \sum_{e_i \in \rho} \frac{1}{\deg_{K(l)} v_i} + \frac{1}{\deg_{K(l)} v_{i+1}}$ be a partial count of the vertices in ρ (all the interior vertices are counted as 1 but the extremal vertices are only partially counted in $\theta_V(\rho)$, as they belong to many different paths). We have $\theta_V(\rho) = t - 1 + \frac{1}{\deg_{K(l)} v_1} + \frac{1}{\deg_{K(l)} v_{t+1}} \leq t - 1 + \frac{2}{3}$. Hence,

$$\frac{\theta_E(\rho)}{\theta_V(\rho)} \geq \frac{t}{t - 1 + \frac{2}{3}} \geq \frac{1}{1 - \frac{1}{6s}} > 1.$$

Furthermore, it is easy to see that

$$\begin{aligned} \sum_{\rho \in P} \theta_E(\rho) &= \text{number of edges in } K(l), \\ \sum_{\rho \in P} \theta_V(\rho) &= \text{number of vertices in } K(l), \end{aligned}$$

which shows that the number of edges in $K(l)$ is at least $\eta = \frac{1}{1 - \frac{1}{6s}} > 1$ times the number of vertices in $K(l)$.

Now, it is classical that any subgraph of a sparse random graph like $G_{m,\widetilde{p}}$ with a number of edges equal to at least $\eta > 1$ times its number of vertices must contain at least a fraction $\epsilon > 0$ of the vertices of $G_{m,\widetilde{p}}$, with probability tending to 1 as $n \to \infty$ (see [66, 53]). Therefore, $K(l)$ contains at least a fraction $\epsilon' > 0$ of the vertices in $L_{m,\widetilde{p}}$.

There exists a natural coupling between $H_{m,p,d}$ and $H_{m,\widetilde{p},d}$: we can obtain $H_{m,p,d}$ from $H_{m,\widetilde{p},d}$ by removing independently each hyperedge with probability $\widetilde{p} - p > 0$. This is equivalent to removing independently with probability $\widetilde{p} - p$ each vertex in $L_{m,\widetilde{p}}$. We let $\mathrm{gap}_{m,\widetilde{p}} = h|L_{m,\widetilde{p}}| - M(G_{m,\widetilde{p}}) = o(m)$. For any uncovered vertex l in $L_{m,\widetilde{p}}$ we can construct a subgraph $K(l)$ as above. If we remove a vertex l' in $L_{m,\widetilde{p}} \cap K(l)$ for such a l, then either this vertex l' is itself uncovered, and then $\mathrm{gap}_{m,\widetilde{p}}$ is decreased by at least 1, or l' is covered and then it must belong to an alternating path starting from l and we can construct a new capacitated matching with size equal to that of $\widetilde{x}$ and in which l' is uncovered and there is one more unit of weight on one of the edges adjacent to l, hence removing l' will also reduce

$\text{gap}_{m,\widetilde{p}}$ by 1. We proceed as follows: we attach independently to each hyperedge l of $H_{m,\widetilde{p},d}$ a uniform $[0,1]$ random variable U_l. To obtain $H_{m,p,d}$ we remove all hyperedges l such that $U_l \leq \widetilde{p} - p$. This can be done sequentially by removing at each step the hyperedge corresponding to the lowest remaining U_l. Then, at each step, assuming there are still uncovered vertices l in $L_{m,\widetilde{p}}$ we can consider the union K of the subgraphs $K(l)$, which has size at least $\epsilon'|L_{m,\widetilde{p}}| \geq \epsilon'\lfloor \tau m \rfloor$. Hence, with positive probability the hyperedge removed will decrease the value of $\text{gap}_{m,\widetilde{p}}$. By Chernoff's bound, the number of hyperedges removed is at least $\lfloor \tau m \rfloor \frac{\widetilde{p}-p}{2}$ with high probability as $m \to \infty$, therefore $\text{gap}_{m,\widetilde{p}}$ will reach 0 with high probability as $m \to \infty$ before we remove all the hyperedges that should be removed. Hence, $H_{m,p,d}$ (and thus $H_{m,\lfloor \tau m \rfloor,d}$) is (k,h,s)-orientable a.a.s. □

3.3 An analysis of double hashing

In this section, we consider the double hashing scheme and show that the same approach via the cavity method can be used to compute its load threshold. More precisely, the process considered to obtain the load threshold for a particular hashing model expected to perform as fully-random hashing can be decomposed in three steps:

1. One shows that the hashing graph converges in the local weak sense to a certain limit. Typically, this limit would be a bipartite unimodular Galton-Watson distribution.

2. Theorem 2.1 then yields the asymptotic size of capacitated matchings –and thus the value of $\widetilde{\tau}^*$– for the hashing graph model of interest.

3. Finally, a step similar to the proof of Theorem 3.1 above needs to be performed to show that $\tau^* = \widetilde{\tau}^*$.

Clearly, the second step is automatic. In this section, we perform the first step for the double hashing model. The only missing part to obtain the double hashing threshold is then a property similar to that taken from [66] in the proof of Theorem 3.1 (about subgraphs with at least $\eta > 1$ times more edges than vertices containing asymptotically a.s. at least a fraction ϵ of the vertices of the graph) for the double hashing model. We do not attempt to address this issue here, as we believe computing the threshold $\widetilde{\tau}^*$ is already enough to show the potential of the approach used.

In this section, we first explain more formally how to construct the double hashing graph, then we state the result about the local weak convergence of such graphs and explain how the proof proceeds by analysing a two-step Breadth-First-Search (BFS) exploration of the hashing graph. Finally, we give the proof of the result.

3.3.1 Double hashing graph

For our analysis of double hashing, it is convenient to see its hashing graph as obtained in the following manner, where we merge all the items which are given

the same choices of buckets and keep track of the multiplicity of the vertices: we start from the bipartite graph $\widetilde{G} = (\widetilde{L} \cup R, E)$ which contains all the information about the choices of buckets offered to every item that can potentially be inserted in the hashtable. We have $|R| = m$ and $|\widetilde{L}| = \binom{m}{2}$, with a left-vertex in $\widetilde{G}$ for each different neighborhood (different choice of buckets) allowed by the local structure constraints of the hashing technique. If we label the vertices in R with the integers from 1 to m and the vertices in $\widetilde{L}$ with the couples (f, g) with $f \in \{1, \dots, m\}$ and $g \in \{1, \dots, \frac{m-1}{2}\}$, there is an edge in $\widetilde{G}$ between vertex $r \in R$ and $l = (f, g) \in \widetilde{L}$ if there exists $i \in \{0, \dots, d-1\}$ such that $r = f + ig \pmod m$. To build the hashing graph, f and g are drawn independently at random for each of the n items, which corresponds exactly to a sampling *with replacement* of n left-vertices in $\widetilde{L}$. We denote by $L' \subset \widetilde{L}$ the set of left-vertices obtained in this way, and by $Z'_l \sim \mathrm{Bin}(n, 1/\binom{m}{2})$ the multiplicity of the left-vertices l. The induced graph $G' = \widetilde{G}[L' \cup R]$ is exactly the graph obtained via double-hashing.

As we are interested in the regime $n = \lfloor \tau m \rfloor \to \infty$, it is equivalent for our problem to suppose $n \sim \mathrm{Poi}(\tau m)$ instead. Indeed, the $\mathrm{Poi}(\tau m)$ random variable is concentrated around τm with only logarithmic fluctuations for $m \to \infty$, therefore the existence and the value of a load threshold τ^* are identical in the two models. Thus, we consider instead the set L obtained by taking $Z_l \sim \mathrm{Poi}(\frac{2\tau}{m-1})$ copies of each vertex in $\widetilde{L}$, independently for each vertex in $\widetilde{L}$, and keeping in L only those vertices with at least one copy, as we did before for L'. We will focus mainly on the random graph $G = \widetilde{G}[L \cup R]$; we denote by G_m a sample of this random graph for a particular value of m (τ being fixed and $n \sim \mathrm{Poi}(\tau m)$).

3.3.2 Main Results and Overview of the Proof

The proof consists in similar steps as in [20], with the difference that in our case the graphs are bipartite and have a more constrained local structure.

Let us call $G_m = (L_m \cup R_m, E_m)$ the random graph obtained at finite m and let ρ_{G_m} be the distribution of $[G_m, r_0]$, where r_0 is a random root in R_m; let $\overline{\rho}_m$ be the average of ρ_{G_m} over the random graph G_m. Let also $\rho = \mathcal{GW}(\mathrm{Poi}(\tau d), d)$ be the law of a bipartite unimodular Galton-Watson tree with laws $\mathrm{Poi}(\tau d)$ and $\mathrm{Deterministic}(d)$, as defined in Section 1.1.3. In a first step, we will show that $\overline{\rho}_m$ converges weakly towards ρ as $m \to \infty$. Then, we will show that ρ_{G_m} is concentrated around $\overline{\rho}_m$ so that ρ_{G_m} almost surely converges weakly towards ρ. Explained differently, there are two sources of randomness involved in the sampling of a rooted graph $[G_m, r_0]$: the first one is artificial and is due to r_0 being a random right-vertex. It has been introduced to turn any fixed graph into a distribution over rooted graphs, which then allows us to consider weak convergence of a sequence of such distributions. The second one is inherent to the hashing scheme used: it comes from the fact the couple (f, g) determining the choice of buckets offered to each item is chosen at random upon arrival of the item, and it turns G_m itself into a random graph, in a way described in Section 1.1.2. We first show that the measure ρ_m obtained by averaging over both sources of randomness at the same time converges weakly towards the law ρ of a two-step unimodular Galton-Watson, before showing that the averaging over the random choices offered to the items is not required for the convergence to hold,

which means we do not need to average over different realization of a large hashtable for the results to hold.

We will thus show the following results: the first one is the intermediate result where we show local weak convergence with both types of averaging together.

Proposition 3.1. *The average distribution $\overline{\rho}_m$ of the double hashing random rooted graph $[G_m, r_0]$ rooted at a random vertex $r_0 \in R_m$ converges weakly towards the distribution of a two-step unimodular Galton-Watson tree. In other words, we have*

$$\overline{\rho}_m \rightsquigarrow \rho.$$

The result above, together with some results on concentration of measure, leads to our main theorem, which is almost sure local weak convergence of the hashing graph:

Theorem 3.2. *The distribution ρ_{G_m} of the double hashing random graph G_m almost surely converges weakly towards the distribution of a two-step unimodular Galton-Watson tree. In other words,*

$$\rho_{G_m} \rightsquigarrow \rho \ \ a.s.$$

Finally, the announced result on the load threshold of double hashing is contained in the following corollary, which is now a direct consequence of Theorem 3.2 and Theorem 2 from [70].

Corollary 3.1. *The load threshold τ^* under which all items can be inserted in the hashtable for fully random hashing w.h.p. is also the load threshold under which all n items but $o(n)$ can be inserted for double hashing w.h.p.*

In the next section, we introduce a main tool for our proofs: a two-step breadth-first search (BFS) exploration of the hashing graph G_m; and illustrate its analysis through the simple example of upper-bounding the number of short cycles in G_m.

3.3.3 Breadth-First Search Exploration

When it is clear from the context, we simply write G instead of G_m. For any $k \in \mathbb{N}^*$ and any graph G^*, we let $\mathcal{N}^k_{G^*}(v)$ denote the k-hop neighborhood of vertex v in the graph G^*; to ease the notation, when $G^* = G$ we simply write $\mathcal{N}^k(v)$, and when $k = 1$ we write $\mathcal{N}_{G^*}(v)$.

We consider the breadth-first search (BFS) exploration of the graph G, starting from a random right-vertex $r_0 \in R$. At each step, we select an active right-vertex and explore all its left-neighbors as well as their own right-neighbors. We define the sets C^R_t and C^L_t of connected right- and left-vertices at time t, the list of active right-vertices A^R_t, and the sets of unexplored right- and left-vertices U^R_t and U^L_t. We have

$$\begin{array}{rcl} C^R_0 = C^L_0 &=& \varnothing, \\ A^R_0 &=& (r_0), \\ U^R_0 &=& R \setminus \{r_0\}, \\ U^L_0 &=& \widetilde{L}. \end{array}$$

At each step t, we proceed as follows:

- we let r_t be the first element in the list A_t^R (so that it is the closest vertex to r_0 in A_t^R);
- if $A_t^R \neq \varnothing$, we do the following updates:

$$\begin{array}{rcl} C_{t+1}^R & = & C_t^R \cup \{r_t\}, \\ C_{t+1}^L & = & C_t^L \cup \mathcal{N}(r_t), \\ A_{t+1}^R & = & (A_t^R \setminus \{r_t\}, \mathcal{N}^2(r_t) \cap U_t^R), \\ U_{t+1}^R & = & U_t^R \setminus \mathcal{N}^2(r_t), \\ U_{t+1}^L & = & U_t^L \setminus \mathcal{N}_{\widetilde{G}}(r_t). \end{array}$$

We let X_t^R be the number of vertices added to $A_t^R \setminus \{r_t\}$ at step t, i.e., $X_t^R = |\mathcal{N}^2(r_t) \cap U_t^R|$, and X_t^L (resp. $\widetilde{X}_t^L$) be the number of left-vertices of G (resp. $\widetilde{G}$) explored at step t, i.e., $X_t^L = |\mathcal{N}(r_t) \cap U_t^L|$ and $\widetilde{X}_t^L = |\mathcal{N}_{\widetilde{G}}(r_t) \cap U_t^L|$, where we identify the vertices in G and $\widetilde{G}$ with the same label. We also let $\theta = \inf\{t \geq 1 : \ A_t \neq \varnothing\}$ be the time at which we complete the exploration of the connected component of r_t in G; θ is a stopping time for the filtration $\mathcal{F}_t = \sigma\Big(\left(C_i^R, C_i^L, A_i^R, U_i^R, U_i^L\right)_{0 \leq i \leq t} \Big)$. With these definitions and for all $t \leq \theta$, it is clear that

$$\begin{array}{rcl} |C_t^R| & = & t, \\ |C_t^L| & = & \sum_{i=0}^{t-1} X_i^L, \\ |A_t^R| & = & 1 + \sum_{i=0}^{t-1} (X_i^R - 1) \leq 1 - t + (d-1) \sum_{i=0}^{t-1} X_i^L, \\ |U_t^R| & = & m - 1 - \sum_{i=0}^{t-1} X_i^R, \\ |U_t^L| & = & \binom{m}{2} - \sum_{i=0}^{t-1} \widetilde{X}_i^L \geq \frac{m-1}{2}(m - dt). \end{array}$$

To get used to reasoning on this model of random bipartite graphs with local structure constraints as well as to understand why double hashing graphs converge to trees, we start off by bounding the number of cycles of a given length in G:

Lemma 3.1. *Let C_k be the number of cycles of length k in G. We have*

$$\mathbb{E}\left[C_{2k}\right] \leq d^{2k}\tau^k + O(1/m)$$

Proof. Let $r_1, \ldots, r_k \in R$ be distinct vertices of R. For $r_1, \ldots, r_k$ to be on a cycle of size $2k$ of G in this order, there must exist distinct vertices $l_1, \ldots, l_k \in L$ such that, for all i, l_i connects r_{i-1} and r_i, where $r_0 = r_k$. Fixing r and r' distinct in R, the number of vertices connecting them in $\widetilde{G}$ is exactly $\binom{d}{2}$ (as setting the set of indices $\{i, i'\}$ such that $f + ig = r$ and $f + i'g = r'$ or $f + i'g = r$ and $f + ig = r'$ leaves exactly one possibility for the couple $(f, g) \in \widetilde{L}$). Therefore, the probability that r and r' are connected by a vertex in L is

$$\begin{array}{rcl} \mathbb{P}\Big(r \text{ and } r' \text{ are connected by a vertex in } L\Big) & = & 1 - e^{-\frac{2\tau}{m-1}\binom{d}{2}} \\ & = & \binom{d}{2}\frac{2\tau}{m} + O(1/m^2). \end{array}$$

Furthermore, it is clear that the probability of existence of *distinct* vertices $l_1, \ldots, l_k$ in L such that, for each i, l_i connects r_{i-1} and r_i, is no larger than the

product of the probabilities of existence of each l_i independently of the others:

$$\begin{aligned}&\mathbb{P}\Big(\forall i,\ \exists l_i \in L \text{ connecting } r_{i-1} \text{ and } r_i;\ l_i \neq l_j,\ \forall j \neq i\Big)\\ &\leq \textstyle\prod_i \mathbb{P}\Big(\exists l_i \in L \text{ connecting } r_{i-1} \text{ and } r_i\Big)\\ &\leq \left(\tfrac{\tau d^2}{m}\right)^k + O(1/m^{k+1}).\end{aligned}$$

We conclude by summing over all the $m \dots (m-k+1) \leq m^k$ possible choices of $r_1, \dots, r_k$. □

3.3.4 Proofs

We now return to the proof of the main theorem. Lemma 3.2 shows that the joint distribution of the numbers of children of the left-vertices explored during the first t steps of BFS is asymptotically very close to that we would obtain in a two-step unimodular Galton-Watson tree.

Lemma 3.2. *On an enlarged probability space, there exists a sequence $(Y_t^L)_{t\geq 0}$ of i.i.d. Poi(τd) variables such that*

$$\begin{aligned}&\mathbb{P}\Big((X_0^L, \dots, X_{(t-1)\wedge\theta}^L) \neq (Y_0^L, \dots, Y_{(t-1)\wedge\theta}^L)\Big)\\ &\leq \tfrac{1}{m-1}(\tau d^2 t^2 + \tau^2 dt + 2\tau t).\end{aligned}$$

We denote by $\mathrm{d_{TV}}(p, q)$ the distance in total variation between the two distributions p and q. To ease the notation, we will also use $\mathrm{d_{TV}}$ for random variables, which will refer to the distance between their distributions. The maximal coupling inequality says that, over an enlarged probability space, there exists a coupling of X and Y such that $\mathbb{P}(X \neq Y) = \mathrm{d_{TV}}(X, Y)$.

