# Greedy approaches to approximation of some NP-hard combinatorial optimization problems

Mathieu Mari

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École normale supérieure

# Greedy approaches to approximation of some NP-hard combinatorial optimization problems


Soutenue par
**Mathieu Mari**
Le 25 septembre 2020


École doctorale n°386

**École doctorale de sciences mathématiques de Paris centre**

Spécialité

**Informatique théorique**

Composition du jury :

| | |
|---|---|
| Christoph Dürr<br>CNRS, Sorbonne Université | *Président* |
| Alantha Newman<br>CNRS, Université Grenoble Alpes | *Examinatrice* |
| Magnús M. Halldórsson<br>Reykjavík University | *Examinateur* |
| Cyril Gavoille<br>Université de Bordeaux | *Rapporteur* |
| Dimitrios M. Thilikos<br>CNRS, LIRMM, Université Montpellier | *Examinateur* |
| Claire Mathieu<br>CNRS, Université de Paris | *Directrice de thèse* |
| Chien-Chung Huang<br>CNRS, ENS, PSL | *Directeur de thèse* |
| Bruce Shepherd<br>University of British Columbia | *Rapporteur* |

ENS ÉCOLE NORMALE SUPÉRIEURE

PSL RESEARCH UNIVERSITY PARIS

à ma grand-mère Rolande,

# Remerciements

Ces trois dernières années ont été pour moi une très belle expérience dans le monde de la recherche.

Cela, je le dois en premier lieu à ma directrice de thèse, Claire Mathieu et mon directeur de thèse, Chien-Chung Huang. Vous avez été un parfait duo pour moi, et je m'estime très chanceux de vous avoir eu comme encadrants.

Merci Claire d'avoir toujours pu trouver du temps et de l'énergie pour moi malgré ton emploi du temps chargé. J'ai beaucoup appris en travaillant à tes côtés. Merci pour ta patience, tes conseils et critiques bienveillantes qui m'ont aidés à progresser dans la bonne direction. Merci aussi de m'avoir donné l'opportunité de pouvoir voyager et de rencontrer de nombreux chercheurs.

Merci Chien-Chung d'avoir été aussi présent. On a régulièrement pu travailler ensemble, c'était agréable et motivant. Je trouve qu'on forme une bonne équipe ! Ta passion pour la résolution de problèmes de recherche a déteint sur moi. J'espère qu'on continuera à travailler ensemble à l'avenir !

I wish to express my gratitude to Cyril Gavoille and Bruce Shepherd who took the time to read and review carefully this thesis. I am also grateful to the other members of my thesis committee: Magnús Halldórsson, Alantha Newman, Dimitrios Thilikos and Christoff Dürr. Thank you for your interest in my work!

J'ai eu de la chance de pouvoir faire ma thèse au sein une équipe aussi sympathique que Talgo; Les nombreuses discussions avec Pierre, Chien-Chung, Tatiana, Kevin et Thang ont beaucoup élargi ma culture informatique et générale. Merci aussi pour les sorties escalades, les coinches et les parties de Carcassone et d'Agricola qui ont su épicer agréablement certaines semaines. Merci à certains membres du DI et de l'équipe administrative qui m'ont souvent apporté une aide précieuse. Merci notamment à David Pointcheval, Pierre Senellart, Sophie Jaudon et Lise-Marie Bivard.

Last year I spent few months at Universidad de Chile to work with new people on different types of algorithms. I am very grateful to José Correa and Andreas Wiese who gave me this opportunity and welcomed me in Chile. Thanks Andy for sharing your knowledge on FPT-algorithms and José, on algorithms related to economics. This was a great time in Chile, also thanks to all nice people who were there: Andrés, Andrew, Boris, Bastian, Felipe, Camilla, and the others.

I would like to thank Piotr Krysta who gave me the opportunity to discover research during my Master internship in Liverpool. I am glad that we continued working together (and with Nan Zhi) during my PhD on questions about the greedy algorithm for Maximum Independent Set. Thanks Piotr also for all the advises that you gave me during the last three years!

Je voudrais remercier Nabil Mustafa pour son soutien, pour avoir partagé ses con-

naissances en géométrie, et pour les discussions inspirantes concernant la recherche en général.

I want to thank Philip Klein for inviting me few days in Providence, and sharing his knowledge on Primal-Dual schemas and planar graphs.

Thank you Jens Vygen for inviting me at the seminar talk in Bonn, that was an honor for me. Also, working with you was very inspiring, and helped me in the way of writing clean and rigorous papers.

Merci David et Simon, votre amitié, nos discussions, et nos sessions de travail collectif ont contribué à la réussite de cette thèse. Et aussi, les conférences HALG et SODA n'auraient probablement pas été aussi fun sans vous !

Merci Quentin pour toutes nos discussions sur la topologie algébrique, ça m'a donné beaucoup d'idées !

J'aimerais remercier tout mes amis qui, par nos discussions en général ont pu être une source d'inspiration. La liste ne sera pas complète, mais je pense en premier lieu à Yohan, Juliette, Renaud, Vadim, Ilyès, Paul, Manu, Pierre, Franzi, Simon, Mathieu, Clémence, Armand, ...

Je voudrais finir en remerciant mes parents, pour leur soutien sans failles durant toutes ces années. Sans vous, je ne serai pas là aujourd'hui. Merci Papa de m'avoir transmis ton goût pour les maths.

À tout ceux qui méritent d'apparaître dans cette section et que je n'ai pas mentionnés, merci !

# Abstract

The greedy approach is natural for the design of algorithms. It is fast, easy to implement, has a good performance on average, and is applicable to many optimization problems in various settings. For those reasons, it is an important heuristic in practice. In this thesis, we present greedy algorithms for three different NP-hard optimization problems. We discuss the relation between those algorithms, their proof techniques and the structure of the problems under study.

In the first part of this thesis, we focus on the Maximum Coverage problem with a connectivity constraint, which comes up for the design of wireless networks. We show that the problem is NP-hard, even when the coverage and the connectivity are induced by a set of unit disks in the plane. For that special case, we show that greedily picking two disks so as to maximize the marginal area covered while correctness achieves a 2-approximation. We further improve the approximation ratio by providing a PTAS for this problem with a small resource augmentation, using Arora's shifted grid dissection technique.

In the second part of this thesis, we focus on the Maximum Independent Set problem. A natural approach is to repeatedly pick a vertex of minimum degree, and remove it and its neighbours from the graph. We present a new technique to analyze the performance of this greedy approach in various classes of graphs and address the following question: if there are several minimun-degree vertices, what vertex should the greedy algorithm pick in order to maximise the size of the final solution ? With this tool, we design an "ultimate tie-breaking" rule that leads to the best possible approximation ratio for sub-cubic graphs and for this type of greedy algorithms. We complement this by lower bound results that show that designing good tie-breaking rules is a difficult task.

The third and last part of the thesis is devoted to the Maximum Integral Multiflows problem. The problem is difficult and well-studied. For instance, a constant factor approximation is unlikely to exist even when the supply graph is planar and cubic. In the special case where the supply graph together with the demand edges form a bounded genus graph, we present a constant factor approximation algorithm. The algorithm consists of a succession of greedy procedures that exploit topological properties of graphs and curves on surfaces.

# Résumé

L'approche gloutonne est naturelle pour concevoir un algorithme. Elle permet la conception d'algorithmes rapides, faciles à implémenter, qui produisent en moyenne des solutions de qualité, pour de nombreux problèmes d'optimisation. Pour ces raisons, c'est une heuristique importante en pratique. Dans cette thèse, nous présentons des algorithmes gloutons pour trois problèmes d'optimisation NP-difficiles. Nous discutons les relations entre ces algorithmes, leurs techniques de preuve et la structure des problèmes étudiés.

Dans la permière partie de cette thèse, on se concentre sur le problème de la couverture maximale avec une contrainte de connexité, contrainte que l'on rencontre lors de la conception de réseaux sans fil. Nous montrons que le problème est NP-difficile, même quand la couverture et la connexité proviennent d'un ensemble de disques unité dans le plan. Pour ce cas particulier, nous montrons que choisir de manière gloutonne deux disques qui maximisent le gain marginal tout en maintenant la solution connexe réalise une 2-approximation. Nous améliorons ce ratio d'approximation en donnant un schéma d'approximation en temps polynomial avec une légère augmentation de ressource, basé sur la technique d'Arora pour le voyageur de commerce euclidien.

Dans la deuxième partie de cette thèse, on se concentre sur le problème du stable maximum. Une approche naturelle est de successivement choisir un sommet de degré minimum, le placer dans la solution en construction, puis le retirer avec ses voisins du graphe. Nous présentons une nouvelle technique pour analyser la performance de cette approche gloutonne dans différentes classes de graphes et abordons la question suivante : s'il y a plusieurs sommets de degré minimum, lequel l'algorithme devrait-il choisir pour maximiser la taille de la solution finale ? Avec cet outil, nous concevons une règle pour briser les cas d'égalité, qui conduit à la meilleure approximation possible dans les graphes sous-cubiques et pour ce type d'algorithmes gloutons. Nous complémentons ces résultats par des résultats négatifs qui suggèrent que la conception de bonnes règles brisant les cas d'égalité est une tâche difficile.

La troisième et dernière partie de la thèse est consacrée au problème du multiflot entier maximum. Ce problème est difficile et a été très étudié. Par exemple, une approximation de facteur constant est probablement impossible même quand le graphe d'offre est planaire et cubique. Dans le cas particulier où le graphe d'offre et les arêtes de demande forment ensemble un graphe de genre borné, nous présentons un algorithme avec un facteur d'approximation constant. L'algorithme consiste en une succession de procédures gloutonnes qui exploitent les propriétés topologiques des graphes et des lacets sur des surfaces.

# Contents

# Publications of the author

- José R. Correa, Mathieu Mari and Andrew Xia, *Dynamic pricing with Bayesian updates from online reviews*, submitted.
- Chien-Chung Huang, Mathieu Mari, Claire Mathieu and Jens Vygen, *Approximating maximum integral multiflows on bounded genus graphs*, submitted, arXiv:2005.00575
- Chien-Chung Huang, Mathieu Mari, Claire Mathieu, Kevin Schewior and Jens Vygen, *An Approximation Algorithm for Fully Planar Edge-Disjoint Paths*, submitted, arXiv:2001.01715
- Andrés Cristi, Mathieu Mari and Andreas Wiese, *Fixed-Parameter Algorithms for Unsplittable Flow Cover*, 37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, 154,42:1–42:17.
- Piotr Krysta, Mathieu Mari and Nan Zhi, *Ultimate greedy approximation of independent sets in subcubic graphs*, Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms SODA 2020, 1436–1455.
- Chien-Chung Huang, Mathieu Mari, Claire Mathieu, Joseph S. B. Mitchell and Nabil H. Mustafa, *Maximizing Covered Area in the Euclidean Plane with Connectivity Constraint*, APPROX/RANDOM 2019, 145,32:1–32:21.

# 1 | General Introduction

## 1.1 Greedy algorithms

Imagine that you are planning a walking tour in Paris to visit several places: the Eiffel tour, the *hôtel des invalides*, the *musée d'Orsay*, the *Panthéon*, the courtyard in *École normale supérieure*, some art galleries in the *Marais* neighborhood, the on-going renovation at *Notre-Dame de Paris*, and then go back to your hotel close to the *jardin du Luxembourg*. You would like to walk but you don't want to walk too much: in which order should you visit these places so that the overall distance is minimized ?

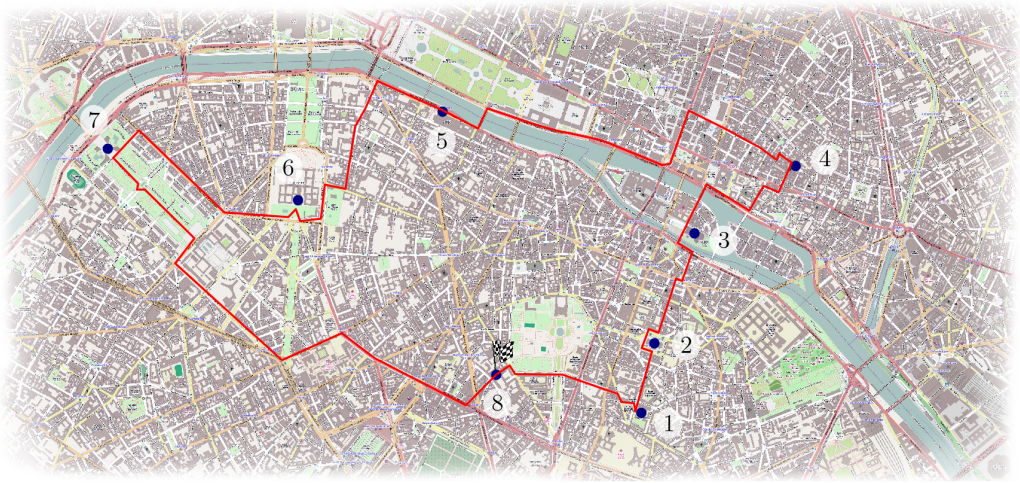



Figure 1.1: What is the shortest tour to visit all the places marked on the above map ?

You could for instance do the following: each time that you have visited a place, you look at the places on your list that you have not visited yet, and go to *the nearest* one from your current location. Luckily, as we can see on Figure 1.1, this strategy would give you here the optimal tour!

In the design of algorithms, we refer to these type of strategies as being *greedy*. Informally, being greedy, it is trying to attain a long-term objective, and at each step of time, make the decision that is the best local decision in order to attain the final goal. Informally, a greedy algorithm, a each step of time, makes the decision that is the best local decision in order to obtain some long-term objective.

Although there is no general definition of what is a greedy algorithm, we can define this notion for some specific classes of problems. In the context of *matroids*, a greedy algorithm is the procedure that builds a maximal independent set by iteratively adding the

cheapest element to the solution, that is initially empty, while maintaining the property that the solution is an independent set (see e.g. [40], chapter 16). In the context of job scheduling problems, Borodin, Nielsen and Rackoff [18] proposed the following definition: a greedy-like algorithm is a procedure that incrementally builds a schedule with each input job being considered exactly once. Each time a job is considered, the algorithm decides to add it or not to the solution. The order in which jobs are considered can be *fixed* before any job is scheduled, or *adaptive*, if the order depends on previous decisions made.

You may wonder if greedy algorithms always give optimal solutions. The answer is no! Take for instance our example and imagine that you had started in *musée des invalides* (point 6 on Figure 1.1). Then the nearest place to visit is *musée d'Orsay* (point 5) and then *Notre-Dame de Paris* (point 3) which is clearly non-optimal, because you should have visited the *Marais* first (point 4). In fact, the combinatorial problem, called *Travelling Salesman problem* (TSP), that models this example is **NP**-hard [125], which means that there is no fast method that systematically finds the shortest tour, unless the famous conjecture $\mathbf{P} \neq \mathbf{NP}$ is disproved.

Being greedy has several merits. First, it is probably the most natural approach for solving a problem. In real life, people use such a strategy sometimes unconsciously. For instance people may use a greedy strategy when they give back the change to mininize the number of coins exchanged[1], or to optimize the overall duration of their travels in a long tourist-tour. In computer science, a greedy approach is often the first idea that comes to mind when facing an optimization problem for the first time.

As a consequence of being natural, greedy algorithms are easy to understand. In our society, algorithms have a growing importance, and have more and more influence on people's lives, for instance for school admissions, job markets, social networks, etc. In a future where algorithms could be used for making social decisions, it seems natural that these algorithms remain simple enough to be understandable by non-experts. A greedy strategy thus seems a natural candidate.

This simplicity of greedy algorithms makes them easy to implement and due to their good time-efficiency, they are attractive heuristics in practice. For instance, for the vast majority of satisfiable 3-CNF formulas, the algorithm that starts at a random truth assignment and repeatedly flips the variables that improves the number of satisfied clauses the most, almost always succeeds in discovering a satisfying truth assignment [99]. Even when no guarantees are known, or when in the worst-case the solution is far from optimal, greedy algorithms can be efficient on average. For instance, even though finding a maximum-size Independent Set[2] cannot be approximated in polynomial time within any constant [80], the natural minimum degree greedy algorithm (Algorithm 2) is on average a 2-approximation[3] in a random graph [116].

Another feature of greedy algorithms is that the decisions are irrevocable. Since greedy algorithms do not re-consider their previous decisions, they are a type of algorithms that can easily implemented *on-line*, i.e., where the input is revealed over time and decisions must be taken at each step without possible go-back.

Sometimes, the greedy algorithm is guaranteed to lead to the optimal solution. In 1935, Hassler Whitney introduced the combinatorial notion of *matroid*, which captures a wide class of problems for which a certain type of greedy algorithms is optimal. This generalises

[1] in fact, for the Euro coin system, the greedy algorithm is optimal.

[2] a set of vertices that are pairwise non-adjacent.

[3] a solution is a 2-approximation if the size of the solution is within a factor of 2 of the optimum.

for instance Kruskal's algorithm for computing a minimum spanning-tree, or the greedy algorithm for scheduling unit length jobs with profits and deadlines.

Even when the greedy algorithm is not optimal, there are some **NP**-hard problems for which a simple greedy algorithm achieves the best possible approximation guarantee in polynomial time, assuming some widely-believed conjectures. The most well-known problem with this characteristic is probably Set Cover, where we are given a universe of $n$ elements and a collection of sets of elements, and the goal is to find the smallest family of sets that covers all the elements. The Greedy algorithm to solve this problem is natural: add to your solution the set that covers a maximum number of uncovered elements, until your solution covers everything.

Starting with an empty solution,
**while** *there are some uncovered elements* **do**
  add to your solution the set that covers the maximum number of uncovered elements.

**Algorithm 1:** A Greedy algorithm for Set Cover.

This algorithm finds a solution of size at most $\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1$ times the size of an optimal solution[4] and Figure 1.2 shows an example where the greedy algorithm returns a solution of size $\log_2 n$ while the optimal solution consists of only two sets. Most importantly, a series of inapproximability results initiated by Lund and Yannakakis in 1994 [112], suggest that no better approximation can be obtained for Set Cover by a polynomial time algorithm, unless $\mathbf{P} = \mathbf{NP}$. In particular, Feige [49] improved the lower bound on the approximability to $(1 - o(1)) \ln n$ under the same assumptions, which essentially matches the approximation ratio achieved by the greedy algorithm. Two other important examples of problems of this flavor are Vertex Cover and Metric $k$-Center where simple greedy algorithms[5] achieve a 2-approximation [144], while $2 - o(1)$ is the best ratio achievable in polynomial time for Vertex Cover (*resp.* Metric $k$-center) unless the Unique Game Conjecture is disproved [93] (*resp.* unless one can solve the Dominating Set problem in polynomial time, i.e., unless $\mathbf{P} = \mathbf{NP}$ [81])

Consider NP-hard problems that have a simple polynomial-time approximation algorithm achieving a best possible approximation ratio. How can the analysis be so tight? Often, that is obtained by finding connections with the problem structure. Because of their simplicity, greedy algorithms are a great place to look for such connections. Sometimes, their approximation ratio can be established by powerful techniques that can be applied more broadly. For instance: looking at the greedy algorithm for Set Cover through the lens of its linear programming relaxation illustrates the technique of *Dual-Fitting*, that suggests a correspondence between the integrality gap and the approximability of Set Cover [111, 37]. For Vertex Cover and more generally Set Cover where the frequency[6] of each element is bounded by a constant $f$, the technique of the *Primal-Dual Schema* provides an algorithm that greedily updates primal and dual variables of the classic LP formulation of the problem, to generate a $f$-approximate solution [144].

[4] A tighter analysis shows that the approximation ratio is exactly $\ln n - \ln \ln n + \Theta(1)$ [140].

[5] for Vertex Cover, the greedy algorithm is: pick two adjacent vertices and remove all incident edges, until de graph is empty.

[6] the number of sets in which it appears. For Vertex Cover, each edge (elements) is incident to $f = 2$ vertices (sets).

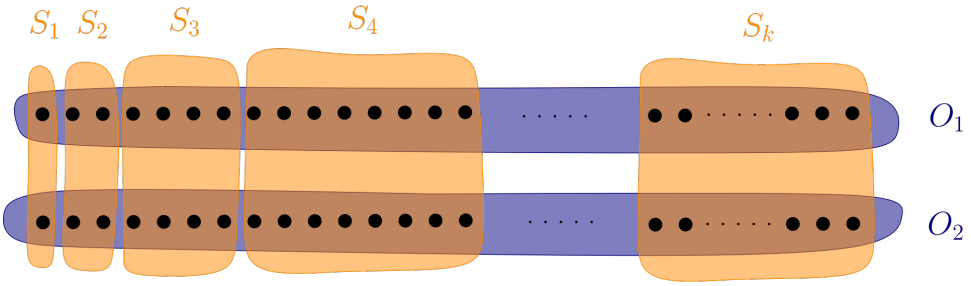



Figure 1.2: A hard example for the Greedy algorithm of Set Cover. Black dots are the elements to be covered. The family of sets consists of $O_1, O_2$ and $S_i$ for $1 \leq i \leq k$. Each set $S_i$ contains $2^i$ elements, so $O_1$ and $O_2$ both contain $2^k - 1$. The greedy algorithm picks iteratively the sets $S_k, S_{k-1}, \ldots, S_2, S_1$ for an overall solution of size $k$ while the optimal solution consists of only two sets $O_1$ and $O_2$. The ratio between these two solutions is $k/2 \approx (\log_2 n)/2$.

For some problems, the existence of a hereditary property of the class of instances considered can suggest a greedy strategy for finding a solution. For instance, the existence of a vertex of degree $d$ in a class of graphs closed by vertex deletion[7], like planar graphs, implies a natural greedy algorithm (Algorithm 3) that properly colors any graph in $\mathcal{G}$ with at most $d+1$ colors. The *method of conditional probabilities* [130] is used to convert probabilistic proofs of the existence of certain structural properties of problems into deterministic algorithms. Applied to the Max-Cut problem, the fact that a (uniformly) random subset of vertices cuts on average half of the edges allows to greedily construct a (deterministic) 2-approximate solution. Applied to the Maximum Independent Set problem, this probabilistic method shows that the greedy algorithm (Algorithm 2) achieves the celebrated Turán bound[8] [142, 47].

Sometimes the same greedy algorithm can be applied to two different problems[9]. The greedy algorithm described for Set Cover (Algorithm 1) can actually be applied to a similar problem called *Maximum Coverage*. Instead of covering every element, with a minimum number of sets, the goal here is to cover a maximum number of elements, given a budget of $k$ sets.

**The Maximum Coverage Problem:** Given a set of elements, a family of sets of elements, and a budget $k \geq 0$, the goal is to find a set of $k$ sets that cover the maximum number of elements.

With the same greedy algorithm[10] that iteratively picks the most profitable set, we obtain a solution for the Maximum Coverage problem that covers at least $1 - (1 - \frac{1}{k})^k > 1 - 1/e \approx 63\%$ of the maximum number of elements that can be covered by $k$ sets [124]. Again, no better approximation can be obtained in polynomial time for this problem, unless $\mathbf{P} = \mathbf{NP}$ [49].

What makes this greedy strategy successful here lies in a structural property of the objective function called *submodularity*. Consider a process that builds a solution by

[7] a class $\mathcal{G}$ such that, for all graph $G \in \mathcal{G}$ and any subset of vertices $U \subseteq V(G)$, the subgraph $G[U]$ induced by $U$ is in $\mathcal{G}$.

[8] any graph with $n$ vertices and average degree $d$ contains an independent set of size at least $n/(d+1)$.

[9] except for the termination criteria.

[10] this greedy algorithm is sometimes called Nemhauser's algorithm. It can be used for maximizing a monotone submodular function.

iteratively adding sets, and consider inserting a set $S$ into that process. The later we insert $S$, the less benefit it brings you[11]. Nemhauser showed the same greedy algorithm applied to maximizing the value of a $k$-set given by any (non-negative and monotone) submodular function achieves the same approximation of $e/(e-1)$ [124].

Greedy methods exploit other type of structural properties. For matroids, the greedy algorithm exploits the *exchange property*[12]. Greedy coloring (Algorithm 3) and the greedy algorithm for Independent Set problem (Algorithm 2) use the *hereditary*[13] existence of a small-degree vertex. The existence of *half-integral* solutions of the fractional relaxation of the classic linear program for Vertex Cover enables us to iteratively construct a 2-approximate solution [144]. Sometimes, when the approximation obtained is the best possible, we are even able to predict the structure of the graphs on which the approximation is the worst, as we will see in chapter 3 (see Section 3.5.3).

## 1.2 Contributions of this thesis

Maximizing submodular functions under additional constraints has drawn a lot of attention, and many variants and generalisations were investigated. For the more general problem of maximizing a monotone submodular function under a matroid constraint, an approximation of $e/(e-1)$ can also be achieved [21]. When the function is no longer monotone, there is a 2-approximation [20].

A natural constraint that has received relatively little attention is the *connectivity* constraint. There is a graph, associated to the input, with sets as vertices, and a solution must induce a connected subgraph to be feasible under the connectivity constraint. The Maximum Coverage problem is motivated by network design (sensors, wireless routers, relay-antennas, etc.) where the connectivity constraint models the ability of communication between the different devices within the network [103]. For these applications, the problem takes place in a geometric setting. Indeed, a sensor, a wireless router or an antenna "covers" all devices present at a certain distance from it, and we can imagine that two devices in the network can communicate if they are within a certain distance from each other, hence the graph structure.

Another motivation for studying connectivity constraints comes from cancer genome studies. Imagine an influence graph where a vertex represents an individual protein (and its associated gene), an edge represents pairwise interactions protein-protein or protein-DNA, and each vertex (gene) has an associated set of samples (patients) in which this gene is mutated. It is widely accepted that cancer is a disease of pathways that can be viewed as specific connected subgraphs of this influence graph. It is important to discover new cancer pathways, by identifying subgraphs of genes that are mutated in a large number of samples. Finding the connected subgraph of $k$ genes that is mutated in the largest number of samples is equivalent to the problem of finding the connected subgraph with $k$ nodes that maximizes the cardinality of the union of the associated sets (see [143]).

This leads us to the first question adressed in this thesis:

[11] Given a set of elements $X$, a set function $f : 2^X \to \mathbb{R}^+$ is *submodular* if $f(\mathcal{A} \cup \{S\}) - f(\mathcal{A}) \leq f(\mathcal{A}' \cup \{S\}) - f(\mathcal{A}')$ for all set $S$ and all collections $\mathcal{A}$ and $\mathcal{A}'$ such that $\mathcal{A}' \subseteq \mathcal{A}$.

[12] If $A$ and $B$ are two independent sets of a matroid such that $|B| > |A|$, then there is an element $x \in B \setminus A$ such that $A \cup \{x\}$ is an independent set.

[13] a property that is true for any subgraph.

Can we design good approximations for the Maximum Coverage problem with the *connectivity constraint* in the *geometric setting* ?

A greedy algorithm would be a natural candidate. Notice that finding a greedy strategy that takes into account the connectivity constraint is not straightforward.

A first idea would be to greedily construct a solution of large value and then connect the different parts of the solution, with the hope that the budget is sufficient to connect everything. In the general (non-geometric) setting, Kuo, Lin and Tsai [103] studied the following strategy. Greedily compute for each vertex in the input graph, a (not necessarily connected) solution of size $\sqrt{k}$ in the $\sqrt{k}$-neighborhood around this vertex and then connect this solution by shortest paths, and return the best of all solutions. They obtained a $O\left(\sqrt{k}\right)$-approximation algorithm.

A second idea would be locally augment the coverage of the solution while maintaining at any step the solution connected. In this thesis we study this greedy strategy and show that when the submodular function and the connectivity are induced by geometry, the approximability of the problem is significantly improved.

▶We introduce the *Maximum Area connected Subset* problem where given a set $\mathcal{D}$ of $n$ unit disks in the plane and an integer $k \leq n$, the goal is to compute a set $\mathcal{D}' \subseteq \mathcal{D}$ of size $k$ that maximizes the area of the union of disks, under the constraint that this union is connected.

We prove that the problem is **NP**-hard (Theorem 2.10) and analyze a greedy algorithm, proving that it is a 2-approximation (Theorem 2.3). We then give a polynomial-time approximation scheme (PTAS) for this problem with resource augmentation, i.e., allowing an additional set of $\varepsilon k$ disks that are not drawn from the input (Theorem 2.4). This algorithm follows Arora's framework for Euclidean TSP [4]. Additionally, for two special cases of the problem we design a PTAS without resource augmentation (Corollary 2.6).

⟶ **see Chapter 2**

The third chapter of this thesis studies tie-breaking rules for greedy algorithms. Sometimes, an algorithm has to make its own choices. Imagine for instance that on Figure 1.2 the two rightmost points were not here, so that $O_1, O_2$ and $S_k$ cover exactly the same number of elements, $2^k - 2$, which are at this step the most profitable sets. So, the greedy algorithm may choose any of them. Picking $O_1$ or $O_2$ leads the algorithm to the optimal solution of size 2, but picking $S_k$ leads the algorithm to a solution of size $\Theta(\log n)$, i.e., to the worst possible greedy solution! Not specifying a tie-breaking rule forces this worst-case performance of $\Theta(\log n)$. For Set Cover, this is unavoidable, but for other problems, tie-breaking can help. For instance, imagine that we want to compute a large independent set in a small degree graph.

**The Maximum Independent Set problem:** Given a graph, find a set of vertices of maximum size, such that no two of those vertices are adjacent.

This is one of the most classic and widely studied **NP**-hard optimization problems. It is known for its inherent hardness of approximation. Indeed, Håstad showed that

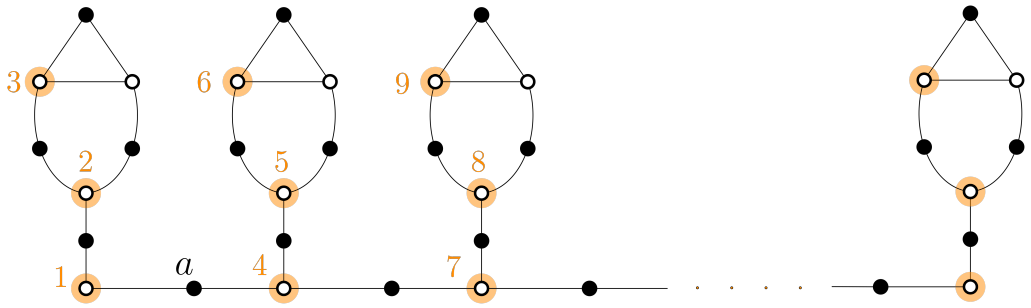



Figure 1.3: An execution of the minimum degree greedy algorithm. The minimum degree is two, and among all vertices of minimum-degree, the algorithm happens to choose vertex 1, then vertices 2,3, etc, leading to a solution consisting of orange vertices. Black vertices form a maximum independent set. Their ratio tends to 5/3 when the size of the graph goes to infinity.

this problem cannot be approximated in polynomial time within a factor $n^{1-\varepsilon}$, unless $\mathbf{P} = \mathbf{NP}$, for all $\varepsilon > 0$ where $n$ is the number of vertices of the input graph [80]. This negative results means that, in the general case, the algorithm that returns one arbitrary vertex already (almost) achieves the best possible approximation in polynomial time. To design interesting algorithms, people focussed on more restricted classes of graphs, such as geometric intersection graphs, bipartite graphs, planar graphs, minor-closed families of graphs. We focus here on the class of bounded degree graphs. For the class of *subcubic* graphs, that are graphs with maximum degree at most three, the Maximum Independent Set problem remains **NP**-hard [59], and even APX-hard [2].

For this problem, one of the best known algorithmic paradigms is the following *minimum degree greedy* method:

Starting with an empty solution,
**while** *the graph is non-empty* **do**
  pick a vertex of *minimum degree*, add it to your solution, and remove it and its neighbors from the graph.

**Algorithm 2:** The *minimum degree* greedy algorithm for the Maximum Independent Set problem.

It is clear that the set returned by this algorithm is an independent set. See Figure 1.3 for an example of an execution of this algorithm.

Halldórsson and Radhakrishnan proved that this greedy algorithm has an approximation ratio of 5/3 in subcubic graphs and more generally an approximation ratio $(\Delta + 3)/2$ on graphs with maximum degree $\Delta \geq 3$ [70], and that all those approximation guarantees are tight. Figure 1.3 shows a asymptotically tight example when $\Delta = 3$. However, in this example, the algorithm made a very poor sequence of choices. Indeed, starting with vertex $a$ would have led the greedy algorithm to an optimal solution. Halldórsson and Yoshihara [74] asked in their paper the following question:

> Can we improve the approximation ratio of the greedy algorithm when we augment it with a *tie-breaking rule* ?

For subcubic graphs, they for instance proved that no tie-breaking rule leads to an approximation ratio better than 5/4 (see Figure 3.1). On the other hand, they provided a

rule that implies a 3/2-approximation (see Theorem 3.2).

▶ In this thesis we present is a novel potential function for the design of greedy algorithms for the independent set problem (Section 3.2). With this new theory we obtain the "ultimate" approximation ratio of 5/4 for greedy on graphs with subcubic graphs (Theorem 3.3), which completely resolves the question from Halldórsson and Yoshihara.

We also obtain a simple and short proof of the $(\Delta + 2)/3$-approximation ratio of any greedy on graphs with maximum degree $\Delta$ (Section 3.3), the result proved previously by Halldórsson and Radhakrishnan. We almost match this ratio by showing a lower bound of $(\Delta + 1)/3$ on the ratio of any greedy algorithm that can use any tie-breaking rule (Theorem 3.22).

We complement our positive results with negative results which prove that the problem of designing good tie-breaking rules for greedy is **NP**-hard (Theorem 3.23) and even hard to approximate on various classes of graphs (Section 3.7.3).

⟶ **see Chapter 3**

In the two last chapters of this thesis, we exploit topological properties of planar graphs and bounded genus graphs to design good approximation algorithms. Designing tight approximation algorithms requires a deep understanding of the structure of the problem that we intend to solve. Planar graphs form a class with many structural properties. Each planar planar has a *dual* planar graph and has small balanced separators [106]; the *Euler characteristic*[14] establishes a relation between the number of vertices, edges and faces; Kuratowski's theorem[15] gives a characterisation in terms of excluded minors; the Jordan's curve theorem[16] implies a connection between cycles and dual cuts, etc.

Eventually, these structural properties imply techniques for the design of good algorithms in planar graphs. In general, the more restricted the instance class is, the stronger and the more diverse the structural properties are, and the better the approximability can potentially be. There is a number of optimization problems that are **APX**-hard in general but admit *polynomial time approximation schemes*[17] (PTAS) in planar graphs. The celebrated technique of Baker [8] provides a general framework for many "ubiquitous" optimization problems including Maximum Independent Set or Minimum Vertex Cover. Other techniques based on planar graphs decomposition imply PTASes for "connectivity" problems such as TSP [5], Steiner Tree [19] or Steiner Forest [11].

For all problems mentionned in the last paragraph, their approximability in planar graphs is well-understood, since the upper bounds (PTASes) match the lower bounds (**NP**-hardness). We now look at the following problem for which there is a gap between the best inapproximability result and the best known approximation, even for planar graphs.

[14] any plane graph satisfies #Faces − #Edges + #Vertices = 2.
[15] a graph is planar if and only if neither $K_5$ nor $K_{3,3}$ appear as a minor.
[16] a closed curve separates the plane into two connected components.
[17] a family of $(1 + \epsilon)$-approximation algorithms, for arbitrarily small values $\epsilon > 0$.

**The Maximum Integral Multi-commodity Flow problem:** Let $G = (V, E)$ be a *supply* graph, with capacities $c : E \to \mathbb{Z}_{\geq 0}$, and $D = \{(s_1, t_1), \ldots, (s_k, t_k)\} \subseteq V \times V$ a family of *demand pairs*. The goal is to find a maximum-cardinality family of paths $\mathcal{P}$ in $G$ whose end-points correspond to demand pairs, such that for each edge $e \in E$, the total capacities of the paths in $\mathcal{P}$ that contain $e$ does not exceed the capacity of $e$.

The well-known *edge-disjoint paths* problem is the special case of this problem where all capacities are 1. Multi-commodity flows, or *multiflows* for short, are well-studied objects in combinatorial optimization. A fractional multiflow of maximum value can be found in polynomial time by linear programming. Often, a multiflow must be integral, and then the problem is much harder.

Assuming that for some $\delta > 0$, not all problems in **NP** can be solved in randomized time $2^{n^\delta}$, there is no $n^{O(1/(\log\log n)^2)}$-approximation even when $G$ is planar and subcubic[18], when $n$ is the number of vertices [36]. On the other land, the first upper bound on the approximability of the edge-disjoint paths problem was $O(\sqrt{|E|})$ and was obtained by a simple greedy algorithm that repeatedly picks any non-connected pair and connect it via a shortest feasible path [96]. The best known approximation guarantee for the edge-disjoint paths and Maximum Integral Multiflow problems is $O(\sqrt{n})$ and is due to Chekuri, Khanna and Shepherd [27]. As far as we know, this is also the best known approximation when $G$ is planar.

In fact, the structure of the edge-disjoint path problem and the Maximum Integral Multiflow problem also depends on how the demand pairs are disposed in the supply graph. In order to design good approximation algorithms for these problems, we will assume that the supply graph $G$ *together with* the demand graph $H = (V, D)$ is planar.

One of the landmark results for this special case is due to Seymour. He proved that if $G + H$ is *planar and Eulerian*, the *cut condition*[19] guarantees a solution to connect all demand pairs; such a solution can be found in polynomial time [137]. Seymour's result has motivated a sequence of follow-up works investigating the edge-disjoint paths problem when $G + H$ is planar. For example, one can decide in polynomial time whether all demand pairs can be connected when $G + H$ is planar and the demand pairs lie on a bounded number of faces of $G$ [117].

Unfortunately the *general* case that $G + H$ is planar is one of these cases in which the edge-disjoint paths problem is **NP**-hard, as Middendorf and Pfeiffer [117] proved. Very little seems to be known about approximation in that setting. Korach and Penn [98] showed that, given that the cut condition holds, one can satisfy all demands except one for each face of $G$, and such a solution can be found in polynomial time. However, this does not imply an approximation ratio in general.

▶ We devise a constant-factor approximation algorithm for the maximization version of the edge-disjoint paths problem if the supply graph together with the demand edges form a planar graph (Theorem 4.2). We also show that the natural linear programming relaxations have constant integrality gap, yielding an approximate max-multiflow min-multicut theorem (Theorem 4.4).

We then generalize these results to higher-genus graphs, by giving the first constant-

[18] in fact, even when $G$ is a wall graph, as in Figure 5.7.

[19] see equation 4.1.

factor approximation algorithm for finding an integral multi-commodity flow of maximum total value for instances where the supply graph together with the demand edges can be embedded on an orientable surface of bounded genus (Theorem 5.1).

⟶ **see Chapters 4 and 5**

Our algorithms are the conjunction of several greedy methods that exploit topological properties of curves and graphs on surfaces. In particular, we establish a connection between the approximation ratio of our algorithm and the coloring number of an intersection graph of a family of curves known in topology as a *1-system*, see Section 5.9.

## 1.3 Preliminaries

*We give here some general definitions and notations that we use all along this thesis. Notions presented in Section 1.3.2 are used in Chapters 4 and 5.*

We say that an algorithm is a **polynomial-time** algorithm if its running time is a polynomial on the size on the input. An algorithm for a maximisation (*resp.* minimisation) problem is an **$\alpha$-approximation**, with **approximation ratio** $\alpha \geq 1$ if it is a polynomial time algorithm and if the value of the solution returned is at least $\mathbf{OPT}/\alpha$ (*resp.* at most $\alpha \cdot \mathbf{OPT}$) where $\mathbf{OPT}$ denotes the value of an optimal solution.

A **polynomial time approximation scheme** is a family of $(1+\epsilon)$-approximation algorithms, for arbitrarily small values $\epsilon > 0$.

### 1.3.1 Graphs notations.

Given a (multi)graph $G = (V, E)$, we denote $V(G) = V$ and $E(G) = E$. When two graphs $G = (V, E)$ and $H = (V, D)$ have the same vertex set $V$ then we denote by $G + H = (V, E \dot\cup D)$ the graph whose edge set is the disjoint union of their edge sets. When a weight function $w : V \to \mathbb{R}$ (or $w : E \to \mathbb{R}$) is given on vertices (or edges), for any subset $U \subseteq V$ (or $U \subseteq E$), we define the weight of $U$ as $w(U) := \sum_{u \in U} w(u)$.

in a graph $G = (V, E)$, the **subgraph induced by** a subset of vertices $U \subseteq V$ is $G[U] := (U, \{uv \in E \mid u, v \in U\})$. To lighten notation, we sometimes write $G \setminus U$ for the subgraph $G[V \setminus U]$. For a vertex $v \in V$, let $N_G(v) := \{u \in V \mid uv \in E\}$ and $N_G[v] := N_G(v) \cup \{v\}$ denote respectively the **open** and **closed neighborhood** of $v$ in $G$. The **degree** of $v$ in $G$ denoted $d_G(v)$ is the size of its open neighborhood. When the context is clear, we simply write $d(v)$. More generally, we define the closed (*resp.* open) neighborhood of a subset $S \subseteq V$, denoted $N_G[S]$ (*resp.* $N_G(S)$) as the union of all closed (*resp.* open) neighborhoods of each vertex in $S$. Given a subset of vertices $U \subseteq V$, we denote $\delta(U)$ the set of edges that are incident to both $U$ and $V \setminus U$. The **degree of a subset** $U \subseteq V$ is the size of its set: $d(U) := |\delta(U)|$.

The maximum degree of a graph $G$ is $\Delta(G) := \max_{v \in V} d(v)$. If every vertex of a graph has degree exactly $k$, then we say that is graph is **$k$-regular**. A 3-regular graph is also called **cubic**. A graph is called **subcubic** if its maximum degree is at most three.

**Independent sets and Coloring.** An **independent set** is a set of vertices $I \subseteq V$ such that every two vertices in $I$ are non-adjacent. We denote by $\alpha(G)$ the **independence number** of $G$, that is, the size of a maximum independent set in $G$.

The **coloring number** $\chi(G)$ is the smallest integer $s$ such that $V(G)$ can be decomposed into a disjoint union of $s$ independent sets, called **color classes**. Such a **coloring** of $G$ with $s$ colors can be viewed as a function $V(G) \to \{1, \ldots, s\}$. Upper bounding the coloring number is a simple tool to prove the existence of large independent sets. Indeed, given a graph $G = (V, E)$ and vertex weight function $w : V \to \mathbb{R}_+$ , there exists an independent set $I$ with weight $w(I) \geq w(V)/s$, that corresponds to a color class of any given coloring with $s$ colors.

The following simple greedy algorithm computes in time $O(|V(G)|^2)$ a coloring of a graph $G$ with at most $\Delta(G) + 1$ colors.

**Input:** a graph $G = (V, E)$.
**Output:** A coloring of $G$.
Let $v$ be a vertex in $G$ of minimum degree;
Compute a coloring of $G[V \setminus \{v\}]$;
Complete this coloring by associating to $v$ the smallest color that does not appear in $N_G(v)$;

**Algorithm 3:** The basic Greedy algorithm for Coloring

More generally, if the input graph belongs to an hereditary class of graphs $\mathcal{G}$ stable by vertex deletion, in the following sense $\forall G \in \mathcal{G}, \forall U \subseteq V(G) : G[U] \in \mathcal{G}$ and such that for all graph in this class, there exists a vertex of degree at most $s$, then Algorithm 3 produces a coloring with at most $s + 1$ colors. For instance, planar graphs form such a hereditary class, with $s = 5$. Thus, with this algorithm we can get a coloring of a planar graph with six colors.

**Cycles and cuts.** A **path** in a graph $G = (V, E)$ is a sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$ for some $k \geq 0$, where $v_0, \ldots, v_k$ are distinct vertices and $e_i = \{v_{i-1}, v_i\}$ is an edge for all $i = 1, \ldots, k$. A **cycle** in a connected graph $G$ is a sequence $(v_0, e_1, v_1, \ldots, e_k, v_k)$ such that $v_1, \ldots, v_k$ are distinct vertices, $\{v_{i-1}, v_i\}$ is an edge for all $i = 1, \ldots, k$, and $v_0 = v_k$. Sometimes we view cycles as edge sets or as graphs.

In a graph $G = (V, E)$, a **cut** is an edge set $\delta(U)$ that contains all edges with one endpoint in $U$ and the other in $V \setminus U$ for some proper subset $\emptyset \neq U \subset V$. A cut $\delta(U)$ is **simple** if both $U$ and $V \setminus U$ are connected in $G$. A cut is simple if and only if it is a minimal cut (for inclusion).

### 1.3.2 Graphs on surfaces

Surfaces are either orientable or non-orientable. A compact orientable surface of genus $g$ can be seen as a connected sum of $g$ tori, or equivalently a sphere with $g$ handles attached on it, where $g$ is called the **genus** of the surface. The **connected sum** of two surfaces $\mathbb{S}$ and $\mathbb{S}'$ is the surface obtained by removing a disk $D$ on $\mathbb{S}$ and a disk $D'$ on $\mathbb{S}'$ and identifying $D$'s boundary with the boundary of $D'$.

Given an integer $g \geq 0$, all compact surfaces with genus $g$ are mutually homeomorphic, and we refer to any one of them as $\mathbb{S}_g$. For instance, $\mathbb{S}_0$ is the sphere and $\mathbb{S}_1$ is the torus.

A (multi)graph has **genus** $g$ or is a **genus-$g$ graph**, if it can be drawn on $\mathbb{S}_g$ without edge crossings, but not on $\mathbb{S}_{g-1}$. A genus-0 graph is also called a planar graph. A genus-$g$ graph may have several non-equivalent embeddings on $\mathbb{S}_g$, but all of them satisfy the same

invariant, called the *Euler characteristic*: #Faces − #Edges + #Vertices = $2 - 2g$. Using this formula one can for instance bound the average degree of a genus-$g$ graph by $O(\sqrt{g})$. In particular we can get:

**Theorem 1.1.** *(Map-coloring theorem) A genus-g graph can be colored in polynomial time with at most* $\chi_g := \lfloor \frac{7+\sqrt{1+48g}}{2} \rfloor$ *colors.*

For $g \geq 1$, the coloring is obtained by Algorithm 3 [77]. For $g = 0$, this is an algorithmic version of the four color theorem. Robertson, Sanders, Seymour, and Thomas [131] improved on the original proof by Appel, Haken, and Koch and showed that a 4-coloring of a planar graph $G$ can be computed in time $O(|V(G)|^2)$. Interestingly Heawood conjectured that $\chi_g$ colors were necessary to color a genus-$g$ graph, and 80 years later Ringel and Youngs proved that the clique of size $\chi_g$ was embeddable on $\mathbb{S}_g$, for any $g \geq 0$.

**Duality.** Given an embedding of a genus-$g$ graph $G$ on $\mathbb{S}_g$, there exists a uniquely defined dual graph, denoted as $G^*$. This graph can be embedded on the same surface as $G$. There exists a bijection between the faces of $G$ and the vertices of $G^*$, a bijection between the vertices of $G$ and the faces of $G^*$, and a bijection between the edge sets of $G$ and of $G^*$. Moreover, the embeddings of $G$ and $G^*$ are consistent: with this bijection, edges only cross their image, faces only contain their image and reciprocally. For notational simplicity, the latter bijection is often implicit.

We say that an edge set $F$ in a graph is a (simple) **dual cut** if the corresponding set of edges $F^*$ in the dual is a (simple) cut. A cycle $C$ in $G$ is called **separating** if it is a dual cut, and **non-separating** otherwise. Note that every separating cycle is a simple dual cut.

**Combinatorial embeddings.** Given a graph, let $\delta(v)$ denote the set of edges incident to a vertex $v$. Given an embedding of a graph on an orientable surface, and an arbitrary orientation of this surface, for each vertex $v$, a clockwise cyclic order can be defined on the edges of $\delta(v)$. Note that contracting an edge $e = \{u, v\}$ results in removing $e$ from $\delta(u)$ and from $\delta(v)$ and concatenating the orders to obtain the clockwise cyclic order of the edges around the vertex created by the contraction. Using these orders together with the incidence relation between edges and faces, embeddings become purely combinatorial objects.

For additional details and results about graphs on surfaces see e.g. [119, 39].

# 2 | Maximum Disk Coverage with Connectivity Constraints

## 2.1 Introduction

Maximizing a submodular function[1] under constraints is a classical problem in computer science and operations research [124, 147]; the most commonly studied constraints are cardinality, knapsack and matroids constraints. A natural constraint that has received little attention is the *connectivity* constraint. In this chapter, we study the following problem: given a set of $n$ unit disks in the plane, select a subset of $k$ disks that maximize the area of their union, under the constraint that this union is connected. We call this problem the *Maximum Area Connected Subset* problem (MACS). Notice that the area covered by a set of disks is a monotone submodular function.

The problem is motivated by wireless router deployment, first introduced in [103], where the goal is to install a given number of routers to maximize the number of clients covered. When the clients are uniformly spread in the plane, the number of clients in a region can be approximated by the area of that region, leading to our problem.

**Our Contributions.** We first analyze a variant of the greedy algorithm and show that it computes a 2-approximation to MACS (Theorem 2.3); further, the analysis is tight. In contrast, the natural algorithm that greedily adds one disk at a time can end up with a solution with area a factor of $\Omega(k)$ worse than the optimal solution.

To improve upon the 2-approximation ratio, we turn to the resource augmentation setting in which the algorithm is allowed to add a few additional disks that need not be drawn from the input. We design a PTAS for the resource augmentation version of the problem using Arora's shifted quadtree technique (Theorem 2.4)[2]. The proof hinges on the existence of a near-optimal solution with $O(\varepsilon k)$ additional disks and with additional structure that allows its computation by dynamic programming.

As a corollary, we show that for two special cases of MACS we can in fact design a PTAS without resource augmentation: *i*) when the Euclidean distances between centers of the input disks are well-approximated by shortest paths in their intersection graph (Corollary 2.6), and *ii*) when every point of the relevant region of the Euclidean plane is covered by at least one input disk (Corollary 2.9).

[1] Given a set $X$, a function $f : 2^X \to \mathbb{R}$ is *submodular* if given any two subsets $A, B \subseteq X$, $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$.

[2] We also developed an alternative algorithm using Mitchell's $m$-guillotine dissection technique [118] with faster running time. See the original paper [82].

On the other hand, via a reduction from the Rectilinear Steiner Tree problem, we show that MACS is **NP**-hard (Theorem 3). We also show that MACS for the input of a set of quadrilaterals instead of disks, the problem is **APX**-hard (Theorem 2.12). We leave open the question whether MACS is **APX**-hard or admits a PTAS without resource augmentation.

### 2.1.1 Related work

Maximizing a monotone submodular function under constraint(s) is a subject that has received a large amount of attention over the years. Kulik et al. designed an approximation algorithm for maximizing a submodular function under multiple linear constraints with an approximation ratio that (almost) matches the bound $e/(e-1)$ [102]. The greedy algorithm gives a $(k+1)$-approximation where the objective function is subject to $k$ matroid constraints [124]. Lee et al. later improved the approximation arbitrarily close to $k$ when $k \geq 2$ using a local-search approach [105]. When a monotone submodular function is subject only one matroid constraint, this is a randomized $\frac{e}{e-1}$-approximation [21].

Our problem can be regarded as maximizing a submodular function under a cardinality (knapsack) constraint and a connectivity constraint. Notice that the connectivity constraint is central to the difficulty of our problem: without connectivity constraints, MACS admits a PTAS even in the more general case of convex pseudodisks [25]. However even without the connectivity constraint the problem remains **NP**-hard[3].

Another motivation for studying the connectivity constraint is related to cancer genome studies. Suppose that a vertex represents an individual protein (and associated gene), an edge represents pairwise interactions, and each vertex has an associated set. Finding the connected subgraph of $k$ genes that is mutated in the largest number of samples is equivalent to the problem of finding the connected subgraph with $k$ nodes that maximizes the cardinality of the union of the associated sets (see [143]).

In the general (non-geometric) setting, there exists a $O\left(\sqrt{k}\right)$-approximation algorithm for maximizing a monotone submodular function [103]. This approximation is obtained by computing for each vertex in the graph, a non-necessarily connected solution of size $\sqrt{k}$ in the $\sqrt{k}$-neighborhood around this vertex and then connect these vertices by shortest paths. Our results show that when the submodular function and the connectivity are induced by a geometric configuration, the approximation ratio can be significantly improved.

We next consider several related problems where the connectivity constraint plays an important role. The goal of the node-cost budget problem [129] is to find a connected set of vertices in a general graph to collect the maximum profit on the vertices while guaranteeing the total cost does not exceed a certain budget. Notice that in this setting the submodular function is a simple additive function of the profits. Another related problem [23] is to assign radii to a given set of points in the plane so that the resulting set of disks is connected and the objective is to minimize the sum of radii.

Khuller et al. [94] study the budgeted connected dominating set problem where given a general undirected graph, the goal is to select $k$ vertices whose induced subgraph is connected and that maximizes the number of dominated vertices. It was pointed out to us that their algorithm can be used to give a $13\frac{e}{e-1}$-approximate solution for MACS. The authors of [79] consider the problem of selecting $k$ nodes of an input node-weighted graph

[3]The reduction is from Maximum independent set problem that is **NP**-hard in unit-disk graphs.

to form a connected subgraph, with the aim of maximizing or minimizing the selected weight.

We now turn to the geometric setting. A logarithmic-factor approximation algorithm is known [67] for the connected sensor coverage problem in which one must select at most $k$ sensors in the plane forming a connected communication network and covering the desired region, where the region covered by each sensor is convex [57, 89]. A $(1+\varepsilon)$-approximation algorithm in time $n^{O(1/\varepsilon)}$ for the maximum independent set problem on unit disk graphs is known [115]. The authors of [114] present a constant-factor approximation algorithm for several problems on unit disk graphs, including maximum independent set. For the geometric set cover problem where the goal is to cover a given set of input points with a minimum number of given disks, a PTAS is possible [120].

### 2.1.2 Our results

The Euclidean distance between two points $x$ and $y$ is denoted by $\|x-y\|$. When there is no confusion, we will refer to a point $x$ in the plane and the unit disk centered at $x$ interchangeably.

**Definition 2.1.** Given a finite set $S$ in the plane, the **unit disk intersection graph** $\mathrm{UDG}(S)$ is a graph on $S$ where $\{x,y\} \subseteq S$ is an edge of $\mathrm{UDG}(S)$ if and only if $\|x-y\| \leq 2$.

A set $S$ of points in the plane are **connected** if $\mathrm{UDG}(S)$ is a connected graph.

**Definition 2.2.** The **Maximum Area Connected Subset** (**MACS**) problem is as follows.
**Input**: a finite set of points $X \subseteq \mathbb{R}^2$ and a non-negative integer $k$, where $k \leq |X|$.
**Output**: a subset $S \subseteq X$ of size at most $k$ such that the unit-disk graph $\mathrm{UDG}(S)$ of $S$ is connected.
**Goal**: maximize the area of the union of the unit disks centered at the points of $S$.
The optimal solution of MACS on input $(X,k)$ is denoted by $\mathbf{OPT}(X,k)$.

When the context is clear, we refer to $\mathbf{OPT}(X,k)$ as $\mathbf{OPT}$, which is also used to denote the area covered by the optimal solution; observe that $\mathbf{OPT}$ is trivially upper-bounded by $\pi k$. Any $S \subseteq X$ with $|S| \leq k$ for which $\mathrm{UDG}(S)$ is connected is called a *feasible solution*.

We state our main results below.

**Theorem 2.3** (Approximation)**.** *There exists a 2-approximation for* MACS *(Algorithm 4).*

With resource augmentation, we obtain a $(1+\varepsilon)$-approximation.

**Theorem 2.4** (Resource augmentation)**.** *Let $\varepsilon > 0$ be a given parameter. Given a set of points $X \subseteq \mathbb{R}^2$ and a non-negative integer $k$, there is an algorithm (Algorithm 5) that computes, in time $n^{O(\varepsilon^{-3})}$, a subset $S \subseteq X$ of size at most $k$ and a set $S_{add} \subseteq \mathbb{R}^2$ of at most $\varepsilon k$ points, such that $\mathrm{UDG}(S \cup S_{add})$ is connected and the area of the union of the unit disks centered at $S$ is at least $(1-\varepsilon)\mathbf{OPT}(X,k)$ .*

Theorem 2.4 can be obtained alternatively by a (deterministic) guillotine cut approach with a faster running time, see [82].

Let $d_G(x,y)$ denote the distance between two vertices $x$ and $y$ of $G$. A set $X$ of points in the plane is called *$\alpha$-well-distributed* if $\mathrm{UDG}(X)$ is an $\alpha$-spanner for $X$ [121]:

**Definition 2.5.** Given $\alpha > 0$, a finite set $X$ of points in the plane is called **$\alpha$-well-distributed** if for all $x, y \in X$, $d_{UDG(X)}(x, y) \leq \lceil \alpha \cdot \|x - y\| \rceil$.

When the input is well-distributed then the set of additional disks $S_{add}$ obtained from Theorem 2.4 can be transformed into a set of $O(|S_{add}|)$ *input disks* achieving the same goal (Lemma 2.26). Thus using a result of "concavity" of the value of optimal solutions (Lemma 2.27) , we get a PTAS without resource augmentation.

**Corollary 2.6.** *There exists a* PTAS *for* MACS *on $\alpha$-well-distributed inputs, where $\alpha$ is a fixed constant (Algorithm 6).*

We apply this result to a particular case of well-distributed inputs:

**Definition 2.7.** A set $X$ is called **pseudo-convex** if the convex-hull of $X$ is covered by the union of the unit disks centered at points of $X$.

**Lemma 2.8.** *A pseudo-convex set $X$ is 3.82-well-distributed.*

The exact constant is $12/\pi < 3.82$ and is obtained by simple geometrical observations.

**Corollary 2.9.** MACS *on pseudo-convex inputs admits a polynomial-time approximation scheme.*

We next turn to the hardness of MACS. By a reduction from Rectilinear Steiner Tree we show:

**Theorem 2.10** (Hardness)**.** MACS *is **NP**-hard.*

For slightly more complicated geometrical object, the problem becomes even hard to approximate.

**Definition 2.11.** The quad-connected-cover is defined as follows.
**Input**: a set $\mathcal{T}$ of $n$ convex quadrilaterals in the plane, and an integer $k$.
**Output**: a subset $T$ of $\mathcal{T}$ of size $k$ such that the intersection graph of $T$ is connected.
**Goal**: Maximize the area covered by the union of quadrilaterals in $T$.

By a reduction from 3-set-cover we show:

**Theorem 2.12.** Quad-connected-cover *is **APX**-hard.*

## 2.2 The Two-by-two greedy algorithm (Proof of Theorem 2.3)

In the section we present a simple 2-approximation for MACS based on a greedy approach: we iteratively add two unit disks that maximize the area covered while maintaining feasibility. Interestingly, the algorithm that adds disks one at a time is not a constant approximation algorithm. See Figure 2.1 for an example. Moreover, trying all possible sets of $s$ disks, for any $s \geq 3$, in the neighborhood of the current solution does not improve the approximation ratio. This can be seen on Figure 2.2 where the first disk chosen by the algorithm is not $x$, but $x_s$.

Let $B_x$ denote the unit disk centered at $x \in \mathbb{R}^2$ and $B(S) = \bigcup_{x \in S} B_x$ denote their union. The area covered by a set $C \subset \mathbb{R}^2$ is denoted by $\mathcal{A}(C)$. When $C = B(S)$, its area is

**Input:** $X \subseteq \mathbb{R}^2, k \geq 0$, where $X$ is finite and $k \leq |X|$.
**Output:** a feasible set of size $k$.
**if** *$k$ is even* **then**
  $S \leftarrow$ any two intersecting disks of $X$;
**else**
  $S \leftarrow$ any one disk of $X$;
**while** $|S| \leq k - 2$ **do**
  $\{x, x'\} \leftarrow \arg\max \{\mathcal{A}(S \cup \{x, x'\}) \colon x, x' \in X,\ S \cup \{x, x'\} \text{ is feasible } \}$;
  $S \leftarrow S \cup \{x, x'\}$;
**return** $S$;

**Algorithm 4:** The Two-by-two algorithm for MACS

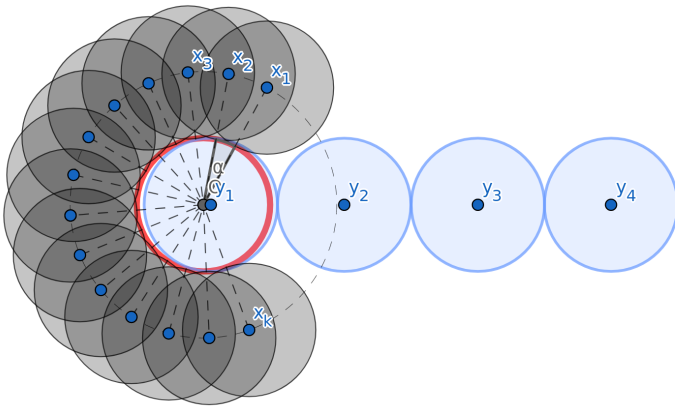



Figure 2.1: The greedy algorithm that adds only one connected disk maximizing the marginal area covered is not a constant factor algorithm. For any $k \geq 0$ and $\varepsilon > 0$, consider the above input where $O = (0, 0)$, and $y_i = (2(i-1) + \varepsilon, 0)$ for all $i$. Then, put all $x_1, \ldots, x_k$ evenly spaced (by an angle $\alpha$) on a circle of radius 2 around $O$ so that none of them intersect $y_2$. Each light grey regions are covered by only one disk $x_i$ so the marginal gain of adding $x_i$ to any solution is at least the area of one of these regions, say $a > 0$. If $\varepsilon$ is chosen such that $\mathcal{A}(B_{y_1} \setminus B_O) < a$, then if the algorithm starts by picking disk $O$, it will then choose all $x_j$, so that the area covered by the solution is upper-bounded by the area of a radius 3 disk, $9\pi$, while the optimal solution (disks $y_i$) has area $\pi k$.

simply written as $\mathcal{A}(S)$. A subset of the vertices of a graph is a **dominating set** if every vertex belongs to the set or is adjacent to some vertex of it.

One can find an example similar to Figure 2.2 to show that optimizing the initial choice of the first disk(s) does not improve the approximation ratio.

**Theorem 2.3** (Approximation)**.** *There exists a* 2*-approximation for* MACS *(Algorithm 4).*

We can assume w.l.o.g. that UDG($X$) is connected; otherwise we return the maximum value over all connected components. For the analysis, we divide the execution of Algorithm 4 in two phases. An iteration belongs to the first phase as long as the current solution $S$ is not a dominating set in the graph UDG($X$).

During the first phase, in each iteration the area covered increases by at least $\pi$. During the second phase, since the current solution is a dominating set, any disk can be added while keeping the solution feasible. Therefore the algorithm reduces to a standard greedy algorithm to maximize a submodular function, and the analysis is similar to the proof that Nemhauser's algorithm is a $\left(\frac{e}{e-1}\right)$-approximation for classic submodular functions.

Figure 2.2 shows a tight example.

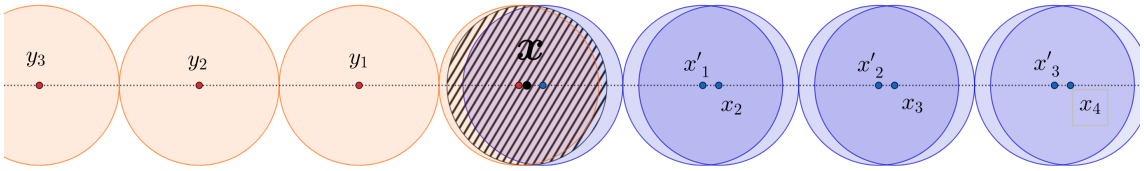



Figure 2.2: A tight example for Algorithm 4. For any $\varepsilon > 0$, $X$ contains $x = (0,0)$ (stripe-shaded), $x_i = (2(i-1)+i\varepsilon, 0)$ and $x'_i = ((2+\varepsilon)i, 0)$ for $1 \le i \le k$ (blue) and $y_i = (-2i-\varepsilon/2, 0)$ for $0 \le i \le k$ (orange). Suppose that $k = 1+2\kappa$ is odd and the algorithm starts with $S_0 := \{x, x\}$. Then the algorithm will add $\{x_i, x'_i\}$ in iteration $i$ since it covers more additional area than $\{y_0, y_1\}$. The solution returned (blue disks) covers an area of $\pi + \kappa(\pi + f(\varepsilon)) \approx \frac{k}{2}\pi$, for some function $f(\cdot)$ with $\lim_{\varepsilon \to 0} f(\varepsilon) = 0$, while **OPT** (orange disks) covers an area $k\pi$.

### 2.2.1 Analysis of the Two-by-Two algorithm

We first analyze the even case where $k = 2\kappa$, and then we reduce the odd case to the even one. Let $S_\kappa = \{x_1, x_2, \ldots, x_{2\kappa}\}$ be the solution returned by the algorithm. Let $S_i = \{x_1, \ldots, x_{2i}\}$ be the set right after the $i$-th iteration and let $d$ be the smallest integer such that $S_d$ is a dominating set in UDG($X$). If such an integer does not exist, i.e., $S_\kappa$ is not a dominating set, then set $d = \kappa$.

**Claim 2.13.** *The area $\mathcal{A}(S_d)$ is at least $\pi d$.*

*Proof.* For $i < d$, $S_i$ is not a dominating set. Then there exist two disks $y, y'$ such that $B(S_i) \cap B_y = \emptyset$ and $S \cup \{y, y'\}$ is connected. Adding such a pair increases the area covered by at least $\mathcal{A}(B_y) = \pi$. Since $(x_{2i+1}, x_{2i+2})$ is chosen to maximize $\mathcal{A}(S_i \cup \{x, x'\})$ among all feasible pairs, $\mathcal{A}(S_{i+1}) \ge \mathcal{A}(S_i \cup \{y, y'\}) \ge \mathcal{A}(S_i) + \pi$. By induction, $\mathcal{A}(S_d) \ge \pi d$. □

Note that when $d = \kappa$, Claim 2.13 immediately implies that $\mathcal{A}(S_\kappa) \ge \frac{\mathbf{OPT}}{2}$. Also regardless of the initial choice, the area covered by the first two disks is at least $\pi$. This observation will be useful when we prove the case where $k$ is odd.

**Claim 2.14.** *For all* $d \le i \le \kappa$, $\mathcal{A}(\boldsymbol{OPT}) \le \mathcal{A}(S_i) + \kappa \cdot (\mathcal{A}(S_{i+1}) - \mathcal{A}(S_i))$.

*Proof.* It is easy to check that the function $\mathcal{A}(\cdot)$ satisfies the following properties for all $H \subseteq H' \subseteq X$ :

**(1) positivity :** $\mathcal{A}(H) \ge 0$.

**(2) monotonicity :** $\mathcal{A}(H) \le \mathcal{A}(H')$.

**(3) submodularity :** $\forall H'' \subseteq X,\ \mathcal{A}(H' \cup H'') \le \mathcal{A}(H \cup H'') - \mathcal{A}(H) + \mathcal{A}(H')$.

Let $\mathbf{OPT} = \{y_1, \ldots, y_{2\kappa}\}$. We have for all $d \le i \le \kappa$ :

$$\begin{aligned}
\mathcal{A}(\mathbf{OPT}) &\le \mathcal{A}(S_i \cup \mathbf{OPT}) \\
&= \mathcal{A}(S_i) + (\mathcal{A}(S_i \cup \{y_1, y_2\}) - \mathcal{A}(S_i)) + \ldots \\
&\qquad + (\mathcal{A}(S_i \cup \{y_1, \ldots, y_{2\kappa}\}) - \mathcal{A}(S_i \cup \{y_1, \ldots, y_{2\kappa-2}\})) \\
&\le \mathcal{A}(S_i) + (\mathcal{A}(S_i \cup \{y_1, y_2\}) - \mathcal{A}(S_i)) + \cdots + (\mathcal{A}(S_i \cup \{y_{2\kappa-1}, y_{2\kappa}\}) - \mathcal{A}(S_i)) \\
&\le \mathcal{A}(S_i) + \kappa \cdot (\mathcal{A}(S_i \cup \{x_{2i+1}, x_{2i+2}\}) - \mathcal{A}(S_i)) \\
&= \mathcal{A}(S_i) + \kappa \cdot (\mathcal{A}(S_{i+1}) - \mathcal{A}(S_i)) .
\end{aligned}$$

The first and the second inequality respectively come from *monotonicity* and *submodularity*, while the third one follows from the fact that for $i \ge d$, $(x_{2i+1}, x_{2i+2})$ is the pair of disks maximizing $\mathcal{A}(S_i \cup \{x, x'\})$ among **all pairs** $(x, x')$ in $X$. As $S_d$ is a connected dominating set in $X$, all pairs $(y_{2j-1}, y_{2j})$ for $1 \le i \le \kappa$ are considered. □

We can now re-write Claim 2.14 as

$$\text{For all } d \le i \le \kappa : \mathcal{A}(S_{i+1}) \ge \left(1 - \frac{1}{\kappa}\right) \mathcal{A}(S_i) + \frac{\mathbf{OPT}}{\kappa}.$$

Combined with Claim 2.13, simple algebra yields that for $d \le i \le \kappa$, we have

$$\mathcal{A}(S_i) \ge \left[1 - \left(1 - \frac{d}{2\kappa}\right)\left(1 - \frac{1}{\kappa}\right)^{i-d}\right] \mathbf{OPT}.$$

Therefore, for $i = \kappa$ we have

$$\mathcal{A}(S) = \mathcal{A}(S_\kappa) \ge \left[1 - \left(1 - \frac{d}{2\kappa}\right)\left(1 - \frac{1}{\kappa}\right)^{\kappa-d}\right] \mathbf{OPT} = \left[1 - \frac{1}{2}(1+t)\left(1 - \frac{1}{\kappa}\right)^{\kappa t}\right] \mathbf{OPT}$$

where $t = \dfrac{\kappa - d}{\kappa} \in [0, 1]$. As $1 + x \le e^x$ for all $x \in \mathbb{R}$, we get

$$\mathcal{A}(S) \ge \left(1 - \frac{1}{2}(1+t)e^{-t}\right) \mathbf{OPT} \ge \left(1 - \frac{1}{2}e^t e^{-t}\right) \mathbf{OPT} = \frac{1}{2}\mathbf{OPT},$$

concluding the proof of the case when $k$ is an even number.

For the odd case $k = 2\kappa - 1$: in the first iteration, instead of adding two disks to $S_1$, we add a single disk of $X$ to $S_1$. This is equivalent to adding two copies of the same disk. This iteration belongs to the first phase, and the only properties we used in the first phase is that each iteration adds an area of $\pi$, and keeps the solution feasible; these are clearly true for the first iteration even with one disk. We have proved Theorem 2.3.

## 2.3 PTAS with resource augmentation (Proof of Theorem 2.4)

**Theorem 2.4** (Resource augmentation)**.** *Let $\varepsilon > 0$ be a given parameter. Given a set of points $X \subseteq \mathbb{R}^2$ and a non-negative integer $k$, there is an algorithm (Algorithm 5) that computes, in time $n^{O(\varepsilon^{-3})}$, a subset $S \subseteq X$ of size at most $k$ and a set $S_{add} \subseteq \mathbb{R}^2$ of at most $\varepsilon k$ points, such that $\mathrm{UDG}(S \cup S_{add})$ is connected and the area of the union of the unit disks centered at $S$ is at least $(1-\varepsilon)\boldsymbol{OPT}(X,k)$ .*

We first summarize the high level ideas; the details are then presented in subsequent sections. Let $(X,k)$ denote an input of MACS and **OPT** be the optimal solution of MACS on input $(X,k)$. When the context is clear **OPT** can also denote the total area covered by the union of the unit disks centered in points of **OPT**.

We start by guessing a bounding box of size $\Theta(k) \times \Theta(k)$ that contains **OPT**. Next, another square of size $L \times L$, where $L = \Theta(k)$, is randomly shifted so that it always contains the bounding box. We remove all disks that are outside the square. That square is then recursively partitioned into smaller squares until they have (large) constant size. This hierarchical dissection induces a grid.

We remove all disks that intersect the lines of the grid. In contrast, we deploy some new disks ($X_{add}$) in some strategic *portal* positions along the lines and near the boundary of all the smallest squares.

Next, we use dynamic programming to build a solution from the smallest squares upwards. The difficulty lies in having to guarantee the connectivity when combining solutions from smaller squares into larger squares using additional disks, while controlling the time complexity and the number of disks added.

The key of our approach lies in Lemma 2.20, in which we argue that with constant probability, there exists a well-structured near-optimal solution that uses at most $\varepsilon k$ additional disks.

### 2.3.1 The grid

The first step is to reduce significantly the size of the input by guessing the position of the optimal solution.

**Lemma 2.15.** *There exists a point $c \in X$ such that **OPT** is contained in an axis-parallel square of side length $4k$ and centered in $c$.*

*Proof.* For $c$, take any disk in **OPT** and recall that **OPT** is connected and has at most $k$ disks, so all the disks in **OPT** are contained in the square centered at $c$ and with side length $4k$. □

Given the randomly shifted hierarchical dissection, we use the same terminology as Vazirani [145, Chapter 11] to define the *root square*, the *shift* of the dissection, the *horizontal* and *vertical lines*, the *levels* of squares and of lines of the dissection, and the *portals*. This precise definitions are given in the next subsection. The recursive dissection stops when a square has side length $L_0 = \Theta(\varepsilon^{-1})$ (*leaf square*). Portals are either at the intersection of grid lines or distributed along the grid lines (with varying density). We make some observations here (all details and proofs are in the following section and the appendix). First, the distance between two consecutive portals on a line at level $\ell$ is $O(L/(m2^{\ell}))$, where $m$ represents the density of portals on the grid. The greater this

parameter, the greater the accuracy of the solution and higher the running time. Choosing $m = \Theta(\varepsilon^{-1}\log(L/L_0)) = O(\varepsilon^{-1}\log(\varepsilon k))$ allows us to compute a near-optimal solution in polynomial time.

**Observation 2.16.** *If an horizontal line of level $\ell$ crosses a vertical line of level greater than or equal to $\ell$ then the intersection point is a horizontal portal.*

We define a set $\mathcal{P}$ of **portal disks** which we position at or near the portals. If a portal $(i, j)$ is on exactly one line of the grid then we add the portal disk $(i, j)$ to $\mathcal{P}$. If a portal $(i, j)$ is at the intersection of two lines of the grid, then $i$) if it is a horizontal portal then we add to $\mathcal{P}$ two portal disks $(i, j + 2)$ and $(i, j - 2)$, and $ii$) if it is a vertical portal then we add to $\mathcal{P}$ two portal disks $(i - 2, j)$ and $(i + 2, j)$.

Given a square $C$ of the dissection, the **potential portal disks** of $C$, denoted by $\mathcal{P}_C$, are the portal disks on the boundary of $C$.

**Observation 2.17.** *For any square, the number of potential portal disks is $O(m) = O(\varepsilon^{-1}\log(\varepsilon k))$.*

The **border** of a leaf square $C$, denoted as $\partial C$, is the set of points in $C$ within distance 1 from $C$'s boundary. The remaining points of $C$ are called the **core** of $C$, written as $core(C)$. A unit disk with its center in $C$ intersects the boundary if and only if its center lies in the border. If two disks are in the core of two different leaf squares, then they do not intersect. We refer to the union of the core of all *leaf squares* as the *core*. In a leaf square $C = [a, b] \times [c, d]$, the set of points formed by the boundary of the square $[a + 2, b - 2] \times [c + 2, d - 2]$ is called the **fence**. We cover the fence of $C$ by **fence disks**, aligned such that each corner of this square is the center of a fence disk. See Figure 2.4. We denote by $\mathcal{F}$ the set of all fence disks for all leaf squares. The set of portal disks and fence disks form the set of **additional disks** $X_{add} = \mathcal{P} \cup \mathcal{F}$.

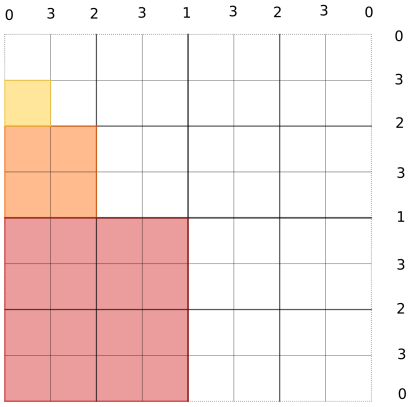



Figure 2.3: An illustration of the grid with $d = 3$. Numbers on the top and the right are the level of the corresponding lines and the red, orange and yellow are respectively the example of square of the dissection at level $1, 2$ and $3$.

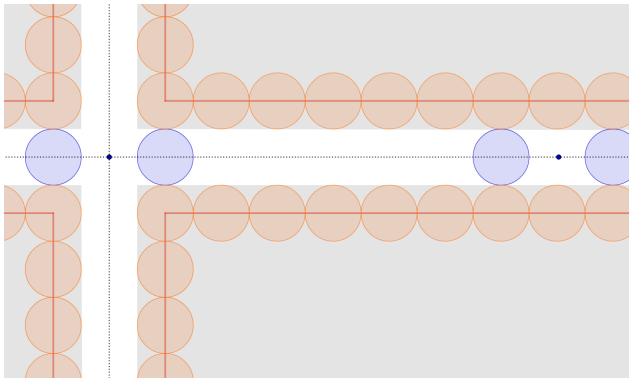


Figure 2.4: The grey and white area are respectively the *core* and the *border*. Dotted lines are from the grid while the orange lines represent the *fence* and orange disks are the *fence disks*. Blue points are (vertical) *portals* and blue disks are *portal disks*.

### 2.3.2 Detailed construction of the grid

Let $L'$ be the sidelength of the box given by the Lemma 2.15, and set $X'$ be the set of points of X lying inside this box. Let $L$ be the smallest power of 2 greater than $2L'$. The **root square** is defined to be the axis-parallel $L \times L$ square with the same left-bottom corner as the bounding-box.

A **shift** is an non-negative integer $a$ smaller than or equal to $L/2$. We say that the root square is **shifted** by $a$ if it is translated by the vector $(-a, -a)$. Notice that any shifted root square contains the bounding-box.

Given a shifted root square, we can define its *dissection* as a recursive partitioning into smaller squares. The $L \times L$ root square is divided into four squares of size $L/2 \times L/2$. Each of these squares is again divided into four $L/4 \times L/4$ squares, so forth. The process stops when the side length of a square is equal to $L_0 = \Theta(\varepsilon^{-1})$. Let $d = \log(L/L_0) = \mathcal{O}(\log(\varepsilon k))$. We can think of this partitioning as 4-ary tree, where each node at level $\ell$ corresponds to a $L_0 2^\ell \times L_0 2^\ell$ square and has four children corresponding to four $L_0 2^{\ell-1} \times L_0 2^{\ell-1}$ squares. The root square is at level 0 and the **leaf squares** are at level $d$. Given two squares of level $\ell$ and level $\ell'$, $\ell > \ell'$, we say the former is of higher level than the latter. So the leaf square is the one with the highest level.

This dissection defines a **grid** composed of $2 \cdot (2^d - 1)$ horizontal and vertical lines of length $L$. We say that a line is at **level** $\ell \in \{1, \ldots, d\}$ if it was added on the grid to divide a square at level $l - 1$ into four squares at level $\ell$. There are $2^\ell$ horizontal (*resp.* vertical) lines at level $\ell$. See figure 2.3.

On each horizontal line of level $\ell \geq 1$, we will place a set of *vertical*—notice the naming asymmetry—**portals** of level $\ell$, near which (not exactly on which) we will deploy the portal disks to facilitate the connection of disks on both sides of this line. We define a set of horizontal portals for each vertical line in an analogous manner. Notice that it is possible that a point is both a vertical portal and a horizontal portal. Let $m = O(\varepsilon^{-1}d)$ be a power of two. Along a line of level $\ell$, there are $m2^\ell + 1$ portals evenly spaced so that the distance between two neighboring portals have distance exactly $\dfrac{L}{m2^\ell}$.

### 2.3.3 Dynamic program

The algorithm uses dynamic programming. The dynamic programming table is indexed by *configurations*.

**Definition 2.18.** A **configuration** is a 5-tuple $\mathcal{C} = [C, t, t_{add}, P, \sim]$, where:

- $C$ is a square of the dissection.
- $0 \leq t \leq k$ is an integer, denoting the number of disks of $S$ used by the solution inside $C$.
- $0 \leq t_{add} \leq \varepsilon k$ is an integer, denoting the number of additional disks used by the solution inside $C$.
- $P \subseteq \mathcal{P}_C$ is a subset of potential portal disks of $C$, those that are used by the solution.
- $\sim$ is a planar connectivity relation on $P$ (described below), representing the connectivity achieved so far by the part of the solution inside $C$.

In the following, to facilitate discussion, we will refer to portals disks as simply portals. An equivalence relation $\sim$ on $P$ is a **planar connectivity relation** if each equivalence class has an associated tree with the portals at the leaves, and there exists a planar embedding of those trees inside the square, such that the trees do not intersect.

The content of the dynamic programming table, the *value* of a configuration $\mathcal{C} = [C, t, t_{add}, P, \sim]$, denoted by $\mathcal{A}(\mathcal{C})$, is the maximum area that can be covered by a set $S \subseteq X$ of $t$ disks in $C \cap core$[4], such that there is a set $S_{add} \subseteq X_{add}$ of $t_{add}$ additional disks such that any $p, p' \in P$ with $p \sim p'$ are in the same connected component induced by $S \cup S_{add} \cup P$. We say that $p$ and $p'$ are **connected in $\mathcal{C}$**. If such sets $\{S, S_{add}, P\}$ do not exist for configuration $\mathcal{C}$, the value $\mathcal{A}(\mathcal{C})$ is set to $-\infty$.

### 2.3.4 Computing leaf entries of the dynamic programming

We first explain how to fill the entries of the table corresponding to the leaf squares. For each leaf square $C$, we enumerate

1. all possible subsets $S \subseteq X' \cap core(C)$ of at most $k_0$ disks, for a parameter $k_0 = O(\varepsilon^{-3})$ (see Lemma 2.20).
2. all possible subsets $S_f \subseteq \mathcal{F} \cap C$,
3. all possible subsets $P \subseteq \mathcal{P}_C$, and
4. all possible planar connectivity relations $\sim$ on $P$.

We say that $(S, S_f, P, \sim)$ is a **guess** in $C$ and that it is **usable** if one of the following two conditions holds:

**Case 1.** if $P = \emptyset$, then $S \cup S_f$ is connected, otherwise

**Case 2.** every connected component of $S \cup S_f \cup P$ contains at least one portal disk in $P$.

Each usable guess $(S, S_f, P)$ in $C$ corresponds to a configuration $\mathcal{C} := [C, |S|, |S_f|, P, \sim]$, where $\sim$ is the planar connectivity relation on $P$ induced by the connected components of $S \cup S_f \cup P$.

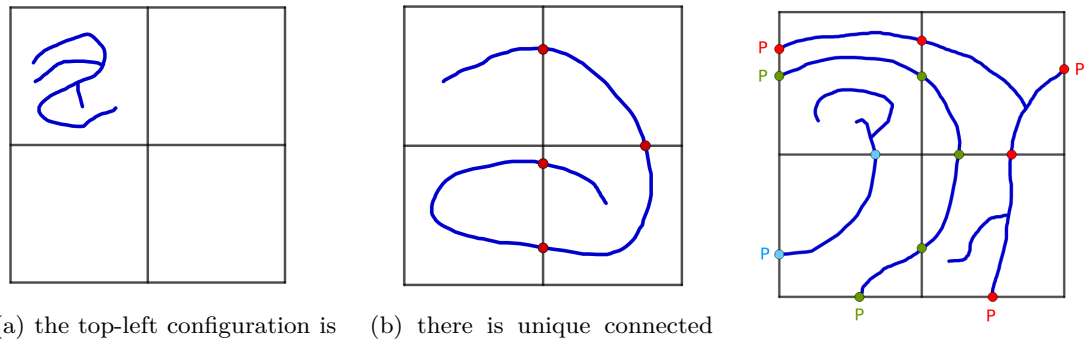
Several usable guesses $(S, S_f, P)$ can potentially correspond to the same configuration $\mathcal{C}$. The value of $\mathcal{C}$ is computed[5] as the maximum value $\mathcal{A}(S)$ over all such guesses $S$.

### 2.3.5 Computing all entries

It remains to show how to compute the solution of a configuration, say $\mathcal{C} = [C, t, t_{add}, P, \sim]$, for a square $C$ at level $\ell$, by combining the solutions $[C^i, t^i, t^i_{add}, P^i, \sim^i]$ of the four child squares $C^i$, $i = 1, 2, 3, 4$, at level $\ell + 1$. Recall that connectivity relations $\sim^i$ capture the information about connectivity in the squares $C^i$. Let $P = \{p_0, \ldots, p_s\}$ be the subset of potential portal disks. We define $\sim'$ as the *transitive closure* of all $\sim^i$: $p \sim' p'$ if and only if there exists a sequence of squares $i_1, \ldots, i_s \in \{1, 2, 3, 4\}$ and a sequence of portals $p = p_0, \ldots, p_s = p'$ such that for all $1 \leq j \leq s$, $p_j$ is a common portal of $P^{i_{j-1}}$ and $P^{i_j}$.

[4] Recall that *core* is the union of the $core(C)$ of all leaf squares $C$.

[5] The area covered by the union of a set of disks is a real number that can be computed exactly. When the desired accuracy is a fixed constant (for instance $\varepsilon$), one can give an approximation of this area with the desired precision in polynomial time.




(a) the top-left configuration is closed while other configurations are empty.

(b) there is unique connected component independent from the outside world.

(c) Each connected component contains a portal in $P$.

Figure 2.5: Illustration of cases (a)-(b)-(c) of point 6. in Definition 2.19.

.

Further, $p_{j-1}$ and $p_j$ must be connected in $\mathcal{C}^{i_j}$. We call $\mathcal{C}$ **empty** if $P = \emptyset$ and $t = 0$, and **closed** if $P = \emptyset$ and $t > 0$.

We now define the notion of compatibility of configurations.

**Definition 2.19.** Five configurations $(\mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3, \mathcal{C}^4)$ with $\mathcal{C} = [C, t, t_{add}, P, \sim]$ and $\mathcal{C}^i = [C^i, t^i, t^i_{add}, P^i, \sim^i]$ are **compatible** if all the following properties are satisfied.

1. all $C^i$ have the same level and their union is the square $C$.
2. $P = \bigcup_{i=1}^4 P^i \cap \partial C$.
3. $\sim$ is the restriction of the transitive closure $\sim'$ of $(\sim^i)_{1 \le i \le 4}$ to $P$.
4. $t = t^1 + t^2 + t^3 + t^4$ and $t \le k$.
5. $t_{add} = t^1_{add} + t^2_{add} + t^3_{add} + t^4_{add} + |\bigcup_{i=1}^4 P^i \setminus P|$ and $t_{add} \le \varepsilon k$.
6. exactly one of following three conditions holds.
   - **(a)** $\mathcal{C}^i$, $i \in \{1, 2, 3, 4\}$, is closed and all $\mathcal{C}^j$, $j \neq i$ are empty.
   - **(b)** $\mathcal{C}$ is closed and there is exactly one equivalence class for $\sim'$.
   - **(c)** all equivalence classes of $\sim'$ contain a portal in $P$.

**Remark.** By condition 2, the set $P$ of portals used by $C$ is obtained by removing from $\bigcup_{i=1}^4 P^i$ the portals not on the border of $C$. Notice that these removed portals in $\bigcup_{i=1}^4 P^i \setminus P$ are now counted as additional disks (in condition 5). Condition 6 attempts to capture all possible situations—either we have a single connected component not connected to the "outside world", which is a feasible solution by itself, (see Condition (6a) and Condition (6b)), or we have several connected components, each of which must be further connected to the outside world in a later stage (see Condition (6c)). See Figure 2.5. Finally, it is easy to see that if all $\sim^i$ satisfy the connectivity relation, then so does $\sim$.

### 2.3.6 The structural Lemma

Let $a$ be a shift chosen uniformly at random in $\left\{0, \frac{L}{2}\right\}$. We consider the grid associated to this shift and the set of additional disks on this grid as defined in the previous section. The following lemma is essential to our main theorem. Recall that $\mathcal{P}$ denotes the set of portal disks and $\mathcal{F}$ the set of fence disks.

**Lemma 2.20** (Structural Lemma)**.** *Given a fixed parameter $\varepsilon > 0$, there exists a subset $S \subseteq$ core of input disks and a set $S_{add} \subseteq \mathcal{P} \cup \mathcal{F}$ of* additional disks*, such that with probability at least* $1/3$*,*

**(i)** *(feasibility) $|S| \leq k$ and $S \cup S_{add}$ is connected,*

**(ii)** *(bounded resource augmentation) $|S_{add}| \leq \varepsilon k$,*

**(iii)** *(near-optimality) $\mathcal{A}(S) \geq (1-\varepsilon)\boldsymbol{OPT}$,*

**(iv)** *(bounded local size) For each leaf square $C$, $|C \cap S| = O(\varepsilon^{-3})$.*

Our dynamic programming aims at finding a solution satisfying all conditions of this Structural Lemma. We show that such a solution can be computed in time $n^{O(\varepsilon^{-3})}$. The *bounded local size* property ensures that we can try all possible configurations in the leaf squares in polynomial time. We also prove that for any square, the number of different planar connectivity relations is upper-bounded by the *Catalan number* of the number of potential portal disks of the square. It follows from Observation 2.17 that this number is polynomially bounded.

### 2.3.7 The algorithm

**Input:** $X, k, \varepsilon$.
**Output:** a real number $maxi \geq (1-\varepsilon)\mathbf{OPT}$.
**forall** $c \in X$ **do**
  let $\mathcal{B}'$ be the $4k \times 4k$ square centered at $c$;
  $X' \leftarrow X \cap \mathcal{B}'$;
  $L \leftarrow$ the smallest power of 2 such that $L \geq 8k$;
  **forall** *shift* $a \in \{0, \ldots, L/2\}$ **do**
    Create a table *tab*;
    **foreach** *configuration* $\mathcal{C}$ **do**
      $tab[\mathcal{C}] \leftarrow -\infty$;
    /* Initialization */
    **foreach** $\mathcal{C}$ *at level* $d$ *(leaf square)* **do**
      $tab[\mathcal{C}] \leftarrow \max\{\mathcal{A}(S) : (S, S_f, P)$ is usable and corresponds to $\mathcal{C}\}$;
    /* Fusion */
    **foreach** *level* $0 \leq i \leq d-1$ *in decreasing order* **do**
      **foreach** *configuration* $\mathcal{C}$ *at level* $i$ **do**
        $tab[\mathcal{C}] \leftarrow \max\left\{\sum_{i=1}^{4} tab[\mathcal{C}^i] : (\mathcal{C}, \mathcal{C}^1, \mathcal{C}^2, \mathcal{C}^3, \mathcal{C}^4)$ are compatible $\right\}$;
**return** $maxi = \max_{\substack{\text{configuration } \mathcal{C} \\ \text{for root square}}} tab[\mathcal{C}]$;

**Algorithm 5:** PTAS for MACS with resource augmentation.

Notice that since the root square has no potential portals (portals are only placed on lines at level at least 1), any configuration that corresponds to the root square has only one connected component. We can easily add information in the table so that the algorithm also outputs the corresponding sets $S$ and $S_{add}$.

Notice that Algorithm 5 tries all possible shift $a$. Our structural Lemma 2.20 ensures that there exists at least one shift such that the output satisfies all expected properties of Theorem 2.4.

**Theorem 2.21.** *Algorithm 5 has a running time* $n^{O(\varepsilon^{-3})}$.

The key ingredient in order to prove that our algorithm is polynomial follows from Observation 2.17. We show that the number of connectivity relations of a set of $O(m)$ portals corresponds to its *Catalan number* which is polynomial when $m = O(\varepsilon^{-1}\log(\varepsilon k))$.

*Proof.* (Theorem 2.21)

**Size of** *tab*. There exists $4^i$ squares at level $i$ so the total of squares is $\sum_{i=0}^{d} 4^i = O(4^{d+1})$. For any square $C$, the number of potential portal disks is at most $4m$. To see this, observe that if $C$ is of level $i$, it is of size $L/2^i \times L/2^i$. Furthermore, it is surrounded by lines of level at most $i$ and two adjacent portals on such a line has distance $\Omega(L/(m2^i))$. Therefore, the number of possible sets $P \subseteq \mathcal{P}_C$ is $2^{4m}$, and for each set $P$ of size $r$ the total number of planar connectivity relations is equal to the $r$-th *Catalan number*: $P(r) = \dfrac{1}{r+1}\binom{2r}{r}$ and then by Stirling formula we get $P(r) = O(4^r) = O(4^{4m})$. To see that $P(r)$ is the $r$-th Catalan number, we check that it satisfies the same recurrence relation :

$$P(r) = \sum_{k=1}^{r} P(k-1) \cdot P(r-k) \tag{2.1}$$

with $P(0) = 1$. Indeed, if $k$ denotes the index of the first portal $p_k$ that is on the connected component of the $r$-th portal disk $p_r$, then the portal disk $p_i$ with $1 \leq i \leq k-1$ cannot be equivalent to a portal $p_j$ disk with $k \leq j \leq n$, and then the equivalence relation can be restricted to the set $\{p_i, 1 \leq i \leq k-1\}$ and there are $P(k-1)$ possible distinct choices. Next observe that since $p_n$ and $p_k$ are connected (i.e., $p_n \sim p_k$), it is enough to count the number of different equivalence relations in $\{p_j, k+1 \leq j \leq r\}$, which is $P(r-k)$. Finally, observe that $k$ can be from 1 to $r$ ($k = r$ means that $p_r$ is alone in its connected component.) We thus concludes (2.1). Therefore, creating *tab* in line 5 can be done in time $O(4^{d+1}\varepsilon k^2 8^{4m}) = k^{O(1/\varepsilon)}$.

**Initialization** There exists $4^d$ leaf squares and for each of them, we try all possible guesses. This can be done in time $n^{O(\varepsilon^{-3})}$.

**Fusion** Trying all possible combinations can be done in time $k^{O(1/\varepsilon)}$

□

## 2.4 Proof of the Structural Lemma

In this section we prove the Structural Lemma.