Proof. For any $t \leq \theta$, X_t^L is a binomial random variable of parameters $|U_t^L \cap \mathcal{N}_{\widetilde{G}}(r_t)|$ and $1 - e^{-\frac{2\tau}{m-1}}$. We have that $|\mathcal{N}_{\widetilde{G}}(r_t)| = \frac{m-1}{2}d$ and, as any two vertices in R (and in particular r_i and r_t for any $i < t$) are connected by exactly $\binom{d}{2}$ vertices in $\widetilde{L}$, we obtain the bound $\frac{m-1}{2}d - \binom{d}{2}t \leq |U_t^L \cap \mathcal{N}_{\widetilde{G}}(r_t)| \leq \frac{m-1}{2}d$. Furthermore, $\frac{2\tau}{m-1} - \frac{2\tau^2}{(m-1)^2} \leq 1 - e^{-\frac{2\tau}{m-1}} \leq \frac{2\tau}{m-1}$. It follows

$$\mathbb{P}\Big((X_0^L, \dots, X_{(t-1)\wedge\theta}^L) \neq (Y_0^L, \dots, Y_{(t-1)\wedge\theta}^L)\Big) \leq \mathbb{E}\Big[\sum_{i=0}^{(t-1)} \mathbf{1}(i \leq \theta)\mathbb{P}(X_i^L \neq Y_i^L \big| \mathcal{F}_i)\Big]$$

and

$$\begin{aligned}\mathbb{E}\Big[\mathrm{d_{TV}}(X_i^L, Y_i^L)|\mathcal{F}_i\Big] &\leq \mathbb{P}\Big(\mathrm{Bin}(\tbinom{d}{2}i, 1 - e^{-\frac{2\tau}{m-1}}) \neq 0\Big)\\ &+ \mathrm{d_{TV}}\Big(\mathrm{Bin}(\tfrac{m-1}{2}d, 1 - e^{-\frac{2\tau}{m-1}}), \mathrm{Bin}(\tfrac{m-1}{2}d, \tfrac{2\tau}{m-1})\Big)\\ &+ \mathrm{d_{TV}}\Big(\mathrm{Bin}(\tfrac{m-1}{2}d, \tfrac{2\tau}{m-1}), \mathrm{Poi}(\tau d)\Big)\\ &\leq \tfrac{2\tau i}{m-1}\tbinom{d}{2} + \tfrac{\tau^2 d}{m-1} + \tfrac{2\tau}{m-1}.\end{aligned}$$

We conclude by using the maximal coupling inequality and then summing over i. □

Next, Lemma 3.3 asserts that w.h.p. the graph explored in the first t steps of BFS is a tree.

Lemma 3.3. *For an integer $t \leq \theta$, the portion of the graph G explored until step t becomes a tree w.h.p. as $m, n \to \infty$. More precisely,*

$$\mathbb{P}\Big(G[(C_t^R \cup A_t^R) \cup C_t^L] \text{ not a tree}\Big) \leq \frac{\tau^2 d^4 t + \tau^2 d^4 t^2}{m-1}.$$

Proof. Loops get formed in the portion of the graph explored at step t when one of the following events occur:

- there exists a vertex $r \in U_t^R$ connected to r_t by at least two distinct vertices in L;
- there exists a vertex $l \in U_t^L \cap L$ which connects r_t and some vertex $r \in A_t^R$.

Indeed, assuming $G[(C_t^R \cup A_t^R) \cup C_t^L]$ is a tree and none of the two events above occur, it is easy to see that $G[(C_{t+1}^R \cup A_{t+1}^R) \cup C_{t+1}^L]$ is still a tree. At step t, if a loop is created, it must contain a newly explored left-vertex (i.e., a vertex in $C_{t+1}^L \setminus C_t^L$), otherwise it would not involve any new vertex or edge and hence would also be a loop in $G[(C_t^R \cup A_t^R) \cup C_t^L]$; furthermore, we can always assume that loop contains r_t. If the second event does not occur, the newly added vertices are connected to those in $(C_t^R \cup A_t^R) \cup C_t^L$ only through r_t, and thus any new loop must be contained in $G[\{r_t\} \cup (A_{t+1}^R \setminus A_t^R)]$, which is prevented since the first event does not occur either. Let us call E_t and E_t' the two events considered.

$$\begin{aligned}
\mathbb{P}(E_t) &\leq \mathbb{E}\Big[\textstyle\sum_{r \in U_t^R} \mathbb{P}(|\mathcal{N}(r) \cap \mathcal{N}(r_t)| \geq 2 \big| \mathcal{F}_t)\Big] \leq \tfrac{\tau^2 d^4}{m-1}, \\
\mathbb{P}(E_t') &\leq \mathbb{E}\Big[\sum_{r \in A_t^R \setminus \{r_t\}} \mathbb{P}(\mathcal{N}(r) \cap \mathcal{N}(r_t) \neq \varnothing \big| \mathcal{F}_t)\Big] \leq \tfrac{\tau d^2}{m-1}(\mathbb{E}[|A_t^R|] - 1).
\end{aligned}$$

We can bound $\mathbb{E}[|A_t^R|]$ as follows

$$\mathbb{E}[|A_t^R|] - 1 \leq -t + (d-1) \sum_{i=0}^{t-1} \mathbb{E}[X_i^L] \leq \tau d^2 t.$$

Summing over t yields the desired result. □

The hashing graph G obtained may be a tree, but remember that we merged all the items with the same choices of buckets into a single left-vertex of G. Lemma 3.4 shows that each left-vertex explored in the first t steps of BFS actually correspond to a single item, i.e., the items explored were given different choices of buckets, w.h.p.

Lemma 3.4. *For an integer $t \leq \theta$, no left-vertex explored until step t has two copies or more (i.e., $Z_l \leq 1$ for all $l \in C_t^L$) w.h.p. as $m, n \to \infty$. We have*

$$\mathbb{P}\Big(\exists l \in C_t^L : \ Z_l \geq 2\Big) \leq \frac{4\tau^3 dt}{(m-1)^2}.$$

Proof. The result follows straightforwardly from the union bound:

$$\begin{aligned}\mathbb{P}\Big(\exists l \in C_t^L :\ Z_l \geq 2\Big) &\leq \mathbb{E}[|C_t^L|](1-\mathbb{P}\big(\mathrm{Poi}(\tfrac{2\tau}{m-1}) \leq 1)\big)\\ &= \textstyle\sum_{i=0}^{t-1} \mathbb{E}[X_i^L]\big(1-e^{-\frac{2\tau}{m-1}} - \frac{2\tau}{m-1}e^{-\frac{2\tau}{m-1}}\big)\\ &\leq \tau dt \tfrac{4\tau^2}{(m-1)^2}.\end{aligned}$$

□

We are now ready to show that the average distribution $\overline{\rho}_m$ of the k-hop neighborhood of a random root-vertex r_0 in a random graph G_m tends to the distribution of a two-step unimodular Galton-Watson tree cut at depth k, for any $k \in \mathbb{N}$ and as $m \to \infty$ (Proposition 3.1).

Proof of Proposition 3.1. Let $k \in \mathbb{N}$. For any finite tree T of depth at most k (or more generally for rooted graphs of radius at most k) and any collection of such trees $\mathcal{T}$, let $A_T = \{[G] \in \mathcal{G}_*,\ (G)_k \simeq T\}$, where $\simeq$ refers to rooted isomorphism, and $A_{\mathcal{T}} = \cup_{T\in\mathcal{T}} A_T$. For every $\epsilon > 0$ and any finite k, there exists a finite set $\mathcal{T}$ of trees of depths at most k in which every odd-depth vertex has degree d and such that $\rho(A_{\mathcal{T}}) = \sum_{T\in\mathcal{T}} \rho(A_T) \geq 1-\epsilon$. Let t be the maximum size of the trees in $\mathcal{T}$; it means that, with probability at least $1-\epsilon$, a BFS run during t steps on a two-step unimodular Galton-Watson tree sampled from ρ will explore at least the k-hop neighborhood of the root.

Let r_0 be a random right-vertex in the random graph G_m. According to Lemmas 3.3 and 3.4 and with probability at least $1 - \frac{\tau^2d^4t+\tau^2d^4t^2}{m-1} - \frac{4\tau^3dt}{(m-1)^2} \underset{m\to\infty}{\to} 1$, the graph induced by $[G_m, r_0]$ on the vertices explored during the first t steps of BFS is a tree, with all left-vertices having exactly one copy. Hence, in particular the offspring of each left-vertex in this tree is of size $d-1$. Furthermore, according to Lemma 3.2 and with probability at least $1 - \frac{\tau d^2t^2+\tau^2dt+2\tau t}{m-1} \underset{m\to\infty}{\to} 1$, there is a coupling between the numbers of newly explored left-vertices in the graph $[G_m, r_0]$ during the first t steps of BFS and a sequence of i.i.d. $\mathrm{Poi}(\tau d)$ random variables. Therefore, on an event of probability tending to 1 as $m \to \infty$, we can couple the exploration during the first t steps of BFS on $[G_m, r_0]$ and on a two-step unimodular Galton-Watson tree sampled from ρ. On the event that these two exploration can be coupled and for any $T \in \mathcal{T}$ such that the BFS exploration on T will have explored at least the k-hop neighborhood of the root within t steps, it follows the BFS exploration on $[G_m, r_0]$ will also have explored at least the k-hop neighborhood of r_0. Then, for any $T \in \mathcal{T}$, we have

$$|\mathbb{P}((G_m, r_0)_k \simeq T) - \rho(A_T)| \leq \frac{\tau^2d^4t + \tau^2d^4t^2 + \tau d^2t^2 + \tau^2dt + 2\tau t}{m-1} + \frac{4\tau^3dt}{(m-1)^2}.$$

Then, it follows $\lim_{m\to\infty} \overline{\rho}_m(A_T) = \rho(A_T)$ for any $T \in \mathcal{T}$.

For any bounded uniformly continuous function f, there exists $k \in \mathbb{N}$ such that $|f((G)_k) - f(G)| \leq \epsilon$ for all rooted graphs $G \in \mathcal{G}_*$. For this k we can define a finite collection of trees $\mathcal{T}$ as before, such that $\rho(\mathcal{T}) \geq 1-\epsilon$. For m large enough, we have $\overline{\rho}_m(\mathcal{T}) \geq 1-2\epsilon$, and

$$\Big|\int f \mathrm{d}\overline{\rho}_m - \int f \mathrm{d}\rho\Big| \leq \epsilon(1+3||f||_\infty) + \sum_{T\in\mathcal{T}} f(T)\big|\overline{\rho}_m(A_T) - \rho(A_T)\big|.$$

Letting $m \to \infty$ and then $\epsilon \to 0$ completes the proof. □

We will now prove *almost sure* local weak convergence of the random graph G_m towards the two-step unimodular Galton-Watson tree (Theorem 3.2). To that end, we use the Azuma-Hoeffding measure-concentration inequality:

Proposition 3.2. *Let $M = (M_t)_{0\le t\le m}$ be a martingale with respect to a filtration $\mathcal{F} = (\mathcal{F}_t)_{0\le t\le m}$. Suppose there exists constants $c_1, \dots, c_m$ such that, for all $1 \le t \le m$, the following holds:*

$$\left|M_t - M_{t-1}\right| \le c_t.$$

Then, for all $\epsilon > 0$,

$$\mathbb{P}\Big(\left|M_m - M_0\right| \ge \epsilon\Big) \le 2e^{-2\epsilon^2/\sum_{0\le t\le m} c_m^2}.$$

Proof of Theorem 3.2. Let $k \in \mathbb{N}^*$ and H be a rooted graph of radius at most k. We define A_H as in the proof of Proposition 3.1. We let h be the number of right-vertices in H; we focus on m large enough such that $\ln m \ge h$. Recall that

$$\rho_{G_m}(A_H) = \frac{1}{m}\sum_{r_0\in R_m} \mathbf{1}\left((G_m, r_0)_k \simeq H\right).$$

For any $r \in R_m = \{1, \dots, m\}$, we define

$$\zeta_r = (Z_l :\ l \in \mathcal{N}_{\widetilde{G}}(r) \text{ and } l \notin \mathcal{N}_{\widetilde{G}}(r'), \forall r' < r)$$

and we let $\mathcal{F}_t = \sigma\Big((\zeta_r)_{1\le r\le t}\Big)$. It is easy to obtain from Chernoff's bound that

$$\mathbb{P}(|\zeta_r| \ge \ln m) \le \mathbb{P}(\sum_{l\in\mathcal{N}_{\widetilde{G}}(r)} Z_l \ge \ln m) \le \frac{me^{-\tau d}}{m^{\ln\frac{\ln m}{\tau d}}},$$

so that

$$\mathbb{P}(\exists r \in R_m \text{ such that } |\zeta_r| \ge \ln m) \le \frac{m^2 e^{-\tau d}}{m^{\ln\frac{\ln m}{\tau d}}} \underset{m\to\infty}{\to} 0.$$

We say that G_m is valid if $|\zeta_r| < \ln m$ for all r, and that ζ_r is valid if $|\zeta_r| < \ln m$. We will sometimes write $G_m(\zeta)$ and $\rho_{G_m(\zeta)}(A_H)$ to avoid confusion. We define $\overline{\rho}_m(A_H|G_m \text{ valid})$ as the expectation of $\rho_{G_m}(A_H)$ over the random graph G_m conditionally on G_m valid.

Now, we let

$$M_t = \mathbb{E}[\rho_{G_m}(A_H)|\mathcal{F}_t, G_m \text{ valid}].$$

We have

$$\mathbb{E}[M_{t+1}|\mathcal{F}_t] = \mathbb{E}[\rho_{G_m}(A_H)|\mathcal{F}_t, G_m \text{ valid}] = M_t,$$

thus M is indeed a martingale with respect to $\mathcal{F}$. Note that $M_0 = \overline{\rho}_m(A_H|G_m \text{ valid})$ and, for G_m valid, $M_m = \rho_{G_m}(A_H)$. Consider two sequences $\zeta = (\zeta_r)_{1\le r\le m}$ and $\zeta' = (\zeta'_r)_{1\le r\le m}$ differing only in one value, say $\zeta_r \ne \zeta'_r$. There are at most $(1+|\zeta_r|(d-1))h$

right-vertices r_0 in $G_m(\zeta)$ such that $(G_m(\zeta), r_0)_k \simeq H$ and r has a 2-hop neighbor in $R \cap (G_m(\zeta), r_0)_k$. Assuming ζ_r and ζ'_r are valid, we obtain

$$|\rho_{G_m(\zeta)}(A_H) - \rho_{G_m(\zeta')}(A_H)| \leq \frac{2dh \ln m}{m}.$$

Therefore, assuming ζ_t is valid for $t < r$, we have

$$\begin{aligned} |M_r - M_{r-1}| &\leq \mathbb{E}\Big[\max_{\zeta_r, \zeta'_r} \big|\rho_{G_m(\zeta)}(A_H) - \rho_{G_m(\zeta')}(A_H)\big| \Big| \mathcal{F}_{r-1}, G_m \text{ valid} \Big] \\ &\leq \tfrac{2dh \ln m}{m}. \end{aligned}$$

Then, the Azuma-Hoeffding inequality yields

$$\mathbb{P}\Big(\big|\rho_{G_m}(A_H) - \overline{\rho}_m(A_H | G_m \text{ valid})\big| \geq \epsilon \Big| G_m \text{ valid} \Big) \leq 2e^{-m\epsilon^2/(2h^2d^2 \ln^2 m)}.$$

Furthermore, it is easy to check that

$$\begin{aligned} \big|\overline{\rho}_m(A_H) - \overline{\rho}_m(A_H | G_m \text{ valid})\big| &\leq 2\mathbb{P}(G_m \text{ not valid}) \\ &\leq \tfrac{2m^2 e^{-\tau d}}{m^{\ln \frac{\ln m}{\tau d}}}, \end{aligned}$$

and it follows that

$$\mathbb{P}\Big(\big|\rho_{G_m}(A_H) - \overline{\rho}_m(A_H)\big| \geq \epsilon \Big) \leq 2e^{-\frac{m\epsilon^2}{2h^2d^2 \ln^2 m}} + \frac{2m^2 e^{-\tau d}}{m^{\ln \frac{\ln m}{\tau d}}}.$$

The term on the right-hand side is the general term of a convergent series, hence we can conclude using the Borel-Cantelli lemma and then the same argument as for Proposition 3.1. □

Chapter 4

Load-balancing and resource-placement in distributed CDNs

This section considers large scale distributed content delivery networks, composed of a data center assisted by many small servers with limited capabilities and located at the edge of the network. The objective in such systems is to offload as much as possible the data center, which operation is comparatively costly. The efficiency critically depends on two factors: (i) content replication within servers, and (ii) how incoming service requests are matched to servers holding the requested content. To inform the corresponding design choices, we study two natural matching policies with extremal features and which should yield fundamental intuition about the system: optimal matching and greedy random matching. Under both matching policies, the inefficiency of underloaded systems decays exponentially fast as storage increases, however the simpler algorithm incurs a severe performance degradation as the system load approaches criticality. For both policies, we explore the impact of the content replication on the performance, and optimize replication in the large system / large storage limits. In a second time, we develop adaptive replication policies for greedy matching designed for minimizing the overall loss rate and reacting quickly to popularity changes.

4.1 Introduction

The surge in consumption of video over the Internet necessitates massive bandwidth provisioning. At the same time, storage is extremely cheap. Extensive replication of content not only within data centers, but also at the periphery of the network, e.g., in users' computers, can thus be envisioned to leverage uplink bandwidth available from users' homes. *People's CDN* is one particular commercial initiative in this direction; the massively popular *PPLive* peer-to-peer system for video-on-demand is another example of this approach.

Such systems feature two key resources, namely storage and bandwidth. Ideally, one would like to utilize storage by pre-loading content replicas at individual servers, in such a way that bandwidth of all servers is available to serve any incoming request. In other words, a challenge in engineering such systems is to create content replicas so that bandwidth can be used maximally. Several strategies for content replication have been considered: e.g., uniform replication does not discriminate between contents; proportional replication tunes the number of replicas to the average number of requests, and automatically arises at cache memories when the so-called random-useful cache eviction method is used (it is also approximately achieved by the classical Least Recently Used (LRU) eviction rule [104]).

Our first objective is to develop a clear understanding of the relative merits of distinct replication strategies in the context of large-scale distributed server platforms. We also aim at characterizing the impact of the amount of storage available on the performance. These properties certainly depend both on the content replication strategy and on the algorithm used to match incoming service requests to servers capable of treating these requests. However, a joint study of the influence of these two factors seems too ambitious, therefore we instead focus on two natural matching algorithms: optimal matching and random greedy matching, which are in a sense extremal and should give a clear view of the range of performance and typical behaviors one can expect in such systems. Under both these matching algorithms, we characterize the influence of the replication of contents, with a particular focus on large system / large storage asymptotics. We notably identify replication policies with the optimal asymptotic performance.