**Lemma 2.20** (Structural Lemma)**.** *Given a fixed parameter* $\varepsilon > 0$*, there exists a subset* $S \subseteq$ *core of input disks and a set* $S_{add} \subseteq \mathcal{P} \cup \mathcal{F}$ *of* additional disks*, such that with probability at least* $1/3$*,*

**(i)** *(feasibility)* $|S| \leq k$ *and* $S \cup S_{add}$ *is connected,*

**(ii)** *(bounded resource augmentation)* $|S_{add}| \leq \varepsilon k$,

**(iii)** *(near-optimality)* $\mathcal{A}(S) \geq (1-\varepsilon)\boldsymbol{OPT}$,

**(iv)** *(bounded local size) For each leaf square* $C$, $|C \cap S| = O(\varepsilon^{-3})$.

We construct $S$ and $S_{add}$ from **OPT** in two steps. In the first step, we build sets $S'$ and $S_{add}$ that satisfy properties *(i)-(iii)*; and in the second step, we construct $S \subseteq S'$ by removing some disks from $S'$ so as to satisfy property *(iv)* while maintaining the validity of the three first properties.

### 2.4.1 Part 1 : Construction of the set of additional disks

Fix any shift, consider its associated grid and dissection and the corresponding set of additional disks $X_{add} = \mathcal{P} \cup \mathcal{F}$. Let $S'$ be the union of disks in **OPT** that are located in the core of a leaf square of the dissection, namely

$$S' = \mathbf{OPT} \cap \mathit{core}.$$

Observe that $S'$ might be disconnected since we have removed from **OPT** all the disks that were intersecting the grid. Letting *border* denote $\bigcup_{C \text{ is leaf}} \partial(C)$, we show how to replace the set of input disks $\mathbf{OPT} \cap \mathit{border}$ by a subset $S_{add} \subseteq \mathcal{F} \cup \mathcal{P}$ of additional disks.

Each leaf square $[a,b] \times [c,d]$ has an associated fence that is the boundary of the square $[a+2, b-2] \times [c+2, d-2]$. For each vertical (*resp.* horizontal) portal disk $(x,y)$, we define a *connection line*, which is $\{x\} \times [y-2, y+2]$ (*resp.* $[x-2, x+2] \times \{y\}$). The set of fences and connection lines naturally partition the set of points which are at distance at most 2 from the lines of the grid into a set of *rectangles* $\mathcal{R}$. See Figure 2.6. Notice that all connections and fences are covered by the union of additional disks. Given a rectangle $R \in \mathcal{R}$, we define $\mathrm{disk}(R) \subseteq X_{add}$ as the minimal set of additional disks that contain $R$.

We construct $S_{add}$ as the union of $\mathrm{disk}(R)$, over all rectangles $R$ that intersect a disk $x \in \mathbf{OPT} \cap \mathit{border}$.

$$S_{add} = \bigcup \{\mathrm{disk}(R) \colon R \in \mathcal{R}, \exists x \in \mathbf{OPT} \cap \mathit{border} \text{ such that } B_x \cap R \neq \emptyset\}$$

Notice that each disk $x \in \mathbf{OPT} \cap \mathit{border}$ intersects at most two rectangles. Furthermore, such a disk does not intersect with any fence and can intersect at most one connection line.

**Claim 2.22.** *Sets* $S'$ *and* $S_{add}$ *are such that* $S' \cup S_{add}$ *is connected,* $S'$ *has size at most* $k$ *and with probability at least* $1/3$ *:* $|S_{add}| \leq O(\varepsilon k)$ *and* $\mathcal{A}(S') \geq (1 - O(\varepsilon))\boldsymbol{OPT}$.

The argument is similar to the one of Arora [4]. We first upper-bound the expectation of $|S_{add}|$ and $\mathcal{A}(\mathbf{OPT}) - \mathcal{A}(S')$, and then use Markov's inequality. To bound the expectation of $|S_{add}|$, we observe that the number of additional disks added in $S_{add}$ for each disk in **OPT** intersecting a line at level $\ell$ is $O(L/(m2^{\ell}))$ while the probability that a disk intersects a line at level $\ell$ is $O(2^{\ell}/L)$.

*Proof.* (Claim 2.22)

Clearly $|S'| \leq |\mathbf{OPT}| \leq k$. We now prove that $S' \cup S_{add}$ is connected. Suppose that there exists a disk $x \in \mathbf{OPT} \cap \mathit{border}$ such that $\mathbf{OPT} \setminus \{x\}$ is split into several

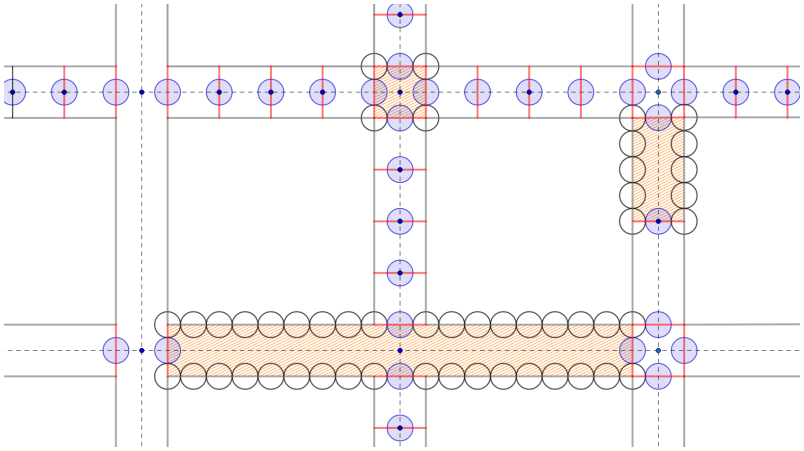


Figure 2.6: Dotted lines are the grid lines. The bottom and top horizontal lines have respectively level 8 and 10, and the vertical lines from left to right have level 5, 10 and 9. Grey continuous line are the *fence*, and the red ones, the *connection lines*. Points and blue disks are *portals* and *portal disks*. Striped orange areas illustrate some rectangles $R \in \mathcal{R}$, and other disks are *fence disks* of the corresponding sets disk$(R)$.

connected components. We know that $x$ intersects only one rectangle $R_1 \in \mathcal{R}$ or two rectangles $R_1, R_2 \in \mathcal{R}$. Since **OPT** is connected, and $B_x$ is contained in the set $U = R_1$ or $U = R_1 \cup R_2$, each connected component intersects the boundary of $U$. Then, $B_x$ intersects a disk in disk$(R_1)$ or disk$(R_2)$. Therefore, $\mathbf{OPT} \setminus \{x\} \cup \text{disk}(R_1) \cup \text{disk}(R_2)$ is connected. By doing so for each $x \in \mathbf{OPT} \cap \mathit{border}$, it follows that $S' \cup S_{add}$ is connected.

It remains to show that, under a uniform random shift $a$, with probability at least one third we have $|S_{add}| \leq O(\varepsilon k)$ **and** $\mathcal{A}(S') = \mathcal{A}(\mathbf{OPT} \cap \mathit{core}) \geq (1 - O(\varepsilon))\mathbf{OPT}$. The proof is very similar to Arora's approach, we first upper-bound the expectation of $|S_{add}|$ and $\mathcal{A}(\mathbf{OPT}) - \mathcal{A}(S')$, and then use *Markov inequality* to conclude.

We first upper-bound the expected number of additional disks. For each $x \in \mathbf{OPT}$ intersecting a line at level $\ell$, we have added at most two sets of additional disks associated to rectangles with side length smaller than the distance between two consecutive portals of this line. It follows that $O(L/(m2^\ell))$ additional disks have been added to $S_{add}$ for each disk in **OPT** intersecting a line of level $\ell$. This can be observed in Figure 2.7. Moreover, the probability that a disk intersects a line at level $\ell$ is $O(2^\ell/L)$. Then,

$$\begin{aligned}\mathbb{E}(|S_{add}|) &\leq \sum_{x \in \mathbf{OPT}} \sum_{\ell=0}^{d-1} \mathbb{P}(x \text{ intersects exactly one line at level } \ell) O(\frac{L}{m2^\ell}) \\ &= \sum_{x \in \mathbf{OPT}} \sum_{\ell=0}^{d-1} O(\frac{2^\ell}{L} \cdot \frac{L}{m2^\ell}) = O(\frac{dk}{m}) = O(\varepsilon k)\end{aligned}$$

We now upper-bound the expectation of $\mathcal{A}(\mathbf{OPT}) - \mathcal{A}(S')$. First we have $\mathcal{A}(\mathbf{OPT}) - \mathcal{A}(S') \leq \mathcal{A}(\mathbf{OPT} \cap \mathit{border})$, and the probability that a point $p \in B(\mathbf{OPT})$ is in $B(\mathbf{OPT} \cap$

*border*) is smaller that $p$ is at distance 2 from the lines of the grid. Therefore

$$\begin{aligned}\mathbb{E}\left(\mathcal{A}(\mathbf{OPT}) - \mathcal{A}(S')\right) &\leq \mathbb{E}(\mathcal{A}(\mathbf{OPT} \cap border)) \\ &\leq \int_{p \in B(\mathbf{OPT})} \mathbb{P}(p \text{ is at distance at most 4 from the grid}) \mathrm{d}p \\ &\leq \int_{p \in B(\mathbf{OPT})} 2 \cdot \frac{4}{L_0} \mathrm{d}p \\ &\leq \frac{8 \cdot \mathbf{OPT}}{L_0} = O(\varepsilon \mathbf{OPT})\end{aligned}$$

By choosing the constant properly in the big O notation and using the Markov inequality, we can show that the probability of $|S_{add}| > O(\varepsilon k)$ and the probability of $A(OPT) - A(S) > O(\varepsilon OPT)$ are both upper bounded by $\frac{1}{3}$. Thus, by a union bound, we conclude the proof. □

### 2.4.2 Part 2 : Sparsification of $S'$

The sets $S' \cup S_{add}$ obtained so far may not satisfy the last property (*bounded local size*). In this section, we show how to remove some disks from $S'$ to guarantee this property while still maintaining the other required properties in Lemma 2.20.

Suppose that there exists a leaf square $C$ such that $S'_C := S' \cap C$ has size greater than $k_0 := (1+\beta^{-1})L_0^2 = O(\varepsilon^{-3})$, where $\beta = \min\{\varepsilon/12, 1\}$. Then the core of $C$ is "overcrowded" and we show how to construct a non-overcrowded subset maintaining connectivity while losing only an $\varepsilon/2$-th fraction of the covered area.

Define a set $S$ to be initially equal to $S'$. Consider each overcrowded leaf square $C$ one by one, and define $S_C = S \cap C$. Start with an empty set $H$ and for each disk $x \in S_C$, add $x$ in $H$ if $\mathcal{A}(H \cup \{x\}) - \mathcal{A}(H) \geq \beta$. Define $\overline{H} = S_C \setminus H$ as the complement of $H$ and then apply Claim 2.23 to $G = UDG(S \cup S_{add})$ and $D = S \cup S_{add} \setminus \overline{H}$ to define $D' \subseteq \overline{H}$. Finally update $S$ to $(S \setminus \overline{H}) \cup D'$.

**Claim 2.23.** *Let $G = (V, E)$ be a connected graph and $D$ a dominating set with $\mu$ connected components. There exists a subset $D' \subseteq V \setminus D$ of size at most $2(\mu - 1)$ such that $G[D \cup D']$ is connected.*

*Proof.* Let $H$ and $H'$ be two connected components in $D$ that minimize $d_G(H, H')$. Then, $d_G(H, H') \leq 3$. Indeed, if $d_G(H, H') \geq 4$, then there exists a vertex $x$ on a shortest path

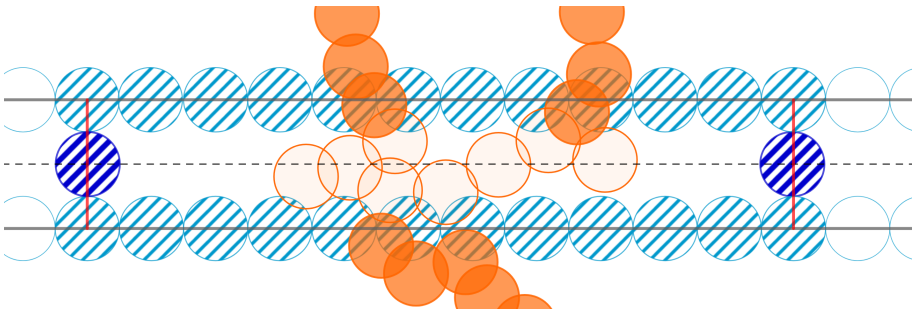


Figure 2.7: **OPT** is represented by orange disks. Disks of **OPT** that intersect the grid (dotted line) are replaced by additional disks (striped blue disks). This operation maintains the connectivity of the set.

from $H$ to $H'$ that is not dominated by $D$. This implies that we can find two vertices that connect $H$ and $H'$. We repeat this operation until there is only one connected component. This requires at most $2(\mu - 1)$ vertices. □

The following claim, together with Claim 2.22 ensures that sets $S$ and $S_{add}$ built in Part 1 and Part 2 satisfy the expected properties of our structural Lemma.

**Claim 2.24.** *The constructed sets $S$ and $S_{add}$ satisfy*

*(i)* *$S \cup S_{add}$ is connected,*

*(ii)* *for each leaf square $C$, $|S \cap C| \leq k_0$, and*

*(iii)* *$\mathcal{A}(S) \geq (1 - \varepsilon/2)\mathcal{A}(S')$.*

This Claim might not be true if the radius of disks considered are arbitrary. The proof of this fact follows from geometrical observations about unit disks.

*Proof.* (claim 2.24)

For (i), we just need to argue that for each leaf square $C$, after $\overline{H}$ is defined, $S \cup S_{add} \setminus \overline{H}$ is a dominating set in $UDG(S \cup S_{add})$ (then the proof follows from Claim 2.23). Indeed if a disk $x$ is in $\overline{H}$ then it means that $\mathcal{A}(H \cup \{x\}) - \mathcal{A}(H) < \beta \leq 1$. In particular, it implies that there exists a disk in $H \subseteq S \cup S_{add} \setminus \overline{H}$ that intersects $x$.

For (ii), observe that the size of $S \cap C$ is the sum of the size of the corresponding sets $H$ and $D'$ built during the "sparsification" of $C$. Since all disks in $H$ increases the area covered by at least $\beta$ and are contained in a square of area $L_0^2$, the number of disks in $H$ is upper-bounded by $\beta^{-1}L_0^2$. Moreover, each connected component of $S \cup S_{add} \setminus \overline{H}$ had a disk contained in $C$ so that the number $\mu$ of connected component is upper-bounded by $L_0^2/\pi < L_0^2/2$. Therefore $|D'| < L_0^2$. Finally $|H \cup D'| < (1 + \beta^{-1})L_0^2 = k_0$.

For (iii), we start by observing that the union $B(S')$ of disks in $S'$ is contained in the set $B^+(S)$, which is defined as

$$B^+(S) := \{z \in \mathbb{R}^2 \mid \exists x \in S \text{ such that } ||z - x|| \leq 1 + \beta\}$$

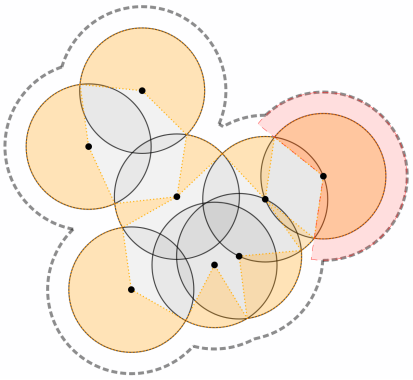


Figure 2.8: $S$ consists of grey disks. The boundary of $B^+(S)$ is the dotted curve. Circular sectors are in orange while the red one represents a circular sector in $B^+(S)$.

Indeed, if there exists a point $p$ covered by a disk $x$ in $S'$ but at distance at least $1+\beta$ from any disk of $S$ then adding $x$ to $S$ would increase the area covered by $S$ by more that $\beta$. Therefore, we have the following inclusion

$$B(S) \subseteq B(S') \subseteq B^+(S), \tag{2.2}$$

and if the following geometrical claim holds, our proof of (iii) will be complete.

**Claim 2.25.** $\mathcal{A}(B(S)) \geq (1-\varepsilon/2)\mathcal{A}(B^+(S))$

The result follows from the fact that $B(S)$ is a union of unit-disks. See Figure 2.8. The boundary of $B(S)$ is made of *circular arcs* and each of these arcs is associated with a *circular sector* $\theta_i$. Circular sectors intersect with other circular sectors only on the extreme points of their corresponding arcs, thus $\mathcal{A}(\cup_i \theta_i) = \sum_i \mathcal{A}(\theta_i)$.

We can associate with each circular sector $\theta_i$ (of a disk of radius 1) its "dilation" $\theta_i^+$ which corresponds to the same circular sector in a disk of radius $1+\beta$. We have $\mathcal{A}(\theta_i^+) = (1+\beta)^2\mathcal{A}(\theta_i)$ and can see that $B^+(S) \setminus B(S) \subseteq \bigcup_i(\theta_i^+ \setminus \theta_i)$. Then

$$\begin{aligned}
\mathcal{A}(B^+(S)) - \mathcal{A}(B(S)) = \mathcal{A}(B^+(S) \setminus B(S)) &= \mathcal{A}\left(\bigcup_i(\theta_i^+ \setminus \theta_i)\right) \\
&\leq \sum_i \mathcal{A}(\theta_i^+ \setminus \theta_i) = \sum_i \mathcal{A}(\theta_i^+) - \mathcal{A}(\theta_i) \\
&\leq \sum_i (1+\beta)^2\mathcal{A}(\theta_i) - \mathcal{A}(\theta_i) \\
&\leq \sum_i 3\beta\mathcal{A}(\theta_i) = 3\beta\mathcal{A}\left(\bigcup_i \theta_i\right) \leq 3\beta\mathcal{A}(B(S))
\end{aligned}$$

Therefore, $\mathcal{A}(B(S)) \geq \dfrac{\mathcal{A}(B^+(S))}{1+3\beta} \geq (1-\varepsilon/2)\mathcal{A}(B^+(S))$. This concludes the proofs of Claims 2.25 and 2.24. □

## 2.5 PTAS for well-distributed inputs

Let us recall the definition of well-distributed input.

**Definition 2.5.** Given $\alpha > 0$, a finite set $X$ of points in the plane is called **$\alpha$-well-distributed** if for all $x, y \in X$, $d_{UDG(X)}(x,y) \leq \lceil \alpha \cdot \|x-y\| \rceil$.

Here $\lceil \cdot \rceil$ is the ceiling function. This ensures that the right-hand side is always at least one. Notice that a well-distributed set is necessarily connected.

One intuitive view of a well-distributed input is to look at the shape of the "holes" of the input, that are the different connected components of the complement of the union of the input disks in the plane. The assumption of well-distribution means that these holes are roughly *fat*.

One particular interesting case arises when there is no hole at all. We call these sets *pseudo-convex*, and we prove that this is a particular case of well-distributed inputs.

**Definition 2.7.** A set $X$ is called **pseudo-convex** if the convex-hull of $X$ is covered by the union of the unit disks centered at points of $X$.

**Lemma 2.8.** *A pseudo-convex set $X$ is 3.82-well-distributed.*

*Proof.* (Lemma 2.8) Let $X$ be a pseudo-convex set, $G$ its unit-disk graph, and $x$ and $y$ be any two disks in $X$ at distance $L = \|x - y\|$. We show that $d_G(x, y) \leq \lceil \alpha L \rceil$ where $\alpha = 12/\pi < 3.82$.

If $L < 2$ then the two unit disks associated to $x$ and $y$ overlap so that $d_G(x, y) = 1 \leq \lceil \alpha L \rceil$. Otherwise suppose that $L \geq 2$. Since $X$ is pseudo-convex, it is connected and any point in the line segment $[x, y]$ is covered by a disk in $X$. Let $S = \{z \in X \mid B_z \cap [x, y] \neq \emptyset, \|x - z\| > 2 \text{ and } \|y - z\| > 2\}$ and let $I$ be any maximal independent set in $S \cup \{x, y\}$. Since $S$ is at distance at least 2 from $x$ and $y$, we deduce that $x, y \in I$ and all disks in $I \setminus \{x, y\}$ are inside a $L \times 4$ rectangle and then $|I| \leq 4L/\pi$. Since $I$ is maximal, it is a dominating set in $S$. Therefore, claim 2.23 implies that there exists a connected subset $D \subseteq X$ such that $I \subseteq D$ and $|D| \leq 3|I| - 2 \leq 12L/\pi - 2$. We deduce that $d_G(x, y) \leq (12L/\pi - 2) + 1 \leq \lceil \alpha L \rceil$. □

### 2.5.1 The algorithm

Our Corollary 2.6 states that the restriction of MACS to well-distributed inputs admits a PTAS. The algorithm works as follows. Given a parameter $0 < \varepsilon \leq 1/2$, and an input $(X, k)$ of MACS, we run Algorithm 5 on input $(X, k', \varepsilon')$ for suitable values $k'$ and $\varepsilon'$ specified below. Next, we transform the set of additional disks obtained into a set of input disks that has roughly the same size while maintaining the connectivity of the solution. See Lemma 2.26 and Algorithm 6 for details. This algorithm naturally applies to pseudo-convex inputs (Corollary 2.9).

**Input:** $X$ an $\alpha$-well-distributed input, $k \geq 0$, $\varepsilon > 0$.
**Output:** A feasible solution to MACS$(X, k)$.
Choose $\varepsilon' > 0$ and $k' \leq k$ such that $(1 - \varepsilon')(1 - 10(22\alpha + 4)\varepsilon') \geq (1 - \varepsilon)$ and $k'(1 + (22\alpha + 4)\varepsilon') = k$;
Let $S, S_{add}$ be the solution of Algorithm 5 on input $(X, k', \varepsilon')$ ;
Let $S'$ be the set obtained from $S_{add}$ by Lemma 2.26;
**return** $S \cup S'$;

**Algorithm 6:** PTAS for MACS for well-distributed inputs.

**Lemma 2.26.** *Given an $\alpha$-well-distributed input $X$ and two finite sets $S \subseteq X$ and $S_{add} \subseteq \mathbb{R}^2$ such that $UDG(S \cup S_{add})$ is connected, there exists a set $S' \subseteq X$ of size at most $(22\alpha + 4)|S_{add}|$ such that $UDG(S \cup S')$ is connected. Moreover, such a set can be computed in polynomial time.*

Since $\varepsilon' = \Theta(\varepsilon/\alpha)$, the previous algorithm runs in polynomial time when $\varepsilon$ and $\alpha$ are fixed constants.

We now explain how to compute the set $S'$ in Lemma 2.26. In this Lemma, the set $S_{add}$ is not supposed to be a set of additional disks as defined in Section 2.3.

*Proof.* (Lemma 2.26) Let us use the same notation as in the statement of Lemma 2.26. We prove how to build $S'$ from $S_{add}$ such that $|S'| \leq (22\alpha + 4)|S_{add}|$ while preserving connectivity.

Let $Y$ be a connected component of $S_{add}$. We prove that we can find a set $Y' \subseteq X$ of input disks such that $|Y'| \leq (4+22\alpha)|Y|$ and $(S_{add} \setminus Y) \cup (S \cup Y')$ is connected. Removing $Y$ might split the solution into several connected components $F_1, \ldots, F_s$. For each connected component $F_i$, pick one disk $x_i$ in $F_i \cap X$ that intersects $Y$.

**Step 1.** Each additional disk $y$ in $Y$ is adjacent to at most 6 disks $x_i$. We can connect the corresponding connected component by using $20\alpha$ disks of the input. Indeed, any two $x_i$ and $x_j$ adjacent to $y$ has a Euclidean distance at most 4. Since $X$ is well-distributed their distance in $UDG(X)$ is at most $4\alpha$. Then, we can find $\lceil 4\alpha - 1 \rceil$ disks in $X$ which connect $x_i$ and $x_j$. In order to connect all the $x_i$ that are adjacent to $y$, it is sufficient to repeat this operation 5 times, which asks at most $20\alpha$ disks. We can perform this operation for each additional disk that was not already considered. Then, in total for this first step we need to use at most $20\alpha|Y|$ disks.

**Step 2.** During step 1, we may have connected some disks $x_i$, so that the number of connected components has decreased. The number of connected components is $s' \leq s$, each of them corresponds to a disk $x_i$, and without loss of generality we can assume that the corresponding indexes are such that $1 \leq i \leq s'$. Let $T$ be a spanning tree on $UDG(Y)$. Without loss of generality, we can suppose that indexes $i$ are such that the sequence $(x_1, \ldots, x_{s'})$ correspond to a $T$ transversal. Note that after step 1, each $x_i$ can be associated to a different $y$ in $Y$. Then, we reconnect each $x_i$ to $x_{i+1}$ for $1 \leq i \leq s-1$. If $x_i$ and $x_{i+1}$ are respectively associated to $y_i$ and $y_{i+1}$, then $||x_i - x_{i+1}|| \leq 2 + 2d_T(y_i, y_{i+1})$ and thus $d_{UDG(X)}(x_i, x_{i+1}) \leq \lceil \alpha(2 + 2d_T(y_i, y_{i+1})) \rceil$. Then, we can find $\lceil \alpha(2 + 2d_T(y_i, y_{i+1})) \rceil - 1$ disks in $X$ to connect $x_i$ and $x_{i+1}$. In order to connect all $x_i$ we need to use at most

$$\sum_{i=1}^{s'-1} \lceil \alpha(2 + 2d_T(y_i, y_{i+1})) \rceil - 1 \leq 2(s'-1)\alpha + 2 \sum_{i=1}^{s'-1} d_T(y_i, y_{i+1})$$

input disks. Since the order corresponds to a $T$ transversal, each edge is visited at most twice and then $\sum_{i=1}^{s'-1} d_T(y_i, y_{i+1}) \leq 2(|Y| - 1)$. Therefore the total number of disks that were added during this second step is bounded by $|Y|(4 + 2\alpha)$.

We proved that there exists a subset $Y' \subseteq X$ of size at most $(4 + 22\alpha)|Y|$ such that $(S \cup S_{add} \setminus Y) \cup Y'$ is connected. By doing so for each connected component of $S_{add}$, we get the result claimed. □

### 2.5.2 Analysis

In this subsection we prove that Algorithm 6 is a PTAS for well-distributed inputs. First, we need to state the following "stability" property over optimal solutions.

**Lemma 2.27.** *Let $\eta < \frac{1}{2}$. Then $\boldsymbol{OPT}(X, k) \geq (1 - 10\eta) \cdot \boldsymbol{OPT}(X, k(1+\eta))$.*

*Proof.* (Lemma 2.27) Let $X$ be a set of points of the plane, $k$ a positive integer and $\eta \leq 1/2$ a parameter. We prove a stronger result. Given any solution feasible solution $S$ to $\mathrm{MACS}(X, k(1+\eta))$, there exists a subset $S'$ of $S$ that is a feasible solution to $\mathrm{MACS}(X, k)$ with value at least $(1 - 10\eta)\mathcal{A}(S)$. Obviously Lemma 2.27 follows when $S$ is optimal. If $\mathcal{A}(S) \geq k/3$, then remove $\eta k$ disks from $S$ without disconnecting $S$. For instance, consider

a spanning tree on $UDG(S)$ and remove the nodes from the leaves to the root until you reach the desired size. Let $S'$ denote the subset obtained.

$$\mathcal{A}(S') \geq \mathcal{A}(S) - \eta\pi k \geq (1 - 3\pi\eta)\mathcal{A}(S) \geq (1 - 10\eta)\mathcal{A}(S)$$

If $\mathcal{A}(S) < k/3$, let $I$ be a maximal independent set in $S$. We have $|I|\pi = \mathcal{A}(I) \leq \mathcal{A}(S) < \frac{k}{3}$. According to claim 2.23, there exists a connected dominating set $I \subseteq D \subseteq S$ in $S$ of size at most $3|I| - 2 < k/\pi < \frac{k}{3}$. Consider a set $H \subseteq S \setminus D$ of size $k - |D| > \frac{2}{3}k$ built by greedily adding a disk $h \in S \setminus (D \cup H)$ maximizing the marginal area $\mathcal{A}(D \cup H \cup \{h\}) - \mathcal{A}(D \cup H)$. Since $D$ is a connected dominating set, the set $S' := D \cup H$ is connected. Since all disk where added greedily in $H$, for all $H \in S \setminus S'$, we have

$$\mathcal{A}(S' \cup \{h\}) - \mathcal{A}(S')) \leq \frac{\mathcal{A}(S) - \mathcal{A}(D)}{|H|} \leq \frac{2\mathcal{A}(S)}{k}.$$

By submodularity, we deduce that $\mathcal{A}(S) - \mathcal{A}(S') \leq \eta k \cdot \frac{2\mathcal{A}(S)}{3k}$. That implies $\mathcal{A}(S') \geq (1 - \frac{3}{2}\eta)\mathcal{A}(S)$. This concludes the proof of lemma 2.27. □

Remark that this proof is constructive and it is easy to check that finding $S'$ from any given set $S$ can be done in polynomial time. We can now prove that Corollary 2.6.

*Proof.* (Claim 2.6) The solution output by Algorithm 5 on input $(X, k', \varepsilon')$ verifies the following properties: $S \cup S_{add}$ is connected, the size of $S$ and $S_{add}$ are respectively upper-bounded by $k'$ and $\varepsilon' k'$ and $\mathcal{A}(S) \geq (1 - \varepsilon')\mathbf{OPT}(X, k')$. Therefore, the set $S'$ given by Lemma 2.26 has size at most $(22\alpha + 4)|S_{add}| \leq (22\alpha + 4)\varepsilon' k'$, and then $|S \cup S'| \leq k' + (22\alpha + 4)\varepsilon' k' \leq (1 + (22\alpha + 4)\varepsilon')k' = k$. Since $S \cup S'$ is connected, this set is a feasible solution to MACS$(X, k)$.

Finally, from Lemma 2.27 with parameter $\eta = (22\alpha + 4)\varepsilon'$, we get that the area covered by this solution is

$$\begin{aligned}\mathcal{A}(S \cup S') &\geq \mathcal{A}(S) \geq (1 - \varepsilon')\mathbf{OPT}(X, k') \geq (1 - \varepsilon')(1 - 10\eta)\mathbf{OPT}(X, k'(1 + \eta)) \\ &\geq (1 - \varepsilon')(1 - 10(22\alpha + 4)\varepsilon')\mathbf{OPT}(X, k'(1 + (22\alpha + 4)\varepsilon')) \\ &\geq (1 - \varepsilon)\mathbf{OPT}(X, k)\end{aligned}$$

which concludes the proof. □

## 2.6 Hardness results

In this section we show our hardness results. First we prove that MACS is **NP**-hard and then we show that the problem even becomes **APX**-hard when we take as input more complicated geometrical objects.

### 2.6.1 NP-Hardness of MACS

We present a reduction from the Rectilinear Steiner Tree (RST) problem, which is **NP**-hard, to prove that MACS is **NP**-hard.

RECTILINEAR STEINER TREE PROBLEM: Given $n$ terminals on a Euclidean plane and a number $L$, decide whether there exists a tree to connect all the $n$ terminals using horizontal and vertical lines of total length at most $L$.

The problem is **NP**-complete [58], even if all terminals have integral coordinates bounded by $V = poly(n)$. In the following, we assume that $n$ is sufficiently large, say $n \geq 8$.

We next explain the reduction. We start from an instance of RST and define an instance of MACS as follows.

- For all $0 \leq i, j \leq V$, place one disk with center with center at $(ni, nj)$. We call them **cardinal disks**.
- Then place $n-4$ disks centered at $\left(in + 2 + (1 + \frac{1}{n-5}) \cdot t, jn\right)$ where $t \in \{0, \ldots, n-5\}$ and $n-4$ disks centered at $\left(in, jn + 2 + t(1 + \frac{1}{n-5})\right)$ where $t \in \{0, \ldots, n-5\}$. We call them **path disks**.
- For each terminal $(i, j)$ in the RST instance, we place $n^2/10$ **bonus disks** : the first one centered at $\left(in + \sqrt{2}, jn + \sqrt{2}\right)$ and the remaining centers forming a connected group in $[in + 2, (i+1)n - 2] \times [jn + 2, (j+1)n - 2]$ in such a way that each bonus disk is tangent[6] to other bonus disks, and can be connected to the first bonus disk. Notice that except the first one, no bonus disk intersects path disks. This defines the set of disks.
- Set $k = 1 + L(n-3) + n^3/10$.

This defines the MACS instance. See Figure 2.9 for an illustration. Note that the interior of a cardinal disk is disjoint from all other disks of the instance.

Notice that as the RST instance has all terminals bounded by a rectangular of polynomial size, the above reduction can be done in polynomial time.

Let $Z$ denote the set of the cardinal disk at $(0, 0)$ and the $n-4$ path disks at $\left(2 + t(1 + \frac{1}{n-5}), 0\right)$ where $t \in \{0, \ldots, n-5\}$ and let $\mathcal{A}(Z)$ denote the area covered by $Z$.

**Lemma 2.28.** *The original* RST *instance has a feasible solution of total length at most $L$ if and only if the derived* MACS *instance has a feasible solution of area of at least $\pi + L \cdot \mathcal{A}(Z) + (\frac{n^3}{10} - \frac{n}{3})\pi$.*

*Proof.* ($\Rightarrow$) This direction is easy to see: We call a set of disks a *segment* if it consists of a cardinal disk and all the $n-4$ path disks between it and one of its four adjacent cardinal disks. Thus the area covered by a segment is exactly $\mathcal{A}(Z)$. Consider a feasible solution for the RST instance, of length exactly $L$ without loss of generality. We root it at an arbitrary integral point, direct it outwards from the root, and view it as a collection of horizontal or vertical directed edges of unit length. In the MACS instance, we take all bonus disks, the cardinal disk associated to the root of the RST solution, and, for each directed edge of the RST solution, all disks of the corresponding segment. The total number of disks is exactly $k$, and the area covered is at least $\pi + L \cdot \mathcal{A}(Z) + \frac{n^3\pi}{10} - 2n\gamma$, where $\gamma$ is the size of the overlapped area of the first bonus disk associated with a terminal and the path disk just

[6] their centers are at distance exactly 2.

tangent to the corresponding cardinal disk of the latter. (Recall the first bonus disk can overlap up to two path disks). The distance between two such centers is $h = \sqrt{8 - 4\sqrt{2}}$. Furthermore, the overlapped area can be expressed as

$$2 \arccos \frac{h}{2} - 2 \left(\frac{h}{2}\right) \sqrt{1 - \left(\frac{h}{2}\right)^2}$$

which is upper-bounded by 0.45. Therefore $2n\gamma \leq 0.9n \leq n\pi/3$. This gives the proof of one direction.

For ($\Leftarrow$) direction, assume that a solution $S$ for the MACS instance is given with area at least $\pi + L \cdot \mathcal{A}(Z) + (\frac{n^3}{10} - \frac{n}{3})\pi$. By our construction, we can modify $S$, while conserving its connectivity and without diminishing covered area, so that the following properties hold.

(i) If any bonus disk corresponding to a terminal is part of $S$, so is the cardinal disk corresponding to this terminal.

(ii) The path and cardinal disks in $S$ form a tree; furthermore, such a tree consists of a cardinal disk, a set of segments, and at most one sub-segment. (A *sub-segment* is a subset of a segment, so that it induces a connected component.)

Indeed, if (i) does not hold, then we add in $S$ the missing terminal disk and remove any bonus disk that does not disconnect the solution. To guarantee (ii), remark that a sub-segment does not contribute to connect terminals.Suppose the path and cardinal disks has a cycle. We can replace one segment of this cycle by any set of disks without disconnecting the solution. Then, if there are at least two sub-segments, we can remove some disks in the shortest sub-segment et replace them by disks in another sub-segment until it is complete or the shortest sub-segment is empty. After this step, the number of sub-segments decreased by at least one. We repeat this operation until the solution has only one sub-segment.

We claim that the number $B$ of bonus disks in $S$ is at least $\frac{n^3}{10} - \frac{9n}{10}$. Suppose not. Observe that the covered area of $S$ can be upper-bounded as

$$\mathcal{A}(S) \leq B\pi + \pi + \frac{L|Z| + \frac{n^3}{10} - B}{|Z|} \mathcal{A}(Z). \tag{2.3}$$

(Here we ignore the possible intersection of a bonus disk with the path disks. The first term is the area covered by bonus disks; the second term is the area covered by a cardinal disk; the third term is the maximum area that can be covered by segments, and possibly the single sub-segment in $S$). Now since $S$ is supposed to be a feasible solution in the MACS instance, its covered area should be at least

$$\mathcal{A}(S) \geq \pi + L \cdot \mathcal{A}(Z) + \left(\frac{n^3}{10} - \frac{n}{3}\right) \pi. \tag{2.4}$$

The difference between the lower bound (2.4) and the upper bound (2.3) is

$$\left(\frac{n^3}{10} - B\right) \left(\pi - \frac{A(Z)}{|Z|}\right) - \frac{n\pi}{3} \geq \frac{9n}{10} \left(\pi - \frac{A(Z)}{|Z|}\right) - \frac{n\pi}{3}.$$

Here in order to reach a contradiction (making the last term greater than 0), we need to calculate $\mathcal{A}(Z)$, which is $(n-3)\pi-(n-5)\gamma'$, where $\gamma'$ is size of the overlapped area of two disks whose centers have distance $1+\frac{1}{n-5}$. $\gamma'$ is easily shown to be at most 1.25. Therefore,

$$0.9n(\pi-\frac{A(Z)}{|Z|})-\frac{n\pi}{3}\geq 0.9n(\pi-\frac{(n-3)\pi-(n-5)1.25}{n-3})-\frac{n\pi}{3}$$
$$=(0.9*1.25-\frac{\pi}{3})n-\frac{2*1.25}{n-3}\geq 0.07n-\frac{2.5}{n-3}\geq 0,$$

which can be verified when $n\geq 8$ by the simple study of a quadratic polynomial.

So we know that $S$ has at least $\frac{n^3}{10}-\delta$ bonus disks, where $\delta\leq n/10$. Ignoring the possible sub-segment of $S$, $S$ includes a cardinal disk, $L'$ segments and $\frac{n^3}{10}-\delta$ bonus disks. As a result,

$$k=1+L|Z|+\frac{n^3}{10}\geq 1+L'|Z|+\frac{n^3}{10}-\delta\geq 1+L'|Z|+\frac{n^3}{10}-\frac{n}{10},$$

implying that $n/10\geq(L'-L)|Z|=(L'-L)(n-3)$. Thus $L'=L$ and the cardinal disks and path disks of $S$ correspond to a tree of length L in the RST instance. The proof follows. □

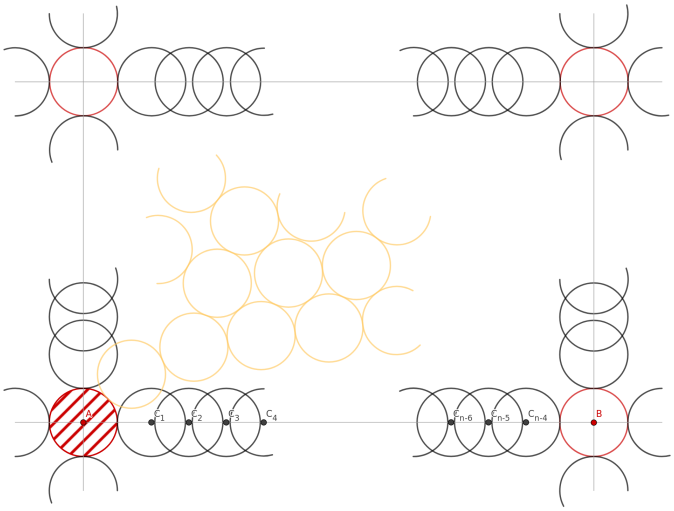



Figure 2.9: Black, red and orange disks respectively represent *path*, *cardinal* and *bonus* disks. The hatched disk is associated to a terminal node.

### 2.6.2 APX-hardness of Quad-Connected-Cover

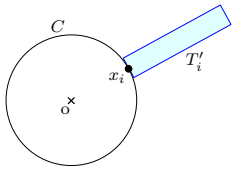
**Theorem 2.12.** QUAD-CONNECTED-COVER *is* ***APX****-hard.*

The reduction will be from the following problem.

> 3-SET-COVER: Given a set $X$ of $n$ elements, and its subsets $\mathcal{S}=\{S_1,\ldots,S_m\}$ such that $|S_i|\leq 3$ for $i=1,\ldots,m$, compute a minimum size subset of $\mathcal{S}$ that covers $X$.

This problem is **APX**-hard (due to the fact that minimum vertex cover on graphs with maximum degree 3 is **APX**-hard).

*Proof.* (Theorem 2.12) The proof is by a reduction from 3-SET-COVER to QUAD-CONNECTED-COVER. In particular, given a set $X = \{x_1, \ldots, x_n\}$ and subsets $\S = \{S_1, \ldots, S_m\}$, we show how to construct, in polynomial time and for any parameter $\varepsilon < \frac{1}{6}$, a $(1+\varepsilon)$-approximation to 3-SET-COVER from a $\left(1 - \frac{\varepsilon}{6}\right)^{-1}$-approximation to the QUAD-CONNECTED-COVER.




Map the $n$ points of $X$ to $n$ points placed uniformly on a circle $C$ of unit area centered at the origin $o$; we will use the notation $x_i$ for these points as well. Our set $\mathcal{T}$ will consist of convex quadrilaterals of two types:

- **center-quads.** These are, for each set $S_j \in \S$, the quadrilateral $T_j = \text{convexhull}\,(S_j \cup \{o\})$.
- **side-quads.** For each element $x_i$, let $T_i'$ be the rectangle with width $\frac{1}{2n}$, length $4n$, containing $x_i$ and tangent to $C$

Note that every pair of center-quads intersect (namely, at $o$), no two side-quads intersect, and $T_j$ intersects $T_i'$ if and only if $x_i \in S_j$. The area of the union of the center-quads is at most 1, and the area of each side-quad is 2.

Let $s$ be the size of an optimal set-cover for $X$ and $\S$. Let $\mathcal{T}'$ be a $(1 - \frac{\varepsilon}{6})$-approximate solution to the quad-connected-cover problem on the set $\{T_1, \ldots, T_m, T_1', \ldots, T_n'\}$ with $k = n + s$. Observe that to maintain connectivity of the intersection graph of $\mathcal{T}'$, if a point $x_i$ is covered by a side-quad of $\mathcal{T}'$, it must also be covered by some center-quad of $\mathcal{T}'$, as a side-quad only intersects center-quads.

One possible solution consists of picking the $s$ center-quads of the set-cover, and all the $n$ side-quads to get the total area of at least $2n$; in particular, an optimal solution has value at least $2n$. Thus the area of the union of the quadrilaterals in $\mathcal{T}'$ is at least $\left(1 - \frac{\varepsilon}{6}\right) \cdot 2n$. This implies that $\mathcal{T}'$ leaves at most $\frac{\varepsilon n}{6}$ elements of $X$ uncovered by center-quads; otherwise at least $\frac{\varepsilon n}{6} + 1$ side-quads are not picked, and so the area covered by $\mathcal{T}'$ can only be $1 + 2\left(n - \frac{\varepsilon n}{6} - 1\right) \leq \left(1 - \frac{\varepsilon}{6}\right) \cdot 2n - 1$. Thus, out of the $n + s$ quadrilaterals in $\mathcal{T}'$, at least $n - \frac{\varepsilon n}{6}$ side-quads are present, and at most $(n + s) - \left(n - \frac{\varepsilon n}{6}\right) = s + \frac{\varepsilon n}{6}$ center-quads are present. Thus one can pick arbitrarily one set for each uncovered point to construct a set cover for $X$ of size at most $\left(s + \frac{\varepsilon n}{6}\right) + \frac{\varepsilon n}{6} \leq s + \frac{2\varepsilon}{6} \cdot 3s \leq (1 + \varepsilon) \cdot s$, where the first inequality follows from the fact that $s \geq \frac{n}{3}$. This completes the proof.

□

We conjecture that by finding a more specific reduction from **APX**-hard geometric covering problems in [76] for instance, the problem QUAD-CONNECTED-COVER remains **APX**-hard even when the quadrilaterals are triangles with area arbitrarily close to one.

# 3 | Greedy approaches for Maximum independent Set

## 3.1 Introduction

We are interested in the Maximum Independent Set (MIS) problem on graphs with maximum degree bounded by $\Delta$. This problem is known to be hard to approximate. The best known asymptotic polynomial time approximation ratio for MIS is $O(\Delta \log\log(\Delta)/\log(\Delta))$ based on a semidefinite programming relaxation [75]. However, the best known asymptotic approximation ratio for MIS is $O(\Delta/\log^2(\Delta))$ with $n^{O(1)} \cdot 2^{O(\Delta)}$ running time [10]. In this chapter we are primarily interested in MIS on graphs with small to moderate values of $\Delta$. The best known polynomial time approximation ratio for this problem for small values of $\Delta \geq 3$ is arbitrarily close to $\frac{\Delta+3}{5}$, see [14, 15, 31]. This is achieved by a local search approach at the expense of huge running time. See Section 3.1.3 for additional details and references about the Maximum Independent set problem.

We focus on greedy approaches to compute large independent sets efficiently in bounded-degree graphs. All the greedy algorithms of this chapter work as follows: given a graph $G$, we *pick* a vertex $v$, add this vertex to the solution, remove $v$ and its neighbors (because they cannot be in any independent set that contains $v$), and repeat until the graph is empty.

$S \leftarrow \emptyset$;
**while** $G \neq \emptyset$ **do**
  Pick a vertex $v \in V(G)$ according to some **rules**;
  $S \leftarrow S \cup \{v\}$;
  $G \leftarrow G \setminus N_G[v]$;
**return** $S$

**Algorithm 7:** the generic greedy algorithm for MIS.

Any maximal (for inclusion) independent set can be produced by this generic greedy approach. We want to compute large independent sets and therefore, our objective is to decide which vertex we should pick at any step in order to maximize the size of the final solution. We will refer to the algorithm used to compute this vertex as the **rules** of the greedy algorithm. We will focus on rules that are implementable in time that is polynomial is in size of the graph.

The simplest rule that comes to mind consists in finding a vertex $v$ in the graph with minimum degree. We call this rule **min-degree**, and by extension we call the greedy procedure implementing this rule as the **min-degree** greedy algorithm.

The **min-degree** rule: Pick a vertex of minimum-degree.

This rule is natural: the size of the complement of the independent set output by the **min-degree** greedy algorithm is equal to the sum of the degrees[1] of the vertices picked. Therefore, we hope that greedily minimizing the number of neighbors removed will produce a large independent set. With this **min-degree** rule, this basic greedy algorithm is profoundly simple and time-efficient and can be implemented to run in linear time. For the class of graphs with maximum degree $\Delta$, the first published approximation guarantee $\Delta + 1$ of this greedy algorithm for MIS we are aware of can be inferred from the proof of the following conjecture of Erdős, due to Hajnal and Szemerédi [68, 13]: every graph with $n$ vertices and maximum degree $\Delta$ can be partitioned into $\Delta + 1$ disjoint independent sets of almost equal sizes. The approximation ratio of greedy has been improved to $\Delta - 1$ by Simon [139]. The best known analysis of greedy by Halldórsson and Radhakrishnan [73, 70] for MIS implies the approximation ratio of $(\Delta + 2)/3$.

Halldórsson and Radhakrishnan proved that this algorithm (Algorithm 5.4) achieves a $\frac{\Delta+2}{3}$-approximation on graphs with maximum degree $\Delta$.