This chapter is organized as follows: in Section 4.2, we explain in detail our model of an edge-assisted CDN, in particular the way replication strategies are captured as well as the two matching policies considered. We then review related work on similar systems and the replication strategies encountered there (Section 4.3). We then turn to the analysis of the system under the two matching algorithms considered: Section 4.4 focuses on optimal matching, for which the expected performance of the system can be obtained via the study of maximum capacitated matchings (Theorem 2.1); and in Section 4.5, we use mean-field heuristics to study the system under random greedy matching. For both matching algorithms, we pay a particular attention to the large storage regime, in which the optimal replications have an explicit expression. Finally, Section 4.6 is devoted to designing adaptive replication algorithms under random greedy matching, which quickly attain the optimal replication of contents by leveraging the detailed analysis of the previous section.

4.2 Edge-assisted CDN model and statistical assumptions

In this section, we describe our basic edge-assisted CDN model. A more complex model is described in Section 4.4.2. However this last model may be less amenable to analysis, therefore we will stick to the basic model most of the time. Indeed, it seems likely that most of the intuition can already be gathered under the simpler model.

The basic components of an edge-assisted CDN are a data center and a large number m of small servers. The CDN offers access to a catalog of n contents of identical size (for simplicity). The data center stores the entire catalog of contents and can serve all the requests directed towards it, whereas each small server can only store a fixed number d of contents and can provide service for at most one request at a time. We can represent the information of which server stores which content in a bipartite graph $G = (S \cup C, E)$, where S is the set of servers, C the set of contents, and there is an edge in E between a content c and a server s if s stores a copy of c; an edge therefore indicates that a server is able to serve requests for the content with which it is linked. We do not allow requests to be delayed or split between many servers. As a result, at any time, the subset of edges of G over which some service is being provided form a capacitated matching $\mathbf{x}$ of the graph G, where the capacities are all 1 except the vertex-capacities of contents c, which are equal to their current number of pending requests R_c.

The contents of the catalog may have different popularities, leading to different requests arrival rates λ_c. We let $\overline{\lambda}$ be the average request rate, i.e., $\overline{\lambda} = \frac{1}{n}\sum_c \lambda_c$. According to the independent reference model (IRM), the requests for the various contents arrive as independent Poisson processes with rates λ_c. In addition, we assume the service times of all the requests are independent Exponential random variables with mean 1 whether a particular request is served by a server or by the data center, so that at each instant the number of pending requests R_c follows a Poisson distribution with mean λ_c. We let ρ denote the expected load of the system, i.e., $\rho = \frac{n\overline{\lambda}}{m}$. The distribution of popularities of the contents is left arbitrary at this stage. In practice, Zipf distributions are often encountered (see [45] for a page-long survey of many studies), where the i-th most popular content in a Zipf distribution of parameter α has popularity $\lambda_i = \overline{\lambda}\frac{i^{-\alpha}}{\sum_{j\le n} j^{-\alpha}}$. As an alternative popularity model, we may often assume that the contents can be grouped into K *popularity classes*, where the class k contains a fraction α_k of all the contents and all of them have the same popularity λ_k. In such a class model, the problem of adapting the replication to the popularities of contents is still almost intact, while the analysis is more tractable and the results easier to interpret.

The large scale of the system as well as the uncertainty ahead of time in the number of requests for each content make it hopeless to design with too much precision the joint constitution of the caches of all the servers, i.e., deciding jointly of each and every edges in the bipartite graph G. Indeed, we are interested in a regime where the number of contents n and the number of servers m tend to infinity together, with $m \sim \tau n$ for some $\tau > 0$. In this regime, it seems reasonable that there will be little fine-tuned cooperation between the servers, and they will thus essentially make

independent choices for their cache based on the available statistics on the requests. In fact, we even assume for simplicity that a same server does independent caching choices for all its memory slots. Then, at best, if the servers have some *a priori* information on all the contents, they can ensure that each content has a given number of copies D_c, with $\sum_{c \in C} D_c = md$, and the content-server graph G is a random bipartite graph with fixed degree d on S and a prescribed degree sequence on C. If less information is available, e.g., the servers behave as if under the popularity class model, then they may decide only of which class to store based on a distribution π and then cache a random content from the corresponding class. Then the distribution of the degree D of a random content is a mixed Poisson distribution, and G is a random bipartite graph with degrees d on S and a prescribed distribution D on C. An extreme of this model is when the servers have no information beforehand on the contents and do fully random caching choices, resulting in a $\mathrm{Poi}(md/n)$ degree distribution for the contents.

We focus on two different matching algorithms. The first one is *optimal matching*, i.e., the system uses a maximum capacitated matching of the content-server graph at any time. For such a matching algorithm to be possible, we make the assumption that one can re-assign requests during their service without losing the work already done, e.g., the same request may at first be routed to the data center for lack of available servers, then re-directed to a small server which has just completed a service and then moved again to another server at some point to allow using a better matching. This assumption that re-allocations are possible allows us to have an optimal matching algorithm which does not require any knowledge of the future requests arrivals. We do not specify a particular implementation for the optimal matching algorithm: the algorithm may be centralized, with an authority knowing the state of the system and the contents of all caches at any time and dispatching the requests, or decentralized like for example the cuckoo hashing random walk insertion algorithm seen in Chapter 3. Note that the problem of finding a maximum capacitated matching in a finite bipartite graph is known to be of strongly polynomial complexity [8], using a strongly polynomial algorithm for maximum flow problems by [102]. There may nonetheless exist lower-complexity methods, that yield comparable performance for the graphs of interest. The purpose of studying the optimal matching algorithm is that it provides upper-bounds on the efficiency for any given replication policy under any other matching algorithm. This allows us to identify the regimes as well as the potential causes for performance degradations under any given candidate matching algorithm, and therefore also determine whether any substantial gain can be obtained by considering a more complex algorithm. The study of the performance of the system and the asymptotics of an optimal replication policy are the object of Section 4.4. In contrast with the optimal matching, which may not be practical in some systems, the second matching algorithm we consider is a very simple one which should be implementable in any system, called *random greedy*. Under this policy, each request is directed upon arrival either to an available server storing the corresponding content or to the data center if no such server exists, and then no re-allocation of the request is possible. As this is the simplest possible policy one could think of, it yields a lower-bound on the average performance of the system under any reasonable matching strategy. The random greedy matching

algorithm is studied in Section 4.5.

Under both matching algorithms, the requests are served without delay, be it by the data center or by a small server, therefore the performance of the system is mainly described by the cost associated with the service. This cost is mostly tied to the number of requests served by the data center, therefore the most relevant performance metric here is the fraction of the load left-over to the data center. Then, it makes sense to view the requests that the small servers cannot handle as *lost*. In fact, the system consisting of the small servers alone with fixed caches is a loss network in the sense of [64]. We call γ_c the rate at which requests for content c are lost, and $\overline{\gamma}$ the average loss rate among contents, i.e., $\overline{\gamma} = \frac{1}{n}\sum_c \gamma_c$. The main goal is to make $\overline{\gamma}$ as small as possible. In underloaded regime $\rho < 1$, we refer to the fraction of lost requests as the inefficiency of the system $\iota = \overline{\gamma}/\overline{\lambda}$. However, such a quantity would not be adapted in overloaded regime (as it would automatically be bounded away from 0 by a positive constant independent of matching and replication), thus we define the inefficiency in this regime as the fraction of servers which are unused. Therefore, a consistent definition of inefficiency in any regime is

$$\iota = 1 - \frac{\rho(1-\overline{\gamma}/\overline{\lambda})}{\min\{1,\rho\}} = \begin{cases} \text{fraction of lost requests} & \text{if } \rho \leq 1, \\ \text{fraction of unused servers} & \text{if } \rho \geq 1. \end{cases} \tag{4.1}$$

In the following section, we review some of the related work, mentioning the different models and performance metrics considered as well as the associated content replication strategies.

4.3 Related models and replication strategies

The edge-assisted CDN model is related to a few other models. A major difference though is the focus on modeling the basic constraints of a realistic system, in particular regarding the limited capacity of servers (in terms of storage, service and coordination) and the way requests are handled, i.e., the matching policy. The same system model is also studied in [101] under the name "distributed server system"; we merely use a different name to emphasize the presence in the core of the network of a data center, which makes it clear what performance metric to consider, while otherwise availability of contents for example may become a relevant metric. Also, [7] studies the same issues of efficient matching and replication under a queueing model of performance; in contrast, we focus on delay-sensitive contents, which must be served without waiting. The edge-assisted CDN model is also related to peer-to-peer (P2P) VoD systems, although in such systems one has to deal with the behavior of peers, and to cache networks, where caches are deployed in a potentially complex overlay structure in which the flow of requests entering a cache is the "miss" flow of another one. Previous work on P2P VoD typically does not model all the fundamental constraints we mentioned, and the cache networks literature tends to completely obliterate the fact servers become unavailable while they are serving requests and to focus on alternative performance metrics such as search time or network distance to a copy of requested contents.

More generally, the question of how to replicate content in distributed systems is related to the general problem of *facility location* (see e.g., Vazirani [108] Chapter

24). The latter has received considerable attention from the standpoint of algorithmic complexity and approximability. The version that we consider here is atypical in that it features capacity constraints on the locations (the servers), and stochastic demand. Also, we aim at characterizing simple, easily implementable strategies with good performance for practical workloads rather than placement algorithms with low worst-case complexity.

There is a rich literature on cache management strategies motivated by server memory and web cache management, an abstract version of which is the so-called *paging problem* (see e.g., Albers [1]). In this context, the main focus has been on characterization of hit rates, focusing on the temporal properties of streams of requests. Capacity limits at the servers are typically not considered in these models, while they are essential for the application scenarios we consider.

Our present motivation, namely efficient use of servers' bandwidth through adequate replication, has been considered in the specific context of ISP-managed Peer-to-Peer networks (as described in [99, 60]). In [104, 24], an argument is made for the proportional replication policy based on the analysis of delay in a queueing model of performance. [99] and [24] propose replication policies that are oblivious to content popularity. The first considers stochastic delay performance models, and the second deterministic conditions on request arrivals to guarantee feasibility of service. The three articles [116, 121, 120] are closer to our motivation in that they revisit the proportional placement strategy, to which they propose alternatives. Their modeling approach however significantly differs from ours. In contrast with the P2P systems above, [106] models the requests in such a network as *internally* generated, and thus they may not require any bandwidth usage if the requested content is already present on the device. This results in an "hot-warm-cold" replication strategy, where the very popular contents are replicated everywhere and the cold ones not at all. Both the ISP-managed P2P system (with only internal requests) and the edge-assisted CDN model (with only external requests) are investigated in [101], which concludes that hot-warm-cold replication is indeed asymptotically optimal for the former model and similarly advocates for *proportional replication* in the latter, which keeps for each content a number of copies proportional to its popularity.

Another strand of research attempts to capture the impact of the particular topology of the network, sometimes at the cost of a less realistic modeling at the level of individual devices (e.g., no bandwidth limitations). The performance criterion is then generally to minimize the total bandwidth consumed to reach a server able to handle the requests. Modeling the geographical aspects only at a coarse level, [62, 95, 94] solve content placement and matching problems between many regions, while not modeling in a detailed way what happens inside a region. In random networks, [105] advocates again for proportional replication to minimize total bandwidth consumption under shortest path routing, while [32] proposes square-root replication (i.e., with a number of copies proportional to $\sqrt{\lambda_c}$) to minimize search-time. Finally, the precise hierarchical structure of storage points is modeled in the literature on cache networks. In this context, [69] addresses the problem of joint dimensioning of caches and content placement. However, it is hard to characterize optimal replications of contents in cache networks, and one often has to resort to heuristics: contents are then cached dynamically as they traverse the network

towards the end-user, thereby replacing older contents based on various strategies. The evicted content may be for example the least frequently used (LFU) content, the least recently used (LRU) one, or even a completely random one. Under such policies, in order to understand which type of content should be stored close to the end-user, [45] studies specifically a two-levels hierarchy of caches using an approximation for LRU performance from [28, 44]: based on the differences in their respective popularity distributions, it advocates that VoD contents are cached very close to the network edge while user-generated contents, web and file-sharing are only stored in the network core. Note that, in a sense, this agrees with the view adopted in edge-assisted CDNs, where the most delay-sensitive contents (i.e., VoD contents) are pushed even further towards the edge of the network up to the users' premises.

4.4 Performance under optimal matching

In this section, we analyse edge-assisted CDNs under an optimal matching policy, which dynamically re-directs requests so that at any time the load on the data center is minimum given the current set of active requests. We begin this section by giving an argument for why proportional placement is popular in such context: under this replication policy, all the requests may be handled by the servers alone provided the storage space grows mildly (logarithmically) with the size of the catalog of contents. Then, we show how the performance of large families of replication policies can be computed in large systems with fixed storage capacity per server, even for a richer model with non-unitary service capacity for the servers and contents of different sizes, using results on the size of maximum capacitated matchings. As it is hard to get intuition from the finite storage expressions for the efficiency of the system, we turn to an asymptotic analysis as storage grows for the basic version of the edge-assisted CDN model; in this regime, the limits of optimal replications can be explicitely computed, which notably shows proportional replication is never optimal, in any load regime.

4.4.1 An argument for proportional placement

For this study of proportional placement, we will allow the memory size d of servers to grow with n; a more precise characterization, at fixed d will be obtained in later sections. Recall that the numbers of requests R_c for content c are mutually independent over all $c \in C$, R_c having a Poisson distribution with parameter $\lambda_c \geq \underline{\lambda}$ for some fixed parameter $\underline{\lambda} > 0$. Based on knowledge of the λ_c's only (and not of the R_c's), the replication problem consists in determining the number of replicas D_c of each item c and their placement onto servers. A candidate strategy consists in taking the number of replicas D_c deterministic and proportional to the *expected* number of requests λ_c: $D_c \approx \tau d \lambda_c / \overline{\lambda}$, where $\overline{\lambda}$ is the average content popularity. The system is sub-critical, i.e., $\rho < 1$, if the expected number of requests is smaller than the number of servers, which reads

$$\delta = \tau - \overline{\lambda} = \tau(1 - \rho) > 0, \tag{4.2}$$

where we used the notation δ for the stability margin $\tau - \overline{\lambda}$. In this context we have the following:

Proposition 4.1. *Assuming $\delta > 0$ and proportional placement is used, all requests can be met with high probability if the storage space per server d satisfies $d \geq \alpha \log(n)(1 + o(1))$, where $\alpha > 0$ satisfies Equation (4.4).*

As a corollary, this result implies that the inefficiency of proportional placement is equal to 0 a.s. in under-loaded regime for d logarithmic in the number of contents in the system.

Proof. For each content c and each of the corresponding R_c requests, we can split the request equally into D_c sub-requests of size $1/D_c$. To a given server s with corresponding collection $\mathcal{N}(s)$ of stored contents, for each content $c \in \mathcal{N}(s)$ we associate R_c sub-requests to that particular server. Then a service of all requests will be feasible provided that for each such server $s \in S$ one has

$$\sum_{c \in \mathcal{N}(s)} \frac{R_c}{D_c} \leq 1. \tag{4.3}$$

Indeed, this mapping of sub-requests to servers would constitute a fractional matching of requests, and an integral matching would therefore exist, by the total unimodularity of the adjacency matrix of the graph G.

Calling $\mathcal{A}_s$ the event corresponding to Condition (4.3) and $\mathcal{A}_s^{\mathbf{c}}$ its complement, Chernoff's inequality yields

$$\begin{aligned} \mathbb{P}(\mathcal{A}_s^{\mathbf{c}}) &\leq \exp\left(-\sup_{\theta>0}\left[\theta - \textstyle\sum_{c\in\mathcal{N}(s)} \log \mathbb{E}[e^{\theta(R_c/D_c)}]\right]\right) \\ &= \exp\left(-\sup_{\theta>0}\left[\theta - \textstyle\sum_{c\in\mathcal{N}(s)} \lambda_c\left(-1 + e^{\theta/D_c}\right)\right]\right). \end{aligned}$$

Note that $D_c \geq d\tau\lambda_c/\overline{\lambda} - 1 \geq \lambda_c d\left(1 + \frac{\delta}{2\overline{\lambda}}\right)$ for d large enough, namely for $d \geq 2\overline{\lambda}/(\delta\underline{\lambda})$. Replacing D_c by this corresponding lower bound in the above expression, we obtain

$$\mathbb{P}(\mathcal{A}_s^{\mathbf{c}}) \leq \exp\left(-\sup_{\theta>0}\left[\theta - \sum_{c\in\mathcal{N}(s)} \lambda_c\left(-1 + e^{\theta'/\lambda_c}\right)\right]\right),$$

where we introduced the notation $\theta' = \theta/(d(1 + \frac{\delta}{2\overline{\lambda}}))$. It is readily checked that the function $\lambda_c \to \lambda_c\left(-1 + e^{\theta'/\lambda_c}\right)$ is decreasing in λ_c, and is thus upper-bounded by its value at $\underline{\lambda}$ for all $\lambda_c \geq \underline{\lambda}$. Using the fact that the cardinality of $\mathcal{N}(s)$ is precisely d, it then entails

$$\begin{aligned} \mathbb{P}(\mathcal{A}_s^{\mathbf{c}}) &\leq e^{-\sup_{\theta>0}\left[\theta - d\underline{\lambda}\left(-1 + e^{\theta/(d\underline{\lambda}(1+\frac{\delta}{2\overline{\lambda}}))}\right)\right]} \\ &= e^{-d\underline{\lambda}h(1+\frac{\delta}{2\overline{\lambda}})}, \end{aligned}$$

where $h(x) = x\log(x) - x + 1$ is the Cramér transform of a unit rate Poisson random variable. It follows that

$$\mathbb{P}(\mathcal{A}_s^{\mathbf{c}}) \leq n^{-\alpha\underline{\lambda}h(1+\frac{\delta}{2\overline{\lambda}})}$$

for $d \geq \alpha \log(n)$. Thus, provided

$$\alpha > \frac{1}{\underline{\lambda} h(1 + \delta/(2\overline{\lambda}))}, \tag{4.4}$$

by the union bound the probability that at least one event $\mathcal{A}_s$ fails is $o(1)$. Consequently, under the stability assumption (4.2), for popularities lower-bounded by $\underline{\lambda} > 0$, all requests are met with high probability as $n \to \infty$ provided

$$d \geq \max\left\{\alpha \log(n), 2\overline{\lambda}/(\delta\underline{\lambda})\right\},$$

where the constant α satisfies (4.4). □

This result indicates that a logarithmic storage size d suffices to meet all the requests, provided one uses the above proportional placement. Note also that proportional placement is robust, in the sense that this feasibility property does not depend on the particular way in which content replicas are co-located within servers so long as one does not replicate content more than once at a single server.