**Theorem 3.1** (Halldórsson and Radhakrishnan [73])**.** *Given a graph $G$ with maximum degree $\Delta$, let $I$ be an independent set output by the **min-degree** greedy algorithm on input $G$. Then, $|I| \geq \frac{3}{\Delta+2} \cdot \alpha(G)$. Moreover, for all $\Delta \geq 3$, this bound is tight.*

If there are several vertices with minimum degree, then the **min-degree** greedy algorithm may pick any of them. Can we improve the approximation ratio of this greedy algorithm by specifying which vertex to pick in case of ties ? Concretely, if two vertices have minimum degree, which vertex should the algorithm pick in order to increase the size of the independent set produced ? Halldórsson and Yoshihara proposed the following tie-breaking rule [74].

The **more-edges** rule: Among all vertices of minimum degree, pick a vertex $v$ such that the degree of its closed neighborhood[a] $d(N_G[v])$ is the largest.

[a] the number of edges that are incident to one of $v$'s neighbors and one vertex at distance two from $v$.

Again this rule is a natural refinement of the first rule: when we pick a vertex $v$ of minimum degree, thinking ahead, we also want to increase the chance to find small degree vertices in the graph $G \setminus N[v]$, hence we want to discard as many edges as possible. Halldórsson and Yoshihara analyzed this algorithm and proved that the **more-edges** greedy algorithm achieves a strictly better approximation on graphs of degree at most three.

**Theorem 3.2** (Halldórsson and Yoshihara [74])**.** *Given a graph $G$ with maximum degree 3, let $I$ be an independent set outputs by the **more-edges** greedy algorithm on input $G$. Then, $|I| \geq \frac{2}{3} \cdot \alpha(G)$*

How far can we hope to go with this approach ? First, Maximum Independent Set is **APX**-hard even for cubic graphs [2]. Since we focus in this chapter on the **min-degree** greedy algorithm, we have a stronger lower bound found by Halldórsson and Yoshihara [74]. Figure 3.1 shows that the approximation ratio of any refinement of the **min-degree** greedy algorithm must be at least 5/4.

[1] at the time they were added to the solution.

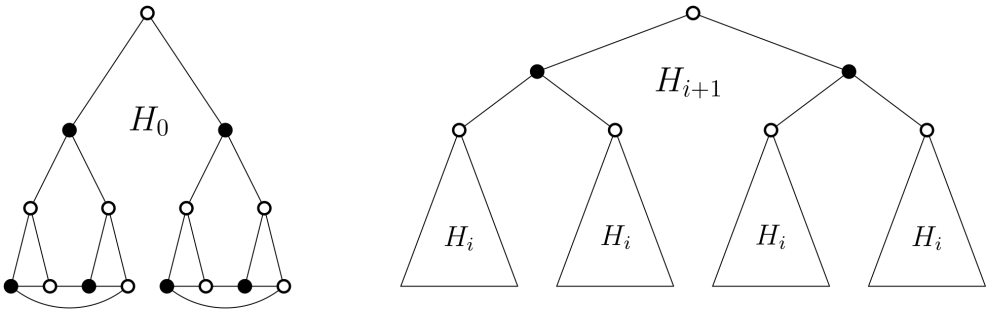



Figure 3.1: For any $i \geq 0$, the graph $H_i$ is subcubic and has only a unique vertex of minimum degree. Picking this vertex creates four copies of $H_{i-1}$. The **min-greedy** greedy algorithm outputs on $H_i$ a solution of size $4^i(5 + 1/3) - 1/3$ while the maximum independent set of $H_i$ has size $\alpha(H_i) = 4^i(6 + 2/3) - 2/3$. Their ratio tends to 4/5. Hence, any refinement of the **min-degree** greedy algorithm has an approximation ratio at least 5/4.

**Motivation.** In addition to its simplicity and time efficiency, the greedy algorithm for MIS is also important in its own right. Following Halldórsson and Radhakrishnan [73], greedy algorithm is known to have several important properties: it achieves the celebrated Turán bound [142, 47], and its generalization in terms of degree sequences [146]. It achieves a good graph coloring approximation when applied iteratively as a coloring method [87]. Finally, the greedy algorithm finds optimal independent sets in trees, complete graphs, series-parallel graphs, co-graphs, split graphs, $k$-regular bipartite graphs, and graphs with maximum degree at most 2 [73, 16]. Another important but non-explicit class of graphs for which greedy is optimal is the class of *well-covered* graphs, introduced by Plummer [126], and widely studied, see [127] for a survey. A graph is well-covered if all its maximal independent sets have the same size. In particular, because any greedy set is maximal, the greedy algorithm is optimal on such graphs. Furthermore, the greedy algorithm finds frequent applications in graph theory, helping to prove that certain classes of graphs have large independent sets, e.g., it almost always finds a 2-approximation to MIS in a random graph [116], or it provides an independent set of size at least $0.432n$ in random cubic graphs with probability tending to 1 as $n$, the number of vertices, tends to $+\infty$ [56].

### 3.1.1 Our results

Our main technical contribution is a new class of "payment schemes" for proving improved and tight approximation ratios of greedy with additional tie-breaking rules. With our new payment schemes we obtain the best possible or the best known analyses of the greedy algorithm on bounded degree graphs. As a warm-up, we first apply these new techniques to MIS on graphs with maximum degree bounded by $\Delta$ to obtain the following results:

- A simple and short proof of the $(\Delta + 2)/3$-approximation ratio of the **min-degree** greedy algorithm, a result proved previously by Halldórsson and Radhakrishnan (Theorem 3.1) [73, 70]. We extend a lower bound construction of Halldórsson and Radhakrishnan [73] to prove that any minimum-degree greedy algorithm (i.e., any algorithm that iteratively picks a vertex of minimum degree) must have an approximation ratio at least $(\Delta + 1)/3 - O(1/\Delta)$.

- A simple proof of the $(\Delta + 6)/4$-approximation ratio of the **min-degree** greedy algorithm on triangle-free graphs with maximum degree $\Delta$ (Theorem 3.9), which

improves the previous best known greedy ratio of $\Delta/3.5 + O(1)$ [70] for MIS on triangle-free graphs. Compared to the proof in [70], which uses a technique of Shearer [138], our proof is extremely simple and short.

- A simple proof that the **more-edges** greedy algorithm achieves a 3/2-approximation in subcubic graphs (Theorem 3.2).

We see that as $\Delta$ increases, there is no hope in obtaining significantly better approximation than $(\Delta + 2)/3$ by using any, even exponential time, tie-breaking rule for greedy (Theorem 3.22). This motivates us to focus on the small values of $\Delta$. Indeed, we have to develop our payment scheme techniques significantly more compared to the above applications to MIS on graphs with maximum degree $\Delta$ for any value of $\Delta$.

We completely resolve the open problem from the paper of Halldórsson and Yoshihara [74] and design a fast, ultimate advice for greedy obtaining a 5/4-approximation, that is, the best possible greedy ratio for MIS on subcubic graphs.

**Theorem 3.3.** *There exists an **ultimate** greedy algorithm (Algorithm 9) such that given a graph $G = (V, E)$ with maximum degree 3, the algorithm outputs a independent set of size at least $\frac{4}{5} \cdot \alpha(G)$.*

Our new greedy 5/4-approximation algorithm has running time $O(n^2)$, where $n$ is the number of vertices in the graph. For comparison, the best known algorithm for this problem is a local search 6/5-approximation algorithm of Berman and Fujito [14], and with an analysis from [71] has a running time no less than $n^{50}$. Specifically, if the approximation ratio of this local search algorithm is fixed to 5/4, then the running time is $n^{18.27}$, see [31].

To prove these results we develop a payment technique to pay for the greedy solution via a specially defined class of potential functions. For this new class of potentials on subcubic graphs, we develop a very specific inductive process, which takes into account "parities" and priorities of the reductions performed by greedy, to prove that the value of the potential is kept locally to be at least $-1$. An additional, global argument is required to show that the global potential is at least 0.

We complement our positive results with impossibility results which suggest that our bounds on the design of good tie-breaking rules for greedy are essentially (close to) best possible, or non-trivial computational problems. This also suggests that the design of rules for the greedy algorithm is a non-trivial task.

A graph is called *well-covered* if all of its maximal independent sets are of the same size, see [126, 127]. Caro et al. [22] study the computational complexity of the problem of deciding if a given graph is well-covered. They prove that this problem is co-**NP**-complete even on $K_{1,4}$-free graphs.

To prove our lower bounds we resort to a notion which captures the essence of greedy. Namely, we study the computational complexity of computing a good greedy rule for MIS. Towards this goal, Bodlaender et al. [16] defined a problem called MaxGreedy, which given an input graph asks for finding the largest possible independent set obtained by any greedy algorithm. Thus, MaxGreedy asks for computing the best rule for greedy, i.e., one that leads to the largest possible greedy independent set. They proved that the problem of computing a rule which finds an $r$-approximate solution to the MaxGreedy problem is

co-**NP**-hard for any fixed rational number $r \geq 1$ and that this problem with $r = 1$ remains **NP**-complete [16].

We significantly improve the previously known results on the hardness of computing good rules for greedy, by obtaining the following new results:

- We prove that the MaxGreedy problem is **NP**-complete even on cubic planar graphs (Theorem 3.23). This significantly strengthens the **NP**-completeness result by Bodlaender et al. [16] who prove it on arbitrary, not even bounded-degree, graphs. This result suggests that the problem of designing and analyzing tie-breaking rules for greedy even on cubic planar graphs is difficult.

- We further prove that MaxGreedy is even **NP**-hard to approximate to within a ratio of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ by a reduction from 3-SAT, and hard to approximate to within $n/\log n$ under the Exponential Time Hypothesis (Theorem 3.27). We extend this construction to the class of graphs with bounded degree $\Delta$. We prove that MaxGreedy remains hard to approximate to within a factor $(\Delta + 1)/3 - O(1/\Delta) - O(1/n)$ on this class (Theorem 3.28), nearly matching the approximation ratio $(\Delta + 2)/3$ of the greedy algorithm in this class.

- We prove that the MaxGreedy problem remains hard to approximate on bipartite graphs (Theorem 3.29). This is quite interesting because it is well known that the MIS problem is solvable in polynomial time on bipartite graphs.

Finally, we extend a lower bound construction of Halldórsson and Radhakrishnan [73] to prove that any greedy algorithm (with any, even exponential time, tie-breaking rule) has an approximation ratio at least $(\Delta + 1)/3 - O(1/\Delta)$ on graphs with maximum degree $\Delta$ (Theorem 3.22).

### 3.1.2 Our technical contributions

Our main technical contributions are a class of potential functions and payment schemes, together with an inductive proof technique that are used to pay for solutions of greedy algorithms for MIS. These new techniques lead to tight analyses of the approximation ratios of greedy algorithms.

Here we will explain intuitions about our proof of the 5/4-approximation ratio of the **ultimate** greedy algorithm on subcubic graphs, detailed in Section 3.4, which uses the full technical machinery of our approach. Let $G$ be a given input graph with an optimal independent set **OPT**. The greedy algorithm executes *reductions* on $G$, i.e., a reduction is to pick a minimum degree vertex $v$ in the current graph (*root* of the reduction) into the solution and remove its neighbors, see Definition 3.4 and Figure 3.2 for examples of reductions. Suppose the first reduction executed by Greedy is $R$ and it is *bad*: its root $v$ has degree 2, $v \notin$ **OPT** and both neighbors of $v$ are in **OPT**. Then, locally, the approximation ratio is 2. To bring the approximation ratio down to 5/4, we must prove that, in the future, there will exist equivalent of at least three reductions, called *good*, each of which adds one vertex to the solution and removes only one vertex from **OPT**. Moreover, for each executed bad reduction, there must exist a unique (equivalent of) three good reductions.

Consider, for example, the family of instances $H_{i+1}$ of MIS in Figure 3.1. There, black vertices belong to **OPT**, while white do not. This class of instances is due to Halldórsson and Yoshihara [73]. The **min-degree** greedy algorithm executes on this instance many

bad reductions, but only at the very end, good reductions, triangles, are executed. It can easily be checked that there are just enough good reductions to uniquely map three of those to any executed bad reduction (in fact in the whole process there is exactly one good reduction that is unused). This essentially shows a lower bound of 5/4 on the ratio of any greedy when $i$ tends to infinity. We see that the "payment" for bad reductions arrives, but very late! Such a "payment" may not only be late, but we also do not know when "good" reductions providing such payment for "bad" reductions are executed. Thus good reductions might be very irregularly distributed. For instance, suppose that the first reduction in $H_0'$ on Figure 3.1, let us call it $R$, has two of its contact edges (these are the four edges going down from $R$'s two black vertices) going to an identical white vertex, creating a follow up reduction of degree one. Then that degree one reduction is good and when executed, it can immediately (partially) pay for the bad reduction $R$.

How do we prove an existence of such a highly non-local and irregular payment scheme? We will define a potential of a reduction, see Definition 3.5, which will imply the existence of two sources of "payments" – in the "past," from the very first executed reduction, and – in the future, from the executed good reductions. Moreover, our potential will be defined in such a way that each executed reduction can in some sense be "almost paid for" locally, so that at every point in time we will keep the value of the potential of each connected component of at least $-1$. For example in the instance from Figure 3.1 the execution of the first bad reduction in graph $H_{i+1}$ creates 4 connected components $H_i$, and then Greedy executes reductions in each of them independently.

We will have an intricate inductive argument, see the Inductive Low-debt Lemma 3.15, showing that an execution of a sequence of reductions in a connected input graph will have the total potential at least $-1$. In the induction step, some reductions $R$ may create multiple components, each with potential $-1$. In such cases when we cannot locally obtain a potential at least $-1$, we will make sure that even before the execution of $R$, such components contain reductions with strictly higher greedy priority than that of $R$, thus leading to a contradiction, or that reduction $R$ can pay for such components. Having proved that the total potential of an execution in the (connected) input graph is at least $-1$, we will finally pay for this $-1$ by the very first executed reduction for which we will show that it will always possess an extra saving of 1, which is a payment from the "past."

Intuitively, our potential will imply that we can ship the payments from good reductions executed *anywhere* in the graph by the greedy algorithm into the places where bad reductions need those payments. Such a shipment is unique, in the above sense that there exists three (equivalent) good reductions per single bad reduction. We will realize this shipment by deferring the need of payment into the future along edges, called *contact edges*, which are incident to the neighbors of the reduction's root vertex. When these contact edges are created by a bad reduction $R$, they will be called *loan* edges. Each loan edge $e$ created by $R$ will have a "dual" edge (physically identical to $e$), called a *debt* edge, which will be inherited by the future reductions directly created by $R$ via its contact edges.

Our potential of a reduction $R$ will account for the number of vertices chosen to the solution, removed from **OPT**, plus the number of loan edges, minus the number of debt edges. Thus, we will precisely account for such edges. This process is complicated by the fact that vertices can be *black* (in **OPT**) and *white* (outside of **OPT**) and whether a reduction is "bad" depends on the distribution of black/white vertices in the reduction. For instance, a reduction like the first one in graph $H_0$ on Figure 3.1 might have a black root and thus the two root's neighbors will be white. Such a reduction will in fact be

"good" when executed.

Some reductions are "bad" and they create "many" contact edges, like the first reduction in graph $H'_0$ on Figure 3.1 (with white root). Those "many" contact edges, called loan edges, will in the future create some good reductions that will pay for that bad one. Observe that such a reduction has more loan edges than debt edges, so it creates a surplus of credit. On the other hand when the greedy process ends, it can end only with terminal reductions. Those terminal reductions do not have any contact edges, but they have the property that for any white vertex added to the solution, they remove only one black vertex. That is why they can "pay" for the previous bad reductions.

This explains only some intuitions of why such a highly non-local and irregular payment scheme can have a chance to succeed, Furthermore, and but most importantly, this approach will enable us to "predict" the precise future graph structure by using the contact edges (see Lemma 3.15). It also enables us to keep track of the past reductions – by keeping track of the debt edges and the current state of the savings. And indeed, we have succeeded in building a theory that delivers a complete and precise such payment scheme.

This approach allows us to achieve a interesting kind of result here – namely, to (essentially) characterize all graphs that can have negative potential! See the Definition 3.14 and Lemma 3.15. These ideas lead to our analysis which is extremely tight, essentially up to an additive unit in the following sense. We prove that (a version of) our 5/4-approximate greedy algorithm finds a solution of size at least $\frac{4}{5}|\mathbf{OPT}| + \frac{1}{5}$ on any subcubic graph, whereas when run on the lower bound instances of Halldórsson and Yoshihara [73], our algorithm finds a solution of size precisely $\frac{4}{5}|\mathbf{OPT}| + \frac{1}{5}$. A somehow unusual aspect of our result is that we can prove that *any* lower bound example that shows exact tightness of our guarantee of $\frac{4}{5}|\mathbf{OPT}| + \frac{1}{5}$ must be an infinite family of graphs, see Section 3.5.3.

### 3.1.3 Further Related work

In this section we survey some further related work on MIS. It will be selective, because of vast existing literature on approximating MIS.

**On general graphs.** The MIS problem is known for its notorious approximation hardness. Håstad [80] provided a strong lower bound of $n^{1-\epsilon}$ on the approximation ratio of the general MIS problem for any $\epsilon > 0$, under the assumption that $\mathbf{NP} \not\subseteq \mathbf{ZPP}$, where $n$ is the number of vertices of the input graph. This hardness result has been derandomized by Zuckerman [148] who showed that the general MIS is not approximable to within a factor of $n^{1-\epsilon}$ for any $\epsilon > 0$, assuming that $\mathbf{P} \neq \mathbf{NP}$. The best known approximation algorithms for general MIS problem achieve the following approximation ratios: $O(n/\log^2 n)$ by Boppana and Halldórsson [17], that was improved to $\widetilde{O}(n/\log^3 n)$ by Feige [50] (this ratio has some additional $\log\log n$ factors).

**On graphs with maximum degree $\Delta$.** Even if $\Delta = 3$, MIS is known to be **APX**-hard, see [2]. There are also explicit constant hardness ratios known for small constant values of $\Delta$ [30]. As $\Delta$ grows, there are stronger, asymptotic, hardness of approximation results known: $\Omega(\Delta/\log^2 \Delta)$, under the Unique Games Conjecture [7], and $\Omega(\Delta/\log^4 \Delta)$, assuming that $\mathbf{P} \neq \mathbf{NP}$ [24].

The first known nontrivial approximation ratio for MIS on graphs with maximum degree $\Delta$ is $\Delta$, achieved by Lovasz's algorithmic proof [109] of Brooks' coloring theorem.

The best known asymptotic polynomial time approximation ratio for MIS, i.e., when $\Delta$ is large, is $O(\Delta \log\log(\Delta)/\log(\Delta))$, based on a semidefinite programming relaxation [75]. However, if we allow for some extra time, there exists an asymptotic $O(\Delta/\log(\Delta))$-approximation algorithm with $O(n^{O(1)} \cdot \exp(\log^{O(1)} n))$ running time (Bansal et al. [9]). And in particular, the best known asymptotic approximation ratio for MIS is $O(\Delta/\log^2(\Delta))$ with $O(n^{O(1)} \cdot 2^{O(\Delta)})$ running time [10]. For small to moderate values of $\Delta$, Halldórsson and Radhakrishnan [72, 71], via subgraph removal techniques, obtain an asymptotic ratio of $\Delta/6(1+o(1))$ with $O(\Delta^{O(1)} \cdot n)$ running time for relatively small $\Delta \geq 5$, and $O(\frac{\Delta}{\log\log\Delta})$ for larger $\Delta$ with linear running time.

Demange and Paschos [44] prove that a $\Delta/6$-approximation ratio can be obtained in time $O(nm)$, but this ratio is asymptotic as $\Delta \to \infty$, where $m$ is the number of edges in the input graph. They also prove that an $\Delta/k$-approximation ratio can be achieved in time $O(n^{\lceil k/2 \rceil})$, for any fixed integer $k$. This second ratio is also asymptotic. Those algorithms do not apply to $\Delta = 3$, but to quite large values of $\Delta$. Furthermore, Khanna *et al.* [92] obtained a $(\sqrt{8\Delta^2 + 4\Delta + 1} - 2\Delta + 1)/2$-approximation for $\Delta \geq 10$, with $O(\Delta^{O(1)} \cdot n)$ running time by local search.

For any values of $\Delta$, Hochbaum [78], using a coloring technique accompanied with a method of Nemhauser and Trotter [123] based on linear programming, obtained an algorithm with a ratio of $\Delta/2$. Berman and Fürer [15] designed an algorithm whose performance ratios are arbitrarily close to $(\Delta + 3)/5$ for even $\Delta$ and $(\Delta + 3.25)/5$ for odd $\Delta$. Berman and Fujito [14] obtained a better ratio which is arbitrarily close to $\frac{\Delta+3}{5}$. Finally, in the latest previously known results from Chlebík and Chlebíková [31], their approximation ratio is arbitrarily close to $\frac{\Delta+3}{5} - \frac{4(5\sqrt{13}-18)}{5}\frac{(\Delta-2)!!}{(\Delta+1)!!}$, which is slightly better than the previous results. These algorithms are based on local search and they have huge running times.

**On subcubic graphs.** For the case of subcubic graphs with $\Delta = 3$, MIS is known to be **NP**-hard to approximate to within $\frac{95}{94}$ [30]. Hochbaum [78] presented an algorithm with a ratio of 3/2. Berman and Fujito [14] obtain a ratio of $\frac{6}{5}$ albeit with a huge running time. Even with a tighter analysis from [71], the time complexity of such local search algorithm appears to be no less than $n^{50}$. Chlebík and Chlebíkova [31] show that their approximation ratio is arbitrarily close to $3 - \frac{\sqrt{13}}{2}$, which is slightly better than $\frac{6}{5}$. Moreover, the time complexity of their algorithm is also better. Specially, if the ratio is fixed to $\frac{5}{4}$, then the running time is $n^{18.27}$. Halldorsson and Radhakrishnan [72] provided another local search approach based on [15] and obtain a ratio of $\frac{7}{5}$ in linear time, and a $(\frac{4}{3} + \epsilon)$-ratio in time $O(ne^{1/\epsilon})$. Finally, Halldórsson and Yoshihara [74] presented a linear time greedy algorithm with an approximation ratio of $\frac{3}{2}$.[2]

## 3.2 Analyzing Greedy: our payment scheme

In this section we present a payment scheme tool to analyze the approximation ratio of (any sophistication of) the **min-degree** greedy algorithm for Maximum independent set.

It is clear that at the end of the execution of the greedy algorithm (Algorithm 7), the solution $S$ returned is an independent set. Let $S = \{v_1, \ldots, v_k\}$ be the ordered output. Let $G_i$ denote the graph after removing vertex $v_i$ and its neighboring vertices. More precisely,

[2]They also claim a better ratio of 9/7 in linear time, however, they have retracted this result [69].

$G_0 = G$ and $G_i = G[V \setminus N_G[\{v_1, \ldots, v_i\}]]$, where $v_i$ is a vertex in $G_{i-1}$ that satisfies $d_{G_{i-1}}(v_i) = \min\{d_{G_{i-1}}(v) : v \in V(G_{i-1})\}$.

### 3.2.1 Basic reductions

Since the greedy algorithm removes iteratively a vertex and its neighbors from the current graph, the whole execution of the algorithm can be seen as a sequence of *reductions* of the input graph. To analyze an execution of the greedy algorithm we will only require local information around the vertices picked at each step of the algorithm.

**Definition 3.4.** Given a graph $G$, the **basic reduction** $R$ associated to a vertex $r \in V(G)$ corresponds to the subgraph induced by vertices at distance at most 2 from $r$. We refer to $r$ as the **root** of the reduction. The root and its neighbors, the **middle vertices**, form the **ground** of the reduction, denoted by $ground(R)$. Vertices at distance two from the root are the **contact vertices**. The set of contact vertices is denoted by $contact(R_i)$. Then, the edges between middle and contact vertices are called **contact edges**. Finally, the **degree** of a basic reduction is defined as the degree of its root vertex.

See Figure 3.2 for an illustration. Each iteration of the greedy algorithm corresponds to a basic reduction, denoted by $R_i$, which can be described by a pair $(v_i, G_{i-1})$. An **execution** $\mathcal{E} := (R_1, \ldots, R_k)$ of the greedy algorithm is the ordered sequence of basic reductions performed by the algorithm. The size $k$ of the solution returned is equal to the number of basic reductions executed.

From now on, we will consider that two basic reductions $R = (v, G)$ and $R' = (v', G')$ are "equivalent" if there exists a one-to-one function $\phi : N_G[v] \longrightarrow N_{G'}[v']$ such that $\phi(v) = v'$, $u$ and $w$ are adjacent in $G$ if and only if $\phi(u)$ and $\phi(w)$ are adjacent in $G'$, and if each middle vertex $u$ is incident in $G$ to the same number of contact edges than $\phi(u)$ in $G'$.

For the **min-degree** greedy algorithm, Figure 3.2 presents a table of all possible basic reductions of degree at most two in subcubic graphs. It is important to notice that the middle vertices must have degrees equal to or greater than the degree of the reduction.

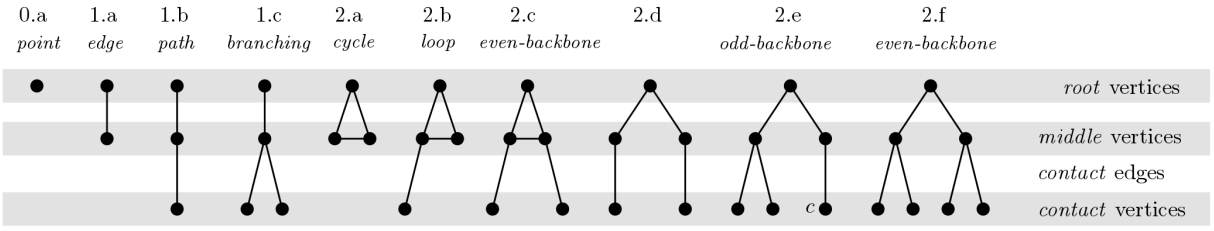



Figure 3.2: Basic reductions of degree at most two in subcubic graphs. We will refer later to these basic reductions by their names, for instance 2.b is a *basic loop reduction*. In this picture, we have drawn contact vertices as distinct vertices, but in a reduction, several contact edges may be incident to the same contact vertex. When the right-most contact vertex $c$ of 2.e has degree three, this reduction is an odd-backbone reduction. Notice that in this case, the middle vertex of degree two is also the root of a basic odd-backbone reduction.

### 3.2.2 Potential function of basic reductions

Let $G = (V, E)$ be a connected graph with maximum degree $\Delta$ and $\mathcal{E} = (R_1, \ldots, R_k)$ an execution of the greedy algorithm on the input graph $G$.

This execution is associated to a decreasing sequence of subgraphs of $G$:

$$G = G_0 \supset \cdots \supset G_k = \emptyset,$$

where $G_i = G\left[V \setminus \bigcup_{j=1}^{i} ground(R_j)\right]$ is the induced sub-graph of $G$ on the set of vertices $V \setminus \bigcup_{j=1}^{i} ground(R_j)$.

We also fix an independent set $I$ and we say that a vertex $v$ is **black** if $v \in I$ and **white** otherwise.

Given a basic reduction $R_i = (v_i, G_{i-1})$, we define **loan edges** of $R_i$ as all contact edges with a white contact vertex. Notice that the middle vertex of a loan edge can either be black or white.

We also define the **debt** of a *white* vertex in the ground of $R_i$ as the number of times this vertex was incident to a loan edge, let us call it $e'$, of a reduction that was previously executed. Such loan edge $e'$ is also called a *debt* edge of reduction $R_i$. It turns out that the debt of a white vertex corresponds exactly to the difference between its degree in the original graph $G$ and in the current graph $G_{i-1}$. We now define the exact potential of basic reduction.

**Definition 3.5.** Given two parameters $\gamma, \sigma \geq 0$, the **exact potential**, *w.r.t.* $\gamma$ and $\sigma$, of a reduction $R_i$, for $1 \leq i \leq k$, as

$$\Phi_{G,I}(R_i) := \gamma - \sigma \cdot |I \cap ground(R_i)| + loan_I(R_i) - debt_{G,I}(R_i),$$

where $loan_I(R_i)$ is the **loan of reduction** $R_i$ that corresponds to its total number of loan edges, and $debt_{G,I}(R_i)$ is the **debt of a reduction** defined as the sum of the debts of the vertices of its ground:

$$debt_{G,I}(R_i) := \sum_{u \in ground(R_i) \setminus I} \left(d_G(u) - d_{G_{i-1}}(u)\right).$$

The **exact potential of an execution** $\mathcal{E} = (R_1, \ldots, R_k)$ is the sum of the exact potentials of all reductions:

$$\Phi_{G,I}(\mathcal{E}) = \sum_{i=1}^{k} \Phi_{G,I}(R_i).$$

The choice of the parameters $\gamma, \sigma$ depends on the approximation ratio that we want to establish. We first obtain the following property.

**Proposition 3.6.** *Given an execution $\mathcal{E} = (R_1, \ldots, R_k)$ of the greedy algorithm on a graph $G$ with an independent set $I$ we have: $\Phi_{G,I}(\mathcal{E}) = \gamma \cdot k - \sigma \cdot |I|$.*

*Proof.* Since the independent set produced by the greedy algorithm is maximal, each vertex in the $G$ is removed exactly once, i.e., is in the ground set of exactly one reduction. Thus, $\sum_{i=1}^{k} |ground(R_i) \cap I| = |I|$. Moreover, the total debt $\sum_{i=1} debt_{G,I}(R_i)$ and the total loan $\sum_{i=1} loan_I(R_i)$ are equal. The equality claimed follows by a by a simple counting argument. □

If we manage to find suitable values $\gamma, \sigma$ such that all possible reductions have non negative potential, then a direct corollary of Proposition 3.6 is that Greedy is an $(\gamma/\sigma)$-approximation algorithm.

In order to measure the potential of each reduction, we now define a new potential, called simply *potential*, that is a lower bound on the exact potential.

As it relies on the notion of debt, the exact potential of a reduction $R_i$ depends on the original graph $G$. This lower bound is obtained by supposing that the debt of each white vertex is maximal, or equivalently, that its degree in the original graph was equal exactly to $\Delta$, the maximum degree in the input graph. Namely,

$$debt_I(R_i) := \sum_{u \in ground(R_i) \setminus I} (\Delta - d_{G_{i-1}}(u)) \geq debt_{G,I}(R_i). \tag{3.1}$$

Then we define the **potential**, *w.r.t.* $\gamma$ and $\sigma$, of reduction $R_i$, which is now independent from the original graph, as

$$\Phi_I(R_i) := \gamma - \sigma \cdot |I \cap ground(R_i)| + loan_I(R_i) - debt_I(R_i).$$

We define the **potential of an execution** similarly. See Figures 3.3, 3.7 and 3.10 for examples of the potential of basic reductions in subcubic graphs with parameters $\gamma, \sigma = 5, 4$. Obviously, this new potential is a lower bound on the exact potential defined previously, and more precisely, we have the following relation, that follows easily from the above definitions.

**Proposition 3.7.** *Let $G$ be a graph with maximum degree $\Delta$, $I$ an independent set in $G$ and $\mathcal{E}$ an execution in $G$. Then,*

$$\Phi_{G,I}(\mathcal{E}) = \Phi_I(\mathcal{E}) + \sum_{v \notin I} (\Delta - d_G(v)).$$

To evaluate the potential of a reduction, we no longer need to know the set of reductions previously executed but simply the structure of the graph formed by the vertices at distance two from the root, and also which vertices are black/white, which reduces to a relatively small number of cases. With this potential we are already able to obtain tight analysis of the approximation ratio of the **min-degree** greedy algorithm, as we will demonstrate in Section 3.3.1. For this, we simply have to find the right parameters $\gamma, \sigma$, then check that all basic reductions have non-negative potential, and conclude by the following result:

**Corollary 3.8.** *Let $\mathcal{G}$ be a family of graphs, and let $\mathcal{R}$ be the set of basic reductions obtained from graphs in $\mathcal{G}$ by a refinement $\mathcal{A}$ of the generic greedy algorithm (Algorithm 7). Then, let $\gamma, \sigma$ be two parameters such that for any reduction $R \in \mathcal{R}$ and any independent set $I$, we have $\Phi_I(R) \geq 0$ then the greedy algorithm $\mathcal{A}$ is a $(\gamma/\sigma)$-approximation on $\mathcal{G}$.*

In the next section, we apply this strategy for three classes of graphs. As we will primarily see in Section 3.3.3 and later to design the ultimate 5/4-approximation in subcubic graphs, negative potential reductions are sometimes inevitable. Fortunately, Corollary 3.8 establishes only a necessary condition, since our initial goal is to guarantee that the exact potential of the whole execution is non-negative (Proposition 3.6). We will see that even if some basic reductions have negative potential, it is still possible to group these "bad" reduction with "good" ones, i.e., reductions with positive potential such that the whole group has non-negative potential.

## 3.3 Application to bounded degree graphs

In this section, we use the potential introduced in the previous section to prove Theorem 3.1 and Theorem 3.2. Our proof is simpler and shorter compared to the proof in [73] and [74]. We additionally prove that the approximation ratio of the **min-degree** greedy algorithm can be improved for triangle-free and bounded-degree graphs.

**Theorem 3.9.** *Given a triangle-free graph $G$ with maximum degree $\Delta$, let $I$ be an independent set outputs by the **min-degree** greedy algorithm on input $G$. Then,* $|I| \geq \frac{4}{\Delta+6} \cdot \alpha(G)$

This new result improves the previous best known greedy ratio of $\Delta/3.5 + O(1)$ [70] for Maximum Independent Set on triangle-free graphs. Compared to the proof in [70], which uses a technique of Shearer [138], our proof is again extremely simple and short.

### 3.3.1 Proof of Theorem 3.1

We prove in this section that the **min-degree** greedy algorithm is a $\frac{\Delta+2}{3}$-approximation algorithm for the Maximum independent set problem on graphs with maximum degree $\Delta$. For this, let us use the potential from the previous section, with parameters

$$\gamma = (\Delta + b)\tfrac{\Delta+2}{3} \text{ and } \sigma = \Delta + b$$

where $b = 1$ if $\Delta \equiv 2 \pmod 3$, and $b = 0$ otherwise. The choice of the value $b$ is simply to ensure that the potential value is integer. We prove that the potential of any reduction is non-negative, and apply Corollary 3.8 so that the approximation ratio of the **min-degree** greedy algorithm in graphs with maximum degree $\Delta$ is $\gamma/\sigma = (\Delta+2)/3$.

Let $\mathcal{R}$ be the set of all possible basic reductions, and let $I$ be a maximum independent set in the input graph. We note that although there are many types of reductions in $\mathcal{R}$, their structure is highly regular. The idea of the proof is to find the worst type reduction and show that its potential is non-negative. Observe that, if we want to find a reduction $R^*$ to minimize the potential, $R^* = \arg\min_{R\in\mathcal{R}} \Phi_I(R)$, such reduction intuitively needs more debt edges and vertices in $I$ and less loan edges. Also, if $v^*$ is the root of reduction $R$, then for each $v \in V(R) \setminus \{v^*\}$, if $d_R(v^*) = k$, then $d_R(v) \geq k$, by the greedy rule. For any reduction $R$, let $i$ be the number of vertices in $I \cap ground(R)$ (black vertices), and let $\ell$ be the number of vertices in $ground(R) \setminus I$ (white vertices). We then have the following bounds:

$$loan_I(R) \geq (i + \ell - 1 - \ell) \cdot i,$$

$$debt_I(R) \leq (\Delta - i - \ell + 1) \cdot \ell.$$

We will justify these bounds now. Let $G'$ be the current graph just before $R$ is executed. Note first that the degree of the root of $R$ is $i + \ell - 1$. The lower bound on $loan_I(R)$ depends on the vertices in $I$, by the definition. By the **min-degree** rule, for each of vertex $v \in I$, $d_{G'}(v) \geq i + \ell - 1$. There are at most $\ell$ vertices not in $I$ which can be connected to $v$, thus, the total number of loan edges of $v$ is at least $(i + \ell - 1 - \ell)$, and we have $i$ such vertices. Note that in this argument we have possibly missed all loan edges that are contact edges of $R$ with both end vertices from $ground(R) \setminus I$. The upper bound on

$debt_I(R)$ depends on $\Delta$, the degree of the root vertex and the number of vertices not in $I$. The number of debt edges is at most $\Delta - i - \ell + 1$ for any white vertex, as otherwise it violates the **min-degree** rule, and we have $\ell$ vertices not in $I$, that is, white vertices.

$$\begin{aligned}\Phi_I(R) &= \frac{\Delta+b}{3}\cdot(\Delta+2) - (\Delta+b)|I \cap ground(R)| + loan_I(R) - debt_I(R)\\ &\geq \frac{\Delta+b}{3}(\Delta+2) - (\Delta+b)i + (i-1)i - (\Delta - i - \ell + 1)\ell\\ &= \ell^2 - (\Delta - i + 1)\ell + \frac{\Delta+b}{3}(\Delta+2) - (\Delta+b)i + (i-1)i\end{aligned}$$

Let $F(\Delta, i, \ell) = \ell^2 - (\Delta - i + 1)\ell + \frac{\Delta+b}{3}(\Delta+2) - (\Delta+b)i + (i-1)i$. Then, the question now is to find the minimum value of $F(\Delta, i, \ell)$ with constrains $\Delta, i, \ell \in \mathcal{Z}^+ \cup \{0\}$.

We will first prove that $F(\Delta, i, \ell) \geq b/3 - b^2/3 - 1/3$ for any $\Delta, i, \ell \in \mathcal{R}^+ \cup \{0\}$. $F(\Delta, i, \ell)$ as a function of $\ell$ is a parabola with the global minimum at point $\ell$ such that $\frac{\partial F}{\partial \ell} = 0$, which gives us that $\ell = (\Delta - i + 1)/2$. Plugging $\ell = (\Delta - i + 1)/2$ into $F(\Delta, i, \ell)$, we obtain the following function:

$$F(\Delta, i, (\Delta - i + 1)/2) = F(\Delta, i) = -\frac{1}{4}(\Delta - i + 1)^2 + \frac{\Delta+b}{3}(\Delta+2) - (\Delta+b)i + (i-1)i$$

$$= \frac{3}{4}i^2 - (\Delta/2 + 1/2 + b)i + \frac{\Delta+b}{3}(\Delta+2) - \frac{1}{4}\Delta^2 - \frac{1}{2}\Delta - \frac{1}{4}.$$

Similarly, function $F(\Delta, i) = \frac{3}{4}i^2 - (\Delta/2 + 1/2 + b)i + \frac{\Delta+b}{3}(\Delta+2) - \frac{1}{4}\Delta^2 - \frac{1}{2}\Delta - \frac{1}{4}$ as a function of $i$ is a parabola with the global minimum at $i$ such that $\frac{\partial F}{\partial i} = 0$, which implies $i = \frac{2}{3}(\Delta/2 + 1/2 + b)$. Plugging $i = \frac{2}{3}(\Delta/2 + 1/2 + b)$ in $F(\Delta, i)$ we obtain the following:

$$F\left(\Delta, \frac{2}{3}(\Delta/2 + 1/2 + b)\right) = F(\Delta) = b/3 - b^2/3 - 1/3.$$

From the above we have that $F(\Delta, i, \ell) \geq b/3 - b^2/3 - 1/3$ for any $\Delta, i, \ell \in \mathcal{R}^+ \cup \{0\}$.

Now, let us observe that if $\Delta \equiv 0, 1 \pmod 3$, then $F(\Delta, i, \ell)$ with $b = 0$ is an integer whenever $\Delta, i$ and $\ell$ are integers. This means that in those cases we have $F(\Delta, i, \ell) \geq -1/3$ which implies that $F(\Delta, i, \ell) \geq 0$. In case when $\Delta \equiv 2 \pmod 3$, we have that $F(\Delta, i, \ell)$ with $b = 1$ is an integer when $\Delta, i, \ell$ are integers. This means that in those cases $F(\Delta, i, \ell) \geq -1/3$, implying $F(\Delta, i, \ell) \geq 0$.

This shows that the potential of any reduction is non-negative for this choice of parameters $\gamma, \sigma$, and in particular that means that the **min-degree** greedy algorithm has an approximation ratio $(\gamma/\sigma) = (\Delta + 2)/3$.

### 3.3.2 Proof of Theorem 3.9

We prove in this section that the **min-degree** greedy algorithm is a $\frac{\Delta+6}{4}$-approximation algorithm for the Maximum independent set problem on triangle-free graphs with maximum degree at most $\Delta$. For this, let us use the potential function here with parameters:

$$\gamma = \Delta \cdot \tfrac{\Delta+6}{4} \text{ and } \sigma = \Delta.$$

We prove that $\Phi_I(R) \geq 0$ for any reduction $R$, and any independent set $I$, so that the approximation ratio of the **min-degree** greedy algorithm on triangle-free graphs with

maximum degree $\Delta$ is $\gamma/\sigma = \frac{(\Delta+6)}{4}$.

Let us first assume that the root vertex of $R$ is white. Then, as in Section 3.3.1, we claim that for triangle-free graphs we have:

$$loan_I(R) \geq (i + \ell - 1 - 1) \cdot i,$$

$$debt_I(R) \leq (\Delta - i - \ell + 1) \cdot \ell.$$

Note that the upper bound on debt edges is the same, however, the lower bound on loan edges is significantly greater for triangle-free graphs. We justify this first lower bound. Observe, that for a triangle-free graph, for any reduction $R$, no two middle vertices of $R$ are adjacent. Note first that the degree of the root of $R$ is $i + \ell - 1$. We obtain a lower bound on $loan_I(R)$ by counting the number of contact edges incident on any middle vertex $v \in I$ of $R$. By the **min-degree** rule, for each such vertex $v$, $d_{G'}(v) \geq i + \ell - 1$, where $G'$ is the current graph. Because the root of $R$ was assumed to be white, there is at most one vertex (the root vertex) not in $I$ which can be connected to $v$. Thus, the total number of loan edges of $v$ is at least $(i + \ell - 1 - 1)$, and we have $i$ such vertices. Then we obtain further that:

$$\begin{aligned}\Phi_I(R) &\geq \frac{\Delta}{4}(\Delta + 6) - \Delta i + (i + \ell - 2)i - (\Delta - i - \ell + 1)\ell \\ &= \ell^2 - (\Delta - 2i + 1)\ell + \frac{\Delta}{4}(\Delta + 6) - \Delta i + (i - 2)i.\end{aligned}$$

Let $F(\Delta, i, \ell) = \ell^2 - (\Delta - 2i + 1)\ell + \frac{\Delta}{4}(\Delta + 6) - \Delta i + (i - 2)i$. Then, we show that $F(\Delta, i, \ell) \geq -i + \Delta - \frac{1}{4}$ for any $\Delta, i, \ell \in \mathcal{R}^+ \cup \{0\}$.

Let now $d \leq \Delta$ be the degree of the root of $R$. If $d \geq i + 1$, then $F(\Delta, i, \ell) \geq 0$ and $\Phi_I(R) > 0$. Suppose now that $d \leq i$, then obviously $d = i$. In such case, because $d = i + \ell - 1$, $\ell = 1$, thus:

$$\Phi_I(R) \geq \frac{\Delta}{4}(\Delta + 6) - \Delta d + (d - 1)d - (\Delta - d) = d^2 - \Delta d + \frac{\Delta^2}{4} + \frac{\Delta}{2} \geq \frac{\Delta}{2},$$

where the last inequality follows by the fact the quadratic function is minimized for $d = \Delta/2$.

Let us now assume that the root of $R$ is black. Noting that $i = 1$ and $\ell \leq \Delta$, we obtain:

$$\begin{aligned}\Phi_I(R) &= \frac{\Delta}{4} \cdot (\Delta + 6) - \Delta \cdot |I \cap ground(R)| + loan_I(R) - debt_I(R) \\ &\geq \frac{\Delta}{4} \cdot (\Delta + 6) - \Delta + 0 - (\Delta - 1 - \ell + 1)\ell = \ell^2 - \Delta\ell + \frac{\Delta^2}{4} + \frac{\Delta}{2} \geq \frac{\Delta}{2}.\end{aligned}$$

As above, the last inequality follows by the fact that the quadratic function is minimized for $\ell = \Delta/2$. This shows that the potential of any reduction is non-negative for this choice of parameters $\gamma, \sigma$, and in particular that means that the **min-degree** greedy algorithm has an approximation ratio $(\gamma/\sigma) = (\Delta + 6)/4$.

### 3.3.3 Proof of Theorem 3.2

We prove in this section that the **more-edges** greedy algorithm is a $\frac{3}{2}$-approximation algorithm for the Maximum independent set problem on subcubic graphs. For this, let us use the potential function here with parameters:

$$\gamma = 6 \text{ and } \sigma = 4.$$

**Input:** A subcubic graph $G$.
**Output:** An independent set $S$ in $G$.
$S \leftarrow \emptyset$;
**while** $G \neq \emptyset$ **do**
  **if** $G$ *has minimum degree* 2 *and maximum degree* 3 **then**
    Let $v$ be a vertex of degree two with a neighbor of degree three.
  **else**
    Let $v$ be any vertex of minimum degree.
  $S \leftarrow S \cup \{v\}$;
  $G \leftarrow G \setminus N_G[v]$;
**return** $S$;

**Algorithm 8:** the **more-edges** greedy algorithm.

We first start by looking at the potential of each basic reduction $R$ (see Figure 3.2) for all possible independent sets in $ground(R)$. This can be easily done "by hand" since there is a small number of such reductions. We realize that all reductions have potential at least $-1$ and that the only reduction with potential $-1$ corresponds to reduction 2.d in Figure 3.2 where the root is white and its two neighbors are black, like in Figure 3.3. Let us call this particular reduction $R'$.

The objective of the **more-edges** rule is to guarantee that when reduction $R'$ is executed by the **more-edges** greedy algorithm, then it must be followed by at least one reduction with potential at least $+1$. Intuitively, this last reduction will "pay" for the reduction of potential $-1$ and this payment is unique so that the potential of the whole execution is non-negative.

Given a graph and an independent set $I$, consider an execution of the **more-edges** greedy algorithm. Suppose that at some point of the execution of the **more-edges** greedy algorithm, the current graph $G$ has minimum degree two and maximum degree three. Then, there is necessarily a vertex $u$ of minimum degree two that has at least one neighbor of degree three. In this situation, according to the **more-edges** rule, we prefer picking the reduction with root vertex $u$ than $R'$ that has two neighbors of degree two. This means that when $R'$ is executed, the current graph must be 2-regular, i.e., a disjoint union of cycles. Let us call $C$ the cycle where $R'$ is executed and $\mathcal{E}'$ the sequence of reductions executed in $C$. We show that the potential of $\mathcal{E}'$, i.e., the sum of the potential of the reductions executed in $C$, is at least one.

Suppose that $C$ has $b$ black vertices and $w$ white vertices, where $b + w = |C|$. Since $R'$ is executed, we can assume that $b \geq 2$ and $b \leq \alpha(C) = \lfloor |C|/2 \rfloor$.

Before the execution, each white vertex has a debt equal to one, so that the total debt is $w$. Also, the loan is zero after the execution. Moreover, the greedy algorithm is optimal in graphs with maximum degree two, which means that in $C$, the solution returned has size $\lfloor |C|/2 \rfloor$. Then,

$$\Phi_I(\mathcal{E}') = 6 \cdot \lfloor |C|/2 \rfloor - 4 \cdot b - w + 0 \geq 2\lfloor |C|/2 \rfloor - w \geq b - 1 \geq 1$$

Thus, the whole execution has non-negative potential which implies that the **more-edges** greedy algorithm is a 6/4-approximation on subcubic graphs.

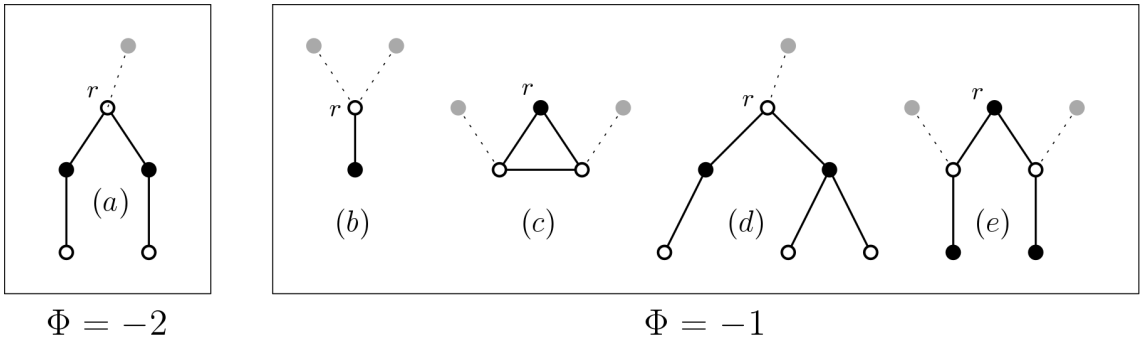



Figure 3.3: Basic reductions with negative potential, where $\gamma, \sigma = 5, 4$. The root vertex of reductions is denoted by the letter $r$. Dotted edges represent the debt of each white vertex. Grey vertex can either be black or white.

## 3.4 The ultimate greedy algorithm for subcubic graphs

In this section, we present a new rule called **ultimate** for subcubic graphs such that the corresponding **ultimate** greedy algorithm achieves an approximation ratio of 5/4.

The exact potential that we use for subcubic graphs is given by the values $\gamma = 5$ and $\sigma = 4$. A table in Figure 3.3 shows the potential of several basic reductions for some different independent sets. Unfortunately, as one can see in Figure 3.3, there exists reductions with negative potential. The goal of the **ultimate** rule will be to deal with these cases.

The first step is to collect some consecutive basic reductions into one *extended reduction*, so that the potential of some basic reductions is balanced by that of the others. For instance, one way to deal with the basic reduction 2.d in Figure 3.2, which can have potential $-2$ (see $(a)$ in Figure 3.3), is to force the greedy algorithm to prioritize a vertex of degree two with a neighbor of degree three. Therefore, if at some point a reduction 2.d is executed, it means that the current graph is an union of disjoint cycles. This allows us to consider that the whole cycle forms an extended reduction — that we will call as **cycle reduction** — and we will see later that its potential will be at least $-1$.

An useful observation in order to define an appropriate extended reduction is to notice that the (basic) path reduction (1.b from Figure 3.2) has potential at least zero. This observation suggests to introduce the following notion. Given a graph $G = (V, E)$ we will say that the set $B = \{w, v_1, \ldots, v_b, w'\} \subset V$ is a **backbone** if the induced subgraph $G[\{v_1, \ldots, v_b\}]$ is a path and if $w$ and $w'$ both have degree three. In this case, $w$ and $w'$ are called the **end-points** of the backbone $B$. Moreover, when $b$ is odd (*resp.* even), we will say that $B$ is an **even** (*resp.* **odd**) **backbone** — notice the asymmetry — which corresponds to the parity of the number of *edges* between the end-points. As an example, the ground of the basic even-backbone reductions (2.f and 2.c in Figure 3.2) are special case of an even-length-backbone (of edge-length two).

### 3.4.1 Extended reductions

An **extended reduction** $\overline{R} = (R_1, \ldots, R_s)$ is a sequence of basic reductions $R_i$ of special type that we will precisely describe in the next paragraph. All different extended reductions are summarized in Proposition 3.10. To facilitate the discussion, when there is no risk of

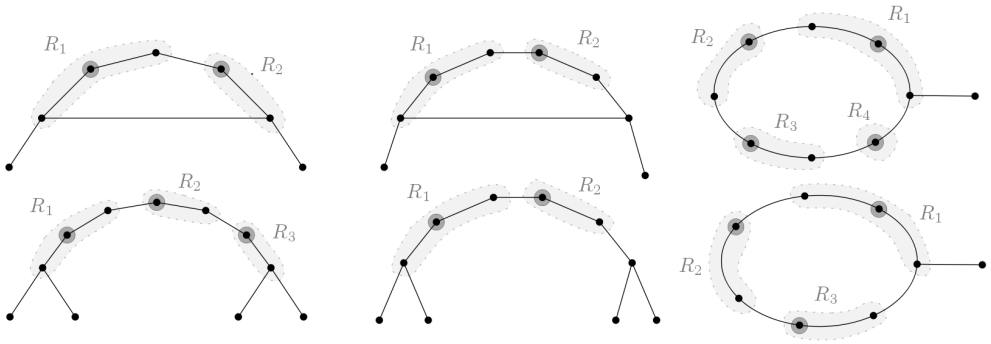



Figure 3.4: Some examples of (extended) reductions of degree two and size at least two. Light grey areas indicate the ground of each executed basic reduction, that together form the ground of the (extended) reduction. Vertices surrounded by grey rings are the roots of the corresponding basic reductions.

confusion, we will simply call it a **reduction**.

The **size** of an extended reduction $\overline{R}$, written $|\overline{R}|$ is the number of executed basic reductions. Its **ground** naturally corresponds to the union of the grounds of its basic reductions, $ground(\overline{R}) := \bigcup_i ground(R_i)$ and its **root** is the same as the root of the first basic reduction. Finally, the **contact vertices** correspond to all contact vertices of its basic reductions that are not in $ground(\overline{R})$. The **degree** of a reduction is the degree of the first executed basic reduction. All basic reductions of Figure 3.2 except 2.d will be considered as (extended) reductions of size one. In particular, all (extended) reductions of degree one considered by the algorithm have size one. Other considered (extended) reductions of degree two have a ground which is a backbone (except the case of odd-backbone where one end-point is excluded). When the two end-points of the backbone are the same vertex, it corresponds to a **loop reduction**. Otherwise, reductions associated to an even and odd backbone are, respectively, called **even-backbone** and **odd-backbone** reductions. When these reductions have size at least two, they correspond to a sequence $\overline{R} = (R_1, \ldots, R_s)$ of basic reductions where:

- The first (basic) reduction $R_1$ is 2.e from Figure 3.2.
- Intermediate reductions $R_i$, with $2 \leq i \leq s-1$, are basic path reduction (1.b from Figure 3.2), where the root vertex of $R_i$ is the contact vertex of $R_{i-1}$.
- The final (basic) reduction $R_s$ corresponds to:
  - branching (1.c) or path (1.b), when $\overline{R}$ is an **even-backbone** reduction. The case $R_s =$ **path**, occurs when the end-points are adjacent.
  - path (1.b), when $\overline{R}$ is an **odd-backbone** reduction.
  - point (0.a) or edge (1.a), when $\overline{R}$ is a **loop** reduction, depending on the parity of the length of the backbone. Recall that the two end-points are identical in this case.

We give examples of different types of extended reductions of degree two in Figure 3.4. Some further remarks are in place here:

- The following basic reductions in Figure 3.2 are special case of (extended) reduction of size one.

  **2.a :** cycle reduction.

  **2.c and 2.f :** even-backbone reduction.

  **2.e :** odd-backbone reduction. This applies only when the right-most contact vertex $c$ has degree three.

  **2.b :** loop reduction.

- The root of an even-backbone, odd-backbone or loop reduction is always the neighbor of one of the end-points of the backbone. For even-backbone and loop reduction of even-length, any of the two choices leads to the same solution. In the case of an odd-loop reduction, the size of the solution — and therefore also the potential — and the ground of the reduction is exactly the same. For loop and even-backbone reductions, the ground of the reduction is the full backbone. However, for odd-backbone reductions, given one backbone, there are *two distinct* possible roots, associated to two distinct grounds. For each odd-backbone reduction, only one end-point is contained in the ground. See Figure 3.4.

- All basic reductions of an extended reduction, except the first one have degree at most one, so that at any given moment, any executed basic reduction has minimum degree in the current graph. This means that we are allowed to execute the full extended reduction without violating our original greedy rule.

In what follows, we will refer to an extended reduction in two different (and equivalent) ways. We will either write its name with the first capital letter or we will write its name with the first lower-case letter followed by the word "reduction". Thus, for example, we will say a loop reduction or just Loop, or an even-backbone reduction or just Even-backbone, etc. Note that basic reductions are special case of extended reductions, and therefore we may also follow this convention for them.

### 3.4.2 The ultimate greedy algorithm

We now describe the **ultimate** rule used to reach the best possible greedy approximation of Maximum Independent Set in subcubic graphs.

This **ultimate** rule can be described as the conjunction of different criteria, and the vertex picked (i.e., the reduction executed) by the **ultimate** greedy algorithm must satisfy all these criteria.

When the input graph $G$ is cubic, i.e., each vertex has degree exactly three, then the first reduction has degree three. However, this is the only degree three reduction executed during the whole execution since there will always be a vertex with degree at most two. We explain how to deal with the very first choice in the last paragraph and assume from now that any step of the algorithm there is a vertex of degree at most two.

The first criterion is to execute basic reductions such that the obtained sequence can be grouped in a sequence of extended reductions as described above. This is justified by Proposition 3.10. This choice is always possible since all basic reductions from Figure 3.2, except 2.d, are special cases of extended reductions. In the case where any minimum degree vertex is the root of a basic reduction 2.d, the graph must be a disjoint union of cycles.

In this case we are able to execute the **ultimate** greedy algorithm so that its execution corresponds to a sequence of cycle reductions. This argument leads to Proposition 3.10.

**Proposition 3.10.** *For each subcubic graph with minimum degree at most two, it is always possible to execute one of the following (extended) reductions:*

Point - Edge - Path - Branching - Loop - Cycle - Even-backbone - Odd-backbone.

When several choices of reductions are possible, we want to pick in priority a reduction with positive potential. The **ultimate** greedy algorithm will have to select one with the **highest priority** extended reduction, according to the following **order** from the highest to lowest priority:

1. Point, Edge, Path, Branching,
2. Cycle or Loop,
3. Even-backbone,
4. Odd-backbone.

Any two reductions among the first group or any two reductions among the second group can be arbitrarily executed first, as soon as both have the minimum degree. We say that a reduction is a **priority reduction** if there exists no reduction in the same graph with strictly higher priority. Thus a priority reduction is any of the highest priority reductions in the current graph. One implication of this order is that when an Even-backbone is executed, it means that the current graph does not contain any degree one vertices, or any loop reduction. Additionally, when the priority reduction is Odd-backbone, the graph does not contain any Even-backbone. These structural observations will be useful later.

**Even-backbone rule.** Suppose that at some point of the algorithm the priority reduction is an Even-backbone. Unfortunately, it turns out that picking an arbitrarily one of these reductions can lead to a solution with poor approximation ratio. For instance, consider the graph $H_i$ in Figure 3.5, highly inspired by [31]. The difficulty here comes from the fact that executing an even-backbone reduction can split the graph into several connected components, each of them having a negative potential, thus implying an approximation ratio worse than 5/4. To address this issue, we want to make sure that we are able to "control" the potential of almost all of these connected components. In order to achieve a 5/4-approximation, we need to pick the right even-backbone reduction, and this can be done by applying the following Lemma.

For any reduction $\overline{R}$ in a graph $G$ we say that $\overline{R}$ *creates* connected components $H_1, \ldots, H_s$ if they are the connected components of the graph $G[V \setminus ground(\overline{R})]$. Intuitively, it suffices to execute an even-backbone reduction $\overline{R}$ such that all other even-backbone reductions are all present in the same connected component created by $\overline{R}$.

**Lemma 3.11.** *Let $G$ be a connected graph, with no degree one vertices, and no loop reduction. Let $\mathcal{B} = \{\overline{R}_1, \ldots, \overline{R}_p\}$ be the set of all even-backbone reductions in $G$. Each even-backbone reduction $\overline{R}_i$ has two root vertices, $r_i$ and $r'_i$. In the case when $\overline{R}_i$ has only one root, we set $r_i = r'_i$.*

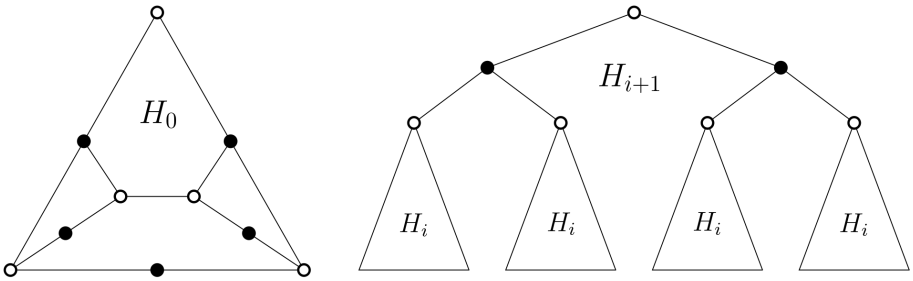



Figure 3.5: For any $i \geq 0$, the highest priority reduction in $H_i$ is Even-backbone, and picking recursively the top vertex leads to a solution where the approximation ratio tends to $17/13 > 5/4$ when $i$ tends to infinity.

*Then, there exists one even-backbone reduction, say $\overline{R}_1$, that satisfies the following property. Let $H_1, \ldots, H_t$ be the connected components created by $\overline{R}_1$, with $1 \leq t \leq 4$. Then either $t = 1$, or $t \geq 2$ and then the following is true. If there exist $r_i, r_j$ for some $i, j \geq 2$ and $i \neq j$, such that $r_i \in V(H_1)$ and $r_j \in V(H_2)$ (in words: $r_i$ and $r_j$ belong to two different connected components among $H_1, \ldots, H_t$), then at least one of $r_i, r_i', r_j, r_j'$ is a contact vertex of $\overline{R}_1$.*

*Proof.* (Lemma 3.11) Let $a \in V(G)$ be any degree three vertex. Consider a graph $\widetilde{G}$ obtained from $G$ by replacing each backbone from $r_i$ to $r_i'$ by a single degree two vertex which is also called $r_i$. On this *contracted* graph, let $d_{\widetilde{G}}(u, v)$ denote the shortest path distance (i.e., with minimum number of edges) between vertices $u, v \in V(\widetilde{G})$ in $\widetilde{G}$. Now, let us pick the root $r_i$ in $\widetilde{G}$ that has the largest distance $d_{\max} := \max_i d_{\widetilde{G}}(r_i, a)$ from $a$. Without loss of generality this is $r_1 : d_{\widetilde{G}}(r_1, a) = d_{\max}$. Denote by $H_1, \ldots, H_t$ the connected components created after executing the corresponding even-backbone reduction $\overline{R}_1$. At most one connected component, say $H_1$, contains $a$. Suppose that there is another connected component, i.e., $H_2$, that contains a vertex $r_j$. Any path from $r_j$ to $a$ intersects $ground(\overline{R}_1)$, including the shortest one, and $d_{\widetilde{G}}(r_j, a) = d_{\widetilde{G}}(r_1, a) = d_{\max}$. It follows that $r_j$ and $r_1$ have one common neighbor $b$, so that $d_{\widetilde{G}}(r_1, r_j) = 2$. In particular, in the original graph $G$, $r_1$ (or $r_1'$) is at distance two from $r_j$ (or $r_j'$) which means that this vertex is a contact vertex of $\overline{R}_1$. □

The above proof is constructive and lets us find the appropriate Even-backbone in time $O(|V|)$.

**Odd-backbone rule.** Suppose now that the priority reduction is the odd-backbone reduction. In this case, the **ultimate** greedy algorithm chooses the one that was created *latest*.

More formally, suppose that we are given a partial execution $\overline{R}_1, \ldots, \overline{R}_j$ in a graph $G$ such that the priority reduction in $G_j$ is an odd-backbone reduction, where $G_i$ is the subgraph of $G$ obtained from $G$ after the execution of $\overline{R}_1, \ldots, \overline{R}_i$, for $i = 1, \ldots, j$. We associate to each vertex $v$ of degree two a *creation time* $t_v \in \{0, \ldots, j-1\}$, such that $t_v$ is the greatest integer such that $v$ had degree three in $G_{t_v - 1}$. Moreover, if $v$ had already degree two in the original graph $G$, then we set $t_v = 0$. When $t_v \geq 1$, this means that $v$ was a contact vertex of $t_v$-th reduction. Then, the *creation time* of an (odd) backbone $B$ is

the greatest creation time over all vertices of degree two in $B$.

The **ultimate** greedy algorithm picks the odd-backbone reduction that has the greatest creation time, among all possible odd-backbone reductions. If several choices are possible, it can pick any of them.

*Remark.* We believe that this rule is not necessary in the sense that it does not improve the approximation ratio. However, this rule makes the algorithm easier to analyses. Intuitively, with this rule, we can not have several successive reductions with negative potential within the same connected component.

**Rule for cubic graphs.** When the input graph $G$ is cubic, we guess the first degree three vertex to pick, so that the potential of the associated execution is positive. By guessing, we mean choosing *any single fixed* vertex $u$ and then trying all *four* executions, each starting from a vertex in the closed neighborhood of vertex $u$. We show later that the first step can only increase the total potential of the whole sequence. When the graph is connected, all following reductions have degree at most two.

*Remark.* This special rule helps to guarantee that the approximation ratio of our algorithm is 5/4. One can show, using Lemma 3.15 that without this rule, i.e., picking initially any degree-three vertex can lead to a solution of size $\frac{4}{5}\alpha(G) - \frac{1}{5}$.

**Input:** A subcubic graph $G$.
**Output:** An independent set $S$ in $G$.
$S \leftarrow \emptyset$;
**if** *all vertices have degree three* **then**
  Let $u$ be any vertex;
  Execute *four* times the while loop just below, starting with $S = \{v\}$ and $G = G \setminus N_G[v]$, for all $v \in N_G[u]$ and output the maximum size solution;
**while** $G \neq \emptyset$ **do**
  **if** *the* priority reduction *(according to the* ***ultimate*** *greedy algorithm order) in* $G$ *is* Even-backbone **then**
    Let $\overline{R}$ be the Even-backbone given by Lemma 3.11 (Even-backbone rule);
  **if** *the* priority reduction *in* $G$ *is* Odd-backbone **then**
    Let $\overline{R}$ be the latest created Odd-backbone (Odd-backbone rule);
  **else**
    Let $\overline{R}$ be any priority reduction;
  Let $v_1, \ldots, v_s$ denote the roots of the basic reductions of $\overline{R}$;
  $S \leftarrow S \cup \{v_1, \ldots, v_s\}$;
  $G \leftarrow G \setminus ground(\overline{R})$
**return** $S$

**Algorithm 9:** the **ultimate** greedy algorithm.

It is clear that the set returned by Algorithm 9 is an independent set. In the next section, we will establish its approximation ratio.

## 3.5 Analysis of the ultimate approximation ratio

In this section, we prove that the **ultimate** greedy algorithm is a 5/4-approximation algorithm for the Maximum Independent Set problem in graphs with maximum degree at most three.

Let $\mathcal{E} = (\overline{R_1}, \ldots, \overline{R_\ell})$ be a sequence of extended reductions performed by the **ultimate** greedy algorithm on an input connected graph $G$. In order to analyze the approximation ratio of the **ultimate** greedy algorithm , we use our potential function in subcubic graphs ($\Delta = 3$) with parameters $\gamma, \sigma = 5, 4$. Given an independent set $I$ in $G$, the potential of an (extended) reduction $\overline{R}$ is

$$\Phi_I(\overline{R}) = 5 \cdot |\overline{R}| - 4 \cdot |I \cap ground(\overline{R})| + loan_I(\overline{R}) - debt_I(\overline{R}).$$

In Section 3.5.1 we present a lower bound on the potential of different extended reductions. As we have seen in Section 3.3.3, we are here able to avoid reduction with potential $-2$ (Figure 3.3 (a)) by grouping this basic reduction with the following ones, so that the resulting (extended) reduction, a cycle reduction, has now only potential $-1$. Unfortunately, we can not use the same trick to avoid reductions with potential $-1$. Moreover, such reductions: edge, (odd) cycle and odd-backbone reductions can not be avoided if we want to respect the original greedy constraint which is to pick a vertex with minimum degree.

In order to prove that the **ultimate** greedy algorithm delivers a 5/4-approximation, we show that the exact potential of any execution on a connected subcubic graph is non-negative. Unfortunately, it is not true that the potential of any execution is non-negative. For instance, picking the top vertex of $H_0$ in Figure 3.5 produces an execution with potential $-1$. We will prove that $-1$ is the worst value for the potential of any execution.

A potential problem arises when an execution creates a lot of connected components where each corresponding execution has negative potential. This could lead to an execution with arbitrarily negative potential. Such connected components can be created by reductions having many contact vertices, such as the even-backbone reduction. Our Even-backbone rule was designed to keep the potential of the created connected components under control, ensuring that at most one (or two) of them have negative potential.

This suggests that to solve our problem we could characterize the type of graphs that can have negative potential, i.e., for which there exists an execution with negative potential. Finding such a characterization is difficult but since we know which reduction's potential is $-1$, we are able to formulate a necessary condition together with a suitable induction hypothesis.

In Section 3.5.2, we describe such a class of graphs called *potentially problematic* graphs. Notice that 'potentially' refers to the potential function and at the same time to the fact that there exist some executions on some potentially problematic graphs with non-negative potential. The other potentially problematic case is when the first reduction of an execution has negative potential, namely the *bad* odd-backbone reduction. Based on these two potentially problematic cases, we present the inductive low-debt Lemma 3.15 that establishes that the potential of any execution is at least $-1$.

### 3.5.1 Potential of extended reductions

We start by looking at the potential of (extended) reductions.

**Claim 3.12.** *For any independent set $I$ we have the following potential estimates for the reductions:*

$$\begin{array}{rcl} \Phi_I(\textsf{Edge}) \geq -1 & \Phi_I(\textsf{Path}) \geq 0 & \Phi_I(\textsf{Point}) \geq 1 \\ \Phi_I(\textsf{Cycle}) \geq -1 & \Phi_I(\textsf{Loop}) \geq 0 & \Phi_I(\textsf{Branching}) \geq 1 \\ \Phi_I(\textsf{Odd-backbone}) \geq -1 & \Phi_I(\textsf{Even-backbone}) \geq 0 & \end{array}$$

For basic reductions, one can easily check by hand all possible cases. Notice that the worst case potential always arises when $I$ is maximum in the ground of a reduction. Figure 3.3 presents these worst cases for basic reductions: Edge, Cycle and Odd-backbone. Figure 3.7 shows the worst potential cases of the remaining basic reductions: Path, Even-backbone, Loop, Point, and Branching. From the worst case potential of basic reductions, we can lower-bound the potential of (extended) reductions.

*Proof.* (Claim 3.12) It remains to prove lower-bounds for reductions of arbitrary size. See Figure 3.6. An odd-backbone reduction is a sequence of basic reductions starting with 2.e (in Figure 3.2), which has a potential at least $-1$ (Figure 3.3 (d)), and a certain number of path reductions, with potential at least zero (Figure 3.7 (a) and (b)), so that the total potential is at least $-1$. More generally, the potential of an extended reduction is lower-bounded by the sum of the potentials of the first and the last basic reduction. For Even-backbone with non-adjacent backbone end-points, these first and last basic reductions are 2.e ($\Phi \geq -1$) and Branching ($\Phi \geq 1$, see Figure 3.7 (g) and (h)) so that the sum is non-negative.

Consider now a cycle reduction $\overline{R}$ of length $n \geq 3$. Let us denote by $b$ and $w$, respectively, the number of black and white vertices, i.e., $b = |I \cap \mathit{ground}(\overline{R})|$ and $b + w = n$. Since **Greedy** is optimal in degree at most two graphs and the size of $I$ is at most $\left\lfloor \frac{n}{2} \right\rfloor$, we have:

$$\Phi_I(\overline{R}) = 5\left\lfloor \frac{n}{2} \right\rfloor - 4b - w = 5\left\lfloor \frac{n}{2} \right\rfloor - 3b - n \geq 5\left\lfloor \frac{n}{2} \right\rfloor - 3\left\lfloor \frac{n}{2} \right\rfloor - n \geq -1. \tag{3.2}$$

For Loop and Even-backbone with adjacent end-points, simply observe that their ground is a cycle with one or two additional edges. Each of these edges is either a debt edge — the loan increases by one — or the corresponding middle white vertex has now degree three — the debt decreases by one. In any case, the potential increases by the number of added edges, so that we proved what we wanted. □

Notice that the worst potential of Cycle and Loop depends on the length of the ground and the worst case corresponds to odd-length cycles. Moreover, notice that when its two end-points are adjacent, the potential of an Even-backbone is at least one.

### 3.5.2 The inductive low-debt Lemma

Given a graph $G$ and an independent set $I$ in $G$, recall that a vertex $v \in V(G) \cap I$ from $I$ is *black*, and *white* otherwise. We say that a backbone $B = \{w, v_1, \ldots, v_b, w'\}$ is **alternating for $I$** (or simply **alternating**) when $v_i$ is black (or white) if and only if $i$ is even. See Figure 3.8 for an example of an odd alternating backbone. Notice that there is no restriction on the types of the end-points of the backbone.

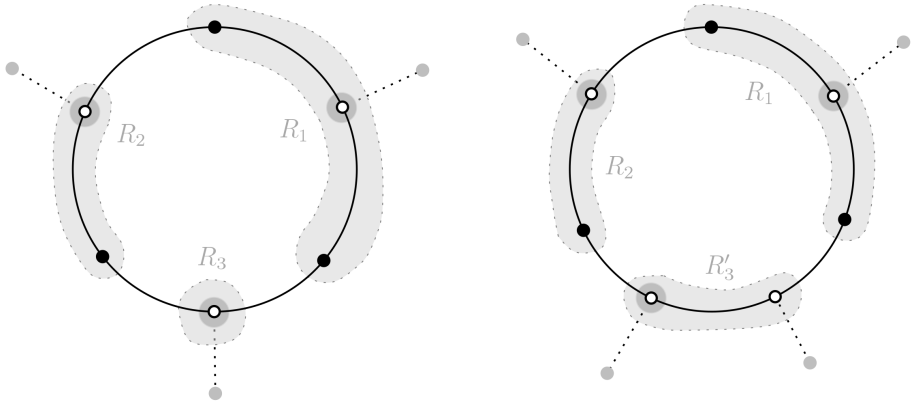



Figure 3.6: A *cycle* reduction. On the left, a even cycle with potential $\Phi_I(R_1, R_2, R_3) = -2+0+2 = 0$ and on the right a odd cycle with potential $-1$. Vertices surrounded by a grey disk are the ones picked by the algorithm and dotted edges are debt edges.

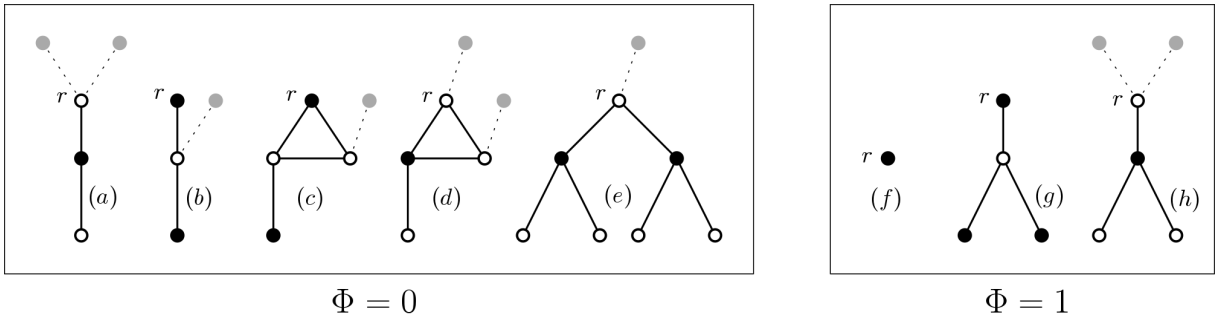



Figure 3.7: Basic reductions with worst potential equal to 0 or 1. Dotted edges translate the debt of each white vertex. Grey vertex can either be black or white.

The next definition of the white and black type reductions is crucial to our proof. Intuitively, these reductions will have the greedy priority strictly higher than that of an odd-backbone reduction, and in many cases, also higher than that of an even-backbone reduction. However we must be careful here because what will matter will be in some sense a "parity" of the reductions. Namely, the first observation is that the potential of an odd-backbone reduction, let us call it $\overline{R}$, can be $-1$ in the case when all its contact vertices are white (see Figure 3.3 (d)). Therefore when such a reduction $\overline{R}$ is executed first, then we need to "pay" for it by showing that the potential of the following reductions in the connected components it creates will in total be zero (as this is the only way of keeping the total value of the potential to be at least $-1$). What we now want is that if a potential of a connected component $H$ created by $\overline{R}$ is negative, then because $\overline{R}$ has only white contact vertices in $H$, even *before* $\overline{R}$ is executed, $H$ must contain a reduction with higher priority than $\overline{R}$, thus leading to a contradiction. Our definition below will ensure this by guaranteeing that such a reduction in $H$ exists with a black root vertex in $H$ by $\overline{R}$'s contact edges.

A similar while high priority reduction is also needed in $H$. Imagine namely that the first executed reduction $\overline{R}$, that created component $H$, is also an odd-backbone reduction but its contact vertices are all black. Then they will "block" the black vertices in $H$. Our definition below still guarantees an existence of a reduction in $H$, before executing $\overline{R}$, which has a white root vertex and has priority strictly higher than $\overline{R}$.

The third possibility is when $\overline{R}$ is an odd-backbone reduction with a white and black contact vertex. This is not a problem because the potential of such a reduction is non-negative or even positive.

It turns out that we can deal with $\overline{R}$ being an even-backbone reduction in a different way. There are some further technicalities and details and they can be read in the details of our full proof.

**Definition 3.13** (Black and white type reductions)**.** Given a graph $G$ and an independent set $I$ in $G$, we define the **black** or **white type** reductions in $G$ by the following rules:
**(1)** Any path reduction or branching reduction in $G$ with black root and white middle vertex (*resp.* with white root vertex and black middle vertex) is a black (*resp.* white) type reduction.
**(2)** Any Loop reduction $\overline{R}$ in $G$, where $I \cap ground(\overline{R})$ is a maximum independent set in $ground(\overline{R})$, whose both root vertices are white (*resp.* whose at least one root vertex is black) is called a white (*resp.* black) type reduction.
**(3)** Any even-backbone reduction $\overline{R}$ in $G$, with an alternating backbone, whose both root vertices are white (*resp.* black) is called a white (*resp.* black) type reduction.

We note here that the black and white reductions correspond to the worst case of the potential (see Figures 3.3 and 3.7). We also say that an an odd-backbone reduction $\overline{R}$ is **bad for** $I$ (or simply **bad**) if $\Phi_I(\overline{R}) = -1$, see Figure 3.3 (d). More generally, given an independent set $I$, we will say that a reduction $\overline{R}$ is **bad** when its potential is minimized by $I$, i.e., $\Phi_I(\overline{R}) = \min\{\Phi_{I'}(\overline{R}) : I' \text{ independent set}\}$.

**Definition 3.14.** Let $G$ be a connected graph with minimum degree at most 2 and $I$ an independent set in $G$. We say that $G$ is **potentially problematic for** $I$ (or just **potentially problematic**), if either:

**(1)** $G$ is an odd cycle or an edge and $I$ is maximum in $G$.
**(2)** or, there exists one reduction of black type *and* one reduction of white type in $G$.

Notice that Cycle (and also Edge) reduction has potential $-1$ if and only if its ground has odd-length and $I$ is maximum (Claim 3.17). In this situation, we will call these graphs *bad* cycle and *bad* edge.

The following Lemma states that any execution has a potential always at least $-1$. We prove this result by induction together with a necessary condition characterizing executions with minimum potential.

**Lemma 3.15** (Inductive low-debt Lemma)**.** *Let $G$ be a connected graph with minimum degree at most two, $I$ an independent set in $G$ and $\mathcal{E} = (\overline{R}_1, \ldots, \overline{R}_\kappa)$ an execution (of the **ultimate** greedy algorithm) on $G$. Then*

*(1)* $\Phi_I(\mathcal{E}) \geq -1$

*(2) If* $\Phi_I(\mathcal{E}) = -1$*, then*

*(a) either the first reduction $\overline{R}_1$ is a bad odd-backbone reduction,*

*(b) or $G$ is potentially problematic.*

### 3.5.3 Proof of Theorem 3.3

We now have the material needed to prove that the **ultimate** greedy algorithm achieves a 5/4-approximation in subcubic graphs. Let $G$ be a connected subcubic graph, $I$ an independent set in $G$, and $\mathcal{E}$ an execution. Our goal is to show that the *exact* potential is non-negative :

$$\Phi_{G,I}(\mathcal{E}) \geq 0 \tag{3.3}$$

Suppose this is true. Then from Proposition 3.6 we have that $5|\mathcal{E}| - 4|I| \geq 0$, which can be re-written as $\frac{|I|}{|\mathcal{E}|} \leq \frac{5}{4}$, and because this is true for any independent set, we have established the desired approximation, thus proving Theorem 3.3. There are two cases depending on whether in input graph is cubic or not.

*Case 1:* The input graph has at least one vertex with degree at most 2. With Lemma 3.15, we know that $\Phi_I(\mathcal{E}) \geq -1$. Suppose that the inequality is tight. Then, the first reduction is a bad odd-backbone reduction or $G$ is potentially problematic. In any case there exists in $G$ a white vertex with degree at most two — that is the root of the first Odd-backbone or of the white reduction in $G$ — so that, using Claim 3.7 we have:

$$\Phi_{G,I}(\mathcal{E}) = \Phi_I(\mathcal{E}) + \sum_{v \notin I}(3 - d_G(v)) \geq -1 + 1 = 0.$$

*Case 2:* all vertices in $G$ have degree three. Assume that $I$ is maximal, so that for any vertex $u$, $|N_G[u] \cap I| \geq 1$. Then, let us consider any degree 3 vertex $u$. One of the four executions of the algorithm will be execution with $v \in N_G[u]$ being black, and let us call this first reduction $R^*$. We detect which of those four executions to consider by taking the one that leads to the greatest size solution. By Claim 3.21 it implies that this execution has the largest potential.

On the other hand, let us consider the execution $\mathcal{E}$ of $R^*$ with its black root $v$. Let $H_1, \ldots, H_s$ denote the connected components created by $R^*$, and $\mathcal{E}_1, \ldots, \mathcal{E}_s$ the corresponding executions. Without loss of generality we have $\mathcal{E} = (R^*, \mathcal{E}_1, \ldots, \mathcal{E}_s)$. To prove that the

exact potential of $\mathcal{E}$ in $G$ is positive, the trick is to consider that the first reduction $R^*$ does not use any loan from its loan edges, so that its potential is exactly 1. This implies also that each connected component will not have any debt edges. Then as we proved before, since each connected component $H_i$ has a vertex with degree at most two, its exact potential in $H_i$ is non-negative. Therefore, we have

$$\Phi_{G,I}(\mathcal{E}) = 1 + \sum_{i=1}^{s} \Phi_{H_i,I}(\mathcal{E}_i) \geq 1$$

which concludes the proof of Theorem 3.3.

*Remark.* Our analysis actually implies that the size of the solution has size at least $(4/5)\mathbf{OPT} + 1/5$. Moreover, if the first reduction $\overline{R}$ is a bad even-backbone reduction, and $G$ is not a problematic graph, then our analysis proves that the **ultimate** greedy solution's size is at least $(4/5)\mathbf{OPT} + 1/5$. This precisely matches the size of the lower bound example of Halldórsson and Yoshihara in Figure 3.1. Our analysis even indicates that this lower bound example has the worst possible approximation ratio for any ultimate greedy algorithm. Indeed, this counter example is actually a sequence of graphs $(G_n)_n$, and the solution returned by any refinement of the **min-degree** greedy algorithm has size $(4/5)\mathbf{OPT}(G_n) + 1/5$, and therefore the corresponding approximation ratio tends to $5/4$ when the size of $G_n$ tends to infinity. However, our previous analysis indicates that there is no (finite) graph $G$, such that the **min-degree** greedy algorithm produces a solution of size exactly $(4/5)\mathbf{OPT}(G)$.

Indeed, this graph must satisfy (2a) or (2b) from Lemma 3.15, otherwise our the **ultimate** greedy algorithm outputs a solution of size at least $(4/5)\mathbf{OPT}(G) + 1/5$. Then, for any maximum independent set, this graph must have at least one *black* minimum degree vertex, and in this situation, we could for instance try all possible minimum degree vertices (only for the first step), and pick the execution of maximum size. This modified greedy algorithm returns a solution at least $(4/5)\mathbf{OPT}(G) + 1/5$, for any input graph $G$.

*Remark.* We can obtain a greedy algorithm with an approximation ratio of $\frac{4}{3}$ but with linear running time. Observe that if we set $\gamma = 4$ and $\sigma = 3$, the minimum potential of an odd-backbone reduction changes from $-1$ to 0. Now, only Cycle has potential value of $-1$. With this observation, we are able to extend Loop in a particular way, which implies, finally, that we are able to exclude even-backbone reductions from the definition of black and white type reductions, and also preserve the induction hypothesis in our inductive proof. It implies that the Even-backbone rule is not necessarily needed. Without this rule this greedy algorithm can be optimized to run in linear time.

## 3.6 Proof of the inductive low-debt Lemma

In this section we prove the Inductive Low-debt Lemma:

**Lemma 3.15** (Inductive low-debt Lemma)**.** *Let $G$ be a connected graph with minimum degree at most two, $I$ an independent set in $G$ and $\mathcal{E} = (\overline{R}_1, \ldots, \overline{R}_\kappa)$ an execution (of the **ultimate** greedy algorithm) on $G$. Then*

*(1)* $\Phi_I(\mathcal{E}) \geq -1$

*(2) If* $\Phi_I(\mathcal{E}) = -1$*, then*

*(a) either the first reduction $\overline{R}_1$ is a bad odd-backbone reduction,*

*(b) or $G$ is potentially problematic.*

Along the proof we will refer to several technical claims about extended reductions proved in Section 3.6.1.

We prove the Inductive low-debt Lemma by induction on the number $\kappa$ of extended reductions in the execution $\mathcal{E}$. First, if $\kappa = 1$, then since $G$ is connected, the reduction is a terminal reduction, i.e., point, edge or cycle reduction, and by Claim 3.12 we know that their potential is at least $-1$. Moreover, if the potential is exactly $-1$, it is not difficult to see from the proof of Claim 3.12 that the reduction is either an edge or odd cycle reduction, which are potentially problematic graphs. For a detailed proof of this fact see Claim 3.17.

Suppose now that $\mathcal{E}$ consists of $\kappa \geq 2$ extended reductions. We will treat all cases depending on the first extended reduction $\overline{R}_1$. We denote its root and the contact vertices by letters $r$ and $c_i$, with $1 \leq i \leq 4$, respectively. In all these cases we will apply the induction hypothesis to each connected component of the graph after executing the first reduction. Recall that we say that $\overline{R}_1$ *creates* connected components $H_1, \ldots, H_s$ if they are connected components of the graph $G[V \setminus \mathit{ground}(\overline{R}_1)]$. Here, $s$ with $1 \leq s \leq 4$, denotes the number of connected components created by $\overline{R}_1$. Reductions executed by the **ultimate** greedy algorithm in distinct connected components are independent. Therefore, without loss of generality we can assume that each execution on $H_i$ corresponds to a sub-execution $\mathcal{E}_i$ of $\mathcal{E}$ so that $\mathcal{E} = (\overline{R}_1, \mathcal{E}_1, \ldots, \mathcal{E}_s)$. Notice that according to Proposition 3.6, we have

$$\Phi_I(\mathcal{E}) = \Phi_I(\overline{R}_1) + \Phi_I(\mathcal{E}_1) + \cdots + \Phi_I(\mathcal{E}_s).$$

We establish hypothesis (1) and (2) in three separated cases depending on whether the first reduction is (1) Path, Branching or Loop, (2) even-backbone or (3) odd-backbone.

**Case (1): $\overline{R}_1$ is Path, Branching or Loop.** These are easy cases because the number of created connected components (and therefore the potential of each corresponding execution) is always balanced by the potential of the reduction. This is precisely captured in the following fact that can be easily verified thanks to Figure 3.2 and Claim 3.12.

For any independent set $I$, and any reduction $\overline{R}$ that is Path, Branching or Loop:

$$\Phi_I(\overline{R}) \geq |\mathit{contact}(\overline{R})| - 1.$$

*Remark.* The inequality is also valid for an even-backbone reduction whose both end-points are adjacent.

Then, if $\overline{R}_1$ is a path, branching, or loop reduction then the number of connected components is at most the number of contact-edges, therefore by the induction hypothesis (1) on each connected component of $G[V \setminus \mathit{ground}(\overline{R}_1)]$, we have

$$\begin{aligned}\Phi_I(\mathcal{E}) &= \Phi_I(\overline{R}_1) + \Phi_I(\mathcal{E}_1) + \cdots + \Phi_I(\mathcal{E}_s) \\ &\geq \Phi_I(\overline{R}_1) + (-1)s \geq \Phi_I(\overline{R}_1) + (-1)|\mathit{contact}(\overline{R}_1)| \geq -1.\end{aligned}$$

Moreover, if $\Phi_I(\mathcal{E}) = -1$, then all these inequalities are tight and in particular, the potential of $\overline{R}_1$ is minimum. This implies that $\overline{R}_1$ must be a reduction of black or white type (see Claim 3.18), and additionally, it must create exactly $|\mathit{contact}(\overline{R}_1)|$ connected components, and each one must have potential $-1$. Applying the inductive assumption (2) to these connected components, together with the following Claim 3.16 implies property (2b) for $G$.

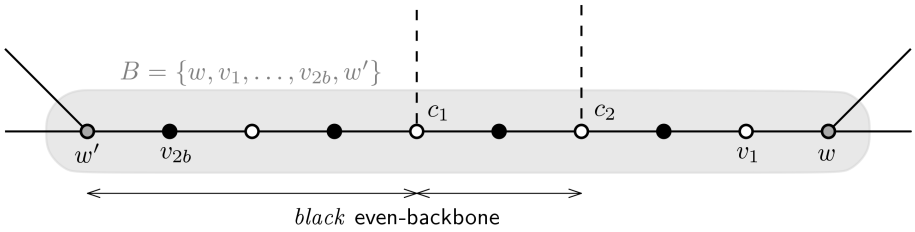



Figure 3.8: Existence of a black reduction in proof of Claim 3.16. $B$ is an alternating odd backbone in $H$. Grey end-points illustrate the fact that these vertices can either be black or white. Dashed edges are contact edges from $\overline{R}_1$. Before the execution, there exists a black (alternating) Even-backbone in $G$ between $w', c_1$ and $c_1, c_2$.

**Claim 3.16.** *Let $G$ be a connected graph, and $I$ an independent set in $G$. Consider an execution $\mathcal{E} = (\overline{R}_1, \ldots, \overline{R}_\kappa)$ in $G$. Let $H$ be a connected component created by the first reduction $\overline{R}_1$, such that all contact vertices of $\overline{R}_1$ that are in $H$ are all white (*resp. *all black). Then, if $H$ is potentially problematic* or *if the first reduction executed in $H$ is a bad odd-backbone reduction, then there exists a black (*resp. *white) reduction $\overline{R}$ in $G$ such that ground$(\overline{R}) \subseteq V(H)$.*

We will prove that in this situation, $\overline{R}_1$ can not be an Odd-backbone, since this would not be the priority reduction according to the **ultimate** greedy algorithm order. This claim is useful in the sense that, when the potential of $\overline{R}_1$ is minimum then $\overline{R}_1$ is white (or black) and its contact vertices are all white (or all black) (Claims 3.17,3.18,3.19), so that $G$ is a potentially problematic graph.

*Proof.* (Claim 3.16) Suppose that $H$ is created by $\overline{R}_1$ with contact vertices $c_1, \ldots, c_t \in V(H)$ that are all white (*resp.* all black), with $1 \leq t \leq 4$. We show that there exists one black (*resp.* white) reduction $\overline{R}$ in $G$ such that *ground*$(\overline{R}) \subseteq V(H)$.

First, assume that the first executed reduction in $H$, say $\overline{R}_2$, is a bad odd-backbone reduction. Denote by $B = \{w, v_1, \ldots, v_{2b}, w'\}$ its backbone, with end-points $w$ and $w'$. Since $\Phi_I(\overline{R}_2) = -1$, its backbone is alternating (Claim 3.20) and without loss of generality we assume that $v_j$ is white if and only if $j$ is odd.

According to the Odd-backbone-rule, at least one of the contact vertices $c_i$ must be one vertex of its backbone. Since all $c_i$ are white (*resp.* black), the distance along $B$ between any pair of $\{c_1, \ldots, c_t, w'\} \cap B$ (*resp.* $\{w, c_1, \ldots, c_t\} \cap B$) is even. Therefore, there is an alternating even-length-backbone between any two consecutive ones. This implies the existence of a black (*resp.* white) even-backbone reduction in $G$, see Figure 3.8.

Now, assume that $H$ is a potentially problematic graph, and suppose first that there is one black (*resp.* white) reduction $\overline{R}_2$ with a black (*resp.* white) root vertex $r$ in $H$. Notice that this vertex is distinct than all $c_i$, and then has the same degree in $G$. Then, if $d_H(r) = d_G(r) = 1$, then $r$ is also the root of a black (*resp.* white) reduction in $G$. Then, suppose that $d_H(r) = d_G(r) = 2$.

If $r$ is the root of a black (*resp.* white) even-backbone reduction $\overline{R}_2$ in $H$, with backbone end-points $w$ and $w'$, then any two consecutive vertices of the set $\{w, w', c_1, \ldots, c_t\}$ along this backbone form an alternating even-length-backbone. In particular, $r$ is the root of a black (*resp.* white) Even-backbone in $G$.

The case when $\overline{R}_2$ is a Loop is similar, but slightly more technical. First, if there is no $c_i$ in the ground of $\overline{R}_2$, the root of this reduction is obviously also the root of an

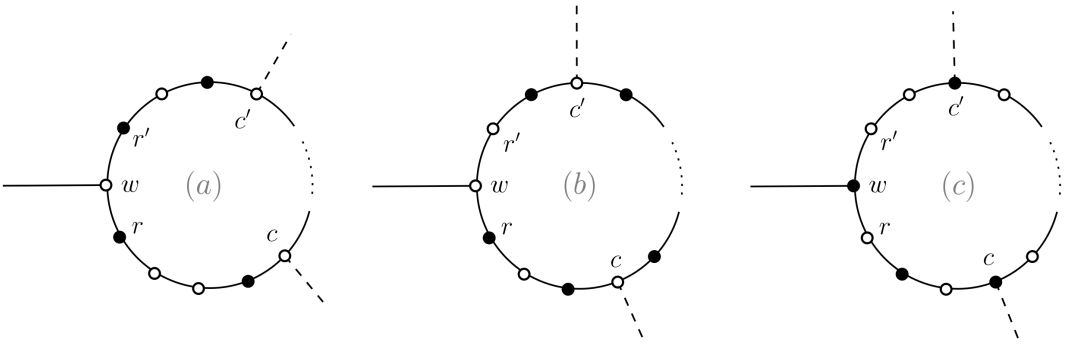



Figure 3.9: Different types of black, $(a)$ and $(b)$, and white Loop $(c)$. Dashed edges are contact edges.

black (*resp.* white) Loop in $G$. Now, suppose that there is at least one contact vertex $c_i$ in $ground(\overline{R}_2)$. Let $w$ be the vertex of degree three in $ground(\overline{R}_2)$, and $r, r'$ its two neighbors. Let us focus on the first two contact vertices $c$ and $c'$ that are met when we sweep the loop from $w$ in each direction (or just $c$ is the only present contact vertex). We claim that at least one of $r$ or $r'$ is the black (*resp.* white) root of an (alternating) black (*resp.* white) Even-backbone in $G$. See Figure 3.9 (a) and (b) (*resp.* (c)). Form left to right on Figure 3.9 this vertex is respectively $r'$, $r$ and $r$.