Nevertheless, several questions of interest remain open. In particular, the above argument does not say what happens when the stability margin δ becomes small: indeed, for small δ, the lower bound (4.4) on α is $\Theta(\delta^{-2})$, and it is thus not clear how this replication policy behaves in a near-critical regime $\delta \to 0$. In addition, it does not say how many requests remain unmatched under storage size restrictions, for instance for constant rather than logarithmic d. Moreover, it states the existence of a matching, but does not address whether such a matching is easy to find. Finally, it only deals with the proposed proportional placement, but leaves open the question of whether alternative placement strategies could be better suited, by maximising the expected fraction of requests that can be matched under fixed storage size d. In the following section, we show how to leverage the results from Theorem 2.1 to characterize the performance at fixed d, not only for proportional placement but also for replication policies as described in Section 4.3.

4.4.2 Performance characterization using maximum capacitated matchings

We first focus here on the basic version of the edge-assisted CDN model, where each server can only serve one request at a time and the contents all have the same size. In this setup, we show how to deduce the inefficiency of the system from Theorem 2.1. Then, we show how to capture a richer model, with non-unitary service capacities and contents of different size fragmented into constant size segments.

As mentioned in Section 4.2, the servers have limited coordination capacities, which does not allow them to control precisely how their caches are jointly constituted. In the popularity class model, the servers know the class to which each content belongs but cannot distinguish between contents of the same class. Therefore, the best the servers can do in this model is to set a specific distribution for the number of replicas of the contents from the same class; the number of replicas D_k and number of requests R_k of a random content of class k would then be independent given the class k. Slightly more generally, for finite m and n, we can assume that the number

of replicas and number of requests follow a joint distribution (D_c, R_c) which may not necessarily be obtained as the mixture of a finite number of pairs of independent variables. The content-server graph $G_{m,n}$ is then chosen uniformly at random among the graphs with the same degree sequences. As explained in Section 1.1.3, such random graphs converge almost surely in the local weak sense towards two-step unimodular Galton-Watson trees with laws $(D^C, R^C) \sim \Phi^C$ and $(d, 1) \sim \Phi^S$, where (D^C, R^C) is the limiting distribution for the joint numbers of replicas and requests of a random content and we did not mention the edge-capacities are they are all unitary in this basic model.

As mentioned in Section 4.2, at any time, the links between the active servers and the contents they are serving form a capacitated matching $\mathbf{x} \in \mathfrak{M}_{G_{m,n}}$ of the content-server graph $G_{m,n}$. In addition, as we consider an optimal matching policy, $\mathbf{x}$ is of maximum size in $\mathfrak{M}_{G_{m,n}}$ and thus $|\mathbf{x}| = M(G_{m,n})$. Consistently with Equation (4.1), the expected inefficiency of the system $\iota_{m,n}$ under the maximum matching policy is then equal to

$$\iota_{m,n} = 1 - \frac{1}{\min\{1, \rho\}} \frac{\mathbb{E}[M(G_{m,n})]}{m}, \tag{4.5}$$

where the expectation is with respect to both the varying number of requests R_c for each content c and the graph $G_{m,n}$, which depends on the replication strategy used. Then, the limit of $\mathbb{E}[M(G_{m,n})]/m$ as $m, n \to \infty$ with $n \sim \tau m$ is given by Theorem 2.1 (or Theorem 3 from [70]) which takes here the simpler form:

Corollary 4.1. *For a sequence of random graphs $G_{m,n}$ converging locally weakly to a two-step unimodular Galton-Watson tree with laws $(D^C, R^C) \sim \Phi^C$ and $(d, 1) \sim \Phi^S$, we have*

$$\lim_{m,n\to\infty} \frac{M(G_{m,n})}{m} = \inf \mathcal{F}^S(p, q),$$

where

$$\mathcal{F}^S(p, q) = \mathbb{P}\left(Bin(d, p) > 0\right) + \tau \mathbb{E}\left[R^C \mathbf{1}\left(Bin(D^C, q) > R^C\right)\right],$$

and the infimum is taken over pairs (p, q) satisfying

$$\begin{aligned} p &= \mathbb{P}\left(Bin(\widetilde{D}^C, q) < \widetilde{R}^C\right) \\ q &= (1-p)^{d-1}. \end{aligned}$$

where $(\widetilde{D}^C, \widetilde{R}^C)$ is drawn from the size-biased joint distribution $\widetilde{\Phi}^C$ (see Section 2.2), i.e., $(\widetilde{D}^C + 1, \widetilde{R}^C)$ is distributed as the joint number of (replicas, request) of the content adjacent to an uniformly random edge of the graph.

The above result can be used to compare different replication policies based on the definition of inefficiency under optimal matching given in Equation 4.5. In Section 4.4.3, we will leverage the particular form this result takes under the popularity class model when servers do independent caching choices based on a distribution π over the different classes of content, but for now let us explain how the same approach can capture a richer model.

First-of-all, with a simple modification, Theorem 2.1 allows us to handle the case when the service capacity of servers is non-unitary: for example, let the storage and

service capacities (D^S, B^S) be drawn from a joint distribution Φ^S, and let the load ρ be defined accordingly as $\rho = \overline{\lambda}\tau/\mathbb{E}[B^S]$. Using infinite edge-capacities, Theorem 2.1 computes the limit $\mathcal{M}(\Phi^S, \Phi^C) = \lim_{m,n\to\infty} \frac{M(G_{m,n})}{m}$, which yields the inefficiency of the system using Equation 4.5.

Another restriction of the basic model which can be relaxed using the flexibility offered by Theorem 2.1 is that the contents all have the same size: assume instead that each content c can be fragmented in a number f_c of constant-size fragments. Assume further that when a server chooses to cache a segment from content c, instead of storing the raw segment it instead stores a random linear combination of all the f_c segments corresponding to content c. Then, when a user requests content c it only needs to download a coded segment from any f_c servers storing segments from c, as any f_c coded segments are sufficient to recover the content c. Note that in this context, it may also be relevant to use network codes such as in [38], however this is beyond the scope of our study here. The storage and upload capacity of servers is more appropriately measured in numbers of fragments, as well as the number of requests for each content: for a random content c, we define the random variables F^C for its number of fragments, $R'^C = R^C F^C$ for its number of fragment-requests, and D'^C for the number of servers storing a fragment of c; for a random server s, we define the random variables B'^S for the number of fragments which s can upload simultaneously and D'^S for the number of contents of which s stores a fragment (asymptotically, s will store fragments of different contents w.h.p.). Let also Φ'^C be the joint distribution of $(D'^C, R'^C, \{R^C\})$ and Φ'^S be the joint distribution of $(D'^S, B'^S, \{R_i^C\}_{i=1}^{D'^S})$, where $\{R_i^C\}_{i=1}^{D'^S})$ is a collection of D'^S independent random variables distributed as R^C. Then, we can again apply Theorem 2.1 to compute the limit $\mathcal{M}(\Phi'^S, \Phi'^C)$. The appropriate definition for the load ρ is $\rho = \tau\frac{\mathbb{E}[R'^C]}{\mathbb{E}[B'^S]}$, and the inefficiency is again obtained via Equation 4.5, although it should now be interpreted as either the fraction of fragment-requests which are directed to the data center or the fraction of servers' upload capacity which is unused.

As one can easily guess from the expressions involved, the basic model is best suited to continue our study and obtain a qualitative understanding of the behavior of the system and of the optimal replications, as we do in the next section for the popularity class model.

4.4.3 Large storage asymptotics and optimal replication

In this section, we focus on the popularity class model, because this model is more suitable to obtain a qualitative understanding of the system and the interpretation of results is more intuitive. Recall that under this model, the contents are grouped into K classes, and the caches of the servers are constituted independently of each other based only on average information over each class: for each memory slot, each server chooses first a random class of contents k according to a distribution π and then a content from class k uniformly at random. Therefore, the degree of each content of class k is a $\mathrm{Poi}(\frac{md\pi_k}{n\alpha_k})$ random variable, where $\alpha_k n$ is the number of contents in class k. Under this model, the expression of $\iota = \lim_{m,n\to\infty} \iota_{m,n}$ obtained

from Corollary 4.1 simplifies into

$$\iota = 1 - \frac{1}{\min\{1,\rho\}} \inf \mathcal{F}^S(p,q),$$

where

$$\mathcal{F}^S(p,q) = 1 - (1-p)^d + \tau \sum_{k=1}^{K} \alpha_k \lambda_k \mathbb{P}\left(\mathrm{Poi}(d_k q) > \mathrm{Poi}(\lambda_k) + 1\right), \tag{4.6}$$

with $d_k = \frac{d\pi_k}{\tau\alpha_k}$ is the expected degree of contents of class k, and the infimum is taken over pairs (p,q) satisfying

$$p = \sum_{k=1}^{K} \pi_k \mathbb{P}\left(\mathrm{Poi}(d_k q) < \mathrm{Poi}(\lambda_k)\right) \tag{4.7}$$

$$q = (1-p)^{d-1}. \tag{4.8}$$

As storage is relatively cheap, the asymptotics for $d \to \infty$ are of particular interest. Furthermore, in this regime, we can solve explicitely the above fixed-point equations on p, q to obtain an explicit expression for the inefficiency:

Proposition 4.2. *Under the multi-class model, in which within class k the number of requests is Poisson with mean λ_k and the number of replicas is Poisson with mean d_k, and assuming that $d_k = \Omega(d)$ for all $k \in 1, \ldots, K$, the inefficiency ι satisfies the following large deviation principle:*

$$\frac{\log(\iota)}{d} \overset{d\to\infty}{\to} \begin{cases} -\inf_k \frac{\pi_k}{\tau\alpha_k} & \text{if } \rho < 1, \\ \log\left(\sum_{k=1}^{K} \pi_k e^{-\lambda_k}\right) & \text{if } \rho > 1, \\ \min\left(-\inf_k \frac{\pi_k}{\tau\alpha_k}, \log\left(\sum_{i=k}^{K} \pi_k e^{-\lambda_k}\right)\right) & \text{if } \rho = 1, \end{cases} \tag{4.9}$$

First-of-all, it is quite remarkable in the proposition above that the asymptotic inefficiency as $d \to \infty$ is explained only by very *local* rare events. Indeed, $\exp(-d_k)$ is the probability that a content of class k is not replicated in any server cache, and similarly $\left(\sum_{k=1}^{K} \pi_k e^{-\lambda_k}\right)^d$ is the probability that a server stores only replicas of contents which have not been requested at all. Hence, asymptotically, the only noticeable cause for inefficiencies in under-loaded regime (when anyway not all servers can be busy) is contents that are stored nowhere; in over-loaded regime (when it is not possible to satisfy all requests) the only visible inefficiency comes from servers that store only unrequested contents; and in critically loaded systems, inefficiency is due almost only to the dominant of these two effects.

Let us then first comment the implications of this result before turning to its proof. First, we observe an exponential decay of the inefficiency in d as soon as $d_k = \Omega(d)$ for all k. This implies that the qualitative behaviour does depend on the distribution π only through its support: as long as no class of content is too much deprived, the inefficiency decays exponentially fast.

Second, it is possible to pick a distribution π that improve upon proportional replication (for which $\pi_k = \alpha_k\lambda_k$). Indeed, it follows at once from (4.9) that in

under-load $\rho < 1$, for large d, inefficiency is minimized by setting $\pi \equiv \alpha$, i.e., by not discriminating between classes. It is also easy to show that in over-load $\rho > 1$, the corresponding exponent in (4.9) is minimized by shifting replicas towards the most popular class k^* such that $\lambda_{k^*} = \max_k \lambda_k$. As for the critical case, it can be shown that the corresponding exponent is minimized by equalizing the coefficients π_k/α_k for $k \neq k^*$ in such a way that the two expressions $-\inf_k \frac{\pi_k}{\tau\alpha_k}$ and $\log\left(\sum_{i=k}^K \pi_k e^{-\lambda_k}\right)$ are equal. We summarize these statements in the following corollary:

Corollary 4.2. *Under the multi-class model, the optimal rate functions for the exponential decay of the inefficiency are given by*

$$\lim_{d\to\infty} \frac{1}{d}\log(\iota) = \begin{cases} -1/\tau & \text{if } \rho < 1, \\ -\sup_k \lambda_k & \text{if } \rho > 1, \\ -1/\tau^- & \text{if } \rho = 1, \end{cases} \tag{4.10}$$

where τ^- is the unique solution x in $[\tau, +\infty)$ of the equation

$$x = \tau \frac{\sum_k \alpha_k \left(e^{-\lambda_k} - e^{-\lambda_{k^*}}\right)}{e^{-x} - e^{-\lambda_{k^*}}}.$$

The corresponding distribution π is given by

$$\begin{array}{ll} \pi \equiv \alpha & \text{if } \rho < 1, \\ \pi_{k^*} = 1 \text{ and } \pi_k = 0 \text{ for } k \neq k^* & \text{if } \rho > 1, \\ \pi_{k^*} = 1 - \frac{\tau}{\tau^-}(1-\alpha_{k^*}) \text{ and } \pi_k = \alpha_k \frac{\tau}{\tau^-} \text{ for } k \neq k^* & \text{if } \rho = 1. \end{array} \tag{4.11}$$

Figure 4.1 illustrates the optimal exponential-decay rate functions for various replication policies in a two-class scenario, with two classes of equal sizes ($\alpha_1 = \alpha_2$), respective popularities $\lambda_1 = 3$ and $\lambda_2 = 1$, and the content/server ratio therefore governs the load of the system according to $\rho = 2\tau$. As we vary π_1, we span replication policies from uniform at $\pi_1 = 1/2$ to proportional at $\pi_1 = 0.75$ to extreme unbalance at $\pi_1 = 1$. The curves represent the inefficiency exponents for various values of d, namely $d = 2, 5, 15$ and the limit $d \to \infty$. We also indicate by a cross the optimal value of π_1 and the corresponding exponent in the limit $d \to \infty$, as characterized in the previous corollary.

We observe in particular how the optimal value for finite d approaches this limiting value: for d as small as 15, the asymptotic evaluations are already reasonably accurate. Note however that for the super-critical case, one should not take $\pi_2 = 0$ for any finite d, as is illustrated by the drop in the exponent of the right-most curve.

Proof. (of Proposition 4.2): The fraction of matched servers is given by Theorem 4.1 with the simplified expressions (4.6),(4.7) and (4.8) appearing at the beginning of this section. We now consider the fixed points of (4.7),(4.8) in the regime of large d. First, if p is bounded away from zero, by (4.7), q is exponentially small in d. Plugging this into (4.8), we find that there is indeed a fixed point such that

$$\begin{array}{rcl} p & \sim & \sum_k \pi_k \left(1 - e^{-\lambda_k}\right), \\ q & = & \left(\sum_k \pi_k e^{-\lambda_k}\right)^{d(1+o(1))}. \end{array} \tag{4.12}$$

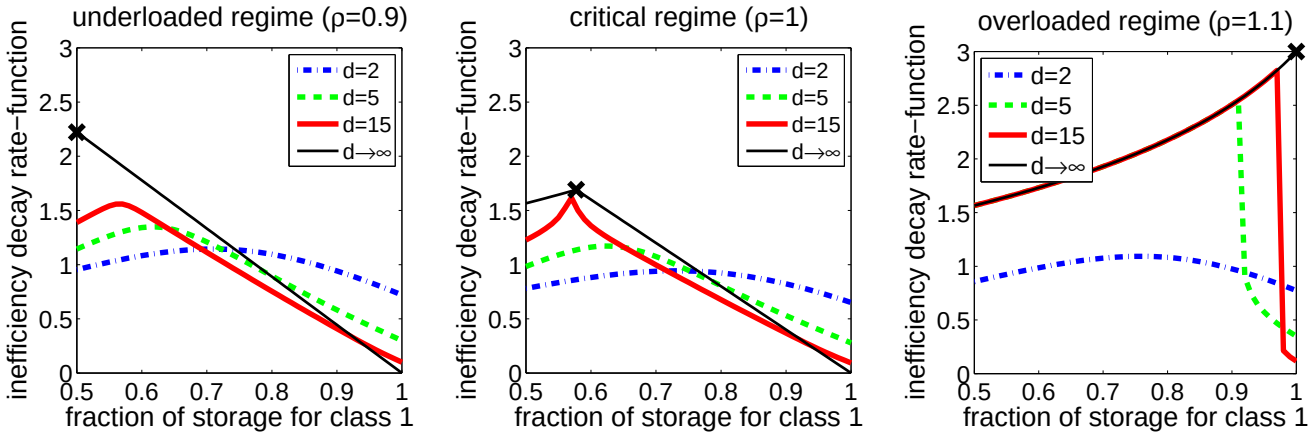

Figure 4.1: Inefficiency exponential-decay rate functions for various fractions of storage space allocated to contents of the first class in underloaded, critical, and overloaded regimes.

Next consider the case where p goes to zero with d. Then, necessarily by (4.8), qd is large, and $p = e^{-\Theta(qd)}$. Plugging this into (4.7) we obtain

$$q = (1 - e^{-\Theta(qd)})^{d-1}.$$

Assume then that $qd = O(\log(d))$. This would entail that $q \to 0$, and hence $(1 - p)^d \to 0$; the corresponding evaluationof (4.6) would be equivalent to $1 + \tau \sum_k \alpha_k \lambda_k$ which is larger than 1, and hence cannot be the minimal evaluation. We can thus assume $qd \gg \log(d)$. Then by (4.8), it follows that $p = o(1/d)$, so that by (4.7), we have $q \sim 1$. Thus, the only other meaningful fixed point to consider satisfies

$$\begin{aligned} p &= \sum_k \pi_k e^{-\frac{d}{\tau}(1+\mathrm{o}(1))} = e^{-d \inf_k \frac{\pi_k}{\tau \alpha_k}(1+\mathrm{o}(1))}, \\ q &= 1 - e^{-d \inf_k \frac{\pi_k}{\tau \alpha_k}(1+\mathrm{o}(1))}. \end{aligned} \tag{4.13}$$

It remains to evaluate Expression (4.6) at the two meaningful fixed-points (4.12) and (4.13). Consider first (4.12): the last term in (4.6) is of order $(dq)^2$, which is negligible compared to the first term $(1 - p)^d$. This yields the first evaluation

$$\mathcal{F}^S(p, q) = 1 - \exp\left(d \log(\sum_k \pi_k e^{-\lambda_k})(1 + o(1)) \right) \tag{4.14}$$

Next, we plug (4.13) into (4.6): the term $(1 - p)^d$ and the last term of (4.6) are equivalent to 1 and ρ respectively, up to corrections of order $e^{-d \inf_k \frac{\pi_k}{\tau \alpha_k}(1+\mathrm{o}(1))}$ adding up to each other, so that the corresponding evaluation is

$$\mathcal{F}^S(p, q) = \rho - \exp\left(-d \inf_k \frac{\pi_k}{\tau \alpha_k}(1 + \mathrm{o}(1)) \right). \tag{4.15}$$

To complete the proof, one then simply need to check that the evaluation of $\mathcal{F}^S(p, q)$ for the fixed-point (4.12) can only be minimal for $\rho \geq 1$, while the evaluation for the fixed-point (4.13) can only be minimal for $\rho \leq 1$. □

To sum up the results on optimal matching, we have characterized the performance of large families of replications strategies (Corollary 4.1) in a random context which is suitable because of the limited coordination capabilities of the small servers. In order to gather more intuition and discriminate between replication policies, we have considered the large storage asymptotics of the performance and observed notably that the inefficiency of the system decays exponentially fast in the storage size under any reasonable replication policy (Proposition 4.2). In this regime, we can identify features of optimal replications (Corollary 4.2) and in particular conclude that proportional replication is never optimal in terms of asymptotic decay rate function, in spite of its many good properties which have been identified in the literature (and exemplified in Proposition 4.1). In the next section, we turn to the analysis of the system under random greedy matching.