We now turn our attention to the case when $H$ is a bad[3] Edge or Cycle. If $H$ is a bad Edge, then its black (*resp.* white) vertex has degree one in $G$, and therefore is the root of black (*resp.* white) path or branching reduction in $G$. Similarly to the case when $\overline{R}_2$ is a Loop, if $H$ is a bad Cycle, then there exists a black (*resp.* white) vertex $r' \in V(H)$ that is the root of a black (*resp.* white) Loop in $G$, when $\overline{R}_1$ has one contact vertex in $H$, or the root of a black (*resp.* white) Even-backbone in $G$, when $\overline{R}_1$ has more than one contact vertex in $H$.

□

We now turn our attention to the case when $\overline{R}_1$ is a backbone reduction.

**Case (2): $\overline{R}_1$ is an even-backbone reduction.** If the first reduction executed in $G$ is an Even-backbone, it means, according to the greedy order, that graph $G$ does not contain any degree one vertices or any loop reductions. All degree two vertices are contained in some backbones linking two distinct degree three vertices. If the end-points of the backbone of $\overline{R}_1$ are adjacent, then $\overline{R}_1$ satisfies Observation 3.6, so that this case was treated in the previous section. From now on, let us assume that these end-points are independent. In the following, we use the same terminology as in Lemma 3.11.

- If all contact vertices of $\overline{R}_1$ are white (*resp.* black), then at most one connected component created by $\overline{R}_1$ has potential $-1$. Indeed, suppose for a contradiction, that $H_i$, with $i \geq 2$, has potential $-1$. By the induction hypothesis, each $H_i$ must satisfy assumptions (2a) or (2b) of Lemma 3.15. According to Claim 3.16, there was a black (*resp.* white) reduction in $H_i$ before $\overline{R}_1$ was executed. This reduction is neither a degree one nor a loop reduction, so it must be an even-backbone reduction. Because these reductions are black (*resp.* white), the root vertices of these reductions are

[3]meaning here that $I$ is maximum in $H$.

distinct from $\overline{R}_1$'s contact vertices, which contradicts Lemma 3.11. We proved (1) when all contact vertices of $\overline{R}_1$ are all white (*resp.* black).

For (2), if $\Phi_I(\mathcal{E}) = -1$, then one created connected component, say $H_1$, has potential $-1$, which by the induction hypothesis satisfies (2a) or (2b) and $\Phi_I(\overline{R}_1) = 0$, so that $\overline{R}_1$ is a white reduction with only white contact vertices (Claim 3.19). Claim 3.16 guarantees the existence of a black reduction in $G$, so that $G$ is potentially problematic.

- If some of $\overline{R}_1$'s contact vertices are both black and white, then we argue that $\Phi_I(\mathcal{E}) \geq 0$. First, the potential of $\overline{R}_1$ is at least two[4] (Claim 3.19). Therefore we should argue that there are *at most two* connected components with potential $-1$. This is true because there are at most two connected components with strictly more than one contact vertex, and at most one connected component with exactly one contact vertex can have potential $-1$. Indeed, for connected components with only one contact vertex, Claim 3.16 applies so that we can use the same argumentation as in the previous paragraph.

**case (3): $\overline{R}_1$ is an odd-backbone reduction.**

- Assume first that $\overline{R}_1$ has potential $\Phi_I(\overline{R}_1) = -1$ (*resp.* $\Phi_I(\overline{R}_1) = 0$). Then, Claim 3.20 indicates that all its contact vertices are white (*resp.* black).

  (1) We prove that all connected components created by $\overline{R}_1$ have potential at least zero. Assume that it is not true for the component $H_1$. By the induction hypothesis, it satisfies (2a) or (2b). By Claim 3.16, there exists a black (*resp.* white) reduction in $H_1$ before $\overline{R}_1$ is executed which contradicts the greedy order, because any black or white reduction has a strictly higher priority than that of an Odd-backbone.

  (2) When $\overline{R}_1$ is supposed to be a bad Odd-backbone by assumption, (2a) is always true, and otherwise, if $\Phi_I(\overline{R}_1) = 0$, we just proved that $\Phi_I(\mathcal{E}) \geq 0$.

- Suppose that $\Phi_I(\overline{R}_1) \geq 1$. We claim that at most one component created by $\overline{R}_1$ can have potential $-1$. First note that $\overline{R}_1$ has three contact edges and thus at most three contact vertices. Indeed, at most one created connected component has at least two contact vertices, and no created connected component $H$ with exactly one contact vertex can have potential $-1$, because Claim 3.16 would imply the existence of a strictly higher priority reduction in $H$.

This concludes the proof of Lemma 3.15.

### 3.6.1 Technical claims

**Claim 3.17.** *Given a connected graph $G$, where any execution consists of* one *extended reduction $\overline{R}$, i.e., Point, Edge or Cycle, and if $\Phi_I(\overline{R}) = -1$, for a given independent set $I$ in $G$, then $G$ is either a bad odd-length cycle, or a bad Edge, and $I$ is* maximum *in $G$.*

*Proof.* First, by Claim 3.12, if $\overline{R}$ is a Point then $\Phi_I(G) \geq 1$. Then, if it is an edge reduction, since it is a basic reduction, it is easy to check that $\Phi_I(\overline{R}) = -1$ only when one vertex is black, i.e., $I$ is maximum, see Figure 3.3($b$).

[4] We assume here that the two end-points of the corresponding backbone are not adjacent.

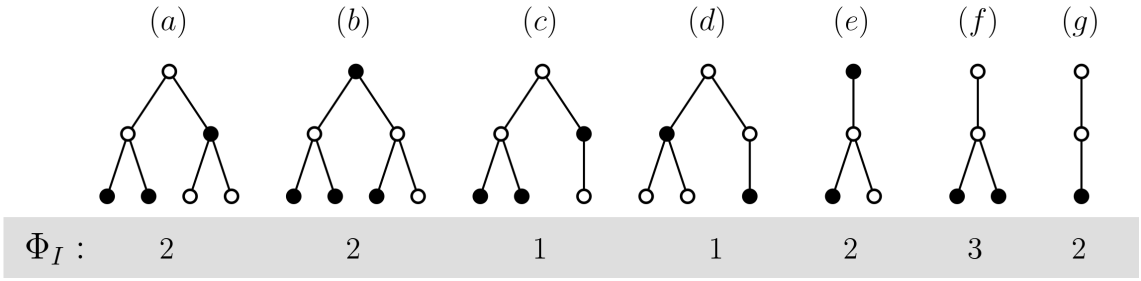



Figure 3.10: Some examples of good case potential value.

Finally, when $G$ is a cycle then, the corresponding cycle reduction has potential $-1$ when all inequalities from Equation (3.2) in the proof of Claim 3.12 are tight. In particular, the number $b$ of black vertices must be maximum, and $\lfloor \frac{n}{2} \rfloor - \lceil \frac{n}{2} \rceil = 1$, that only arises when the length $n$ of the cycle is odd. □

**Claim 3.18.** *Any Path, Branching, or Loop with minimum potential have type black (or white,* resp.*), and have contact vertices that are all black (all white,* resp.*).*

*Proof.* For (basic) path and branching reductions, with the worst case potential (Figure 3.7 (a),(h) — *resp.* (b),(g)) are white (*resp.* black) reductions, with white (*resp.* black) contact vertices.

Furthermore, as noticed before, the potential of a Loop, is correlated to the potential of the cycle reduction obtained by removing its contact edge. That is, because adding an edge to a ground of the reduction either increases by one the loan or decreases by one the debt, so that the potential increases by one or by two. In particular, when Loop has minimum potential, the corresponding cycle has also minimum potential. By Claim 3.17, we know that the independent set must be maximum, and its backbone must have odd-length, so that the loop reduction must have type black or white. Moreover, when the potential is minimum, the contact edge can not be incident to two white vertices, as otherwise, making the contact vertex black would decrease the potential. Therefore, when $\Phi_I(\overline{R}) = -1$, the contact vertex has the same type as the reduction. □

**Claim 3.19.** *Let $\overline{R}$ be Even-backbone, then for each independent set $I$ we have $\Phi_I(\overline{R}) \geq 0$, and*

1. *if $\Phi_I(\overline{R}) = 0$, then $\overline{R}$ is a white reduction with only white contact vertices.*
2. *if the end-points of $\overline{R}$'s backbone are not adjacent, and if its contact vertices are not all white or all black, then $\Phi_I(\overline{R}) \geq 2$.*

*Proof.* 1. Suppose that the potential of an Even-backbone $\overline{R} = (R_1, \ldots, R_t)$ is minimum : $\Phi_I(\overline{R}) = 0$. We know that its end-points are not adjacent, as otherwise it has potential at least one, and the potential of all basic reductions $R_i$ must be minimum. When $\overline{R}$ has size one, the worst case arises only when $I$ is such as in Figure 3.7(e), that is, a particular case of white reduction with white contact vertices. For greater sizes, i.e., $t \geq 2$, this implies that the first basic reduction $R_1$ is like Figure 3.3(d). Then, $R_2$ must be a Path with minimum potential and white root, like in Figure 3.7(a), and so on for all path reductions. The final branching reduction $R_t$ has

minimum potential and white root, as in Figure 3.7(h). Finally, we proved that this backbone is alternating and the root is white, so that $\overline{R}$ is a white reduction. Moreover, all its contact vertices must be white.

2. In the following, we will say that a reduction is *mixed* is it has two different type contact vertices. When $\overline{R}$ has size one and is mixed, we can easily check by hand, that its potential is at least two (Figure 3.10(a),(b)). Consider now a mixed even-backbone reduction $\overline{R} = (R_1, \ldots, R_t)$, and without loss of generality[5], assume that the last branching reduction $R_t$ has at least one black contact vertex.

   First, if $R_1$ or $R_t$ are mixed, then their potential is respectively at least 1 and 2 (see Figure 3.10 (c),(d) and (e)) so that $\Phi_I(\overline{R}) \geq 2$.

   Otherwise, assume that $R_1$ and $R_t$ have only respectively white and black contact vertices. In particular, the root $r$ of the first Path $R_2$ is white. If the root $r'$ of the last branching reduction is white then[6], its potential is at least 3 (Figure 3.10(f)), so that $\Phi_I(\overline{R}) \geq 2$. Then, if $r'$ is black, since the distance between $r$ and $r'$ is even, we must find a Path $R_i$ with both root and middle white vertices. Such a reduction has potential at least 2 (Figure 3.10(g)), so that $\Phi_I(\overline{R}) \geq 2$.

□

**Claim 3.20.** *Let $\overline{R}$ be Odd-backbone, then for each independent set $I$ we have $\Phi_I(\overline{R}) \geq -1$, and*

1. *if $\Phi_I(\overline{R}) = -1$, then it has an alternating backbone, white root and only white contact vertices.*

2. *if $\Phi_I(\overline{R}) = 0$, then it has an alternating backbone, black root and only black contact vertices.*

*Proof.* These facts can easily be checked by hand for odd-backbone reduction $\overline{R}$ of size one, see Figure 3.3(d) and Figure 3.10(h). Suppose now that $\overline{R} = (R_1, \ldots, R_t)$ has size at least two, where $R_1$ is the basic odd-backbone reduction, and $R_i$ are path reductions, for $i \geq 2$.

1. If $\Phi_I(\overline{R}) = -1$, then necessarily, $\Phi_I(R_1) = -1$ and for all Path, $\Phi_I(R_i) = 0$. The first reduction has only white contact vertices (Figure 3.3(d)), and then $R_2$ has white root, so it must be as in Figure 3.7(a). In particular, it must have a white contact vertex, so that $R_3$ has white root, and so on and so forth. This implies that the corresponding backbone is alternating, and the root vertex and all contact vertices are white.

2. If $\Phi_I(\overline{R}) = 0$, then either **case 1 :** all basic reductions have potential zero, or **case 2 :** $\Phi_I(R_1) = -1$ and there is one Path $R_j$ with potential one.

   **case 1 :** $R_1$ must be like in Figure 3.10(h), and using exactly the same argument as in the previous paragraph, we show that all following path reductions are like in Figure 3.7(b), so that $\overline{R}$'s backbone is alternating, $\overline{R}$ has black root and contact vertices.

[5] Recall that we are free to choose the root of Even-backbone.

[6] We assume here that at least one contact vertex is black.

**case 2 :** we show that this case is impossible. Indeed, as before all path reductions $R_i$, with $i < j$ have a white contact vertex (Figure 3.7(a)), so that $R_j$ has white root. Since it has potential one, its middle vertex must be white (otherwise its potential is zero, see Figure 3.7(a)), and in this case its potential is at least two (Figure 3.10(g)).

□

**Claim 3.21.** *Let $G$ be a connected graph and $I$ an independent set in $G$. Let $\mathcal{E} = (\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_\kappa)$ be an execution of greedy on $G$, and let this execution produce a solution of size $s_r$. Let $\mathcal{E}' = (\overline{R}'_1, \overline{R}'_2, \ldots, \overline{R}'_{\kappa'})$ correspond to another execution of greedy on $G$ producing a solution of size $s_{r'}$. Then if $s_r > s_{r'}$ then $\Phi_I(\mathcal{E}) > \Phi_I(\mathcal{E}')$, and otherwise, if $s_r < s_{r'}$ then $\Phi_I(\mathcal{E}') > \Phi_I(\mathcal{E})$.*

*Proof.* We know that $\Phi_I(\mathcal{E}) = 5s_r - 4|I \cap V(G)|$, and $\Phi_I(\mathcal{E}') = 5s_{r'} - 4|I \cap V(G)|$. Therefore, $\Phi_I(\mathcal{E}) < \Phi_I(\mathcal{E}')$ if and only if $s_r < s_{r'}$. □

## 3.7 Limits of the greedy approach.

In the section we focus on the limitations of the greedy approach for finding large independent sets. Given a graph $G = (V, E)$, we say that $I$ is a **greedy set** of $G$, if $I$ is an independent set, and its elements can be ordered, $I = \{v_1, \ldots, v_k\}$ in such a way that, for all $1 \leq i \leq k$, the vertex $v_i$ has minimum degree in the subgraph $G_i$, where $G_i := G[V \setminus N_G[\{v_1, \ldots, v_{i-1}\}]]$. The size a maximum (*resp.* minimum) greedy set in $G$ is denoted by $\alpha^+(G)$ (*resp.* $\alpha^-(G)$). Any solution returned by the **min-degree** greedy algorithm on input graph $G$ has size at least $\alpha^-(G)$ and at most $\alpha^+(G)$. We have $\alpha^-(G) \leq \alpha^+(G) \leq \alpha(G)$.

We call MaxGreedy the maximization problem that consists in finding a maximum size greedy set in a given graph $G$. This problem was shown to be **NP**-hard [16].

### 3.7.1 Limitations of the min-degree rule.

Halldórsson and Radhakrishnan showed in [73] constructions of graphs with maximum degree $\delta$ where the *minimum* greedy set is small compared to the maximum independent set. More precisely, for these graphs the the ratio between $\alpha(G)$ and $\alpha^-(G)$ is $\frac{\Delta+2}{3} - \mathcal{O}(\Delta^2/n)$. In particular, this implies that the analysis of the **min-degree** greedy algorithm is tight, for all $\Delta \geq 3$.

However, on these examples there exist several vertices with minimum degree and picking the right minimum degree vertex leads greedy to an optimal solution, i.e., $\alpha^+(G) = \alpha(G)$. We prove that we can extend these examples such that we now compare the maximum independent set to the *maximum* greedy set while keeping roughly the same ratio. This suggests that any (even exponential time) tie-breaking rules for the **min-degree** greedy algorithm cannot improve significantly the approximation ratio when $\Delta$ is large.

**Theorem 3.22.** *There exists a graph $G$ with maximum degree $\Delta$ such that*

$$\frac{\alpha(G)}{\alpha^+ G} \geq \frac{\Delta + 1}{3} - \mathcal{O}(1/\Delta).$$

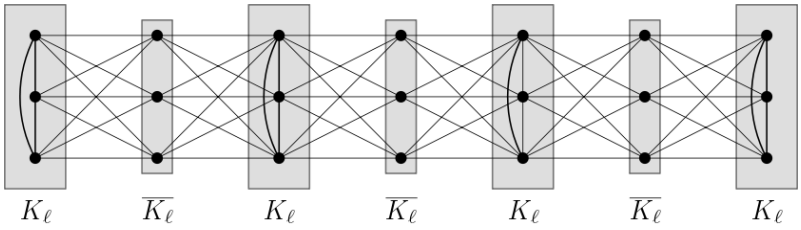



Figure 3.11: An example when $\ell = 3$. $K_\ell$ and $\overline{K_\ell}$ respectively denotes a clique and an independent set of size $\ell$.

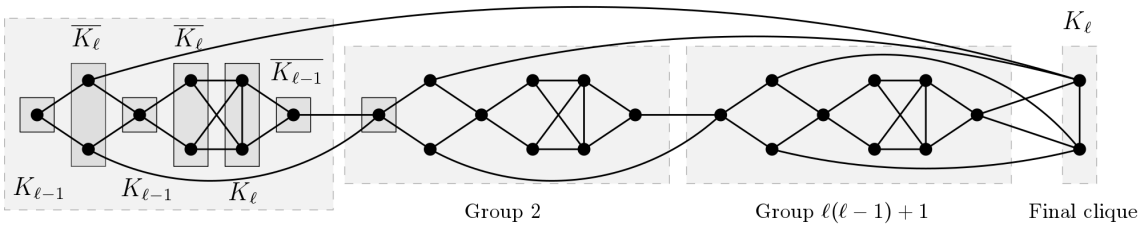



Figure 3.12: The construction when $\Delta = 3\ell - 2$.

*Proof.* We show this construction for the case $\Delta \equiv 2 \pmod 3$. See Figure 3.11. Let $\ell$ be the integer such that $3\ell - 1 = \Delta$. The graph consists in a chain of subgraphs, alternating a clique on $\ell$ vertices and an independent set of size $\ell$. Each subgraph is completely connected with the adjacent subgraphs in the chain. This structure ends with a complete graph on $\ell$ vertices. The degree of the vertices in the first and the last clique is $2\ell - 1$, while the degree of vertices of other cliques and independent sets are respectively $\Delta = 3\ell - 1$ and $2\ell$. The greedy algorithm picks one vertex in each clique while the optimal solution is the union of all vertices in the independent sets. If $n$ denotes the number of vertices in the graph, the ratio between the size of the optimal solution and the size of the solution returned is

$$\frac{(n-\ell)/2}{(n-\ell)/2\ell+1} = \ell - \frac{\ell}{(n-\ell)/2\ell+1} = \ell - \mathcal{O}(\ell^2/n) = \frac{\Delta+1}{3} - \mathcal{O}(\Delta^2/n)$$

In particular for any instance where $n = \Omega(\Delta^3)$, we get the expected result.

For the case $\Delta \equiv 1 \pmod 3$, we need a more complicated graph that can be describe as a chain of groups of six subgraphs. Consider the integer $\ell$ such that $3\ell - 2 = \Delta$. Each group is formed by a chain of subgraphs of size $\ell$ or $\ell - 1$ that are alternatively a clique and an independent set. The total graph consists in a chain of $\ell(\ell-1)+1$ such groups where the last independent set and the first clique of the next group are completely connected. Then, this chain ends with a clique of size $\ell$ totally connected with the last independent set of the last group. Additionally, add a matching of size $\ell - 1$ between the first independent set of each group to the first clique of the next group. Since these independent sets have size $\ell$, there is one unmatched vertex per each such independent set. Finally add an edge from each of these vertices to the final clique. It is not difficult to see that this can be done so that all vertices of the final clique have degree $3\ell - 2 = \Delta$. See Figure 3.12. We can see that on this graph, the maximum degree is $\Delta = 3\ell - 2$, the vertices of the first clique of the first group have degree $2\ell - 2$, while all independent set vertices have degree $2\ell - 1$. It is not difficult to check that any greedy set contains one vertex from each clique for a total of $3(\ell(\ell-1)+1)+1$ vertices while the maximum independent set consists of the union of all independent sets from each group. This number is $(3\ell-1)(\ell(\ell-1)+1)$. The

corresponding ratio is

$$\frac{3\ell-1}{3} - \frac{3\ell-1}{9(\ell(\ell-1)+1/3)} = \frac{\Delta+1}{3} - \mathcal{O}(1/\Delta)$$

The case $\Delta \equiv 0 \pmod 3$ is treated similarly than the previous one, using instead the following group

$$K_{\ell-1} - \overline{K_{\ell+1}} - K_\ell - \overline{K_\ell} - K_\ell - \overline{K_\ell}$$

and where matchings are between the first independent set and the first clique of the following group and between the last independent set and the last clique of the next group. Details of the construction and calculation are left to the curious reader. □

### 3.7.2 Hardness in cubic planar graphs

As we know now, there are examples of graphs where the maximum greedy set can be small compare to the maximum independent set. In this section, we show that computing this maximum greedy set is **NP**-hard even in the restricted class of cubic planar graphs.

**Theorem 3.23.** MaxGreedy *remains* ***NP****-complete for planar cubic graphs.*

The proof is a reduction from Maximum Independent Set in cubic planar graphs, which is **NP**-hard [59].

*Proof.* Let $G = (V, E)$ be a cubic planar graph with $m$ edges. Let us construct a graph $G'$ by replacing each edge $uv \in E$ by the structure $\mathcal{H}_{uv}$ described in Figure 3.13. We call

$$V' := V(G') = V \cup \bigcup_{e \in E} V(\mathcal{H}_e)$$

and

$$E' := E(G') = \bigcup_{uv \in E} (E(\mathcal{H}_e) \cup \{au, gv\})$$

where $au$ and $gv$ correspond to the edges connecting $u$ and $v$ to the graph $\mathcal{H}_{uv}$.

$G'$ has order $|V| + 22m$ and can be computed in polynomial time.

**Claim 3.24.** *Let $S' \subseteq V'$ be an independent set in $G'$ and $e = uv \in E$. Then, $|V(\mathcal{H}_e) \bigcap S'| \leq 9$. Moreover, if both $u$ and $v$ belong to $S'$ then $|V(\mathcal{H}_e) \bigcap S'| \leq 8$.*

We can easily check that $|\mathcal{A} \cap S'| \leq 2$, $|\mathcal{D} \cap S'| \leq 2$, $|\mathcal{C} \cap S'| \leq 3$ and $|\{a, b, d, g\} \cap S'| \leq 2$. Thus, $|V(\mathcal{H}_e) \bigcap S'| \leq 9$. Moreover, if both $u$ and $v$ belong to $S'$ then $\{a, g\} \cap S' = \emptyset$ and then $|\{a, b, d, g\} \cap S'| \leq 1$ which gives $|V(\mathcal{H}_e) \bigcap S'| \leq 8$.

**Claim 3.25.** $\alpha(G') \leq \alpha(G) + 9m$

Let $S' \subseteq V'$ be an independent set in $G'$. Denote by $F$ the set of edges of $G$ which have both end nodes in $S'$. We have

$$|S' \cap V| - |F| \leq \alpha(G)$$

Indeed, $(S' \cap V) \setminus \{x_e, e \in F\}$ is an independent set in $G$ where $x_e$ is one of the two vertices incident to an edge $e \in F$. Then,

$$|S' \cap V| - |F| \leq |(S' \cap V) \setminus \{x_e, e \in F\}| \leq \alpha(G)$$

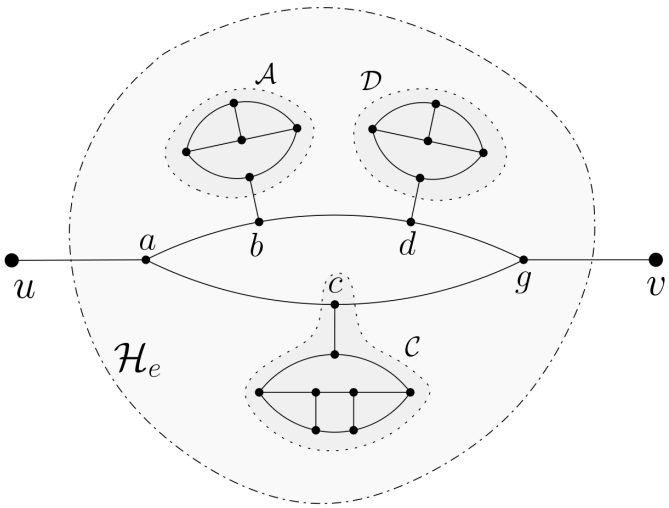



Figure 3.13: Each edge $e = uv$ is replaced by this gadget $\mathcal{H}_e$.

It follows that

$$\begin{aligned}
|S'| &= |S' \cap V| + \sum_{e \in E} |\mathcal{H}_e \cap S'| \\
&\leq (\alpha(G) + |F|) + (\sum_{e \in F} |\mathcal{H}_e \cap S'| + \sum_{e \notin F} |\mathcal{H}_e \cap S'|) \\
&\leq (\alpha(G) + |F|) + (\sum_{e \in F} 8 + \sum_{e \notin F} 9) \\
&= (\alpha(G) + |F|) + (\sum_{e \in E} 9 - \sum_{e \in F} 1) \\
&= \alpha(G) + 9m
\end{aligned}$$

Since this inequality is true for any independent set $S'$, we have $\alpha(G') \leq \alpha(G) + 9m$.

**Claim 3.26.** *There exists a greedy set $S'$ in $G'$ of size $\alpha(G) + 9m$.*

Let $S$ be a maximum independent set in $G$. Construct the set $S'$ as follows

- While there exists some unpicked nodes in $G'$ do
  1. If there exists an unpicked vertex $u \in S$ with minimum degree, add $u$ to $S'$ and nodes $b$ and $g$ in all adjacent gadgets $\mathcal{H}_{uv}$ (see Figure 3.13)
  2. Otherwise, there exists a vertex of type $a$ with minimum degree in some gadget $\mathcal{H}_{uv}$.
     - If $v \in S$, add $a$ and $d$ to $S'$
     - If $v \notin S$, add $a$ and $g$ to $S'$
- Run **Greedy** on the remaining connected components $\mathcal{A}, \mathcal{D}$ and $\mathcal{C}$ of graphs $\mathcal{H}_e$ which have not been picked yet.

At the end, we have $S' \cap V = S$ and $|S' \cap V(\mathcal{H}_e)| = 9$ for all $e$ in $E$. Then the greedy set $S'$ has the desired size.

Therefore, $\alpha^+(G') = \alpha(G') = \alpha(G) + 9m$ and then for any integer $k$ we have

$$\alpha^+(G') \geq k + 9m \quad \text{if and only if} \quad \alpha(G) \geq k.$$

□

### 3.7.3 Hardness of approximation

In this section we establish hardness of approximation for finding maximum greedy sets in three classes of graphs: general graphs, graphs with maximum degree $\Delta$ and bipartite graphs.

For any class of graphs, one can find worst case examples where the size of any greedy set is small compare to the size of the maximum independent set. In the following, we will call such examples *hard graphs*. For such graphs, there is an unique greedy set, meaning that at any stage of the algorithm the choice of the minimum degree vertex is unique, and the ratio of its size to that of the maximum independent set is minimized. For a given class of graphs, we will call this ratio an *ultimate lower bound* w.r.t the class of graphs considered, and it shows the limitation of our initial greedy rule. However, since our original motivation was to design additional tie-breaking rules for the **min-degree** greedy algorithm, in order to measure the difficulty of designing such rules, we need to compare ourselves to the maximum greedy set in the given graph. And for these examples, since the size of the greedy solution is unique, the **min-degree** greedy algorithm is optimal when we compare to the maximum size of a greedy set.

In what follows, we show that MaxGreedy is hard to approximate in different classes of graphs, within to an inapproximability factor that matches the ultimate lower bound.

**Theorem 3.27.** MaxGreedy *is hard to approximate within a factor of* $n^{1-\varepsilon}$*, for any constant* $\varepsilon > 0$*, assuming* $\boldsymbol{P \neq NP}$ *and hard to approximate within a factor of* $n/\log n$*, assuming the Exponential Time Hypothesis.*

**Theorem 3.28.** *For graphs with maximum degree* $\Delta \geq 7$*,* MaxGreedy *is hard to approximate within a factor of* $(\Delta+1)/3 - \mathcal{O}(1/\Delta) - \mathcal{O}(1/n)$*, assuming* $\boldsymbol{P \neq NP}$*.*

**Theorem 3.29.** *For bipartite graphs,* MaxGreedy *is hard to approximate within a factor of* $n^{1/2-\varepsilon}$*, for any constant* $\varepsilon > 0$*, assuming* $\boldsymbol{P \neq NP}$*.*

Here, $n$ refers here to the number of vertices of the graph considered. Notice that since the size of a maximum greedy set is upper bounded by the size of a maximum independent set, a lower bound on the approximability of MaxGreedy is necessarily smaller than the best approximation ratio achieved by one particular tie-breaking greedy algorithm. This suggests that our inapproximability results are (almost) tight.

A way of proving these inapproximability lower bounds is the following. Given a class of graphs, we first find a hard graph $B$, where the greedy set is unique. In particular, there exists exactly one minimum degree vertex $r$ in $B$, that we call the *root* of the hard graph. We ask additionally these hard graphs to have the following property. If the root is removed from $B$, then the **min-degree** greedy algorithm outputs a maximum independent set: $\textbf{Greedy}(B \setminus r) = \alpha(B)$.

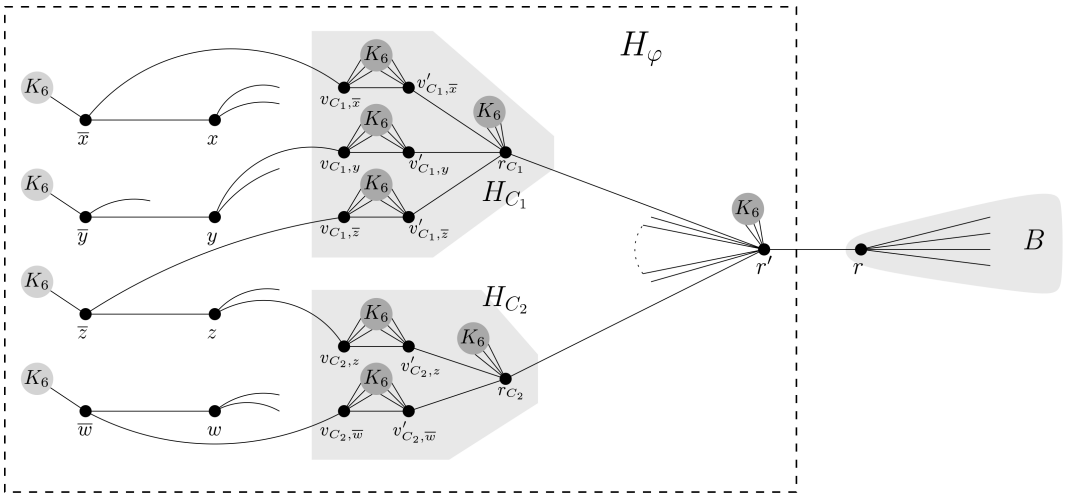



Figure 3.14: A example of the construction of $H_\varphi$, where $\varphi$ contains clauses $C_1 = \overline{x}y\overline{z}$ and $C_2 = z\overline{w}$. Cliques $K_6$ ensure that vertices of this graph are executed by **Greedy** in the right order.

Now, we add an **anchor graph** $H$ connected to $B$ by the root vertex $r$. Intuitively, the sub-graph encodes an instance of an **NP**-hard problem, in such way that there exists a greedy set that does not contain $r$ if and only if the instance is positive.

When the size of $H$ is arbitrarily small compared to $\alpha(B)$, then the gap introduced will be arbitrarily close to the approximation ratio of the greedy algorithm in $B$, i.e., the ultimate lower bound for the class of graphs considered.

Precisely, let $\varphi$ be a formula of SAT, with $n$ variables and $m$ clauses. Without loss of generality, we can assume that :

(1) Each clause contains two or three literals.

(2) Each positive (*resp.* negative) literal appears in exactly two (*resp.* one) clauses.

Indeed, given a 3-SAT formula, if a variable $y$ occurs $k$ times, then we replace each occurrence by new variables $y_1, \ldots, y_k$, and add the new clauses $y_1 \Rightarrow y_2, \ldots, y_k \Rightarrow y_1$. Now, (1) is satisfied and each variable appears exactly three times. Finally, if a variable does not satisfy (2), simply exchange its positive and negative literal. This new formula has polynomial size in $n, m$ and is satisfiable if and only if the original formula is satisfiable.

Now we build the anchor graph $H_\varphi$ as follows. First create a vertex $r'$. Create two adjacent **literal vertices** $x$ and $\overline{x}$, for each variable $x$ and create a **gadget** $H_C$ (see Figure 3.14) for each clause $C$. In particular, this gadget has one vertex $v_{C,\ell}$ for each literal $\ell$ in $C$ and a vertex $r_C$. Connect $v_{C,\ell}$ to the corresponding literal vertex $\ell$ and each $r_C$ to $r$. Finally, we increase the degree of negative literal vertices by one, by adding a clique of 6 vertices with one connected to the literal vertex. Then, each literal vertex has now exactly degree three. Let $G$ be the graph obtained by connecting $H_\varphi$ and $B$ with their respective vertices $r'$ and $r$. See Figure 3.14.

Any execution of the **min-degree** algorithm in $G$, resulting in a greedy set $S$ at the end, consists of three phases. During phase 1, the minimum degree vertices are the literal vertices, with degree three. **Greedy** has to decide either to pick $x$ or $\overline{x}$, for each literal $x$. This is exactly choosing a valuation $\nu$ such that $x$ is in $S$ if and only if $\nu(x) = true$. During the second phase, all remaining vertices in $H$ are removed, such that at the end, $r'$

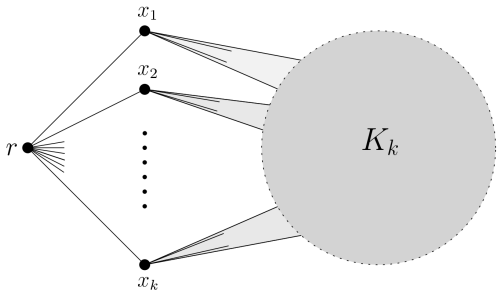



Figure 3.15: hard general graph. The greedy algorithm returns $r$ and one vertex from the clique, while the maximum independent is $\{x_1, \ldots, x_k\}$. The ratio is $\frac{k}{2} = \frac{n-1}{4}$.

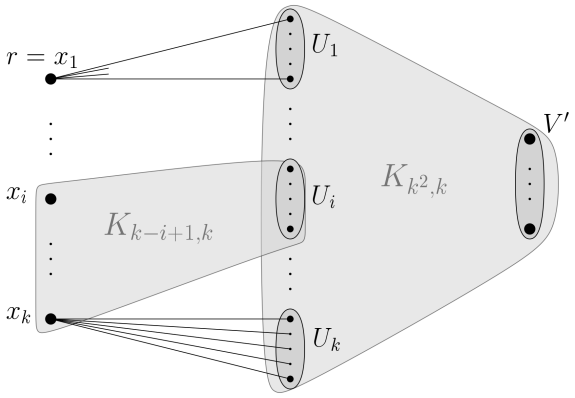



Figure 3.16: hard bipartite graph. $|U_1| = \cdots = |U_n| = |V'| = k$. Any vertex is $U_i$ is adjacent to $V' \cup \{x_i, \ldots, x_n\}$. The greedy algorithm outputs $V' \cup \{x_1, \ldots, x_n\}$ while the maximum independent set is $\bigcup_i U_i$. The gap is $k/2 = \Theta(\sqrt{n})$.

is in $S$ if and only if $\varphi$ is true for $\nu$. To see this, notice that $v_{C,\ell}$ or $v'_{C,\ell}$ have now minimum degree four, and the clause $C$ is not satisfied by $\nu$ if and only if for all literals $\ell$ in $C$, the corresponding vertices $v_{C,\ell}$ are still in the graph. This implies picking all these vertices, and later the vertex $r_C$ in $S$, so that $r' \notin S$. Finally, the last phase consists in executing the greedy algorithm in $B$: if the formula is not satisfied then, the root $r$ is picked and the number of vertices picked during this phase is $\mathbf{Greedy}(B)$, and otherwise $\alpha(B)$ by the assumption.

To summarize, we build a graph $G$ such that

- If $\varphi$ is satisfiable then, there exists a greedy set of size at least $\alpha(B)$.
- otherwise, if $\varphi$ is not satisfiable then, any greedy set has size at most $|V(H_\varphi)| + \mathbf{Greedy}(B)$,

where $|V(H_\varphi)| = \Theta(m+n)$. Then, when the hard graph has arbitrarily large size, one can introduce a gap arbitrarily close to the ratio $\alpha(B)/\mathbf{Greedy}(B)$. We use hard graphs in Figure 3.15 (general graphs), Figure 3.16 (bipartite graphs) and the hard graphs of bounded degree presented in Theorem 3.22, to obtain the above results that are are detailed subsequently below.

**Hardness of approximation for general graphs.** Consider the hard graph $B$ given by Figure 3.15. For any $\varepsilon > 0$, choose its size $n$ such that $|\varphi| < n^\varepsilon$. The size of $G$ is a polynomial in $n$. This implies that MaxGreedy is hard to approximate to within $n^{1-\varepsilon}$. Interestingly, Håstad proved that Maximum Independent Set is also hard to approximate to within the same factor [80].

Further, assuming the *Exponential Time Hypothesis*, that there exists a constant $\epsilon > 0$, such that 3-SAT cannot be solved in time $2^{\epsilon n}$, implies that MaxGreedy is hard to approximate to within a factor of $n/\log n$. To see this, one may use the same construction with a bad graph of size $2^{\epsilon'|\varphi|}$ for a suitable value $\epsilon' > 0$. In particular this suggests that a greedy algorithm will not do as well as Feige's algorithm that achieves an $O(n/(\log n)^3)$ approximation [50].

**Hardness of approximation for graphs with maximum degree $\Delta$.** We use hard graphs described in the proof of Theorem 3.22, see Figures 3.11 and 3.12. Recall that the structure of these graphs depends on the value $\Delta \mod 3$. As defined before, these hard graphs have several vertices of minimum degree, that make a clique $K$. Then, add a vertex $r''$, adjacent to all vertices in $K$ and a root vertex $r$ adjacent to $r''$. Now, these graphs have a unique root and the size of the greedy set and the maximum independent set only grew up by an additive constant factor. Since $\Delta \geq 7$, all vertices in $B$ have degree at least 5 so that the greedy algorithm will execute first all vertices in the anchor graph.

All vertices in $G$ have degree at most $\Delta$, except the root $r'$ that has degree equal to a least the number of clauses. To overcome this issue, we can replace edges $(r_C, r)$ by a tree of bounded degree, rooted in $r'$, such that the distance from and $r_C$ to $r$ is odd, and such that vertices at odd distance from $r'$ have smaller degree than the ones at even distance. We can finally add several cliques to increase the degree of vertices of this tree so that, they all have degree at least five. If at some point a vertex $r_C$ is picked by the greedy algorithm — meaning that the formula is not satisfied — then all vertices at odd distance from $r'$ on the path from $r_C$ to $r$ will be picked in $S$, so that $r' \notin S$.

**Hardness of approximation for bipartite graphs.** Figure 3.16 presents an example of an hard bipartite graph, with a ratio $\Theta(\sqrt{n})$. When the root $r$ is removed from $B$, then the vertices in $U_1$ have minimum degree $2k-1$ while all vertices $x_1$ have degree at least $2k$. Therefore, in this situation the greedy set returned is the union of all $U_i$, that is a maximum independent set in $B$.

To make the anchor graph bipartite, we first change the structure of gadgets $K_6$ used to increases the degree of some vertices. For instance, a simple bipartite complete graph $K_{6,6}$ fulfils exactly the same function. Then, one might still find some odd-cycles left due to edges between literal vertices and gadgets. These can be avoided using a **NP**-hard version of the satisfaction problem called Monotone 3-SAT-4, where each variable appears four times and additionally that containing three literals, each clause contains only positive, or only negative literals. See [43] for more details.

Since an optimal independent set can be computed in polynomial time in a bipartite graph, the greedy algorithm is not a good algorithm for this class. However, this negative result suggests that even knowing a maximum independent set may not be helpful in order to design good tie-breaking rules.

## 3.8 Conclusion

Our main technical contribution is a non-local payment scheme together with an inductive argument that can be embedded with greedy-style algorithms for the Maximum Independent Set problem on bounded degree graphs. These techniques imply best possible approximation guarantees of greedy on subcubic graphs. We have also shown versatility of these techniques by proving (via simple proofs) that they imply close to best possible greedy guarantees on graphs with maximum degree $\Delta$, for any $\Delta$.

Furthermore, we showed that in our original paper [101] that these techniques also imply improved fast approximation algorithms for the minimum vertex cover problem on bounded degree graphs. We obtained a fast $O(n^2)$-time 6/5-approximation for the Minimum Vertex Cover problem on subcubic graphs. The best algorithm for this problem is a 7/6-approximation with a running time of at least $n^{50}$ [71]. Even obtaining the

6/5-approximation for the Minimum Vertex Cover problem on subcubic graphs required a running time of $n^{18.27}$ [31].

We have complemented these results by hardness results, showing that it is hard to compute good tie-breaking rules for the greedy algorithms.

Our techniques have a potential for further generalizations and applications. Our potential function and the inductive argument are quite general and they could be applied to other related problems on bounded degree graphs. Such general problem should have the following features: given a graph, the optimal solution should be ubiquitously "distributed" over the input graph, and therefore also a feasible solution should be computable sequentially/locally by "choosing parts of the graph", debt and loan should be definable on such a problem as problem specific, depending on the problem's constraints. Possible candidates are, for instance, the set packing and set covering problems with sets of bounded size and bounded element occurrences.

Finally, we also mention some more specific directions for further study. Can we obtain a greedy rule to design a $(\Delta + 1)/3$-approximation for any value of $\Delta$? We think that it should be possible with our techniques, by a careful and refined analysis. For instance, we can already easily prove an 9/5-approximation when $\Delta = 4$, and a 13/6-approximation when $\Delta = 5$. This already improves on the previous ratios that follow from the known bound of $(\Delta + 2)/3$ on any greedy: 2 for $\Delta = 4$, and 7/3 for $\Delta = 5$.

# 4 | Approximating Integral Multiflows on the Plane

## 4.1 Introduction

The edge-disjoint paths problem (EDP) is a fundamental problem in combinatorial optimization, consisting of connecting as many demand pairs as possible in a graph via edge-disjoint paths. There is a large body of literature studying this problem in various settings. A primary goal has been to find conditions under which there is a solution satisfying all demands, e.g. when the *cut condition* is sufficient; see Frank's survey [54] or part VII of Schrijver's book [133].

Unfortunately many cases of EDP are **NP**-hard, so it is natural to look for approximation algorithms. However there is no general theory and constant-factor approximations can only be expected in special cases.

One of the landmark results in this area is due to Seymour. Let $G+H$ denote the union (of the edge sets) of the supply graph $G$ and the demand graph $H$. Seymour [137] proved that if $G+H$ is *planar and Eulerian*, the cut condition guarantees a solution to connect all demand pairs; such a solution can be found in polynomial time. Seymour's result has motivated a sequence of follow-up works investigating EDP when $G+H$ is planar. We refer the readers to Frank's surveys [54, 55] and Schrijver's book [133, in particular Chapter 74.2] for an overview of these results. For example, one can decide in polynomial time whether all demand pairs can be connected when $G+H$ is planar and the demand pairs lie on a bounded number of faces of $G$ [117].

Unfortunately the *general* case that $G+H$ is planar is one of these cases in which EDP is **NP**-hard, as Middendorf and Pfeiffer [117] proved. Very little seems to be known about approximation in that setting. Korach and Penn [98] showed that, given the cut condition, one can satisfy all demands except one for each face of $G$, and such a solution can be found in polynomial time. However, this does not imply an approximation ratio in general.

### 4.1.1 Our Results

Before we present our main results, let us define the edge-disjoint paths problem formally:

**Definition 4.1.** The **EdgeDisjointPaths** problem (**EDP**) takes as input a **supply** graph $G = (V, E)$ and a **demand** graph $H = (V, D)$ and asks for a maximum-cardinality set of pairwise edge-disjoint cycles such that each cycle consists of an edge $\{u, v\}$ in $D$ and a path between $u$ and $v$ in $E$.

The conditions that the set $\mathcal{C}$ of cycles in $G+H$ must satisfy can equivalently be

written as: (1) $\forall C \in \mathcal{C}$, $|C \cap D| = 1$, (2) $\forall e \in D \cup E$, $|\{C \mid C \in \mathcal{C}, C \ni e\}| \leq 1$. Our results extend to the natural generalization when edges have integral capacities (the Integer Multiflow problem; see Section 4.5.2); for the sake of simplicity, we present our algorithms for the uncapacitated case. We consider the case when $G + H$ is planar and present a constant-factor approximation algorithm:

**Theorem 4.2.** *There is a polynomial-time* 24*-approximation algorithm for* EdgeDisjointPaths *if* $G + H$ *is planar.*

We prove this theorem in Sections 4.2 and 4.3. Our proof is based on rounding a fractional solution to a certain LP relaxation to obtain a subset of demand edges for which the cut condition is satisfied, i.e., no cut has more of these demand edges than supply edges: in the planar dual, this translates into what we call the NonNegativeCycles problem.

**Definition 4.3.** The **NonNegativeCycles** problem (**NNC**) takes as input a supply graph $G = (V, E)$ and a demand graph $H = (V, D)$ and asks for a maximum-cardinality set $D' \subseteq D$ of demand edges such that

$$|C \cap D'| \leq |C \cap E| \text{ for every cycle } C \text{ in } G + H.$$

In other words, if edges in $E$ have weight 1, edges in $D'$ have weight $-1$, and edges in $D \setminus D'$ have weight 0, then every cycle must have non-negative total weight. The NonNegativeCycles problem is **NP**-hard even when $G + H$ is planar (Section 4.6), but we give an approximation algorithm (see Theorem 4.6). Once this rounding is done and we have an approximate solution of NonNegativeCycles, we can obtain edge-disjoint paths for at least half of these demand edges (see Theorem 4.8). We use the four color theorem in several places.

Section 4.5 analyzes the integrality gaps of various LPs. By comparing the output of our algorithm to the value of the natural LP relaxation, we get an upper bound of 24 on its integrality gap. A special case, when every demand edge has an infinite (or large enough) number of parallel copies, has been called *maximum integer multiflow*. In this case the dual LP is a relaxation of the multicut problem, which asks for a smallest set of supply edges whose deletion destroys all paths for all demand edges. We show that this dual LP has an integrality gap of at most 2 if $G + H$ is planar. This yields:

**Theorem 4.4.** *If* $G + H$ *is planar, the minimum cardinality of a multicut is at most* 48 *times the maximum value of an integer multiflow.*

This is one of the few cases with a constant upper bound on the ratio of the smallest multicut and the largest *integer* multiflow (see, e.g., [144]). Another such case is when $G$ arises from a tree by duplicating edges; then this ratio is 2 [63]; in general the ratio can be as large as $\Theta(|D|)$, even when $G$ is planar (see Figure 5.7).