4.5 Performance under random greedy matching

This section concerns edge-assisted CDNs under a random greedy matching procedure, which simplicity guarantees its feasibility in most systems in contrast with the optimal matching policy considered before. However, the online nature of the random greedy matching algorithm, with no re-allocation of requests, may lead to performance degradations in certain regimes, which we first point out by studying a very simple model where replication strategies do not come into play as all contents have identical statistical properties. This first approach tells us that random greedy matching is only suitable in underloaded regime as it undergoes a severe loss of efficiency neer criticality, where the inefficiency does not decay exponentially fast anymore as storage increases. In underloaded regime, we use a mean-field approximation guided by large storage asymptotics to capture the impact of the replication strategy on the loss rates of the contents. This analysis then allows us to identify the large storage asymptotics of optimal replications, as well as a very simple and intuitive efficiency principle: the best way to take into account the differences in popularity of the various contents is to equalize the loss rates for all contents. Following this direction, the last section of this thesis focuses on designing adaptive replication algorithms, which use both the loss and usage information to attain the optimal replication of contents.

4.5.1 The phase transition at criticality

In order to decouple the effect on the system performance of the matching algorithm and that of the replication policy, we consider the single-class model, where all the contents have the same popularity and servers choose uniformly at random which content they store. For simplicity, we will adopt a "batch arrival" view of the system, similar to that used in [10] for example and typical in the study of balls-and-bins processes, rather than studying the stationary regime. Under this approach, for a given load $\rho = \frac{\overline{\lambda} n}{m}$ of the system, we start from an idle system and use the random greedy algorithm to handle $\overline{\lambda} n$ requests arriving successively, with no departures. Adopting a temporal view of this process, upon arrival of a request, one chooses a content uniformly at random to be the object of the request and then checks

whether there is an available server storing that particular content; if there are such servers, one of them is chosen at random and assigned to the service of the request, else the request is directed to the data center and the content is marked as being *depleted*, i.e., no further requests can be served for that content as there are no ends of service in the batch arrival model. This process stops after $\mathrm{Poi}(\overline{\lambda}n)$ arrivals of requests, so that the number of requests for each content is $\mathrm{Poi}(\overline{\lambda})$, as in stationary regime. Taking the limit $m, n \to \infty$ with $n \sim \tau m$ and focusing on the large storage asymptotics, we obtain the following result:

Proposition 4.3. *For $\rho \neq 1$, the inefficiency ι verifies*

$$\iota \leq e^{-\frac{d}{\tau}|\rho-1|(1+o(1))},$$

and in the critical regime $\rho = 1$, we have

$$\iota = \frac{\tau \log 2}{d} + o(1/d).$$

Thus, online matching incurs severe performance degradation at criticality, compared to the under-loaded and over-loaded regimes. This is in sharp contrast with optimal matching which does not suffer this singularity at $\rho = 1$. Such a result suggests that it may be necessary in certain cases to use more complex matching algorithms. Indeed, even though our analysis is oblivious of the geographical aspects of the real network, it seems likely that in practice one would want to ensure a *local* service of the requests, for reasons both of delay and bandwidth consumption. In that scenario, some areas may be overloaded, even though the global network is underloaded. Then, some requests from the heavily loaded regions will have to be transfered to servers in underloaded areas, and the overloaded regions will be artificially maintained very close to criticality. The performance in the near-critical regime is thus important, as parts of the network may be mainted in this regime due to higher level optimizations and a preference for local service.

We now turn to the proof of Proposition 4.3, which relies on approximation by differential equations of the Markov chain describing the evolution of the system (see e.g., [33]). We let the time be discrete, $t = 0$ corresponding to an empty system, and one request arriving at each unit of time. Note that due to concentration of the Poisson random variable with mean $\overline{\lambda}n$ around its mean for $n \to \infty$, it is equivalent to stop the process at the deterministic time $\overline{\lambda}n$. We let X_t^n be the number of matched servers at time t and Y_t^n be the number of contents for which at least one request has already been discarded by time t. It can be easily verified that $(X_t^n, Y_t^n)_{t\in\mathbb{N}}$ is a Markov chain. Indeed, it is enough to check that the induced subgraph on the $n - Y_t^n$ available contents and the $m - X_t^n$ unmatched servers is still distributed according to the same random graph model, i.e., every unassigned server stores exactly d available contents (because it cannot store depleted contents as those would not be depleted otherwise) and the set of contents stored by an unmatched server is chosen uniformly at random among the subsets of size d of the set of available contents.

The one-step transition probabilities of the Markov chain are as follows:

$$(X^n_{t+1}, Y^n_{t+1}) = \begin{cases} (X^n_t, Y^n_t) & \text{w.p. } \frac{Y^n_t}{n} \\ (X^n_t, Y^n_t + 1) & \text{w.p. } \frac{n-Y^n_t}{n} p_{x,y} \\ (X^n_t + 1, Y^n_t) & \text{w.p. } \frac{n-Y^n_t}{n}(1 - p_{x,y}) \end{cases}$$

where $p_{x,y}$ is the probability that no unmatched server stores a particular available content given $X^n_t = x$ and $Y^n_t = y$. We have

$$p_{x,y} = \left(\frac{\binom{n-y-1}{d}}{\binom{n-y}{d}} \right)^{m-x} \underset{n\to\infty}{\sim} e^{-d\frac{m-x}{n-y}}.$$

In the limit $n \to \infty$, we use mean field techniques to approximate the Markov chain $(X^n_t, Y^n_t)_{t\in\mathbb{N}}$ by the solution of differential equations. We define $x(t)$ and $y(t)$ as candidate approximations of $\frac{X^n_{tn}}{n}$ and $\frac{Y^n_{tn}}{n}$ respectively, by setting $x(0) = y(0) = 0$, and $x(t), y(t)$ are given by the following differential equations:

$$\dot{x} = (1-y)\left(1 - e^{-d\frac{1/\tau - x}{1-y}}\right) \tag{4.16}$$

$$\dot{y} = (1-y)e^{-d\frac{1/\tau - x}{1-y}} \tag{4.17}$$

Classical results of Kurtz [68] then imply the following

Lemma 4.1. *Almost surely, we have*

$$\lim_{n\to\infty} \frac{1}{n}(X^n_{\overline{\lambda}n}, Y^n_{\overline{\lambda}n}) = (x(\overline{\lambda}), y(\overline{\lambda})).$$

According to Equation (4.1), the limiting inefficiency ι of the random greedy algorithm is thus given by

$$\iota = 1 - \frac{\tau x(\overline{\lambda})}{\min\{1, \rho\}}.$$

While the above differential equations do not admit closed form solutions, it is possible to derive explicit upper and lower bounds based on more tractable ODE's, which become accurate for large d. Let thus $u = d\frac{1/\tau - x}{1-y}$, we obtain

$$\dot{u} = ue^{-u} + de^{-u} - d \tag{4.18}$$

$$\iota = 1 - \frac{1}{\min\{1,\rho\}}\left(1 + \frac{\tau u(\overline{\lambda})}{d}(1 - y(\overline{\lambda}))\right)$$

To find a good surrogate for u, note first that u is a decreasing function, because $e^u\dot{u} = u + d - de^u \le -(d-1)u$. Also, the term ue^{-u} is always small compared to the others when d is large.

It suggests that the behavior of u for large d should be captured by the solution to the following equation:

$$\dot{v} = d(e^{-v} - 1) \tag{4.19}$$

$$v(0) = u(0) = d/\tau$$

A change of variables leads to

$$v(t) = \log\left(1 + e^{d(1/\tau - t)}(1 - e^{-d/\tau})\right)$$

Note that, at all time, we have $v \leq u$, because the functions are continuous and $\dot{v} \leq \dot{u}$ whenever $v = u$. We then have

Lemma 4.2. *In the regime $\rho < 1$, the solution of Equations (4.16)-(4.17) satisfies*

$$\iota \leq (1 + 1/\tau)e^{-\frac{d}{\tau}(1-\rho)}.$$

Proof. At all time, $\frac{d}{\tau} \geq u \geq \frac{d}{\tau}(1-\rho)$, thus $\dot{u} \leq d\left((1+1/\tau)e^{-\frac{d}{\tau}(1-\rho)} - 1\right)$. We immediately obtain

$$\begin{array}{l} u(\overline{\lambda}) \leq \frac{d}{\tau}(1-\rho) + d\overline{\lambda}(1+1/\tau)e^{-\frac{d}{\tau}(1-\rho)}, \\ \iota = 1 - \frac{1}{\rho}\left(1 + \frac{\tau}{d}u(\overline{\lambda})(1 - y(\overline{\lambda})) \leq (1+1/\tau)e^{-\frac{d}{\tau}(1-\rho)}\right). \end{array}$$

□

Lemma 4.3. *In the regime $\rho = 1$, the solution of Equations (4.16)-(4.17) satisfies*

$$\iota \leq \frac{\tau \log 2}{d} + o\left(1/d\right).$$

Proof. Let $c > 1$. $u(0) = d/\tau$ so there exists $0 < T_d \leq \overline{\lambda}$ such that $u \geq c \log d$ on $[0, T_d]$ and $u \leq c \log d$ on $(T_d, \overline{\lambda}]$ for large enough d. Furthermore, $\dot{u} \geq -d$, so we always have $T_d \geq 1/\tau - \frac{c \log d}{d}$ for d large enough. Then, on $[0, T_d]$, we have

$$\dot{u} \leq d^{1-c}/\tau + d(e^{-u} - 1) \leq d^{1-c}/\tau + \dot{v}$$

and, on $(T_d, \rho]$,

$$\dot{u} \leq c \log d + \dot{v}$$

We obtain an upper-bound on $u(\overline{\lambda})$:

$$\begin{array}{rcl} u(\overline{\lambda}) & \leq & v(\overline{\lambda}) + T_{\frac{d}{\tau}} d^{1-c} + (\overline{\lambda} - T_d)c \log d \\ & \leq & \log\left(2 - e^{-\frac{d}{\tau}}\right) + \frac{d^{1-c}}{\tau}\overline{\lambda} + \frac{c^2 \log^2 d}{d}. \end{array}$$

As $c > 1$,

$$\begin{array}{rcl} \iota & \leq & \frac{\tau \log 2}{d} + d^{-c}\rho + \frac{c^2\tau^2 \log^2 d}{d^2} \\ & = & \frac{\tau \log 2}{d} + o(1/d) \end{array}$$

□

Lemma 4.4. *In the regime $\rho = 1$, the solution of Equations (4.16)-(4.17) satisfies*

$$\iota \geq \frac{\tau \log 2}{d} + o\left(1/d\right).$$

Proof. As $y + x \leq \overline{\lambda}$, it follows that $y \leq \frac{\iota}{\tau} \leq \frac{\log 2}{d} + o(1/d)$. Hence,

$$\begin{aligned} \iota &\geq \frac{\tau}{d} \log\left(2 - e^{-\frac{d}{\tau}}\right) \left(1 - \frac{\log 2}{d} + o(1/d)\right) \\ &= \frac{\tau \log 2}{d} + o(1/d). \end{aligned}$$

□

As upper- and lower-bound coincide, we actually have that $\iota = \frac{\tau \log 2}{d} + o(\frac{1}{d}) \underset{d\to\infty}{\sim} v(\overline{\lambda})$.

Lemma 4.5. *In the regime $\rho > 1$, the solution of Equations (4.16)-(4.17) satisfies*

$$\iota \leq \frac{\tau \log 2}{d^2} e^{-\frac{d-1}{\tau}(\rho-1)}(1 + o(1)).$$

Proof. We already know that $u(1/\tau) = \frac{\tau \log 2}{d} + o(\frac{1}{d})$, so we can focus on the time interval $[1/\tau, \overline{\lambda}]$. For d large enough and for all $t \geq 1/\tau$, as u is decreasing, u is strictly less than 1. Then,

$$\dot{u} \leq u + d(1 - u + \frac{u^2}{2} - 1) = -(d-1)u + d\frac{u^2}{2}$$

As u is actually much smaller than 1 in the range $[1/\tau, \overline{\lambda}]$, we can guess that the influence of the term in u^2 will be small compared to that of the term in u, and that we did not lose much by neglecting higher order terms.

Let z such that $z(1/\tau) = u(1/\tau)$ and $\dot{z} = -(d-1)z + d\frac{z^2}{2}$. At $u = z$, we have $\dot{u} \leq \dot{z}$, so $u \leq z$ on $[1/\tau, \overline{\lambda}]$, and z is decreasing on $[1/\tau, \overline{\lambda}]$ for d large enough.

As long as $z \neq 0$ and $z \neq 2(1 - \frac{1}{d})$,

$$\dot{z} = -(d-1)z + d\frac{z^2}{2} \Leftrightarrow \frac{\dot{z}}{d-1}\left(\frac{1}{z - 2(1-1/d)} - \frac{1}{z}\right) = 1$$

Integrating from $\overline{\lambda}$ to $1/\tau$, we obtain

$$\begin{aligned} \tfrac{1}{d-1}\left[\log\left(z - 2(1-1/d)\right)\right]_{1/\tau}^{\overline{\lambda}} - \tfrac{1}{d-1}\left[\log z\right]_{1/\tau}^{\overline{\lambda}} &= \overline{\lambda} - 1/\tau \\ \tfrac{1}{d-1}\log\frac{1 - \frac{z(\overline{\lambda})}{2(1-1/d)}}{1 - \frac{u(1/\tau)}{2(1-1/d)}} + \tfrac{1}{d-1}\log\frac{u(1/\tau)}{z(\overline{\lambda})} &= \overline{\lambda} - 1/\tau. \end{aligned}$$

As $0 < z(\overline{\lambda}) < u(1/\tau) = \frac{\log 2}{d} + o(\frac{1}{d})$, the first term is $o(1)$. Thus, we obtain

$$\begin{aligned} u(\overline{\lambda}) &\leq z(\overline{\lambda}) = u(1/\tau)e^{-\frac{d-1}{\tau}(\rho-1)+o(1)} \\ &= \frac{\log 2}{d} e^{-\frac{d-1}{\tau}(\rho-1)}(1 + o(1)). \end{aligned}$$

and also

$$\iota \leq \frac{\tau \log 2}{d^2} e^{-\frac{d-1}{\tau}(\rho-1)}(1 + o(1))$$

This completes the proof of the last lemma of Proposition 4.3. □

Given the lengthy calculations involved to compute the asymptotics of the inefficiency for large storage under a single-class model, we do not pursue this approach under a more complex model, with different content popularities. Instead, in the next section, we turn to mean-field approximations to understand the relationship between the replication of a content and its loss rate in the stationary regime.

4.5.2 Mean-field approximation for the loss-rate

In this section, we propose an approximation to understand in a precise manner the relation between any fixed replication of contents and the loss rates in the system. This analytical step has many advantages: it allows us to formally demonstrate that to optimize the system one needs to make the loss rates equal for all the contents; as a consequence we obtain an explicit expression for the optimal replication (Section 4.5.3); finally, in Section 4.6.2, we will leverage our analytical expression for the full distribution of the number of available replicas of a content to propose a mechanism enhancing the speed of adaptive algorithms. We validate our approximation and show that our optimized replication strategy largely outperforms proportional replication through extensive simulations.

For a given fixed constitution of the caches (i.e., a fixed graph G), the system is Markovian, the minimum state space indicating which servers are busy (it is not necessarily to remember which content they serve). We want to understand how the loss rate γ_c for a particular content c relates to the graph G, but using too detailed information about G, such as the exact constitution of the caches containing c, would have the drawback that it would not lead to a simple analysis when considering adaptive replication policies (as the graph G would then keep changing). Therefore, we need to obtain a simple enough but still accurate approximate model tying γ_c to G. Here, we make no assumption on the popularity distributions of the contents. We work in underloaded regime $\rho < 1$ as this is the most interesting regime and furthermore the previous section showed random greedy matching is not suited for critical regime.

The expected loss rate γ_c of content c is equal to its requests arrival rate λ_c multiplied by the steady state probability that c has no replicas available. Let D_c be the total number of replicas of content c and Z_c be its number of *available* replicas, i.e., those stored on a currently idle server. We thus want to compute $\pi(Z_c = 0)$ to get access to γ_c. However, the Markov chain describing the system is too complicated to be able to say much on its steady state. In order to simplify the system, one can remark that in a large such system the state of any fixed number of servers (i.e., their current caches and whether they are busy or idle) are only weakly dependent, and similarly the number of available replicas Z_c of any fixed number of contents are only weakly dependent. Therefore, it is natural to approximate the system by decoupling the different contents and the different servers (similar phenomenon are explained rigorously in [100]). In other words, this amounts to forgetting the exact constitution of the caches; as a result, the correlation between contents which are stored together is lost. Then, the evolution of Z_c becomes a one dimensional Markov chain, independent of the values of $Z_{c'}$ for other contents, and we can try to compute its steady-state distribution.

For any $z < D_c$, the rate of transition from $Z_c = z$ to $z+1$ is always $D_c - z$: it is the rate at which one of the $D_c - z$ occupied servers storing c completes its current service; we do not need to distinguish whether a server is actually serving a request for c or for another content c' as we assume the service times are independent and identically distributed across contents. For any $z > 0$, the transitions from $Z_c = z$ to $z - 1$ are more complicated. They happen in the two following cases:

- either a request arrives for content c and as $Z_c = z > 0$ it is assigned an idle servers storing c;
- or a request arrives for another content c' and it is served by a server which also stores content c.