### 4.1.2 Further Related Work

**Hardness.** The decision version of EDP is one of Karp's original **NP**-complete problems [90], and remains **NP**-complete even in many special cases [122], including the case of interest in this paper, namely even when $G + H$ is planar [117]. In terms of approximation, EDP is **APX**-hard [3]. Assuming that $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log n)})$, where $n = |V|$, there is

no $2^{o(\sqrt{\log n})}$-DTIMEapproximation for EDP, even when $G$ is planar and subcubic and each demand edge has one of its endpoints on the outer face of $G$ [35]. Assuming that for some $\delta > 0$, not all problems in **NP** can be solved in randomized time $2^{n^\delta}$, there is no $n^{O(1/(\log\log n)^2)}$-approximation even when $G$ is a wall graph [36]. As far as we know, no stronger hardness result is known for integral mutliflows.

The difficulty is further illustrated by the integrality gap of a natural LP relaxation: even when $G$ is planar and subcubic, the integrality gap is already in the order of $\Theta(\sqrt{n})$ [63], see Figure 5.7.

**Positive results.** EDP can be solved in polynomial time when the number of demand edges is bounded by a constant [132]. The same holds for integral multiflows when $G + H$ is planar [135]. The best known approximation guarantee is $O(\sqrt{n})$ [27], even when $G$ is planar.

Nonetheless, there are some special cases for which significantly better approximation ratios are known: for instance, when $G$ is planar and Eulerian, or planar and 4-edge-connected, there is an $O(\log n)$-approximation [91], improving on a previous $O(\log^2 n)$-approximation algorithm for Eulerian planar graphs [97]. When $G$ is a grid, there is an $O(\log^2 n)$-approximation [6]. When $G$ is a wall graph, there is a $\tilde{O}(n^{1/4})$-approximation [33]. When in addition all demand edges have one endpoint at the boundary of the wall, there is an $2^{O(\sqrt{\log n}\cdot\log\log n)}$-approximation [36]. Yet none of those results gets to the range of a constant-factor approximation.

**Variants.** One way to relax EDP is to allow for congestion $c$, that is, in the solution, up to $c$ paths may share the same edge. It is known that EDP becomes significantly easier with congestion: with congestion 2, there is a polylogarithmic approximation [32], and when in addition $G$ is planar, there is a constant-factor approximation [136].

A closely-related but more difficult problem is the Node-Disjoint Paths (NDP) problem, in which two paths may not even share a common *node*. EDP can be reduced to NDP by taking the line graph of the supply graph, but this reduction does not preserve planarity. If $G + H$ is planar, however, Middendorf and Pfeiffer [117] showed how to reduce EDP to NDP while preserving planarity. Hence the $\tilde{O}(n^{9/19})$-approximation algorithm for NDP with planar supply graphs [34] implies the same approximation ratio for EDP with $G + H$ planar.

Another more general version of EDP is its directed version (one can reduce EDP to its directed version by replacing each edge by a simple gadget). The directed version seems to be strictly harder than the undirected EDP; for example it is even **NP**-hard for two demand edges [53].

### 4.1.3 Preliminaries

We assume throughout, without loss of generality, that $G + H$ is connected. We recall here some notions and well-known properties related to cuts, cycles, planar duality, and $T$-joins.

A ***D*-cycle** is a cycle that contains exactly one edge of $D$. Then EdgeDisjointPaths asks for a maximum number of pairwise disjoint $D$-cycles.

We say an instance of EDP has a **complete solution** if there are $|D|$ pairwise disjoint $D$-cycles. For the existence of a complete solution, a necessary condition is the **cut**

**condition**:

$$|C \cap D| \leq |C \cap E| \text{ for every cut } C. \tag{4.1}$$

It is necessary because every cycle intersects every cut in an even number of edges. In fact it is equivalent to requiring Equation 4.1 only for *simple* cuts, i.e., edge sets $\delta(U)$ where both $U$ and $V \setminus U$ induce connected subgraphs.

The cut condition is very useful although in general it is **NP**-hard to check (this is an unpublished observation of Sebő). The reason is that in many special cases, e.g. when $G + H$ is planar and Eulerian [137], it is sufficient and then can also be checked in polynomial time. However, the cut condition is not sufficient in general, even when $G + H$ is planar. For instance, if $G + H$ is $K_4$, the complete graph on four vertices, and $D$ is a perfect matching, then the cut condition is satisfied but there is no complete solution., see Figure 4.5 (a).

Let us call a simple cut that contains exactly one edge of $D$ a **$D$-cut**. Then an equivalent formulation of EDP, if $G + H$ is planar, asks for finding a maximum number of pairwise disjoint $D$-cuts in the planar dual graph $(G + H)^*$. The cut condition translates to the **cycle condition** in the planar dual graph:

$$|C \cap D| \leq |C \cap E| \text{ for every cycle } C. \tag{4.2}$$

Deleting an edge in a planar graph is equivalent to contracting the corresponding edge in the planar dual. These operations preserve planarity.

***T*-Joins.** Condition Equation 4.2 is well known in the context of $T$-joins. For an edge set $F$, let $\text{odd}(F)$ denote the set of vertices whose degree is odd in $(V, F)$. For a subset $T \subseteq V$, a **$T$-join** is a set $F$ of edges with $\text{odd}(F) = T$. If $F$ is a minimum-cardinality $\text{odd}(F)$-join, then $F$ is called a **join**. Guan [66] observed that $F$ is a join if and only if

$$|C \cap F| \leq |C \setminus F| \text{ for every cycle } C. \tag{4.3}$$

For any connected graph, any $T$ with $|T|$ even, and any real edge weights, a minimum-weight $T$-join can be computed in polynomial time [45].

If we have a set of pairwise disjoint $D$-cuts and $D' \subseteq D$ is the set of demand edges that belong to one of these cuts, then we must have

$$|C \cap D'| \leq |C \cap E| \text{ for every cycle } C. \tag{4.4}$$

This is equivalent to saying that $D'$ is a join after contracting the edges in $D \setminus D'$. Although this condition is again not sufficient, we will first find a set $D' \subseteq D$ satisfying Equation 4.4 (in the planar dual graph).

## 4.2 Roadmap

### 4.2.1 Satisfying the Cycle Condition

The preceding considerations motivate studying the NonNegativeCycles problem in the planar dual graph.

The NonNegativeCycles problem is **NP**-hard even when $G + H$ is planar; see Section 4.6. We devise a 16-approximation algorithm if $G + H$ is planar. It begins by

solving the following linear programming relaxation, which we call the *non-negative cycle LP*:

$$\begin{aligned} \max \sum_{e \in D} x_e & \qquad (4.5) \\ \sum_{e \in C \cap D} x_e \leq |C \cap E| & \qquad \forall \text{ cycles } C \\ 0 \leq x_e \leq 1 & \qquad \forall e \in D \end{aligned}$$

**Lemma 4.5.** *The non-negative cycle LP (4.5) can be solved in polynomial time.*

*Proof.* By the equivalence of optimization and separation [65], it is sufficient to solve the separation problem. To this end, given $x \in \mathbb{R}^D$, define weights $w(e) := -x_e$ for $e \in D$ and $w(e) := 1$ for $e \in E$. Then the separation problem reduces to finding a negative-weight cycle or deciding that there is none. This can be done by computing a minimum-weight $\emptyset$-join $J$. If $w(J) \geq 0$, then there is no negative-weight cycle. Otherwise $J$ can be decomposed into cycles, and at least one of them will have negative weight. □

The second step of our algorithm will round the LP solution. In Section 4.3 we will prove:

**Theorem 4.6.** *Let $x$ be a feasible solution to the non-negative cycle LP (4.5). Then there is a polynomial-time algorithm to construct an integral feasible solution that has value at least $\frac{1}{16} \sum_{e \in D} x_e$.*

### 4.2.2 Packing $T$-Cuts

Let $x$ be an integral feasible solution to the non-negative cycle LP (in the planar dual), and let $J \subseteq D$ contain those demand edges $e$ for which $x_e = 1$. So $J$ is an NNC solution. If we delete the other demand edges (or contract them in the planar dual), the resulting instance now satisfies the cut condition (the cycle condition in the planar dual). Let $G' = (V, J \dot\cup E)^*$ be the graph that arises from the planar dual by contracting the edges in $D \setminus J$. Then $J$ is a join in $G'$. Our goal is now to find as many pairwise disjoint $J$-cuts in $G'$ as possible because after uncontracting, these will correspond to pairwise disjoint $D$-cuts in $(G + H)^*$ and hence to pairwise disjoint $D$-cycles in $G + H$.

Edmonds and Johnson [45] and Lovász [108, 110] showed that a perfect half-integral packing of $J$-cuts exists and can be computed in polynomial time:

**Theorem 4.7.** *For every graph $G$ and every join $J$ in $G$ there exist vertex sets $U_1, \ldots, U_{2|J|}$ (not necessarily distinct) that form a laminar family, such that $|\delta(U_i) \cap J| = 1$ for all $i$ and for every edge $e$ of $G$ there are at most two indices $i$ with $e \in \delta(U_i)$. Such sets can be computed in polynomial time.*

This theorem has been called the $T$-cut packing theorem because if $J$ is a $T$-join (for some *vertex* set $T$) then the cuts will be $T$-cuts, i.e., cuts $\delta(U)$ with $|U \cap T|$ odd. However, we prefer to continue speaking of $J$-cuts because this is what we need. See [55] and Theorem 4.15 for extensions of this result. Using planarity one can deduce from this a large number of pairwise disjoint $J$-cuts. The following theorem follows from the proof of Theorem 1 in Fiorini et al. [51] (that theorem is originally due to Král' and Voss [100]).

**Theorem 4.8.** *Let $G$ be a planar graph and $J$ a join in $G$. Then there is a family of at least $\frac{|J|}{2}$ pairwise disjoint $J$-cuts in $G$, and such a family can be computed in polynomial time.*

For completeness, we give a proof of this result in Section 4.4.

### 4.2.3 A 32-approximation algorithm

We have now all ingredients to prove a constant-factor approximation. The overall algorithm can be summarized as follows. The input consists of two graphs $G = (V, E)$ and $H = (V, D)$ such that $G + H$ is planar.

*Step 0:* Construct a planar dual graph $(G + H)^*$

*Step 1:* Solve the non-negative cycle LP Equation 4.5 in the planar dual graph $(G + H)^*$, to get a fractional solution $x$;

*Step 2:* Use the algorithm of Theorem 4.6 (proved in Section 4.3) to deduce an integral feasible solution $x'$ with $\sum_{e \in D} x'_e \geq \frac{1}{16} \sum_{e \in D} x_e$. Let $J := \{e \in D : x'_e = 1\}$ be the corresponding NNC solution.

*Step 3:* Contract the edges in $D \setminus J$ (in the planar dual graph) and use the algorithm of Theorem 4.8 to compute at least $\frac{|J|}{2}$ pairwise disjoint $J$-cuts. By planar duality, these correspond to at least $\frac{|J|}{2}$ pairwise disjoint $D$-cycles in $G + H$. Output these.

**Theorem 4.9.** *The above is a polynomial-time 32-approximation algorithm for* EdgeDisjointPaths *when $G + H$ is planar.*

*Proof.* A $J$-cut in $(V, J \dot\cup E)^*$ indeed corresponds to a $J$-cycle in $(V, J \dot\cup E)$ and hence in $G + H$. Therefore the output is a feasible solution. By Lemma 4.5, Theorem 4.6, and Theorem 4.8, the algorithm runs in polynomial time. The number of $D$-cycles that it computes is at least $\frac{1}{32}$ times the LP value.

On the other hand, consider an optimum solution, say consisting of OPT many $D$-cycles. Then there is a set $\bar{D} \subseteq D$ such that $|\bar{D}| = \text{OPT}$ and $(G, (V, \bar{D}))$ has a complete solution. This instance must therefore satisfy the cut condition, in other words we have $|C \cap \bar{D}| \leq |C \cap E|$ for every cut $C$ in $G + H$. Therefore, in the planar dual, setting $\bar{x}_e := 1$ for $e \in \bar{D}$ and $\bar{x}_e := 0$ for $e \in D \setminus \bar{D}$ defines a feasible solution to the non-negative cycle LP. Hence the LP value is at least OPT. □

See Section 4.3.3 for a slight variant of this algorithm reducing the approximation ratio from 32 to 24, hence Theorem 4.2.

## 4.3 Rounding the Non-negative Cycle LP

In this section we show how to round the non-negative cycle LP and prove Theorems 4.6 and 4.2. Let us first recall the statement of Theorem 4.6.

**Theorem 4.6.** *Let $x$ be a feasible solution to the non-negative cycle LP (4.5). Then there is a polynomial-time algorithm to construct an integral feasible solution that has value at least $\frac{1}{16} \sum_{e \in D} x_e$.*

### 4.3.1 Algorithm

Starting from a feasible solution $x$ of the non-negative cycle LP (4.5), we first contract all cycles in $(V, D)$. Every edge $e \in D$ that vanishes in this contraction has $x_e = 0$, therefore this preprocessing will not change $\sum_{e \in D} x_e$. Now $(V, D)$ is a forest. Then we execute the following two algorithms and pick the better of the two solutions $S_1$ and $S_2$ obtained from Algorithms 10 and 11 respectively.

The first algorithm is independent of the fractional solution $x$ and simply picks a subset of edges incident to leaves. It is the better choice if $(V, D)$ has many leaves (degree-1 vertices).

**Input:** an NNC input $G + H$ such that $(V, D)$ is a forest.
**Output:** a feasible solution $S_1$ to NonNegativeCycles.
Find a 4-coloring of $G + H$, consider a color class containing the largest number of leaves of $(V, D)$, and let $I$ be the set of leaves of that color.
Let $S_1$ be the set of edges in $D$ that are incident to a leaf in $I$.
**return** $S_1$

**Algorithm 10:** The leaf algorithm.

The second algorithm does a careful rounding of the fractional LP solution; see Figure 4.1 for an example.

**Input:** an NNC input $G + H$ such that $(V, D)$ is a forest, and a feasible solution $x$ to the non-negative cycle LP *Equation* 4.5.
**Output:** a feasible solution $S_2$ to NonNegativeCycles.
For each connected component of the forest $(V, D)$: Root the tree arbitrarily. Let $B$ denote the set of vertices with at least two children, and let $L$ denote the set of leaves.
For each $b' \in B \cup L$ and $b \in B \setminus \{b'\}$ such that $b$ is an ancestor of $b'$ and each inner vertex of the $b$-$b'$-path has exactly one child: subdivide the first edge on this path, thus inserting an artificial edge $a_{b,b'}$ incident to $b$, and set $x_{a_{b,b'}} = 0$. Let $A$ denote the set of artificial edges.
Define a budget function $y : D \cup A \to \mathbb{R}_{\geq 0}$. Initially $y_e = x_e$ for all $e \in D \cup A$.
For each artificial edge $a_{b,b'} \in A$ follow the path from $b$ down to $b'$. For each traversed edge $e \in D$, decrease $y_e$ and increase $y_{a_{b,b'}}$ by the same amount. Do this until $y_{a_{b,b'}} = 1$ or we reach $b'$.
Consider the edges $e \in D$ in an order of non-increasing distance from the root (in the graph $(V, D \cup A)$), and for each such edge $e = (u, v)$, if $y_e > 0$ then: Follow the path from $u$ up to the root. For each traversed edge $e' \in D \cup A$, decrease $y_{e'}$ and increase $y_e$ by the same amount. Do this until $y_e = 2$ or we reach the root.
$S_2^0 \leftarrow \{e \in D : y_e > 0\}$.
Contract each tree of $(V, D)$, find a 4-coloring of the resulting graph, inducing a 4-coloring of the trees, choose the color class maximizing the number of edges of $S_2^0$ in trees of that color, and let $S_2$ denote that subset of $S_2^0$.
**return** $S_2$

**Algorithm 11:** The internal algorithm.

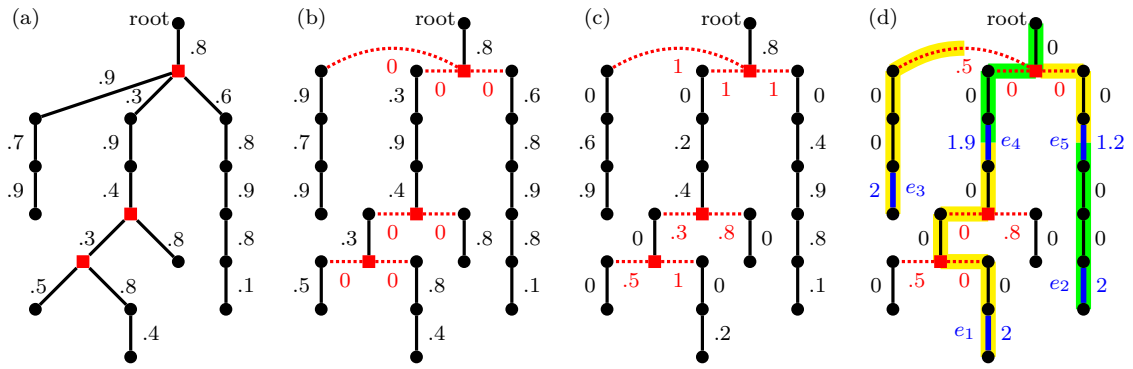



Figure 4.1: Illustrating Algorithm 11. (a): the original tree with an arbitrarily chosen root; vertices of $B$ are red squares; next to each edge $e$ the value $x_e$ is shown. (b): the rooted tree after inserting the artificial edges (red, dotted) and the initial budgets $y_e$ (after step 3). (c): each artificial edge has collected up to one unit from below (in step 4). (d): edges $e_1, \dots, e_5$ (shown in blue, bold) are collecting up to two units from above (in step 5) and are included into $S_2^0$ (in step 6). We see the final distribution if the edges are considered in this order. The green and yellow shades show from where the final budgets $y_{e_1}, \dots, y_{e_5}$ come from.

### 4.3.2 Analysis

These algorithms are well-defined and polynomial time by Theorem 1.1. We will prove that $S_1$ and $S_2$ are feasible NonNegativeCycles solutions (Lemmas 4.10 and 4.12, respectively) and that $\max\{|S_1|, |S_2|\} \geq \frac{1}{16} \sum_{e \in D} x_e$ (Lemma 4.13), thus establishing Theorem 4.6.

**Lemma 4.10.** *$S_1$ is a feasible solution for* NonNegativeCycles.

*Proof.* Let $C$ be a cycle in $G + H$. By definition of $S_1$ each edge of $C \cap S_1$ is incident to at least one vertex of $I$, and since vertices of $I$ are leaves of $(V, D)$ those vertices are all distinct, so $|C \cap S_1| \leq |V(C) \cap I|$. Each vertex of $V(C) \cap I$ is incident to at most one edge of $C \cap D$, hence to at least one edge of $C \cap E$. Since $I$ is an independent set in $G$, those edges are all distinct, so $|V(C) \cap I| \leq |C \cap E|$. □

To prove that $S_2$ is also a feasible solution for NonNegativeCycles, we first show the following key inequality:

**Lemma 4.11.** *Let $P$ be the set of edges of a path in $(V, D)$. Then*

$$\sum_{e \in P} x_e \geq 2|S_2 \cap P| - 2. \tag{4.6}$$

*Proof.* We may assume that $S_2 \cap P \neq \emptyset$, for the assertion is trivial otherwise. Let $v$ be the vertex in $P$ that is closest to the root (picked in step 1 of Algorithm 11) of the connected component of $(V, D)$ that contains $P$. Then $P$ is the union of two paths $P_1$ and $P_2$ that both begin in $v$ and go down the tree. One of these paths may be empty. Let $i \in \{1, 2\}$ with $S_2 \cap P_i \neq \emptyset$, and let $e_i^h$ be the edge of $S_2 \cap P_i$ that is closest to $v$ (and to the root). After step 5 of Algorithm 11, every edge $e$ of $(S_2 \cap P_i) \setminus \{e_i^h\}$ has $y_e = 2$, and these budgets come from edges of $P_i$. The latter is because

(i) $e_i^h$ is the only edge in $S_2 \cap P_i$ to which budget that is originally from closer to the root than $v$ can be moved, and

(ii) budget that was moved towards the root (to an artificial edge) in step 4 remains at this artificial edge or is moved in step 5 to an edge that is not closer to the root than the edge where the budget was initially.

Thus,

$$\sum_{e \in P_i} x_e \geq \sum_{e \in S_2 \cap P_i, e \neq e_i^h} y_e = 2(|S_2 \cap P_i| - 1).$$

This already implies Equation 4.6 if $S_2 \cap P = S_2 \cap P_i$.

It remains to consider the case that neither $S_2 \cap P_1$ nor $S_2 \cap P_2$ is empty. Then $v \in B$, and after step 2 there is an artificial edge incident to $v$ on the path from $v$ to $e_i^h$ (for each $i \in \{1, 2\}$). Let $f_i$ be the last artificial edge on the path from $v$ to $e_i^h$. Since $e_i^h \in S_2$, we had $y_{e_i^h} > 0$ after step 4 (i.e., even before we started augmenting $y_{e_i^h}$ in step 5). Therefore $y_{f_i} = 1$ after step 4 (and, by the same argument as before, this budget comes from edges of $P_i$), and $y_{e_i^h} \geq 1$ after step 5. Thus

$$\sum_{e \in P_i} x_e \geq 1 + \sum_{e \in S_2 \cap P_i, e \neq e_i^h} y_e = 1 + 2(|S_2 \cap P_i| - 1) = 2|S_2 \cap P_i| - 1.$$

Adding this for $P_1$ and $P_2$ yields Equation 4.6. □

**Lemma 4.12.** *$S_2$ is a feasible solution for* NonNegativeCycles.

*Proof.* Let $C$ be a cycle in $G + H$ that contains at least one demand edge. Since $(V, D)$ is a forest, $C$ contains at least one supply edge. The cycle $C$ alternates between $D$-segments (maximal subpaths of $C$ all whose edges belong to $D$) and $E$-segments. Suppose there is some cycle $C$ with $|C \cap S_2| > |C \cap E|$. Among those cycles $C$, choose one such that $C$ has as few $D$-segments as possible.

We claim that for every tree $T$ of $D$, $C \cap T$ is connected. Assume not; then there is a path $P \subseteq D$ whose endpoints $u$ and $v$ belong to $V(C)$, but no edge of $P$ belongs to $C$. The two endpoints $u$ and $v$ of $P$ partition $C$ into two $u$-to-$v$ paths, $P_1$ and $P_2$. One of the two cycles $C_1 = P_1 \cup P$ and $C_2 = P_2 \cup P$ must have $|C_i \cap S_2| > |C_i \cap E|$ and has strictly fewer $D$-segments than $C$, contradicting the choice of $C$. This proves the claim.

Thus each $D$-segment of $C$ belongs to a different tree. By Lemma 4.11, every $D$-segment $P$ satisfies Equation 4.6. Moreover, since $x$ is a feasible solution to the non-negative cycle LP, $\sum_{e \in C \cap D} x_e \leq |C \cap E|$.

If $|C \cap E| = 1$, then Equation 4.6 yields $2|C \cap S_2| \leq 2 + \sum_{e \in C \cap D} x_e \leq 2 + |C \cap E| = 3$, implying $|C \cap S_2| \leq 1 = |C \cap E|$. Otherwise, thanks to selecting an independent set in the contracted graph in step 7, if there is more than one $D$-segment, every $E$-segment has at least two edges. Summing Equation 4.6 over all $D$-segments $P$ yields $2|C \cap S_2| \leq |C \cap E| + \sum_{e \in C \cap D} x_e \leq |C \cap E| + |C \cap E|$. Therefore, $|C \cap S_2| \leq |C \cap E|$ as required. □

**Lemma 4.13.** $\sum_{e \in D} x_e \leq 8(|S_1| + |S_2|)$.

*Proof.* The construction of Algorithm 11 maintains the invariant $\sum_{e \in D} x_e = \sum_{e \in D \cup A} y_e$. Since edges of $S_2^0$ have $y_e \leq 2$, artificial edges have $y_e \leq 1$, and other edges have $y_e = 0$, we

can write $\sum_{e\in D\cup A} y_e \leq 2|S_2^0| + |A|$. Observe that the number of artificial edges is at most $2|L| - 2$. By construction $|L| \leq 4|S_1|$ and $|S_2^0| \leq 4|S_2|$. Combining, we conclude

$$\sum_{e\in D} x_e \leq 2|S_2^0| + 2|L| \leq 8|S_2| + 8|S_1|.$$

□

Lemma 4.13 immediately implies $\max\{|S_1|, |S_2|\} \geq \frac{1}{16}\sum_{e\in D} x_e$, finishing the proof of Theorem 4.6.

### 4.3.3 Proof of Theorem 4.2: Improving the approximation ratio from 32 to 24

Lemma 4.13 also implies $\max\{|S_1|, \frac{1}{2}|S_2|\} \geq \frac{1}{24}\sum_{e\in D} x_e$. This yields Theorem 4.2: we have $|S_1|$ pairwise disjoint $D$-cuts from Algorithm 10, the singleton cuts of the leaves, and we also get $\frac{1}{2}|S_2|$ pairwise disjoint $D$-cuts by applying Theorem 4.8 to $S_2$. We output the larger of the two.

## 4.4 Packing Cuts (Proof of Theorem 4.8)

In this section we prove Theorem 4.8, whose statement we now recall.

**Theorem 4.8.** *Let $G$ be a planar graph and $J$ a join in $G$. Then there is a family of at least $\frac{|J|}{2}$ pairwise disjoint $J$-cuts in $G$, and such a family can be computed in polynomial time.*

Recall that a *$J$-cut* is a simple cut that contains exactly one edge from $J$. An example by Korach and Penn [98] shows that the factor 2 in Theorem 4.8 is best possible. Without planarity we cannot hope for any constant factor, e.g.if $G$ is a complete graph and $J$ a perfect matching in $G$, then there are no two disjoint $J$-cuts.

Let $J$ be a join in a planar graph $G$. Middendorf and Pfeiffer [117] showed that it is NP-complete to decide whether there are $|J|$ pairwise disjoint $J$-cuts. Hence also finding the maximum number of pairwise disjoint $J$-cuts is NP-hard. However, we give an approximation algorithm in the following. Our proof is inspired by the paper of Korach and Penn [98].

Recall that a family of subsets of $V$ is *laminar* if for any two of those sets either they are disjoint or one is a subset of the other. It is *cross-free* if for any two of those sets, either they are disjoint, or one is a subset of the other, or their union is $V$. We will show the following:

**Lemma 4.14.** *Let $G = (V, E)$ be a connected planar graph. Let $\mathcal{L}$ be a laminar family of (not necessarily distinct) vertex sets of $G$ such that $\delta(U)$ is a simple cut for all $U \in \mathcal{L}$ and every edge is contained in at most two of these cuts. Then there exists a polynomial-time algorithm to compute a sub-family $\mathcal{L}' \subseteq \mathcal{L}$ such that $|\mathcal{L}'| \geq \frac{1}{4}|\mathcal{L}|$ and the cuts $\delta(U)$ for $U \in \mathcal{L}'$ are pairwise disjoint.*

Again, the example by Korach and Penn [98] shows that the factor 4 in Lemma 4.14 is best possible. Without planarity we cannot hope for any constant factor, e.g.if $G$ is a complete graph and $\mathcal{L}$ contains all singletons.

Before we prove Lemma 4.14, let us first see how it implies Theorem 4.8.

*Proof.* (Theorem 4.8) Let $J$ be a join in $G$. By Theorem 4.7, we can compute vertex sets $U_1, \ldots, U_{2|J|}$ (not necessarily distinct) that form a laminar family $\mathcal{L}$, such that $|\delta(U_i) \cap J| = 1$ for all $i$ and every edge is contained in at most two of the cuts $\delta(U_i)$, $i = 1, \ldots, 2|J|$. Before applying Lemma 4.14, we want to make sure that these cuts are all simple cuts, and this will be achieved in the following by a sequence of transformations.

For $i = 1, \ldots, 2|J|$ let $U_i' \subseteq U_i$ be the vertex set of the connected component of $G[U_i]$ for which $|\delta(U_i') \cap J| = 1$. Then $\delta(U_i') \subseteq \delta(U_i)$. We claim that the sets $U_1', \ldots, U_{2|J|}'$ form a laminar family $\mathcal{L}'$. Let $1 \leq i < j \leq 2|J|$. If $U_i \cap U_j = \emptyset$, then $U_i' \cap U_j' = \emptyset$. If $U_i \subseteq U_j$, then the vertex sets of the connected components of $G[U_i]$ are subsets of the vertex sets of the connected components of $G[U_j]$, so $U_i'$ is either disjoint from $U_j'$ or a subset of $U_j'$. The case $U_j \subseteq U_i$ is symmetric.

Now $G[U_i']$ is connected for all $i$. For $i = 1, \ldots, 2|J|$ let $U_i'' \subseteq V \setminus U_i'$ be the vertex set of the connected component of $G[V \setminus U_i']$ for which $|\delta(U_i'') \cap J| = 1$. Then $\delta(U_i'') \subseteq \delta(U_i') \subseteq \delta(U_i)$. Note that for each $i$, $G[U_i'']$ and $G[V \setminus U_i'']$ are connected and $|\delta(U_i'') \cap J| = 1$. In other words, $\delta(U_1''), \ldots, \delta(U_{2|J|}'')$ are $J$-cuts.

We claim that the sets $U_1'', \ldots, U_{2|J|}''$ form a cross-free family $\mathcal{L}''$. Let $1 \leq i < j \leq 2|J|$. If $U_i' \subseteq U_j'$, then the vertex sets of the connected components of $G[V \setminus U_j']$ are subsets of the vertex sets of the connected components of $G[V \setminus U_i']$, so $U_j''$ is either disjoint from $U_i''$ or a subset of $U_i''$. The case $U_j' \subseteq U_i'$ is symmetric.

It remains to consider the case $U_i' \cap U_j' = \emptyset$. Suppose $X := V \setminus (U_i'' \cup U_j'')$ is nonempty. Since $G$ is connected, we have $\emptyset \neq \delta(X) = \delta(U_i'' \cup U_j'') \subseteq \delta(U_i' \setminus U_j'') \cup \delta(U_j' \setminus U_i'')$, where the last step is because all edges with exactly one endpoint in $U_i''$ have the other endpoint in $U_i'$, and all edges with exactly one endpoint in $U_j''$ have the other endpoint in $U_j'$. Hence at least one of $U_i' \setminus U_j''$ or $U_j' \setminus U_i''$ is nonempty. Suppose, without loss of generality, $U_i' \setminus U_j'' \neq \emptyset$. Because $G[U_i']$ is connected, $U_i'$ is a subset of the vertex set of a connected component of $G[V \setminus U_j']$. So $U_i' \cap U_j'' = \emptyset$. But then $U_j''$ is a subset of the vertex set of a connected component of $G[V \setminus U_i']$, and thus it is a subset of $U_i''$ or disjoint from $U_i''$.

So indeed $\mathcal{L}''$ is cross-free. To obtain a laminar family, choose $v \in V$ arbitrarily, and for every $U'' \in \mathcal{L}''$ with $v \in U''$, replace $U''$ by $V \setminus U''$. We obtain a laminar family $\mathcal{L}'''$ of $2|J|$ (not necessarily distinct) vertex sets such that $\delta(U)$ is a $J$-cut for all $U \in \mathcal{L}'''$ and every edge is contained in at most two of these cuts. It is obvious that all the above steps can be performed in polynomial time. Applying Lemma 4.14 concludes the proof. □

To prove Lemma 4.14, consider the following recursive algorithm:

**Input:** a planar graph $G = (V, E)$ and a laminar family $\mathcal{L}$ of (not necessarily distinct) nonempty subsets of $V$ such that $G[U]$ and $G[V \setminus U]$ are connected for all $U \in \mathcal{L}$ and every edge is contained in at most two of the cuts $\delta(U)$, $U \in \mathcal{L}$.
**Output:** a subset $\mathcal{L}' \subseteq \mathcal{L}$ such that the cuts $\delta(U)$, $U \in \mathcal{L}'$, are pairwise disjoint
**if** *the elements of* $\mathcal{L}$ *are pairwise disjoint* **then** /* case 1 */
  Consider the planar graph $P$ obtained by contracting each $U \in \mathcal{L}$ into a single vertex and deleting all other vertices. Compute a 4-coloring on $P$.
  **return** a subset of $\mathcal{L}$ that corresponds to a maximum-cardinality color class
**if** *there exist* $U_1, U_2 \in \mathcal{L}$ *such that* $U_1 = U_2$ **then** /* case 2 */
  **return** $\{U_1\} \cup$ CutPacking$(G, \mathcal{L} \setminus \{U_1, U_2\})$
/* case 3 */
Let $\overline{U} \in \mathcal{L}$ be a set that is not minimal but all sets $U_1, \ldots, U_l \in \mathcal{L}$ that are proper subsets of $\overline{U}$ are minimal (inclusionwise).
Consider the planar graph $P$ obtained by contracting each $U_i$ into a *normal* vertex, deleting all other vertices in $\overline{U}$, and contracting all vertices outside $\overline{U}$ into a single *special* vertex.
Find a 4-coloring of $P$ and choose a color class $\mathcal{K}$ with the largest number of normal vertices. If all color classes contain $\frac{l}{4}$ normal vertices, choose the one that contains the special vertex. Let $\mathcal{L}' \subset \mathcal{L}$ be the set of $U \in \mathcal{L}$ that correspond to normal vertices in $\mathcal{K}$.
**if** $\mathcal{K}$ *contains more than* $\frac{l}{4}$ *normal vertices* **then** /* case 3a */
  **return** $\mathcal{L}' \cup$ CutPacking$(\mathcal{L} \setminus \{\overline{U}, U_1, \ldots, U_l\})$
**else** /* case 3b */
  **return** $\mathcal{L}' \cup$ CutPacking$(\mathcal{L} \setminus \{U_1, \ldots, U_l\})$

**Algorithm 12:** CutPacking .

*Proof.* (Lemma 4.14) We show the assertion by induction on $|\mathcal{L}|$. We say here that a subset $\mathcal{L}' \subseteq \mathcal{L}$ is *stable* if all $\delta(U)$ for $U \in \mathcal{L}'$ are pairwise disjoint. We apply Algorithm 12. Note that the graphs $P$ in step 2 and step 7 are indeed planar because we only contract connected vertex sets. By Theorem 1.1, the algorithm runs in polynomial time. We start with the base case of our induction.

**Case 1:** The elements of $\mathcal{L}$ are pairwise disjoint. Let $\mathcal{L}' \subseteq \mathcal{L}$ be the solution returned. $\mathcal{L}'$

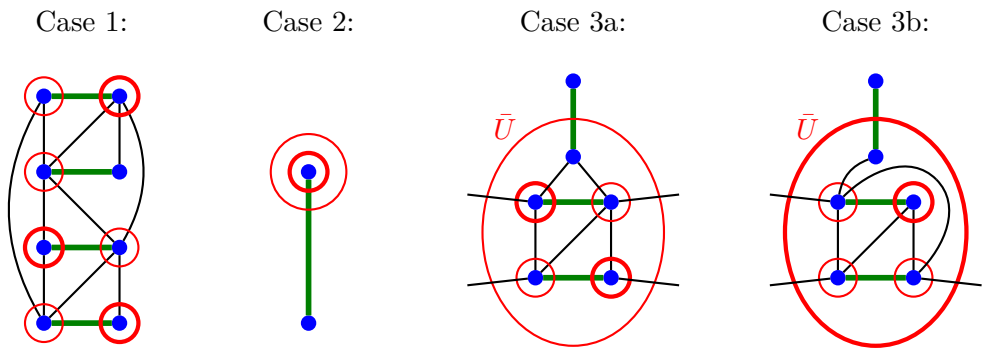



Figure 4.2: The different cases in the proof of Lemma 4.14.

corresponds to an independent set in $P$, so it is stable and has size at least $|V(P)|/4 = |\mathcal{L}|/4$. Notice that when $\mathcal{L} = \emptyset$, case 1 applies, and the empty set is returned.

**Case 2:** Some $U_1, U_2 \in \mathcal{L}$ are equal. By the induction hypothesis CutPacking$(G, \mathcal{L} \setminus \{U_1, U_2\})$ is stable and CutPacking$(G, \mathcal{L} \setminus \{U_1, U_2\})$ has size at least $(|\mathcal{L}|-2)/4$. First, the solution returned has size $\frac{|\mathcal{L}|-2}{4}+1 = \frac{|\mathcal{L}|+2}{4} > \frac{|\mathcal{L}|}{4}$. To prove that $\{U_1\} \cup$ CutPacking$(G, \mathcal{L} \setminus \{U_1, U_2\})$ is stable, we only have to check that $\delta(U_1) \cap \delta(U) = \emptyset$ for all $U \in \mathcal{L} \setminus \{U_1, U_2\}$. This holds since any edge in $\delta(U_1)$ is also contained in $\delta(U_2)$ and is contained in at most two of the cuts.

**Case 3:** There exists a set $\overline{U} \in \mathcal{L}$ that is not minimal but all sets $U_1, \ldots, U_l \in \mathcal{L}$ that are proper subsets of $\overline{U}$ are minimal. Therefore the sets $U_i$ $(i = 1, \ldots, l)$ are pairwise disjoint. Then consider the graph $P$ constructed in step 7 of the algorithm and a partition of its vertex set into four independent sets. Now we consider two subcases.

If one of these independent sets contains at least $\frac{l+1}{4}$ normal vertices (case 3a), then by induction hypothesis, the solution has size at least $\frac{l+1}{4} + \frac{|\mathcal{L}|-(l+1)}{4} = \frac{|\mathcal{L}|}{4}$. Otherwise (case 3b) all these independent sets contain exactly $\frac{l}{4}$ normal vertices and we break ties by taking the one that contains the special vertex. Using the induction hypothesis again, the solution returned has size at least $\frac{l}{4} + \frac{|\mathcal{L}|-l}{4} = \frac{|\mathcal{L}|}{4}$.

After using induction hypothesis, this solution is stable because any edge in $\delta(U_i) \cap \delta(U)$ for $U \in \mathcal{L}$ such that $U \subseteq V \setminus \overline{U}$ would also be contained in $\delta(\overline{U})$, but no edge is contained in three of the cuts. If $\overline{U} \in$ CutPacking$(\mathcal{L} \setminus \{U_1, \ldots, U_l\})$, then the special vertex is in the independent set $\mathcal{K}$, and hence $\delta(\overline{U}) \cap \delta(U_i) = \emptyset$ for all $U_i \in \mathcal{L}'$. □

## 4.5 Max-Multiflow Min-Multicut gap (Proof of Theorem 4.4)

### 4.5.1 Linear Programming Relaxations and Integrality Gaps

EdgeDisjointPaths consists of finding as many $D$-cycles as possible. If $\mathcal{C}$ denotes the set of all $D$-cycles, then we look for a maximum subset of $\mathcal{C}$ whose elements are pairwise disjoint. The natural fractional relaxation is (along with its linear programming dual):

$$\begin{aligned} \max \sum_{C \in \mathcal{C}} f_C & \\ \sum_{C \in \mathcal{C}: e \in C} f_C \leq 1 & \quad \forall e \in D \dot\cup E \\ f_C \geq 0 & \quad \forall C \in \mathcal{C}. \end{aligned} \qquad \begin{aligned} \min \sum_{e \in D \dot\cup E} y_e & \\ \sum_{e \in C} y_e \geq 1 & \quad \forall C \in \mathcal{C} \\ y_e \geq 0 & \quad \forall e \in D \dot\cup E \end{aligned}$$

Figure 4.3: The $D$-cycle packing LP and its dual, the $D$-cycle covering LP.

If we go to the planar dual graph and let $\mathcal{U}$ denote the set of vertex sets inducing $D$-cuts, these LPs become

It turns out that the $D$-cut packing LP is equivalent to the non-negative cycle LP. We need the following well-known generalization of Theorem 4.7 (see, e.g., [134] and the references therein):

$$\begin{aligned} \max \sum_{U\in\mathcal{U}} f_U & & \qquad \min \sum_{e\in D\,\dot\cup\, E} y_e & \\ \sum_{U\in\mathcal{U}:e\in\delta(U)} f_U \le 1 & \quad \forall e\in D\,\dot\cup\, E & \qquad \sum_{e\in\delta(U)} y_e \ge 1 & \quad \forall U\in\mathcal{U} \\ f_U \ge 0 & \quad \forall U\in\mathcal{U}. & \qquad y_e \ge 0 & \quad \forall e\in D\,\dot\cup\, E \end{aligned}$$

Figure 4.4: The $D$-cut packing LP and its dual, the $D$-cut covering LP.

**Theorem 4.15.** *For every graph $G=(V,E)$ with capacities $u: E\to\mathbb{R}_{\ge 0}$ and every set $J$ of edges such that $u(C\cap J)\le u(C\setminus J)$ for every cycle $C$, one can compute in strongly polynomial time a laminar family $\mathcal{L}$ of vertex sets with weights $w:\mathcal{L}\to\mathbb{R}_{>0}$ such that $|\delta(U)\cap J|=1$ for all $U\in\mathcal{L}$, $\sum_{U\in\mathcal{L}} w(U)=\sum_{e\in J} u(e)$, and $\sum_{U\in\mathcal{L}:e\in\delta(U)} w(U)\le u(e)$ for all $e\in E$. If $u$ is integral, than $w$ can be chosen half-integral.*

Using this, we can show:

**Lemma 4.16.** *The $D$-cut packing LP is equivalent to the non-negative cycle LP (4.5). Their values are the same, from an optimum solution to the former we can get an optimum solution to the latter in polynomial time, and vice versa.*

*Proof.* For any feasible solution $f$ of the $D$-cut packing LP define a vector $x$ by $x_e := \sum_{U\in\mathcal{U}:e\in\delta(U)} f_U$ for $e\in D$. Then $x$ is a feasible solution to the non-negative cycle LP because for every cycle $C$ in $G+H$ we have

$$\begin{aligned} \sum_{e\in C\cap D} x_e &= \sum_{e\in C\cap D} \sum_{\substack{U\in\mathcal{U}:\\ e\in\delta(U)}} f_U = \sum_{U\in\mathcal{U}} f_U |C\cap D\cap\delta(U)| \\ &\le \sum_{U\in\mathcal{U}} f_U |C\cap E\cap\delta(U)| = \sum_{e\in C\cap E} \sum_{\substack{U\in\mathcal{U}:\\ e\in\delta(U)}} f_U \le |C\cap E|. \end{aligned}$$

Here the first inequality holds because a cycle $C$ and a cut $\delta(U)$ intersect in an even number of edges, and for $U\in\mathcal{U}$ at most one edge in the intersection belongs to $D$. We also have $\sum_{e\in D} x_e = \sum_{U\in\mathcal{U}} f_U$.

Conversely, let $x$ be a feasible solution to the non-negative cycle LP. Define $u(e):=x_e$ for $e\in D$ and $u(e):=1$ for $e\in E$. By Theorem 4.15 (applied to $G+H$ and $J:=D$), one can compute a laminar family $\mathcal{L}$ of vertex sets with weights $w:\mathcal{L}\to\mathbb{R}_{>0}$ such that $|\delta(U)\cap D|=1$ for all $U\in\mathcal{L}$, $\sum_{U\in\mathcal{L}} w(U)=\sum_{e\in D} u(e)$, and $\sum_{U\in\mathcal{L}:e\in\delta(U)} w(U)\le u(e)$ for all $e\in D\,\dot\cup\, E$. Set $f_U:=w(U)$ for $U\in\mathcal{L}$ and $f_U:=0$ otherwise. Then $f$ is a feasible solution to the $D$-cut packing LP with $\sum_{U\in\mathcal{U}} f_U=\sum_{U\in\mathcal{L}} w(U)=\sum_{e\in D} u(e)=\sum_{e\in D} x_e$. □

This implies:

**Corollary 4.17.** *If $G+H$ is planar, the integrality gap of the $D$-cut packing LP (and hence the integrality gap of the $D$-cycle packing LP) is at least 2 and at most 24.*

*Proof.* We showed in the proof of Theorem 4.2 that there is an algorithm that computes an integral solution of the $D$-cut packing LP of at least $\frac{1}{24}$ times the value of the non-negative cycle LP. By Lemma 4.16 this implies the upper bound.

The lower bound of 2 is attained for $G+H=K_4$ (the complete graph on 4 vertices) if $D$ is a perfect matching in $G+H$, see Figure 4.5 (a). □

For the non-negative cycle LP the integrality gap could be smaller. We know that it is between $\frac{3}{2}$ (shown by $G+H=K_4$ and $D=\delta(u)$ for an arbitrary vertex $u$, see Figure 4.5 (b)) and 16 (shown by Theorem 4.6).

We now observe that the dual LPs have integrality gap at most 2.

**Lemma 4.18.** *The integrality gap of the $D$-cut covering LP (and hence the integrality gap of the $D$-cycle covering LP if $G+H$ is planar) is at most 2, and it is at least $\frac{3}{2}$ even if $G+H$ is planar.*

*Proof.* The lower bound follows from an example by Cheriyan, Karloff, Khandekar, and Könemann [29] for the tree augmentation problem (let $D$ contain the tree edges and $E$ contain the links of this example, then their LP is equivalent to the $D$-cut covering LP). See Figure 4.5 (c).

We now show the upper bound. The $D$-cut covering LP is equivalent to

$$\begin{aligned} \min \sum_{e\in D\,\dot\cup\, E} y_e & & (4.7)\\ \sum_{e\in\delta(U)} y_e + \sum_{e\in\delta(U)\cap D} z_e \geq 2 & \qquad \forall U\subseteq V \text{ with } D\cap\delta(U)\neq\emptyset \\ y_e \geq 0 & \qquad \forall e\in D\,\dot\cup\, E \\ 0\leq z_e\leq 1 & \qquad \forall e\in D \end{aligned}$$

because in Equation 4.7 we can assume that $z_e=1$ for all $e\in D$, and then it is exactly the same as the $D$-cut covering LP. Now Equation 4.7 is a survivable network design LP of the type

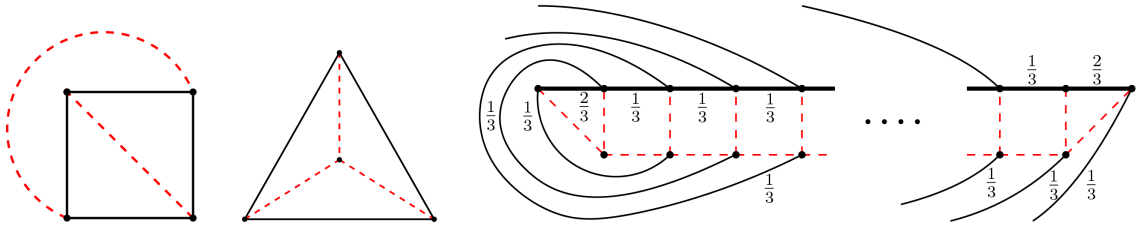
$$\begin{aligned} \min \sum_{e\in E'} c(e)x_e & \\ \sum_{e\in\delta(U)} x_e \geq f(U) & \qquad \forall U\subseteq V \\ 0\leq x_e\leq 1 & \qquad \forall e\in E' \end{aligned}$$

for the graph whose edge set $E'$ consists of an edge $e$ of cost $c(e)=1$ for each $e\in D\,\dot\cup\, E$ and another parallel edge $e'=\{v,w\}$ with $c(e')=0$ for each $\{v,w\}\in D$. The requirement function $f:2^V\to\mathbb{Z}_{\geq 0}$ is given by $f(U)=2$ if $D\cap\delta(U)\neq\emptyset$ and $f(U)=0$ otherwise. This function $f$ is easily seen to be proper (and thus weakly supermodular), and hence Jain's iterative rounding theorem [86] tells that there is an integral feasible solution $(y,z)$ to Equation 4.7 of cost at most twice the LP value. Then $y$ is an integral feasible solution to the $D$-cut covering LP of the same cost. □

We remark that even if just $G$ is planar (not $G+H$) the integrality gap of the $D$-cycle covering LP is bounded by a (large) constant [141].