The first event occurs at rate λ_c and the second one at expected rate

$$\sum_{c' \neq c} \lambda_{c'} \mathbb{E}\left[\frac{|\{s \in S : \ s \text{ is idle and } c, c' \in s\}|}{|\{s \in S : \ s \text{ is idle and } c' \in s\}|}\right],$$

where $c \in s$ indicates that the content c is stored on the server s. At any time and for any c', $|\{s \in S : \ s \text{ is idle and } c' \in s\}|$ is equal to $Z_{c'}$. The term $|\{s \in S : s \text{ is idle and } c, c' \in s\}|$ is equal in expectation to the number of idle servers storing c' (i.e., Z'_c) times the probability that such a server also stores c. As we forget about the correlations in the caching, this last probability is approximated as the probability to pick one of the $d-1$ remaining memory slots in an idle servers storing c' when we dispatch at random the Z_c available replicas of content c between all the remaining slots in idle servers. Thus,

$$\mathbb{E}\left[|\{s \in S : \ s \text{ is idle and } c, c' \in s\}|\right] \approx \frac{Z_{c'}(d-1)}{(1-\rho_{\text{eff}})md} Z_c,$$

where $\rho_{\text{eff}} = \rho(1 - \overline{\gamma}/\overline{\lambda})$ is the average load effectively absorbed by the system, so that the total number of memory slots on the idle servers is $(1-\rho_{\text{eff}})md$. We also neglected the $Z_{c'}$ memory slots occupied by c' in these idle servers when computing the total number of idle slots $(1-\rho_{\text{eff}})md$. We obtain the following approximation for the expected rate at which the second event occurs:

$$Z_c \sum_{c' \neq c} \lambda_{c'} \frac{d-1}{(1-\rho_{\text{eff}})md} \approx Z_c \frac{\rho_{\text{eff}}}{1-\rho_{\text{eff}}} \frac{d-1}{d} = Z_c \theta_{\text{eff}},$$

where we neglected the rate λ_c at which requests arrive for content c compared to the aggregate arrival rate of requests, and we let $\theta_{\text{eff}} = \frac{\rho_{\text{eff}}}{1-\rho_{\text{eff}}} \frac{d-1}{d}$. Note that the interesting regime, with reasonably high effective load ρ_{eff}, corresponds to large values of θ_{eff}, as $\rho_{\text{eff}} \to 1$ implies $\theta_{\text{eff}} \to \infty$. The Markov chain obtained satisfies the local balance equations: for $z < D_c$,

$$(D_c - z)\pi(Z_c = z) = (\lambda_c + (z+1)\theta_{\text{eff}})\pi(Z_c = z+1),$$

This yields the following steady-state probability:

$$\pi(Z_c = z) = \pi(Z_c = D_c)\theta_{\text{eff}}^{D_c - z} \binom{D_c + \lambda_c/\theta_{\text{eff}}}{D_c - z}, \tag{4.20}$$

where the binomial coefficient does not really have a combinatorial interpretation as one of its arguments is not an integer, but should only be understood as $\binom{k+x}{l} = \frac{1}{l!}\prod_{i=k-l+1}^{k}(i+x)$, for $k, l \in \mathbb{N}$ and $x \in \mathbb{R}_+$.

We now have an approximation for the full distribution of the number Z_c of available replicas of content c, which yields an approximation for the loss rate $\gamma_c = \lambda_c \pi(Z_c = 0)$. We can also compute the mode $\widehat{z}_c$ of this distribution: $\widehat{z}_c \approx \frac{D_c - \lambda_c}{1+\theta_{\text{eff}}}$, which can be used as an approximation for the expectation $\mathbb{E}[Z_c]$ (in fact, simulations show the two are nearly indistinguishable). We further simplify the expression obtained by providing a good approximation for the normalization factor $\pi(Z_c = D_c)$ in Equation (4.20):

$$\pi(Z_c = D_c) = \left(\sum_{x=0}^{D_c} \theta_{\text{eff}}^x \binom{D_c + \lambda_c/\theta_{\text{eff}}}{x} \right)^{-1}.$$

To that end, we use the fact that we aim at small loss rates, and thus most of the time at small probabilities of unavailability. Therefore, the bulk of the mass of the distribution of Z_c should be away from 0, which means that the terms for x close to D_c should be fairly small in the previous expression. We thus extend artificially the range of the summation in this expression and approximate $\pi(Z_c = D_c)$ as follows:

$$\begin{aligned} \pi(Z_c = D_c)^{-1} &\approx \sum_{x=0}^{D_c + \lfloor \lambda_c/\theta_{\text{eff}} \rfloor} \theta_{\text{eff}}^x \binom{D_c + \lambda_c/\theta_{\text{eff}}}{x} \\ &\approx (1+\theta_{\text{eff}})^{D_c + \lambda_c/\theta_{\text{eff}}}. \end{aligned}$$

We obtain the following approximation for $\pi(Z_c = 0)$:

$$\pi(Z_c = 0) \approx \left(\frac{\theta_{\text{eff}}}{1+\theta_{\text{eff}}} \right)^{D_c} (1+\theta_{\text{eff}})^{-\frac{\lambda_c}{\theta_{\text{eff}}}} \binom{D_c + \lambda_c/\theta_{\text{eff}}}{D_c}.$$

Finally, as we are interested in large systems and large storage / replication, we can leverage the following asymptotic behavior:

$$\binom{k+x}{k} \approx \frac{e^{x(H_k - \mathrm{C}_{\text{Euler}})}}{\Gamma(x+1)},$$

for $k \in \mathbb{N}$ large enough and where Γ is the gamma function: $\Gamma(1+x) = \int_0^{+\infty} t^x e^{-t} \mathrm{d}t$; H_k is the k-th harmonic number: $H_k = \sum_{i=1}^{k} \frac{1}{i}$; and $\mathrm{C}_{\text{Euler}}$ is the Euler-Mascheroni constant: $\mathrm{C}_{\text{Euler}} = \lim_{k\to\infty}(H_k - \ln k) \approx 0.57721$. For large number of replicas D_c, we thus obtain the approximation:

$$\pi(Z_c = 0) \approx \mathrm{C}(\lambda_c) \left(1 + \frac{1}{\theta_{\text{eff}}} \right)^{-D_c} D_c^{\frac{\lambda_c}{\theta_{\text{eff}}}},$$

where we let $\mathrm{C}(\lambda_c) = \frac{e^{-\mathrm{C}_{\text{Euler}} \lambda_c/\theta_{\text{eff}}}}{(1+\theta_{\text{eff}})^{\frac{\lambda_c}{\theta_{\text{eff}}}} \Gamma(1+\frac{\lambda_c}{\theta_{\text{eff}}})}$ and the term $D_c^{\frac{\lambda_c}{\theta_{\text{eff}}}}$ is an approximation of $e^{\frac{\lambda_c}{\theta_{\text{eff}}} H_{D_c}}$. The approximation for $\pi(Z_c = 0)$ immediately yields an approximation for the loss rate γ_c:

$$\gamma_c = \lambda_c \pi(Z_c = 0) \approx \lambda_c \mathrm{C}(\lambda_c) \left(1 + \frac{1}{\theta_{\text{eff}}} \right)^{-D_c} D_c^{\frac{\lambda_c}{\theta_{\text{eff}}}}. \tag{4.21}$$

Note that the expression of θ_{eff} involves the average effective load ρ_{eff}, which itself depends on the average loss rate $\overline{\gamma}$. We thus have a fixed point equation in $\overline{\gamma}$ (just as in [28, 44]), which we can easily solve numerically. Indeed, the output value $\overline{\gamma}$ of our approximation is a decreasing function of the input value $\overline{\gamma}$ used in ρ_{eff}, which implies simple iterative methods converge exponentially fast to a fixed-point value for $\overline{\gamma}$.

4.5.3 Large storage asymptotics and optimized replication

In this section, we exploit the approximation obtained in Equation (4.21) to understand which replication strategy minimizes the total loss rate in the system. In other words, we approximately solve the optimization problem:

$$\min \quad \overline{\gamma} \quad \text{s.t.} \begin{array}{l} \sum_c D_c \leq md \\ D_c \in \mathbb{N},\ \forall c \end{array} \tag{4.22}$$

Note that the approximation from Equation (4.21) is consistent with the intuition that γ_c is a decreasing, convex function of D_c. Indeed, letting $x = \frac{\theta_{\text{eff}}}{1+\theta_{\text{eff}}} e^{\frac{\lambda_c}{\theta_{\text{eff}}(D_c+1)}} \leq 1$ since we have $D_c + 1 \geq \frac{\lambda_c}{\theta_{\text{eff}} \ln\left(1+\frac{1}{\theta_{\text{eff}}}\right)}$ in the regime of interest, we compute

$$\begin{aligned} \gamma_c(D_c+1) - \gamma_c(D_c) = \Delta\gamma_c(D_c) &= -\gamma_c(D_c)\,(1-x) \leq 0, \\ \Delta\gamma_c(D_c-1) &= -\gamma_c(D_c)\left(\tfrac{1}{x}-1\right) \leq 0. \end{aligned}$$

The loss rate γ_c is a convex function of D_c as shown by

$$\Delta\gamma_c(D_c) - \Delta\gamma_c(D_c-1) \geq \gamma_c(D_c)\left(x + \frac{1}{x} - 2\right) \geq 0.$$

As a consequence, the optimization problem (4.22) is approximately convex, and we thus obtain an approximate solution by making the first derivatives of the loss rates $\Delta\gamma_c$ equal. Keeping only the dominant orders in D_c, we have

$$\Delta\gamma_c(D_c) = -\frac{\gamma_c}{1+\theta_{\text{eff}}}\left(1 - \frac{\lambda_c}{D_c}\right). \tag{4.23}$$

In the first order, equalizing the first derivatives in (4.23) using the expression in (4.21) leads to equalizing the number of replicas for every content, i.e., setting $D_c = \overline{D} + o(\overline{D}))$ where $\overline{D} = \frac{md}{n}$ is the average number of replicas of contents. Going after the second order term, we get

$$D_c = \overline{D} + (\lambda_c - \overline{\lambda})\frac{\ln \overline{D}}{\theta_{\text{eff}} \ln\left(1 + 1/\theta_{\text{eff}}\right)} + o(\ln \overline{D}). \tag{4.24}$$

We therefore obtain that the asymptotically optimal replication is uniform with an adjustment due to popularity of the order of the logarithm of the average number of replicas. Finally, inserting back this expression for D_c into Equation (4.21) yields

$$\gamma_c = \left(1 + 1/\theta_{\text{eff}}\right)^{-\overline{D}}\, \overline{D}^{\frac{\overline{\lambda}}{\theta_{\text{eff}}}(1+o(1))}, \tag{4.25}$$

which shows that the average loss rate $\overline{\gamma}$ under optimized replication behaves as the loss rate of an imaginary average content (one with popularity $\overline{\lambda}$ and replication $\overline{D}$).

Properties	class 1	class 2	class 3
number of contents n_i	200	400	400
popularity λ_i	9	3	1
number of replicas D_i	200	67	23
simulation data / approximation	class 1	class 2	class 3
nb. of available replicas $\mathbb{E}[Z_i]$	21.7 / 21.6	7.28 / 7.25	2.51 / 2.50
$10^3\times$ loss rates γ_i	0 / 10^{-5}	3.31 / 2.36	79.4 / 76.3
simulation data / approximation	Zipf, $\alpha = 0.8$	Zipf, $\alpha = 1.2$	
proportional: $10^3\times$ inefficiency	5.69 / 5.46	11.8 / 11.6	

Table 4.1: Properties of the content classes, and numerical results on the accuracy of the approximation.

4.5.4 Assessment of the approximation accuracy via simulations

In the process of approximating the loss rate γ_c of a content c, we performed many approximations based on asymptotic behaviors for large systems and large storage / replication. It is therefore necessary to check whether the formula of Equation (4.21) is not too far off, which we do in this section via simulation. The systems we simulate are of quite reasonable size (with a few hundred contents and servers). However, the accuracy of the approximation should only improve as the system grows. We use two scenarios for the popularity distribution of the contents: the popularity class model, which allows us to compare between many contents with similar characteristics, and a Zipf popularity model, which is deemed more realistic. We evaluate the accuracy of the approximation using proportional replication (other replications were also simulated and yield similar results). To compute the approximate expressions, we solve numerically the fixed point equation in $\overline{\gamma}$. We simulate systems under a reasonably high load of 0.9. As often, it is mainly interesting to know how the system behaves when the load is high, as otherwise its performance is almost always good. However, as requests arrive and are served stochastically, if the average load were too close to 1 then the instantaneous load would exceed 1 quite often, which would automatically induce massive losses and mask the influence of the replication. In fact, it is easy to see that we need to work with systems with a number of servers m large compared to $\frac{\rho}{(1-\rho)^2}$ in order to mitigate this effect.

The setup with the class model is the following: there are $n = 1000$ contents divided into $K = 3$ classes; the characteristics of the classes are given in the first part of Table 4.1. The popularities in Table 4.1 together with $n = 1000$ and $\rho = 0.9$ result in $m = 3800$ servers. Each server can store $d = 20$ contents, which is 2% of the catalog of contents. We let the system run for 10^4 units of time, i.e., contents of class 3 should have received close to 10^4 requests each.

Figure 4.2 and the second part of Table 4.1 show the results for the class model: the left part of Figure 4.2 shows the distributions of the numbers of available replicas for all the contents against the approximation from Equation (4.20) for each class; the right part shows the loss rates for all the contents against the approximation

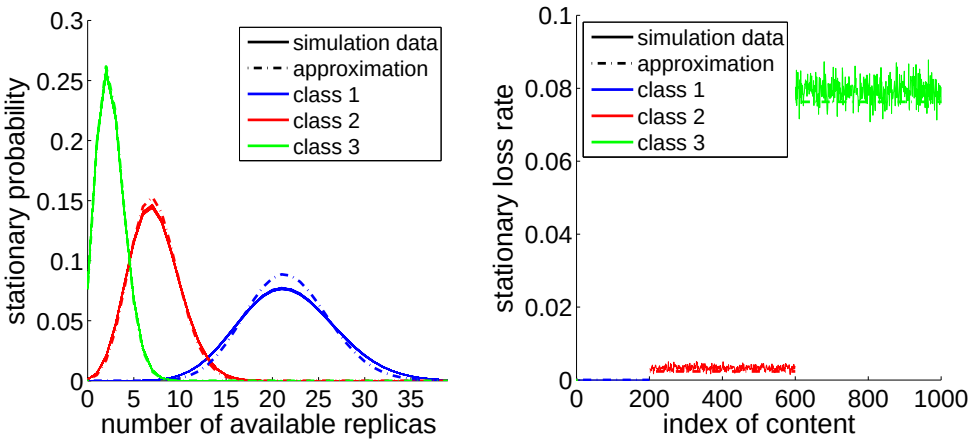

Figure 4.2: Class model: distribution of number of available replicas and loss rates Vs approximation.

from Equation (4.21) for each class. Although it is not apparent at first sight, the left part of Figure 4.2 actually displays a plot for each of the 1000 contents, but the graphs for contents of the same class overlap almost perfectly, which supports our approximation hypothesis that the stationary distribution of the number of available replicas is not very dependent on the specific servers on which a content is cached or on the other contents with which it is cached. This behavior is also apparent on the right part of Figure 4.2, as the loss rates for the contents of the same class are quite similar. From Figure 4.2 and Table 4.1, it appears that the approximations from Equations (4.20) and (4.21) are quite accurate, with for example a relative error of around 5% in the inefficiency of the system (9.68×10^{-3} Vs 9.20×10^{-3}). We consider such an accuracy is quite good given that some of the approximations done are based on a large storage / replication asymptotic, while the simulation setup is with a storage capacity of $d = 20$ contents only and contents of class 3 (responsible for the bulk of losses) have only 23 replicas each.

We now turn to the Zipf popularity model. In this model, the contents are ranked from the most popular to the least; for a given exponent parameter α, the popularity of the content of rank i is given by $\lambda_i = \frac{i^{-\alpha}}{\sum_{j \leq n} j^{-\alpha}} \overline{\lambda}$. We use two different values for the Zipf exponent α, 0.8 and 1.2, as proposed in [45]. The exponent 0.8 is meant to represent correctly the popularity distribution for web, file sharing and user generated contents, while the exponent 1.2 is more fit for video-on-demand, which has a more accentuated popularity distribution. We simulate networks of $n = 200$ contents and $m \approx 2000$ servers of storage capacity $d = 10$ under a load $\rho = 0.9$. This yields an average content popularity $\overline{\lambda} = \rho \frac{m}{n} \approx 9$. Note that under proportional replication the numbers of replicas of the most popular contents are actually larger than the number of servers m for $\alpha = 1.2$; we thus enforce that no content is replicated in more than 95% of the servers. As expected and confirmed by the simulations, this is anyway benefical to the system. Each setup is simulated for at least 10^4 units of time.

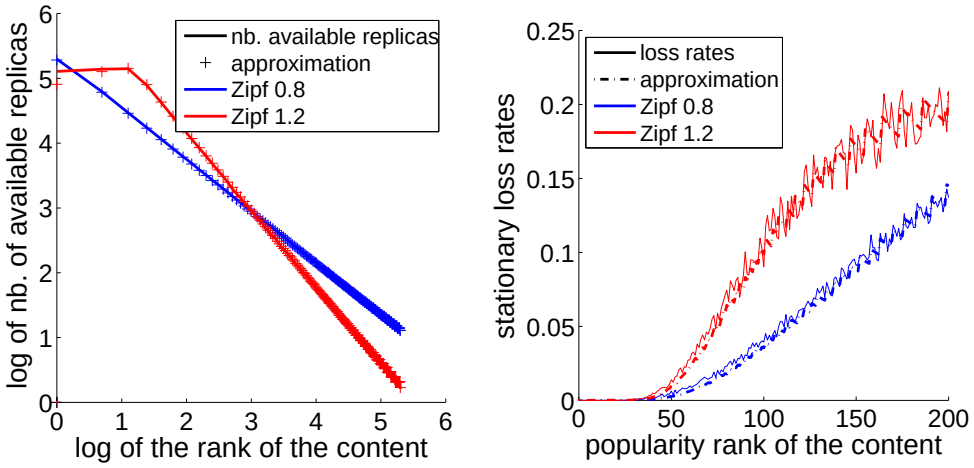

Figure 4.3: Zipf model: expected number of available replicas and loss rates Vs approximation.

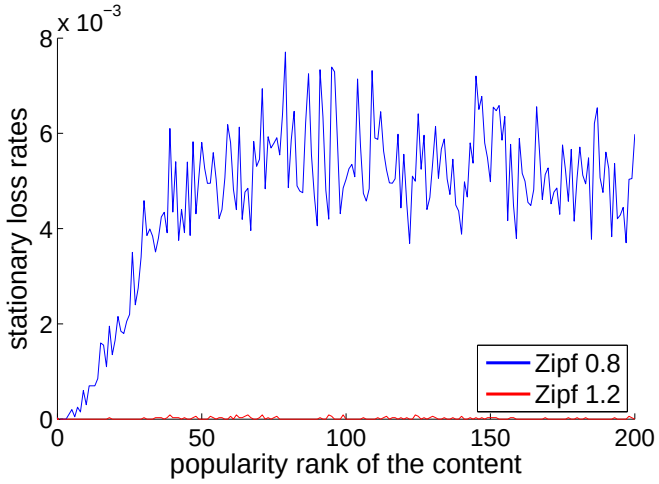

Figure 4.4: Zipf model: loss rates under optimized replication.

We show the results for the Zipf model with both exponent values in Figure 4.3 and in the third part of Table 4.1: the left part of Figure 4.3 shows the expected number of available replicas for all the contents against the approximation from Equation (4.20); the right part shows the loss rates for all the contents against the approximation from Equation (4.21). Again, the results from both Figure 4.3 and Table 4.1 confirm the accuracy of the approximations.

We now turn to evaluating the performance of the optimized replication from Equation (4.24). Figure 4.4 shows the resulting loss rates under the optimized replication, with both Zipf exponents. This figure is to be compared with the right part of Figure 4.3, which showed the same results for proportional replication. It is clear that the optimized replication succeeds at reducing the overall inefficiency $\overline{\gamma}/\overline{\lambda}$ compared to proportional replication (from 5.7×10^{-3} to 5.1×10^{-4} for $\alpha = 0.8$, and from 1.2×10^{-2} to 1.6×10^{-6} for $\alpha = 1.2$). Note that in the case of Zipf exponent

$\alpha = 1.2$ the popularity distribution is more "cacheable" as it is more accentuated, and the optimized replication achieves extremely small loss rates. However, the loss rates for all the contents are not trully equalized, as popular contents are still too much favored (as can be seen for Zipf exponent $\alpha = 0.8$). This may be because the expression for optimized replication is asymptotic in the system size and storage capacity. Hence, there is still room for additional finite-size corrections to the optimized replication.

As an outcome of this section, we conclude that the approximations proposed are accurate, even at reasonable system size. In addition, the optimized scheme largely outperforms proportional replication. In the next section, we derive adaptive schemes equalizing the loss rates of all the contents and achieving similar performances as the optimized replication.

4.6 Adaptive replication schemes to minimize losses

In a practical system, we would want to adapt the replication in an online fashion as much as possible, as it provides more reactivity to variations in the popularity of contents. Such variations are naturally expected for reasons such as the introduction of new contents in the catalog or a loss of interest for old contents. Thus, blindly enforcing the replication of Equation (4.24) is not always desirable in practice. Instead, we can extract general principles from the analysis performed in Section 4.5.2 to guide the design of adaptive algorithms.

In this section, we first show how basic adaptive rules from cache networks can be translated into our context based on the insights from Section 4.5.2 to yield adaptive algorithms minimizing the overall loss rate. Then, we show how the more detailed information contained in Equation (4.20) allows the design of faster schemes attaining the same target replication. Finally, we validate the path followed by evaluating through simulations the adaptive schemes proposed.

4.6.1 Simple caching + eviction algorithms reacting to losses

The analysis in Section 4.5.3 shows that the problem of minimizing the average loss rate $\overline{\gamma}$ is approximately a convex problem, and that therefore one should aim at equalizing the derivatives $\Delta\gamma_c(D_c)$ of the stationary loss rates of all the contents. In addition, Equation (4.23) points out that these derivatives are proportional to $-\gamma_c$ in the limit of large replication / storage, and thus equalizing the loss rates should provide an approximate solution for the optimization problem. An immediate remark at this point is that it is unnecessary to store contents with very low popularity λ_c if the loss rate of the other contents is already larger than λ_c.

An adaptive replication mechanism is characterized by two rules: a rule for creating new replicas and another one for evicting contents to make space for the new replicas. In order to figure out how to set these rules, we analyse the system in the fluid limit regime, with a separation of timescales such that the dynamics of the system with fixed replication have enough time to converge to their steady-state between every modification of the replication. Achieving this separation of timescales in practice would require slowing down enough the adaptation mechanism,

which reduces the capacity of the system to react to changes. Therefore, we keep in mind such a separation of timescales as a justification for our theoretical analysis but we do not slow down our adaptive algorithms.

When trying to equalize the loss rates, it is natural to use the loss events to trigger the creation of new replicas. Then, new replicas for content c are created at rate γ_c, and we let η_c be the rate at which replicas of c are deleted from the system. In the fluid limit regime under the separation of timescale assumption, the number of replicas of c evolves according to $\dot{D}_c = \gamma_c - \eta_c$, where all the quantities refer to expectations in the steady-state of the system with fixed replication. At equilibrium, we have $\dot{D}_c = 0$ and $\gamma_c = \eta_c$ for all the contents c, thus we need $\eta_c = \text{Cste}$ for all c to equalize the loss rates. This would be achieved for example if we picked a content for eviction uniformly at random among all contents. However, the contents eligible for eviction are only those which are available (although some systems may allow modifying the caches of busy servers, as serving requests consumes upload bandwidth while updating caches requires only download bandwidth). Therefore, the most natural and practical eviction rule is to pick an *available* content uniformly at random (hereafter, we refer to this policy simply as the RANDOM policy). Then, the eviction rate for content c is given by $\eta_c \propto \pi(Z_c > 0) = 1 - \frac{\gamma_c}{\lambda_c}$. So, at equilibrium, we can expect to have $\frac{\gamma_c}{1-\frac{\gamma_c}{\lambda_c}} = \text{Cste}$, $\forall c \in C$. In a large system, with large storage / replication and loss rates tending to zero, the difference with a replication trully equalizing the loss rates is negligeable. If we are confronted to a system with a large number of very unpopular contents though, we can compensate for this effect at the cost of maintaining additional counters for the number of evictions of each content.

Once the rule for creating replicas if fixed, we immediately obtain a family of adaptive algorithms by modifying the eviction rules from the cache network context as we did above for the RANDOM policy. Instead of focusing on "usage" and the incoming requests process as in cache networks with the LFU and LRU policies, we react here to the loss process. This yields the LFL (least frequently lost) and LRL (least recently lost) policies. RANDOM, LRL, and LFL are only three variants of a generic adaptive algorithm which performs the following actions at every loss for content c:

1. create an empty slot on an available server, using the eviction rule (RANDOM / LRL / LFL);

2. add a copy of the content c into the empty slot.

The three eviction rules considered here require a very elementary centralized coordinating entity. For the RANDOM rule, this coordinator simply checks which contents are available, picks one uniformly at random, say c', and then chooses a random idle server s storing c'. The server s then picks a random memory slot and clears it, to store instead a copy of c. In the case of LRL, the coordinator needs in addition to maintain an ordered list of the least recently lost content (we call such a list an LRL list). Whenever a loss occurs for c, the coordinator picks the least recently lost available content c' based on the LRL list (possibly restricting to contents less recently lost than c) and then updates the position of c in the LRL list. It then picks a random idle server s storing c', which proceeds as for RANDOM.

Finally, for LFL, the coordinator would need to maintain estimates of the loss rates of each content (by whatever means, e.g., exponentially weighted moving averages); when a loss happens, the coordinator picks the available content c' with the smallest loss rate estimate and then proceeds as the other two rules. This last rule is more complicated as it involves a timescale adjustement for the estimation of the loss rates, therefore we will focus on the first two options. Note that the LFL policy does not suffer from the drawback that the eviction rate is biased towards popular contents due to their small unavailability, and this effect is also attenuated under the LRL policy. We point out that it is possible to operate the system in a fully distributed way (finding a random idle server first and then picking a content on it by whatever rule), but this approach is biased into selecting for eviction contents with a large number of available replicas (i.e., popular contents), which will lead to a replication with reduced efficency. It is of interest for future work to find a way to unbias such a distributed mechanism.

4.6.2 Adapting faster than losses

In the algorithms proposed in the previous section, we use the losses of the system as the only trigger for creating new replicas. It is convenient as these events are directly tied to the quantity we wish to control (the loss rates), however it also has drawbacks. Firstly, it implies that we want to generate a new replica for a content precisely when we have no available server for uploading it, and thus either we send two requests at a time to the data center instead of one (one for the user which we could not serve and one to generate a new copy) or we must delay the creation of the new replica and tolerate an inedaquate replication in-between, which also hinders the reactivity of the system. Secondly, unless in practice we intend to slow down the adaptation enough, there will always be oscillations in the replication. If losses happen almost as a Poisson process, then only a bit of slow-down is necessary to moderate the oscillations, but if they happen in batches (as it may very well be for the most popular contents) then we will create many replicas in a row for the same contents. If in addition we use the LRL or LFL rules for evicting replicas, then the same contents will suffer many evictions successively, fuelling again the oscillations in the replication. Finally, it is very likely that popularities are constantly changing. If the system is slow to react, then the replication may never be in phase with the current popularities but always lagging behind. For all these reasons, it is important to find a way to decouple the adaptation mechanisms from the losses in the system to some extent, and at least to find other ways to trigger adaptation.

We propose a solution, relying on the analysis in Section 4.5.2. A loss for content c occurs when, upon arrival of a request for c, its number of available replicas is equal to 0. In the same way, whenever a request arrives, we can use the current value of Z_c to estimate the loss rate of c. Indeed, Equation (4.20) tells us how to relate the probability of $Z_c = z$ to the loss rate γ_c, for any $z \in \mathbb{N}$. Of course, successive samples of Z_c are very correlated (note that losses may also be correlated though) and we must be careful not to be too confident in the estimate they provide. A simple way to use those estimates to improve the adaptation scheme is to generate *virtual* loss events, to which any standard adaptive scheme such as those introduced in the

previous section may then react. To that end, whenever a request for c arrives, we generate a virtual loss with a certain probability $p_c(Z_c)$ depending on the current number of available replicas Z_c. The objective is to define $p_c(Z_c)$ so that the rates $(\widetilde{\gamma}_c)_{c\in C}$ of generation of virtual losses satisfy $\widetilde{\gamma}_c = \widetilde{\text{Cste}} \times \gamma_c$ for all $c \in C$ (so that the target replication still equalizes loss rates) and $\widetilde{\text{Cste}}$ is as high as possible (to get fast adaptation).

As a first step towards setting the probability $p_c(Z_c)$, we write Equation (4.20) as follows:

$$\pi(Z_c = 0) = \pi(Z_c = z) \prod_{i=1}^{z} \frac{\lambda_c + i\theta_{\text{eff}}}{D_c - i + 1}.$$

This shows that, for any fixed value z with $\pi(Z_c = z) \geq \pi(Z_c = 0)$, we can generate events at rate γ_c by subsampling at the time of a request arrival with $Z_c = z$ with a first probability

$$q_c(z) = \prod_{i=1}^{z} \frac{\lambda_c + i\theta_{\text{eff}}}{D_c - i + 1}.$$

If on the contrary z is such that $\pi(Z_c = z) < \pi(Z_c = 0)$, then the value of $q_c(z)$ given above is larger than 1 as we cannot generate events at rate γ_c by subsampling even more unlikely events. If we generated virtual losses at rate γ_c as above for each value of z, then the total rate of virtual losses for content c would be $\lambda_c \int_{z=1}^{D_c} \min\{\pi(Z_c = z), \pi(Z_c = 0)\}$, which clearly still depends on c. We thus proceed in two additional steps towards setting $p_c(Z_c)$: we first restrict the range of admissible values of Z_c, for which we may generate virtual losses, by excluding the values z such that $\pi(Z_c = z) < \pi(Z_c = 0)$. In the regime $\theta_{\text{eff}} \geq 1$, this can be done in a coarse way by letting $z_c^* = \frac{D_c - \lambda_c}{\theta_{\text{eff}}}$ and rejecting all the values $z > z_c^*$. Indeed,

$$q_c(z_c^*) = \prod_{i=0}^{z_c^*} \frac{D_c - i\theta_{\text{eff}}}{D_c - i} \leq 1,$$

and the distribution of Z_c is unimodal with the mode at a smaller value $\frac{D_c - \lambda_c}{\theta_{\text{eff}}} \approx \rho z_c^*$. Now, subsampling with probability $q_c(Z_c)$ when a request arrives for content c and $Z_c \leq z_c^*$ would generate events at a total rate $z_c^* \gamma_c$. Thus, it suffices to subsample again with probability $\frac{\min_{c' \in C} z_{c'}^*}{z_c^*}$ to obtain the same rate of virtual losses for all the contents (another approach would be to restrict again the range of admissible values of z, e.g., to values around the mode $\widehat{z}_c$).

To sum up, our approach is to generate a virtual loss for content c at each arrival of a request for c with probability

$$p_c(Z_c) = \frac{\min_{c' \in C} z_{c'}^*}{z_c^*} \mathbf{1}(Z_c \leq z_c^*) q_c(Z_c).$$

The rate $\widetilde{\gamma}_c$ at which virtual losses are generated for content c is then given by $\gamma_c \times \min_{c' \in C} z_{c'}^*$, which is independent of c as planned. Whenever a virtual loss occurs, we can use whatever algorithm we wanted to use in the first place with real losses; there is no need to distinguish between the real losses and the virtual ones. For example, if we use the LRL policy, we update the position of c in the LRL

list and create a new replica for c by evicting a least recently lost available content (from a server which does not already store c). If we choose to test for virtual losses at ends of service for c (which yields the same outcome in distribution, as the system is reversible), the new replica can simply be uploaded by the server which just completed a service for c. Furthermore, in practice, we advocate estimating the values of z_c^* and $p_c(Z_c)$ on the fly rather than learning these values for each content: θ_{eff} can be computed from ρ_{eff}, which is naturally approximated by the ratio of the current number of busy servers to the total number of servers; similarly, we can approximate λ_c by the current number of requests for c being served. From these, we can compute z_c^* and $p_c(Z_c)$; it is only necessary to maintain an estimate for $\min_{c' \in C} z_{c'}^*$, which can be for example an average over the few least popular contents updated whenever a service ends for one of them.

Next, we evaluate the performance of the adaptative schemes proposed and the virtual losses mechanism.

4.6.3 Evaluation of the Performance through Simulations

Before getting to the simulation results, let us just mention that the complexity of simulating the adaptive algorithms grows very fast with the system size (with n, m and d). Indeed, it is easy to see that simulating the system for a fixed duration t requires $\Omega(md\overline{\lambda}t)$ operations. Furthermore, the time needed for the RANDOM algorithm to converge, when started at proportional replication, is roughly of the order of $\max_{c \in C} D_c/\overline{\gamma}$, where $\overline{\gamma}$ is the average loss rate for the limit replication, which decreases exponentially fast in d as seen in Equation (4.25). Therefore, if we want to compare all the adaptive schemes, we cannot simulate very large networks. Anyway, our intention is to show that our schemes work even for networks of reasonable size.

As in Section 4.5.4, we used Zipf popularity distributions with exponents 0.8 and 1.2 and a class model to evaluate the performance of the various schemes. The results are qualitatively identical under all these models, so we only show the results for Zipf popularity with exponent $\alpha = 0.8$. We compare the various schemes in terms of the replication achieved and its associated loss rates, as well as the speed at which the target replication is reached. We do not slow down the dynamics of the adaptive schemes even though this necessarily induces some oscillations in the replication obtained. Nonetheless, this setup is already sufficient to demonstrate the performance of the adaptive schemes. It would be interesting to quantify the potential improvement if one reduces the oscillations of the replication obtained (e.g., by quantifying the variance of the stationary distribution for the number of replicas for each content); we leave this out for future work. Also, we did not account for the load put on the data center to create new copies of the contents; one can simply double the loss rates for the adaptive schemes to capture this effect. Note that if adaptation speed is not a priority, one can trade it off to almost cancel this factor of 2. Finally, concerning the virtual loss mechanism, we estimate all the quantities involved on the fly, as recommended in the previous section.

In Figure 4.5, we show results for the various adaptive schemes. On the left part of the figure, we show the stationary loss rates of all the contents; on the right part we show in log-log scale the stationary expectation of the numbers of replicas for each

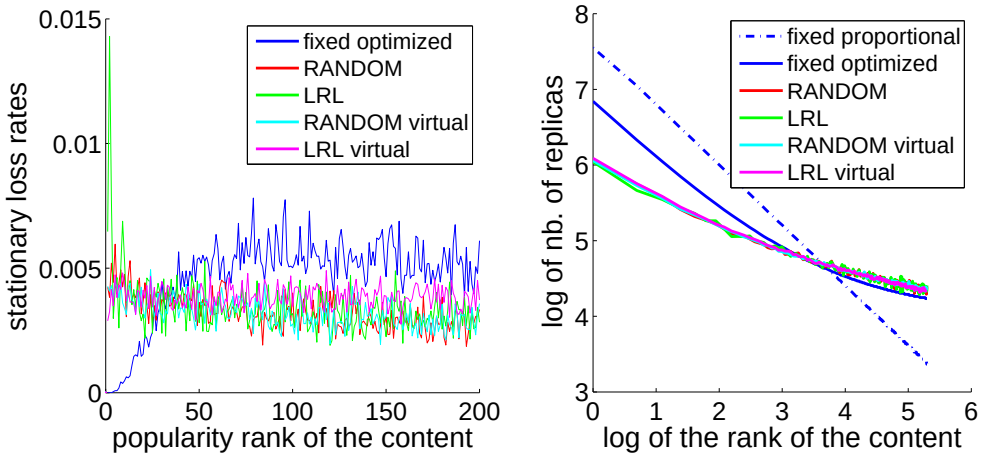

Figure 4.5: Adaptive schemes: loss rates and replication.

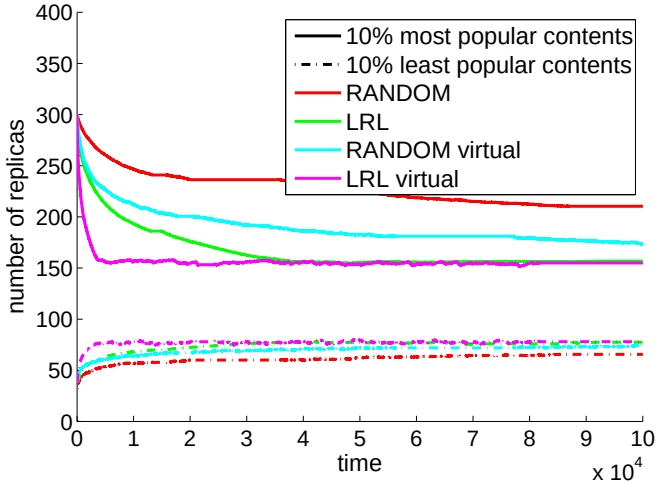

Figure 4.6: Adaptive schemes: time evolution.

Replication strategy	$10^3\times$ inefficiency $\overline{\gamma}/\overline{\lambda}$
fixed proportional	5.69
fixed optimized	0.51
fixed uniform	67.3
adaptive RANDOM	0.36
adaptive LRL	0.38
RANDOM + virtual losses	0.35
LRL + virtual losses	0.42

Table 4.2: Performance of the fixed and adaptive schemes.

content. This plot shows firstly that all the adaptive schemes converge to the same replication and secondly that this replication equalizes the loss rates, as intended. In addition, the adaptive schemes perform even better than the optimized static replication, which suffers from finite network / finite storage effects, as they manage to find the right balance between popular and unpopular contents. In Table 4.2, we provide the inefficiency of the various schemes in the stationary regime for the Zipf exponent $\alpha = 0.8$. One can see that the adaptive schemes consistently outperform the fixed optimized replication, which is already an order of magnitude better than the fixed proportional replication. One can also note that a quick reaction speed generally goes with a small penalty in average loss rate, as the replication attained fluctuates slightly around the optimal replication, and the preventive replication mechansim also artificially increases the load by a small amount (in practice, of course, such a mechanism should be refrained when the load approaches 1).

We compare the adaptation speed of the various schemes on Figure 4.6, where we plot both the evolution of the average number of replicas of the 10% most popular contents and that of the 10% least popular ones, starting from proportional replication. As expected, the LRL schemes are faster than the RANDOM ones, but more importantly this plot clearly demonstrates the speed enhancement offered by the virtual loss method of Section 4.6.2. Regarding the benefits of such an enhanced reaction capability, there is an interesting property which we did not point out nor illustrate with the simulations: the virtual loss scheme has the potential to follow a constantly evolving popularity profile at no cost, as the required creations of replicas to adapt to the changing popularities can be done without requesting copies of the contents to the data center.

Bibliography

[1] S. Albers, L. M. Favrholdt, and O. Giel. On paging with locality of reference. *Journal of Computer and System Sciences*, 70(2):145–175, 2005.

[2] D. Aldous. The ζ (2) limit in the random assignment problem. *Random Structures & Algorithms*, 18(4):381–418, 2001.

[3] D. Aldous and A. Bandyopadhyay. A survey of max-type recursive distributional equations. *Annals of Applied Probability 15*, 15:1047–1110, 2005.

[4] D. Aldous and R. Lyons. Processes on unimodular random networks. *Electron. J. Probab.*, 12:no. 54, 1454–1508, 2007.

[5] D. Aldous and J. M. Steele. The objective method: probabilistic combinatorial optimization and local weak convergence. In *Probability on discrete structures*, volume 110 of *Encyclopaedia Math. Sci.*, pages 1–72. Springer, Berlin, 2004.

[6] N. Alon and J. H. Spencer. *The probabilistic method*. Wiley. com, 2004.

[7] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying. Content-aware caching and traffic management in content distribution networks. In *INFOCOM*, pages 2858–2866, 2011.

[8] R. P. Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters*, 24(3):153–157, 1987.

[9] K. B. Athreya and P. E. Ney. *Branching processes*, volume 28. Springer-Verlag Berlin, 1972.

[10] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[11] A. Bandyopadhyay and D. Gamarnik. Counting without sampling: Asymptotics of the log-partition function for certain statistical physics models. *Random Structures & Algorithms*, 33(4):452–479, 2008.

[12] M. Bayati, C. Borgs, J. Chayes, and R. Zecchina. On the exactness of the cavity method for weighted b-matchings on arbitrary graphs and its relation to linear programs. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(06):L06001, 2008.

[13] M. Bayati, D. Gamarnik, D. Katz, C. Nair, and P. Tetali. Simple deterministic approximation algorithms for counting matchings. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 122–127. ACM, 2007.

[14] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307, 1978.

[15] I. Benjamini and O. Schramm. Recurrence of distributional limits of finite planar graphs. *Electron. J. Probab.*, 6:no. 23, 13 pp. (electronic), 2001.

[16] P. Billingsley. *Convergence of probability measures*, volume 493. Wiley. com, 2009.

[17] B. Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311 – 316, 1980.

[18] B. Bollobás. *Random graphs*, volume 73. Cambridge university press, 2001.

[19] B. Bollobás and O. Riordan. Sparse graphs: metrics and random models. *Random Structures & Algorithms*, 39(1):1–38, 2011.

[20] C. Bordenave. Lecture notes on random graphs and probabilistic combinatorial optimization!! draft in construction!! 2012.

[21] C. Bordenave, M. Lelarge, and J. Salez. Matchings on infinite graphs. *Probability Theory and Related Fields*, pages 1–26, 2012.

[22] C. Borgs, J. Chayes, and D. Gamarnik. Convergent sequences of sparse graphs: A large deviations approach. *arXiv preprint arXiv:1302.4615*, 2013.

[23] C. Borgs, J. Chayes, J. Kahn, and L. Lovász. Left and right convergence of graphs with bounded degree. *Random Structures & Algorithms*, 42(1):1–28, 2013.

[24] Y. Boufkhad, F. Mathieu, F. De Montgolfier, D. Perino, L. Viennot, et al. Achievable catalog size in peer-to-peer video-on-demand systems. In *Proceedings of the 7th Internnational Workshop on Peer-to-Peer Systems (IPTPS)*, pages 1–6, 2008.

[25] A. Braunstein, M. Mézard, and R. Zecchina. Survey propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[26] T. Britton, M. Deijfen, and A. Martin-Löf. Generating simple random graphs with prescribed degree distribution. *Journal of Statistical Physics*, 124(6):1377–1397, 2006.

[27] J. A. Cain, P. Sanders, and N. Wormald. The random graph threshold for k-orientiability and a fast algorithm for optimal multiple-choice allocation. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 469–476, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[28] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305–1314, 2002.

[29] N. Chen and M. Olvera-Cravioto. Directed random graphs with given degree distributions. *arXiv preprint arXiv:1207.2475*, 2012.

[30] M. Chertkov. Exactness of belief propagation for some graphical models with loops. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10016, 2008.

[31] Cisco White Paper. Cisco visual networking index: Forecast and methodology, 2012-2017. May 2013.

[32] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 177–190, 2002.

[33] R. Darling, J. R. Norris, et al. Differential equation approximations for markov chains. *Probability surveys*, 5:37–79, 2008.

[34] A. Dembo and A. Montanari. Gibbs measures and phase transitions on sparse random graphs. *Brazilian Journal of Probability and Statistics*, 24(2):137–211, 2010.

[35] M. Dietzfelbinger, A. Goerdt, M. Mitzenmacher, A. Montanari, R. Pagh, and M. Rink. Tight thresholds for cuckoo hashing via xorsat. In *Proceedings of the 37th international colloquium conference on Automata, languages and programming*, ICALP'10, pages 213–225, Berlin, Heidelberg, 2010. Springer-Verlag.

[36] M. Dietzfelbinger and C. Weidling. Balanced allocation and dictionaries with tightly packed constant size bins. *Theoretical Computer Science*, 380(1):47 – 68, 2007.

[37] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[38] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *Information Theory, IEEE Transactions on*, 56(9):4539–4551, 2010.

[39] R. Durrett. *Random graph dynamics*, volume 20. Cambridge university press, 2007.

[40] P. Erdos and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[41] D. Fernholz and V. Ramachandran. The k-orientability thresholds for $G_{n,p}$. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 459–468, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[42] N. Fountoulakis, M. Khosla, and K. Panagiotou. The multiple-orientability thresholds for random hypergraphs. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1222–1236. SIAM, 2011.

[43] N. Fountoulakis, K. Panagiotou, and A. Steger. On the insertion time of cuckoo hashing. *CoRR*, abs/1006.1231, 2010.

[44] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. In *Proceedings of the 24th International Teletraffic Congress*. International Teletraffic Congress, 2012.

[45] C. Fricker, P. Robert, J. Roberts, and N. Sbihi. Impact of traffic mix on caching performance in a content-centric network. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 310–315, 2012.

[46] A. Frieze and P. Melsted. Maximum matchings in random bipartite graphs and the space utilization of cuckoo hash tables. *Random Structures & Algorithms*, 41(3):334–364, 2012.

[47] A. Frieze, P. Melsted, and M. Mitzenmacher. An analysis of random-walk cuckoo hashing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 490–503. Springer, 2009.

[48] R. Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[49] D. Gamarnik, D. Goldberg, and T. Weber. Correlation decay in random decision networks. *arXiv preprint arXiv:0912.0338*, 2009.

[50] D. Gamarnik and D. Katz. Correlation decay and deterministic fptas for counting list-colorings of a graph. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1245–1254. Society for Industrial and Applied Mathematics, 2007.

[51] D. Gamarnik and S. Misra. Giant component in random multipartite graphs with given degree sequences. *arXiv preprint arXiv:1306.0597*, 2013.

[52] D. Gamarnik, T. Nowicki, and G. Swirszcz. Maximum weight independent sets and matchings in sparse random graphs. exact results using the local weak convergence method. *Random Structures & Algorithms*, 28(1):76–106, 2006.

[53] P. Gao and N. C. Wormald. Load balancing and orientability thresholds for random hypergraphs. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 97–104, New York, NY, USA, 2010. ACM.

[54] E. N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[55] G. H. Gonnet. Expected length of the longest probe sequence in hash code searching. *Journal of the ACM (JACM)*, 28(2):289–304, 1981.

[56] L. J. Guibas and E. Szemeredi. The analysis of double hashing. *Journal of Computer and System Sciences*, 16(2):226–274, 1978.

[57] I. Heller and C. Tompkins. An extension of a theorem of Dantzig's. In H. Kuhn and A. Tucker, editors, *Linear inequalities and related systems*, pages 247–254. Princeton University Press, 1956.

[58] T. Heskes. On the uniqueness of loopy belief propagation fixed points. *Neural Computation*, 16(11):2379–2413, 2004.

[59] A. T. Ihler, J. Fisher III, and A. S. Willsky. Loopy belief propagation: Convergence and effects of message errors. In *Journal of Machine Learning Research*, pages 905–936, 2005.

[60] S. James and P. Crowley. Isp managed peer-to-peer. In *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 167–168. ACM, 2009.

[61] S. Janson, T. Luczak, and V. Kolchin. *Random graphs*. Cambridge Univ Press, 2000.

[62] W. Jiang, S. Ioannidis, L. Massoulié, and F. Picconi. Orchestrating massively distributed cdns. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 133–144. ACM, 2012.

[63] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[64] F. P. Kelly. Loss networks. *The annals of applied probability*, pages 319–378, 1991.

[65] M. Khosla. Balls into bins made faster. In *ESA*, Lecture Notes in Computer Science. Springer, 2013.

[66] J. H. Kim. Poisson Cloning Model for Random Graphs. *ArXiv e-prints*, May 2008.

[67] F. Krząkała, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, and L. Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proc. Natl. Acad. Sci. USA*, 104(25):10318–10323 (electronic), 2007.

[68] T. Kurtz. *Approximation of Population Processes*. CBMS-NSF Regional Conference Series in Applied Mathematics, 1981.

[69] N. Laoutaris, V. Zissimopoulos, and I. Stavrakakis. On the optimization of storage capacity allocation for content distribution. *Computer Networks*, 47(3):409–428, 2005.

[70] M. Lelarge. A new approach to the orientation of random hypergraphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 251–264. SIAM, 2012.

[71] L. Lovász and M. D. Plummer. *Matching theory*. AMS Chelsea Publishing, Providence, RI, 2009. Corrected reprint of the 1986 original [MR0859549].

[72] G. Lueker and M. Molodowitch. More analysis of double hashing. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 354–359. ACM, 1988.

[73] R. Lyons. Asymptotic enumeration of spanning trees. *Combinatorics, Probability & Computing*, 14(4):491–522, 2005.

[74] E. Maneva, E. Mossel, and M. J. Wainwright. A new look at survey propagation and its generalizations. *J. ACM*, 54(4), July 2007.

[75] B. D. McKay and N. C. Wormald. Uniform generation of random regular graphs of moderate degree. *Journal of Algorithms*, 11(1):52–67, 1990.

[76] M. Mezard and A. Montanari. *Information, Physics, and Computation*. Oxford University Press, Inc., New York, NY, USA, 2009.

[77] M. Mézard and G. Parisi. The bethe lattice spin glass revisited. *The European Physical Journal B-Condensed Matter and Complex Systems*, 20(2):217–233, 2001.

[78] M. Mézard and G. Parisi. The cavity method at zero temperature. *Journal of Statistical Physics*, 111(1-2):1–34, 2003.

[79] M. Mézard, G. Parisi, and M. A. Virasoro. *Spin glass theory and beyond*, volume 9 of *World Scientific Lecture Notes in Physics*. World Scientific Publishing Co. Inc., Teaneck, NJ, 1987.

[80] M. Mézard, G. Parisi, and R. Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002.

[81] M. Mitzenmacher. Some open questions related to cuckoo hashing. In *Algorithms-ESA 2009*, pages 1–10. Springer, 2009.

[82] M. Mitzenmacher. Balanced allocations and double hashing. *CoRR*, abs/1209.5360, 2012.

[83] M. Mitzenmacher, A. W. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. In *in Handbook of Randomized Computing*, pages 255–312. Kluwer, 2000.

[84] M. Mitzenmacher and J. Thaler. Peeling arguments and double hashing. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1118–1125. IEEE, 2012.

[85] J. M. Mooij and H. J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Transactions on Information Theory*, 53(12):4422–4437, Dec. 2007.

[86] R. Mulet, A. Pagnani, M. Weigt, and R. Zecchina. Coloring random graphs. *Physical review letters*, 89(26):268701, 2002.

[87] A. Müller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks*. Wiley, 2009.

[88] T. Nelson. Literary machines, 93.1 edn, 1993.

[89] R. Pagh and F. F. Rodler. Cuckoo hashing. In *ESA*, pages 121–133, 2001.

[90] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *AAAI*, pages 133–136, 1982.

[91] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. The Morgan Kaufmann Series in Representation and Reasoning. Morgan Kaufmann, San Mateo, CA, 1988.

[92] R. Pemantle. Towards a theory of negative dependence. *Journal of Mathematical Physics*, 41:1371, 2000.

[93] T. J. Richardson and R. L. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *Information Theory, IEEE Transactions on*, 47(2):599–618, 2001.

[94] Y. Rochman, H. Levy, and E. Brosh. Max percentile replication for optimal performance in multi-regional p2p vod systems. In *Ninth International Conference on Quantitative Evaluation of Systems (QEST)*, pages 238–248. IEEE, 2012.

[95] Y. Rochman, H. Levy, and E. Brosh. Resource placement and assignment in distributed network topologies. In *Proceedings IEEE INFOCOM*, 2013.

[96] J. Salez. Weighted enumeration of spanning subgraphs in locally tree-like graphs. *Random Structures & Algorithms*, 2012.

[97] A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, jul 2004.

[98] J. G. Shanthikumar and D. D. Yao. The preservation of likelihood ratio ordering under convolution. *Stochastic Processes and their Applications*, 23(2):259–267, 1986.

[99] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello. Push-to-peer video-on-demand system: Design and evaluation. *Selected Areas in Communications, IEEE Journal on*, 25(9):1706–1716, 2007.

[100] A.-S. Sznitman. Topics in propagation of chaos. In *Ecole d'Eté de Probabilités de Saint-Flour XIX—1989*, pages 165–251. Springer, 1991.

[101] B. Tan and L. Massoulié. Optimal content placement for peer-to-peer video-on-demand systems. In *Proceedings IEEE INFOCOM*, pages 694–702, 2011.

[102] E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[103] S. C. Tatikonda and M. I. Jordan. Loopy belief propagation and gibbs measures. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 493–500. Morgan Kaufmann Publishers Inc., 2002.

[104] S. Tewari and L. Kleinrock. On fairness, optimal download performance and proportional replication in peer-to-peer networks. In *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 709–717. Springer, 2005.

[105] S. Tewari and L. Kleinrock. Proportional replication in peer-to-peer networks. In *Proceedings IEEE INFOCOM*, 2006.

[106] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez. Greening the internet with nano data centers. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT)*, pages 37–48. ACM, 2009.

[107] R. Van Der Hofstad. Random graphs and complex networks. *Available on http://www. win. tue. nl/rhofstad/NotesRGCN. pdf*, 2009.

[108] V. V. Vazirani. *Approximation algorithms*. springer, 2001.

[109] S. Vembu and S. Verdú. Generating random bits from an arbitrary source: Fundamental limits. *Information Theory, IEEE Transactions on*, 41(5):1322–1332, 1995.

[110] H. W. Watson and F. Galton. On the probability of the extinction of families. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 4:138–144, 1875.

[111] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural computation*, 13(10):2173–2200, 2001.

[112] D. Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149. ACM, 2006.

[113] W. Whitt. Uniform conditional stochastic order. *Journal of Applied Probability*, pages 112–123, 1980.

[114] W. Whitt. Multivariate monotone likelihood ratio and uniform conditional stochastic order. *Journal of Applied Probability*, pages 695–701, 1982.

[115] N. C. Wormald. *Some problems in the enumeration of labelled graphs.* PhD thesis, Newcastle University, 1978.

[116] W. Wu and J. C. Lui. Exploring the optimal replication strategy in p2p-vod systems: Characterization and evaluation. *Parallel and Distributed Systems, IEEE Transactions on*, 23(8):1492–1503, 2012.

[117] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282 – 2312, july 2005.

[118] L. Zdeborová and F. Krząkała. Phase transitions in the coloring of random graphs. *Physical Review E*, 76(3):031131, 2007.

[119] L. Zdeborová and M. Mézard. The number of matchings in random graphs. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(05):P05003, 2006.

[120] Y. Zhou, T. Z. Fu, and D. M. Chiu. A unifying model and analysis of p2p vod replication and scheduling. In *Proceedings IEEE INFOCOM*, pages 1530–1538, 2012.

[121] Y. Zhou, T. Z. Fu, and D. M. Chiu. On replication algorithm in p2p vod. *IEEE/ACM Transactions on Networking*, pages 233 – 243, 2013.

[122] D. Zuckerman. Simulating bpp using a general weak random source. *Algorithmica*, 16(4-5):367–391, 1996.