### 4.5.2 Multiflows and Multicuts

A natural generalization of EDP takes as input, in addition to $G$ and $H$, a capacity function $u:D\,\dot\cup\, E\to\mathbb{Z}_{>0}$ and asks for a maximum number of $D$-cycles such that each edge $e\in D\,\dot\cup\, E$ belongs to at most $u(e)$ of them. If $u(e)=1$ for all $e\in D\cup E$, this is EDP.




(a) The integrality gap of the $D$-cut packing LP is at least 2.

(b) The integrality gap of the non-negative cycle LP is at least $\frac{3}{2}$.

(c) The integrality gap of the $D$-cut covering LP is at least $\frac{3}{2}$. Bold black edges form an optimum integral solution. Values $\frac{1}{3}$ and $\frac{2}{3}$ on the black edges form a feasible fractional solution.

Figure 4.5: Examples for lower bounds of integrality gaps. Solid black edges belong to $E$, dashed red edges belong to $D$.

This generalization reduces to EDP by replacing each edge $e$ by $u(e)$ parallel edges of unit capacity. In the planar dual, this corresponds to replacing $e$ by a path with $u(e)$ edges. If $u$ is given in binary representation, this reduction should of course not be performed explicitly. Nevertheless our algorithm can easily be generalized to run in polynomial time for general capacities. A dual of a weighted planar graph can be coded by a weighted planar graph where an edge $e$ of weight $w_e$ represents a path of length $w_e$.

All algorithms can be easily extended to weighted instances, as we elaborate now. Solving the weighted version of the non-negative cycle LP is straightforward. Algorithm 10 does not change. In Algorithm 11, an edge $e \in D$ with weight $u(e)$ has initial budget $y_e = u(e)x_e$. If, during step 5, an edge has budget more than 2, we pick multiple copies of it, namely $\lceil \frac{y_e}{2} \rceil$ many copies. Once we have a weighted feasible solution $J$ that respects the non-negative cycle condition, we apply Theorem 4.15 to get a weighted laminar family of $J$-cuts. This can still be viewed as a half-integral packing of $J$-cuts in the graph in which every edge $e$ is replaced by a path of length $u(e)$, and the proof of Theorem 4.8 works as before, without the need to store that graph explicitly. The final feasible solution of EDP can be represented by an integral flow of a certain value for some demand edges. These flows can be decomposed into (and hence represented by) at most $|E|$ paths by standard flow decomposition.

If $u(e) = \infty$ (or large enough) for all $e \in D$, the problem has been called the maximum integer multiflow problem. (Note that $u(e) = \infty$ for all $e \in D$ can in fact be assumed without loss of generality because a demand edge $e = \{v, w\}$ with capacity $u(e)$ can be replaced equivalently by a demand edge $\{v, x\}$ of infinite capacity and a supply edge $\{x, w\}$ of capacity $u(e)$, where $x$ is a new vertex.)

If we do not require integrality, the maximum multiflow problem is of course a linear program:

$$\begin{aligned} \max \sum_{C \in \mathcal{C}} f_C & \\ \sum_{C \in \mathcal{C}, C \ni e} f_C \leq u(e) & \quad \forall e \in E \\ f_C \geq 0 & \quad \forall C \in \mathcal{C}. \end{aligned} \qquad \begin{aligned} \min \sum_{e \in E} u(e) y_e & \\ \sum_{e \in C \cap E} y_e \geq 1 & \quad \forall C \in \mathcal{C} \\ y_e \geq 0 & \quad \forall e \in E \end{aligned}$$

Figure 4.6: The multiflow LP and its dual, the multicut LP.

The feasible solutions to the multiflow LP are called multiflows (here we use the form after flow decomposition). The integral feasible solutions to the dual LP correspond to edge sets $F \subseteq E$ such that deleting $F$ destroys all $D$-cycles. They are called *multicuts*. The *capacity* of a multicut is the total capacity of its edges. Of particular interest is the worst ratio of the minimum capacity of a multicut and the maximum value of an integer multiflow. The (integer version of the) famous max-flow min-cut theorem says that this ratio is 1 if $|D| = 1$. However in general, even if $G$ is sub-cubic and planar the ratio can be as large as $\Theta(|D|)$ [63]. Besides when $G$ is a tree (then the ratio is 2 [63]), when $G$ is planar and has bounded tree-width [12] or when $G + H$ is series-parallel [41] (then the ratio is 1), very few cases are known where the ratio can be bounded by a constant. We can now show a constant upper bound when $G + H$ is planar. This is Theorem 4.4, which we restate here:

**Theorem 4.4.** *If $G + H$ is planar, the minimum cardinality of a multicut is at most 48 times the maximum value of an integer multiflow.*

*Proof.* The integrality gap of the multiflow LP equals the integrality gap of the $D$-cycle packing LP, as the two problems can be reduced to each other: the gap cannot be smaller because given an instance of the $D$-cycle packing LP, as said above, we can replace a demand edge $\{v, w\}$ by a demand edge $\{v, x\}$ of infinite (or very large) capacity and a supply edge $\{x, w\}$ of capacity 1 to form a multiflow instance.

Conversely, the gap also cannot be larger since we can reduce the multiflow problem to $D$-cycle packing by replacing every edge $e$ by $u(e)$ parallel edges with unit capacity. Note that both reductions preserve planarity. The integrality gap of the multicut LP equals the integrality gap of the $D$-cycle covering LP by the same argument. Now Corollary 4.17 and Lemma 4.18 imply the assertion. □

Determining the exact integrality gaps remains an open question. Without the planarity assumption, all the LPs except for the $D$-cut covering LP have unbounded integrality gap (for the $D$-cycle packing LP, a well-known example in [63] shows that the gap is in the order of $\Omega(\sqrt{n})$, even when $G$ is planar, subcubic, and all demand pairs lie in the boundary of the outer face of $G$; for the $D$-cut packing LP and the non-negative cycle LP, consider $G + H = K_n$ and $D = \delta(v)$ for some vertex $v$; for the $D$-cycle covering LP, let $G$ be a bounded-degree expander graph with $n$ vertices and $D$ the set of $\frac{n^2}{4}$ vertex pairs with largest distance [62]).

## 4.6 NP-completeness of NonNegativeCycles

In this section we prove that NONNEGATIVECYCLES is **NP**-hard. In fact, we consider the NONNEGATIVECYCLES decision problem, which takes as input an instance of NONNEGATIVECYCLES and an integer $k$, and asks whether there exists a solution of cardinality $k$.

**Theorem 4.19.** *The* NONNEGATIVECYCLES *decision problem is **NP**-complete even when $G + H$ is planar.*

To prove membership in **NP**, here is a polynomial-time algorithm that, given an instance $G = (V, E)$ and $H = (V, D)$ and a subset $D' \subseteq D$ of cardinality $k$, verifies whether $D'$ is a solution: As in the proof of Lemma 4.5, assign weight 1 to edges of $E$, $-1$ to edges

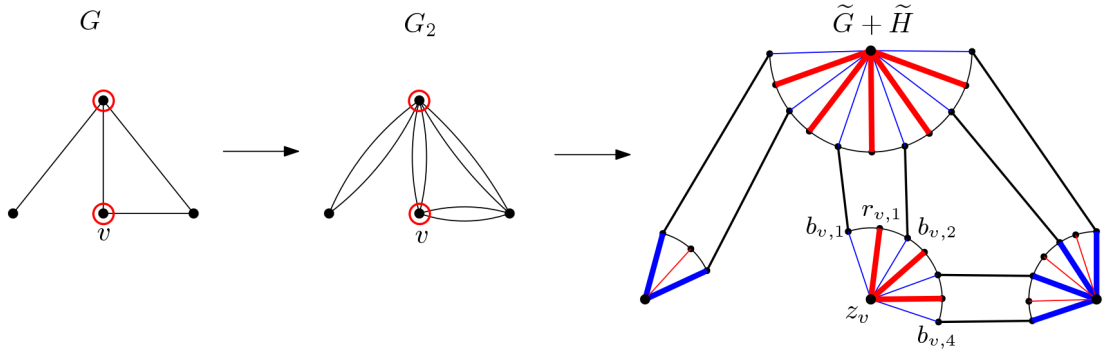



Figure 4.7: The reduction from vertex cover in planar graphs. On the left-hand side picture, the vertices surrounded by a circuit form a minimum vertex cover $X$ in $G$. On the right-hand side, the supply edges are shown in black, and the demand edges are colored (red or blue). The solution to NNC corresponding to $X$ is the set of thicker edges. It is obtained by taking the red edges in each gadget associated to a vertex in $X$ and the blue edges otherwise.

of $D'$, and 0 to edges of $D \setminus D'$; compute a minimum-weight $\emptyset$-join; then $D'$ is a solution if and only if that minimum weight is 0.

To prove **NP**-completeness, recall the vertex cover problem: given a graph $G$ and an integer $k$, it asks whether $G$ has a vertex cover of size $k$. This problem is well-known to be **NP**-complete even in planar graphs [60]. We give a polynomial-time transformation from that problem.

Let $G = (V, E)$ be a planar graph. Let $G_2 := (V, E \dot\cup E)$ denote the (multi)graph where each edge of $G$ is duplicated. We construct an instance $\widetilde{G} + \widetilde{H} = (\widetilde{V}, \widetilde{E} \dot\cup \widetilde{D})$ of NonNegativeCycles as follows, illustrated in Figure 4.7.

For each vertex $v \in V$ of degree $d = d_G(v)$ in $G$, there is a *gadget* $(\widetilde{V}_v, \widetilde{E}_v \dot\cup \widetilde{D}_v)$ built as follows. The vertex set $\widetilde{V}_v$ consists of $4d$ vertices $z_v, b_{v,1}, r_{v,1}, b_{v,2}, \ldots, r_{v,2d-1}, b_{v,2d}$. The supply edge set $\widetilde{E}_v$ consists of edges $\{b_{v,i}, r_{v,i}\}$ and $\{r_{v,i}, b_{v,i+1}\}$, for $1 \le i \le 2d-1$. The demand edge set $\widetilde{D}_v$ consists of $\widetilde{B}_v \cup \widetilde{R}_v$, where $\widetilde{B}_v := \{\{z_v, b_{v,i}\} \mid 1 \le i \le 2d\}$ and $\widetilde{R}_v := \{\{z_v, r_{v,i}\} \mid 1 \le i \le 2d-1\}$.

To complete the construction of $(\widetilde{G}, \widetilde{H})$, starting from a planar embedding of $G_2$, replace each vertex $v$ by the corresponding gadget, and each edge $e = \{u, v\}$ in $G_2$ by a supply edge $\{b_{u,i}, b_{v,j}\}$ in $\widetilde{E}$, where $i$ and $j$ are chosen so that $\widetilde{G} + \widetilde{H}$ is planar.

The construction of $\widetilde{G} + \widetilde{H}$ can be done in polynomial time. Hence the following lemma completes the proof of Theorem 4.19.

**Lemma 4.20.** *Let $G$ be a graph. There exists a vertex cover in $G$ of size $k$ if and only if there exists a subset $D' \subseteq \widetilde{D}$ of size $4|E| - k$ that is a feasible solution to* NonNegative-Cycles *in $\widetilde{G} + \widetilde{H}$.*

*Proof.* ($\Rightarrow$) Let $X$ be a vertex cover of size $k$ in $G$. Define $D' := (\bigcup_{v \in X} \widetilde{R}_v) \cup (\bigcup_{v \notin X} \widetilde{B}_v)$. It is easy to check that $D'$ has size $4|E| - k$. We show now that $D'$ is a feasible solution for NonNegativeCycles. Let $\widetilde{I}$ denote the set of $|D'|$ vertices naturally associated to $D'$, taking the endpoint of each edge of $D'$ that is not $z_v$ for any $v \in V$. By construction and since $X$ is a vertex cover in $G$, $\widetilde{I}$ is an independent set in $\widetilde{G} + \widetilde{H}$. Hence $\{\delta(\{\widetilde{v}\}) \mid \widetilde{v} \in \widetilde{I}\}$ is a $D'$-cut packing. For each cycle $\widetilde{C}$ in $\widetilde{G} + \widetilde{H}$, and for each $\widetilde{v} \in \widetilde{I}$, we have

$|\widetilde{C} \cap \delta(\{\widetilde{v}\}) \cap D'| \leq |\widetilde{C} \cap \delta(\{\widetilde{v}\}) \cap \widetilde{E}|$ and thus

$$|\widetilde{C} \cap D'| = \sum_{\widetilde{v} \in \widetilde{I}} |\widetilde{C} \cap \delta(\{\widetilde{v}\}) \cap D'| \leq \sum_{\widetilde{v} \in \widetilde{I}} |\widetilde{C} \cap \delta(\{\widetilde{v}\}) \cap \widetilde{E}| \leq |\widetilde{C} \cap \widetilde{E}|.$$

($\Leftarrow$) Assume now that we are given a feasible solution $D'$ for NonNegativeCycles in $\widetilde{G} + \widetilde{H}$ of size $4|E| - k$. We define $X := \{v \in V \mid D' \cap \widetilde{D}_v \neq \widetilde{B}_v\}$. We show that $X$ is a vertex cover in $G$ with size at least $k$. If there was an edge $\{u, v\} \in E$ with $u \notin X$ and $v \notin X$, then there would be a cycle, induced by vertices $\{z_u, b_{u,i}, b_{v,j}, z_v, b_{v,j+1}, b_{u,i+1}\}$ in $\widetilde{G} + \widetilde{H}$ for some indices $i, j$, that contains four edges in $D'$ but only two supply edges. We now prove that $|X| \leq k$.

Since $D'$ is a feasible solution, we have $|D' \cap \widetilde{D}_v| \leq 2d_G(v)$ for all $v \in V$, for otherwise one could find a triangle in $\widetilde{G}_v + \widetilde{H}_v$ with two edges in $D'$. Moreover, $|D' \cap \widetilde{D}_v| = 2d_G(v)$ only if $D' \cap \widetilde{D}_v = \widetilde{B}_v$, i.e., only if $v \notin X$. This implies

$$|X| \leq \sum_{v \in V} \left(2d_G(v) - |D' \cap \widetilde{D}_v|\right) \leq 4|E| - |D'| = k.$$

□

## 4.7 Conclusion

We designed a constant-factor approximation algorithm for EdgeDisjointPaths if $G + H$ is planar.

If all demand edges lie on a single face of a planar embedding of $G$, then the planar dual $H^*$ has only one nontrivial connected component (and isolated vertices). In this case, any instance satisfying the cut condition has a complete solution [98]. Then a 3-coloring of the outerplanar graph $G[L]$ in Algorithm 10 and a small modification of Algorithm 11 (reducing the upper bounds on $y$ from 1 and 2 to $\frac{1}{2}$ and 1) yields a 4-approximation. If we could solve NonNegativeCycles in this case, we would even obtain an exact algorithm. This motivates the following open question: can NonNegativeCycles be solved in polynomial time if $H$ has only one nontrivial connected component?

One attempt, also to reduce the constant factor that we lose when rounding a solution to the non-negative cycle LP (4.5) in general, would be to find a characterization of optimal LP solutions. However, this seems to be difficult. Figure 4.8 shows that there are instances in which the unique optimum LP solution is not half-integral.

**Note:** After finishing this work, we learned from Naveen Garg, Nikhil Kumar, and András Sebő that they had independently discovered a 4-approximation for the edge-disjoint paths problem when $G + H$ is planar [61].

They proceed by first obtaining a half-integral multiflow and then using the four color theorem to round it to an integral solution. The main difference of the two works is the way such half-integral multiflows are obtained. In the paper of Garg et al. [61], it is constructed by uncrossing a fractional multiflow (see Section 5.4 in the next chapter for a definition) to construct a certain network matrix, which is known to be totally unimodular.

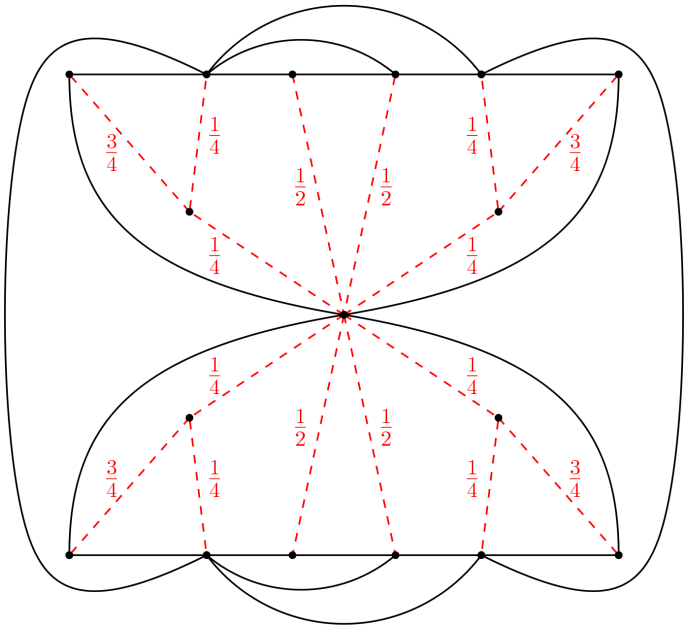



Figure 4.8: This instance shows that the non-negative cycle LP (4.5) does not always have half-integral optimal solutions. Black (solid) edges are supply edges, red (dashed) edges represent demand edges together with their value in the unique optimum fractional solution.

# 5 | Approximating Integral Multiflows on Orientable Surfaces

## 5.1 Introduction

Multi-commodity flows, or *multiflows* for short, are well-studied objects in combinatorial optimization; see, e.g., Part VII of [133]. A multiflow of maximum total value can be found in polynomial time by linear programming. Oftentimes, a multiflow must be integral, and then the problem is much harder; the well-known edge-disjoint paths problem is a special case. In Chapter 4, we gave a constant-factor approximation algorithm for maximum edge-disjoint paths and integral multiflows in *fully planar instances*, i.e., when $G + H$, the supply graph together with the demand edges, can be embedded in the plane. Garg et al. independently discovered a 4-approximation for the problem [61]. We generalize these results to surfaces of bounded genus and devise the first constant-factor approximation algorithm for that case.

Beyond using some ideas of Chapter 4 and the paper of Garg et al. [61], we need several new ingredients. We start by computing an optimal (fractional) multiflow. If this multiflow has constant value then we use the algorithm in [132] to compute an optimal solution. Otherwise like [61] we "uncross" the cycles in its support as much as possible, but uncrossing is significantly more complicated on general surfaces than in the plane. Next, we need to deal with two cases separately: depending on whether most of the fractional multiflow is on separating cycles (that case is similar to the planar case) or on non-separating cycles. In the latter case we partition the cycles into free homotopy classes and pick a set of free homotopy classes that form a family of pairwise non-crossing cycles. We define a cyclic order in each such free homotopy class, which is possible due to the uncrossing and allows for a simple greedy algorithm.

### 5.1.1 Our results

The (fractional) *maximum multiflow problem* can be described as follows. An instance consists of an undirected graph $(V, D \dot\cup E)$ whose edge set is partitioned into *demand edges*, in $D$, and *supply edges*, in $E$. We write $G = (V, E)$, $H = (V, D)$, and $G + H = (V, D \dot\cup E)$. Moreover we have a function $u : D \dot\cup E \to \mathbb{Z}_{>0}$ which defines a *capacity* $u(e)$ for each supply edge $e \in E$ and a *demand* $u(d)$ for each demand edge $d \in D$. The goal is to satisfy as much of the demand as possible by routing flow on supply edges. More precisely, we ask for an $s$-$t$-flow $f^d$ of value at most $u(d)$ for every demand edge $d = \{t, s\}$ such that the total flow on each supply edge is at most its capacity and the total value of all those flows is maximum.

It is well known that every $s$-$t$-flow can be decomposed into flows on $s$-$t$-paths and on cycles, and for integral flows there is an integral decomposition. The cycles in such a decomposition do not contribute to the value of the $s$-$t$-flow and can be ignored. An $s$-$t$-path in $(V, E)$ together with the demand edge $d = \{t, s\}$ forms a *$D$-cycle*: a cycle in $G + H$ that contains exactly one demand edge. Letting $\mathcal{C}$ denote the set of all $D$-cycles in $G + H$, we can write the maximum multiflow problem equivalently as

$$\max \sum_{C \in \mathcal{C}} f_C \text{ s.t. } \begin{cases} \sum_{C \in \mathcal{C}: C \ni e} f_C \leq u(e) & \text{for all } e \in D \dot\cup E \\ f_C \geq 0 & \text{for all } C \in \mathcal{C} \end{cases} \tag{5.1}$$

In some previous work, the problem has been defined with $u(d) = \infty$ for $d \in D$, and this variant is easily seen to be equivalent. We call the linear program (5.1) the *maximum multiflow LP*. The *maximum integral multiflow* problem is identical, except that the flow must be integral:

$$\max \sum_{C \in \mathcal{C}} f_C \text{ s.t. } \begin{cases} \sum_{C \in \mathcal{C}: C \ni e} f_C \leq u(e) & \text{for all } e \in D \dot\cup E \\ f_C \in \mathbb{Z}_{\geq 0} & \text{for all } C \in \mathcal{C} \end{cases} \tag{5.2}$$

The special case where $u(e) = 1$ for every edge $e \in D \dot\cup E$ is known as the *maximum edge-disjoint paths* problem. Even that special case is unlikely to have a constant-factor approximation algorithm for general graphs (see Section 5.1.2). Our main result is a constant-factor approximation algorithm in the case when $G + H$ can be embedded on an orientable surface of bounded genus.

**Theorem 5.1.** *There is a polynomial-time algorithm which takes as input an instance $(G, H, u)$ of the maximum integral multiflow problem such that $G + H$ is embedded on an orientable surface of genus $g$, and which outputs an integral multiflow whose value is at most a factor $O(g^2)$ smaller than the value of any integral multiflow.*

See Section 5.2 for an outline of the algorithm and the proof. We will use the maximum (fractional) multiflow as a bound for **OPT** when the value of maximum (fractional) multiflow is $\Omega(g^3)$ (see step 1). In particular this algorithm implies a bound $O(g^3)$ on the integrality gap of the LP ( 5.1) when $G + H$ has bounded genus $g$. By a slight modification of the algorithm, we can obtain a better bound on the integrality gap.

**Theorem 5.2.** *There is a polynomial-time algorithm which takes as input an instance $(G, H, u)$ of the maximum integral multiflow problem such that $G + H$ is embedded on an orientable surface of genus $g$, and which outputs an integral multiflow whose value is at most a factor $O(g^2 \log g)$ smaller than the value of any* fractional *multiflow.*

It is worth pointing out that almost all known hardness results for the maximum edge-disjoint paths problem hold even when $G$ is planar (see Section 5.1.2). Theorem 5.1, along with the previous chapter and the recent paper [61], highlight that for tractability one needs more than the planarity of $G$ alone. The topology of $G + H$ together plays an important role.

The dual LP of (5.1) is:

$$\min \sum_{e \in D \dot\cup E} u(e) y_e \text{ s.t. } \begin{cases} \sum_{e \in C} y_e \geq 1 & \text{for all } C \in \mathcal{C} \\ y_e \geq 0 & \text{for all } e \in D \dot\cup E \end{cases} \tag{5.3}$$

and this may be called the *minimum fractional multicut problem*. The *minimum multicut problem* results from replacing the inequality $y_e \geq 0$ in (5.3) by $y_e \in \{0, 1\}$ for all edges $e \in D \dot{\cup} E$. Again, many previous work considered the equivalent special case where $u(d) = \infty$ for $d \in D$, in which case no dual variable for demand edges is needed. By weak duality, the value of any multiflow is at most the capacity of any multicut. Using Theorem 5.2 and a previous result of [141], we obtain (in Section 5.8.2):

**Corollary 5.3.** *For any instance $(G, H, u)$ of the maximum integral multiflow problem such that $G+H$ is embedded on an orientable surface of genus $g$, the minimum capacity of a multicut is at most $O(g^{3.5} \log g)$ times the maximum value of an integral multiflow.*

In general the integral multiflow-multicut gap[1], and even the integrality gap of Equation 5.1, can be as large as $\Theta(|D|)$, even when $G$ is planar and $G + H$ is embedded in the projective plane [63] (see Figure 5.7). In this paper we consider orientable surfaces only. Corollary 5.3 states that the gap becomes constant when $G + H$ has bounded genus. So far very few such constant integral multiflow-multicut gaps are known, for example when $G$ is a tree [63], or when $G + H$ is planar, as recently shown in the previous chapter and [61].

### 5.1.2 Related Work

See Section 4.1.2 for approximation algorithms and inapproximability results for the edge-disjoint paths problem and the maximum integral multiflow problem.

**Minimum multicut problem.** The minimum multicut problem is **NP**-hard even when there are only three demand edges [42]. In general, assuming that the Unique Games conjecture holds, there is no $O(1)$-approximation [26], but a $O(\log |D|)$-approximation algorithm [62]. Better approximations also have been shown for special cases; see [63, 141] and the references therein. In particular, when $G + H$ is planar, Klein et al. [95] gave an approximation scheme. When $G$ has genus $g$, an FPT-approximation scheme with parameters of $g$ and $|D|$ has been proposed [38].

**Tools from topology.** The design of multiflows on surfaces is closely related to the properties of systems of curves on surfaces. More specifically, a set of essential curves that are any two curves cross at most once and are not freely homotopic is called a *1-system* [88]. In a recent breakthrough, Przytycki [128] proved that the size of a 1-system on a closed orientable surface of genus $g$ is $O(g^3)$, improving on the previous exponential upper bound by [113]. Very recently, this number was shown to be $O(g^2 \log g)$ (Theorem 5.24) by Greene [64], which almost matches the lower bound $\Omega(g^2)$ on the size of 1-systems [113]. We will use in Section 5.6 the fact that any curve in a 1-system crosses at most $O(g^2)$ other curves [128] (Theorem 5.17).

### 5.1.3 Preliminaries

Consider an instance $(G, H, u)$ of the maximum integral multiflow problem. Throughout the chapter, we assume that the graph $G + H$ is connected, otherwise we can run the

[1]There is a closely related, but different, notion of integral flow-cut gap introduced in [28]: they study the smallest constant $c$ such that whenever $u(C \cap E) \geq u(C \cap D)$ for every cut $C$ (the cut condition), there is an integral multiflow satisfying all demands and violating capacities by at most a factor $c$.

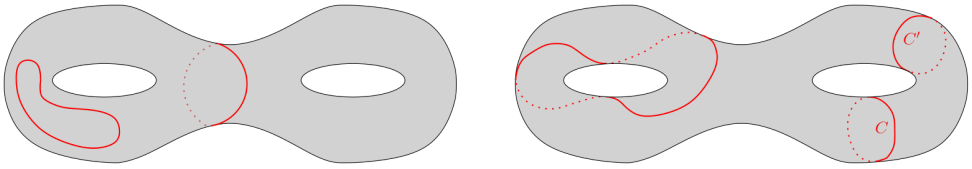



Figure 5.1: Some cycles on an orientable surface of genus 2. On the left, two *separating* cycles. On the right, three *non-separating* cycles. $C$ and $C'$ are *freely homotopic* and their union disconnects the surface.

algorithm on each of its connected components.

**Homotopy.** Given a surface $\mathbb{S}$, a (simple) **topological cycle** is a continuous injective map $\gamma$ from the unit cycle $S^1 := \{z \in \mathbb{C},\ ||z|| = 1\}$ to $\mathbb{S}$. Two topological cycles $\gamma_1$ and $\gamma_2$ are **freely homotopic** if there exists a continuous function $\varphi : [0,1] \times S^1 \to \mathbb{S}$ such that $\varphi(0,\cdot) = \gamma_1$ and $\varphi(1,\cdot) = \gamma_2$. Intuitively, cycle $\gamma_1$ is transformed into cycle $\gamma_2$ by continuously moving it along the surface. Free homotopy is an equivalence relation.

Given an embedding of the graph $G + H$ on $\mathbb{S}$, we say that a cycle $C$ in $G + H$ *is represented by* a topological cycle $\gamma$ of $\mathbb{S}$ if the image of $\gamma$ is the embedding of $C$ on $\mathbb{S}$[2].

In the sequel, we use the following well-known fact.

**Fact 5.4.** *If two cycles $C$ and $C'$ are freely homotopic, then their symmetric difference is a dual cut. If $C$ and $C'$ are additionally disjoint and non-separating, then their union is a simple dual cut.*

Intuitively, the image of the continuous homotopy function from $C$ to $C'$ on the surface forms an annulus [46]. See Figure 5.1 for an illustration.

## 5.2 Overview

In this section, we give an overview of our constant-factor approximation algorithm for the maximum integral multiflow problem when $G + H$ is embedded on an orientable surface $\mathbb{S}_g$ of genus $g$, where $g$ is a constant (Theorem 5.1). Again, without loss of generality, we assume that $G + H$ is connected. Here is the main algorithm. Steps 1,2,3,4 will be described in detail in Sections 5.3, 5.4, 5.5, 5.6, respectively.

1. Solve the linear program (5.1) to obtain a (fractional) multiflow $f^*$. If its value is $|f^*| < \beta \cdot g^3$ where $\beta > 0$ is a universal constant to be defined later, then we can compute a maximum integral multiflow in polynomial time [132]. Otherwise, we can assume for the following steps that $|f^*| \geq \beta \cdot g^3$.

2. Construct another multiflow $\overline{f}$ such that any two cycles in the support of $\overline{f}$ cross at most once (Lemma 5.7). See Definition 5.6 for the definition of "crossing."

3. If at least half of the total value of $\overline{f}$ is contributed by separating cycles, these cycles now form a laminar family. Construct a half-integral multiflow $f^{\text{half}}$ (Theorem 5.9),

[2]Topological cycles are considered up to orientation-preserving reparameterization. Therefore, a cycle in $G + H$ may be represented by a topological cycle from two classes, one for each orientation: the class of $\gamma$ and the class of $\gamma'$ where $\gamma'(e^{i\theta}) = \gamma(e^{-i\theta})$.

and from there, using the map-coloring theorem (Theorem 1.1), compute an integral multiflow $f'$ (Lemma 5.10), which is the output.

4. Otherwise, compute a large set of non-separating and pairwise non-crossing cycles in the support of $\overline{f}$ (Lemma 5.13). Partition this set into free homotopy classes. For each class, define a new capacity function, and greedily construct an integral multiflow (Lemmas 5.22 and 5.21). Output the union of these multiflows.

We will prove that we lose only a constant factor the approximation at every step of the algorithm: see Section 5.7 for the analysis of the above algorithm.

## 5.3 Finding a large fractional multiflow (Step 1)

A feasible solution $f$ to the maximum multiflow LP (5.1) will be simply called a **multiflow**. Recall that $\mathcal{C}$ denotes the set of all $D$-cycles, i.e., all cycles in $G+H$ that contain precisely one demand edge. We denote by $|f| = \sum_{C\in\mathcal{C}} f_C$ the **value** of $f$, and by $\mathcal{C}(f) := \{C \in \mathcal{C} \mid f_C > 0\}$ the **support** of $f$. Although formulation (5.1) has an exponential number of variables, it is well known that it can be reformulated by polynomially many flow variables and constraints (see, e.g., [52, 1]) and thereby solved in polynomial time:

**Proposition 5.5.** *There is an algorithm that finds an optimal solution $f^*$ to the maximum multiflow LP (5.1) such that $|\mathcal{C}(f^*)| \le |D||E|$. Its running time is polynomial in the size of the input graph.*

*Proof.* By introducing flow variables $x_e^d := \sum_{C\in\mathcal{C}:d,e\in C} f_C$ for all $d \in D$ and $e \in D \dot\cup E$ we can maximize the total value $\sum_{d\in D} x_d^d$ subject to non-negativity and flow conservation constraints (for each $d \in D$ and for each vertex). This is a linear program of polynomial size. By flow decomposition, one can then construct a feasible solution to Equation 5.1 of the same value and with support at most $|D||E|$. □

Suppose that $|f^*| < \beta \cdot g^3$. Then the maximum integral multiflow $f^{\mathbf{OPT}}$ is constant, so we can guess in time $|D|^{O(g^3)}$ the value of the flow $f^{\mathbf{OPT}}(d)$ through each demand edge $d \in D$. For each guess, we create an instance of EdgeDisjointPaths problem replacing each demand edge $d \in D$ by $f^{\mathbf{OPT}}(d)$ parallel demand edges of demand one, and each supply edge $e \in E$ by $\min(u(e), \lceil \beta g^3 \rceil)$ parallel edges of unit capacity. It is easy to see that this graph has polynomial size. Since the number of demand edge is bounded by a constant, we can apply the polynomial time algorithm from [132] or [85] to decide whether it is a yes instance or not for the decision question.

In the following we can assume that $|f^*| \ge \beta \cdot g^3$. Later we will restrict a multiflow to subsets of $D$-cycles. For $\mathcal{C}' \subseteq \mathcal{C}$ we define a multiflow $f'$ by $f'_C := f_C$ for $C \in \mathcal{C}'$ and $f'_C := 0$ for $C \in \mathcal{C} \setminus \mathcal{C}'$, and write $f(\mathcal{C}') := f'$.

## 5.4 Making a fractional multiflow minimally crossing (Step 2)

In this section we show that for a given embedding, we can "uncross" a multiflow in such a way that any two $D$-cycles in the support cross at most once. While doing this we will lose only an arbitrarily small fraction of the multiflow value.

Uncrossing is a well-known technique in combinatorial optimization, but in most cases it is applied to families of subsets of a ground set $U$. Such a family is said to be cross-free if, for any two of its sets, $A$ and $B$, at least one of the four sets $A \setminus B$, $B \setminus A$, $A \cap B$, and $U \setminus (A \cup B)$ is empty. Here we want to uncross $D$-cycles in the topological sense, and this can be reduced to the above (with some extra care) only if all these cycles are separating (which, for example, is always the case if $G + H$ is planar; cf.[61]).

**Definition 5.6.** We say that two $D$-cycles $C_1$ and $C_2$ **cross** if there exists a path $P$ (possibly a single vertex), which is a subpath of both $C_1$ and $C_2$, and such that in the embedding, after contracting the edges of $P$, the vertex $v$ thus obtained is incident to two edges of $C_1$ and to two edges of $C_2$, all distinct, and in the embedding the restriction of the cyclic order of $\delta(v)$ to those four edges alternates between an edge of $C_1$ and an edge of $C_2$.

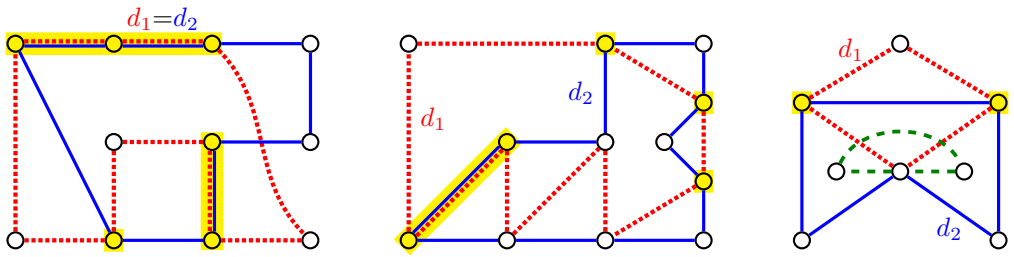



Figure 5.2: Each of the two figures on the left show two $D$-cycles, $C_1$ (red, dotted) and $C_2$ (blue, solid). The edges belonging to $D$ are marked as $d_1$ and $d_2$. Edges are arranged at every vertex in the order of their embedding. Crossings are marked by yellow shade. The two $D$-cycles on the left cross three times. The two $D$-cycles in the middle cross four times. The figure on the right shows two $D$-cycles $C_1$ and $C_2$ that cross twice, and a third $D$-cycle $C_3$ (green, dashed) that crosses neither $C_1$ nor $C_2$. Uncrossing $C_1$ and $C_2$ here generates a crossing of $C_3$ with a new $D$-cycle.

Two cycles may cross multiple times. We denote by $cr(C, C')$ the number of times that $C$ and $C'$ cross. See Figure 5.2 for three examples. In contrast to the planar case, it is possible that two cycles cross exactly once and cannot be uncrossed. The third example in Figure 5.2 shows another difficulty: when uncrossing two $D$-cycles it might be unavoidable to generate new crossings with other cycles.

**Lemma 5.7.** *Let $\epsilon > 0$ be fixed. Given a multiflow $f$ whose support has size at most $|E||D|$, there is a polynomial-time algorithm to construct another multiflow $\overline{f}$, of value at least $|\overline{f}| \geq (1-\epsilon)|f|$, and such that any two cycles in the support of $\overline{f}$ cross at most once.*

*Proof.* First we discretize the multiflow, losing an $\epsilon$ fraction in value; then we iteratively modify it, without changing its value, to reduce the number of crossings or the total amount of flow on all edges; finally we analyze the process and argue that the number of iterations is polynomially bounded.

**Discretization.** The statement is trivial if $|f| = 0$. Otherwise, before uncrossing, we round down the flow on every $D$-cycle to integer multiples of $\frac{\epsilon|f|}{|E||D|}$. That is, we define $f'_C := \frac{\epsilon|f|}{|E||D|} \left\lfloor \frac{|E||D|f_C}{\epsilon|f|} \right\rfloor$ for all $C \in \mathcal{C}$. Note that $f'$ is a multiflow. We claim that

$|f'| \geq (1-\epsilon)|f|$. Indeed,

$$|f'| = \sum_{C \in \mathcal{C}} f'_C \geq \sum_{C \in \mathcal{C}(f)} \left( f_C - \frac{\epsilon |f|}{|E||D|} \right) = |f| - |\mathcal{C}(f)| \frac{\epsilon |f|}{|E||D|} \geq |f| - \epsilon |f|.$$

The discretized multiflow $f'$ can be represented by a multi-set $\mathcal{S}$ of unweighted $D$-cycles: if $f'_C = k \frac{\epsilon |f|}{|E||D|}$, then $k$ identical copies of cycle $C$ are added to $\mathcal{S}$. The number of cycles in $\mathcal{S}$ (counting multiplicities) is at most $\frac{|E||D|}{\epsilon}$ because $|\mathcal{S}| = \sum_{C \in \mathcal{C}} f'_C \frac{|E||D|}{\epsilon |f|} \leq \sum_{C \in \mathcal{C}} f_C \frac{|E||D|}{\epsilon |f|} = \frac{|E||D|}{\epsilon}$.

**Uncrossing.** To construct $\overline{f}$, we perform a sequence of transformations of the multiflow. We will modify $\mathcal{S}$ while maintaining the following invariants:

(a) The number of elements of $\mathcal{S}$ (counting multiplicities) remains constant.

(b) For every $e \in D \dot{\cup} E$, the number of elements of $\mathcal{S}$ (counting multiplicities) that contain $e$ never increases.

Thanks to (b), at any stage, $\overline{f}$ is a multiflow, where $\overline{f}$ is defined by $\overline{f}_C = k \frac{\epsilon |f|}{|E||D|}$ for $C \in \mathcal{C}$, where $k$ is the multiplicity of $C$ in $\mathcal{S}$. Initially $\overline{f} = f'$. Thanks to (a), the value of the multiflow is preserved. In the following we work only with $\mathcal{S}$.

While there exist two cycles $C_1$ and $C_2$ in $\mathcal{S}$ that cross at least twice, do the following *uncrossing* operation (on one copy of $C_1$ and one copy of $C_2$). Let $d_1$ be the edge in $C_1 \cap D$, and let $d_2$ be the edge in $C_2 \cap D$. Let $P$ and $Q$ be two paths where $C_1$ and $C_2$ cross, such that $Q$ contains only edges of $E$. Orient $C_1$ so that in that orientation, when traversing the entirety of $P$ and then walking towards $Q$, edge $d_1$ is traversed before reaching $Q$. Let $\vec{C_1}$ denote the resulting directed cycle. Let $a$ be the first vertex on $P$ in the orientation of $\vec{C_1}$, and let $b$ be an arbitrary vertex on $Q$. Vertices $a$ and $b$ partition $\vec{C_1}$ into a path $C_1^+$ from $a$ to $b$ that contains $d_1$ and a path $C_1^-$ from $b$ to $a$ that does not contain $d_1$.

**Case 1:** $P$ contains an edge of $D$. Then this edge is $d_1 = d_2$. We orient $C_2$ so that the orientation on $P$ agrees with the orientation of $\vec{C_1}$ on $P$. Let $\vec{C_2}$ denote the resulting directed cycle. Then the vertices $a$ and $b$ also partition $\vec{C_2}$ into a path $C_2^+$ from $a$ to $b$ that contains $d_2$ and a path $C_2^-$ from $b$ to $a$ that does not contain $d_2$.
**Case 2:** $P$ contains edges of $E$ only. Then we orient $C_2$ so that in that orientation, when traversing the entirety of $P$ and then walking towards $Q$, edge $d_2$ is traversed before reaching $Q$. Let $\vec{C_2}$ denote the directed cycle. With that orientation, vertices $a$ and $b$ also partition $\vec{C_2}$ into a path $C_2^+$ from $a$ to $b$ that contains $d_2$ and a path $C_2^-$ from $b$ to $a$ that does not contain $d_2$.

To obtain $C'_1$, we concatenate $C_1^+$ and $C_2^-$, remove any cycle that does not contain $d_1$, and remove the orientation. To obtain $C'_2$, we concatenate $C_2^+$ and $C_1^-$, remove any cycle that does not contain $d_2$, and remove the orientation. Note that $C'_1$ and $C'_2$ are $D$-cycles because each of $C_1^+$ and $C_2^+$ contains exactly one demand edge, and $C_1^-$ and $C_2^-$ contain no demand edge.

See Figure 5.3 for two examples, one for each case.

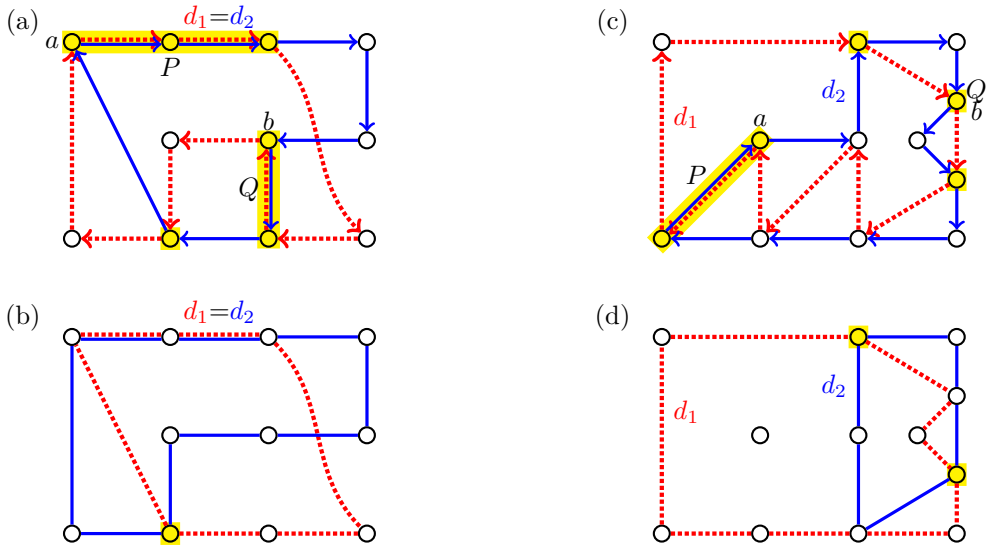



Figure 5.3: Uncrossing the pairs of $D$-cycles from Figure 5.2. (a) and (b) show an example for Case 1, (c) and (d) an example for Case 2. The initial situation ($C_1$ red, dotted, and $C_2$ blue, solid) and a possible choice of $P, Q, a, b$ and the resulting orientation is shown in (a) and (c). As the result of the uncrossing operation, shown in (b) and (d), we have the new $D$-cycles $C_1'$ (red, dotted) and $C_2'$ (blue, solid) with fewer crossings among each other.

**Analysis.** From the construction it follows that $C_1'$ and $C_2'$ are $D$-cycles and $C_1' \dot\cup C_2' \subseteq C_1 \dot\cup C_2$. Hence removing one copy of $C_1$ and $C_2$ from $\mathcal{S}$ and adding one copy of $C_1'$ and $C_2'$ to $\mathcal{S}$ maintains the invariants (a) and (b).

To show that the after a polynomial number of uncrossing operations any pair of cycles in $\mathcal{S}$ crosses at most once, we consider the total number of edges $\Phi_1 = \sum_{C\in\mathcal{S}} |C|$ (counting multiplicities) and the total number of crossings $\Phi_2 = \sum_{C,C'\in\mathcal{S}} cr(C,C')$ (where we again count multiplicities). Note that $|\mathcal{S}|$ remains constant by invariant (a), and $\Phi_1$ never increases by invariant (b). Moreover $0 \le \Phi_1 \le |V||\mathcal{S}|$ and $0 \le |\Phi_2| \le |V||\mathcal{S}|^2$. We claim:

Each uncrossing operation either decreases $\Phi_1$ or leaves $\Phi_1$ unchanged and decreases $\Phi_2$. (5.4)

This will conclude the proof because $\Phi_1$ decreases at most $|V||\mathcal{S}|$ times, and while $\Phi_1$ remains constant, $\Phi_2$ decreases at most $|V||\mathcal{S}|^2$ times, so the total number of uncrossing operations is at most $|V|^2|\mathcal{S}|^3 \le \frac{|V|^2|E|^3|D|^3}{\epsilon^3}$.

To prove Equation 5.4, consider an uncrossing operation that replaces $C_1$ and $C_2$ by $C_1'$ and $C_2'$, and suppose that $\Phi_1$ remains the same, so $C_1'$ consists of $C_1^+$ plus $C_2^-$, and $C_2'$ consists of $C_2^+$ plus $C_1^-$. We first observe that $cr(C_1', C_2') < cr(C_1, C_2)$. Indeed, the crossings at $P$ and at $Q$ go away, and no new crossing arises.

Finally we need to show that for any cycle $C \in \mathcal{C}$,

$$cr(C, C_1') + cr(C, C_2') \le cr(C, C_1) + cr(C, C_2). \tag{5.5}$$

To show Equation 5.5, consider a crossing of $C$ and $C' \in \{C_1', C_2'\}$ at a path $R$. Let $e_1' = \{v_0, v_1\}, \dots, e_k' = \{v_{k-1}, v_k\}$ be the edges of $R$ ($k \ge 0$), and let $e_0, e_{k+1}, e_0', e_{k+1}'$ be edges such that $e_0, e_1', \dots, e_k', e_{k+1}$ are subsequent on $C$ and $e_0', e_1', \dots, e_k', e_{k+1}'$ are

subsequent on $C'$. After contracting $R$, the incident edges $e_0, e'_0, e_{k+1}, e'_{k+1}$ are embedded in this cyclic order. (Note that $e_0 = e_{k+1}$ or $e'_0 = e'_{k+1}$ is possible if $k \geq 1$, then contracting $R$ yields a loop.) See Figure 5.4 (a).

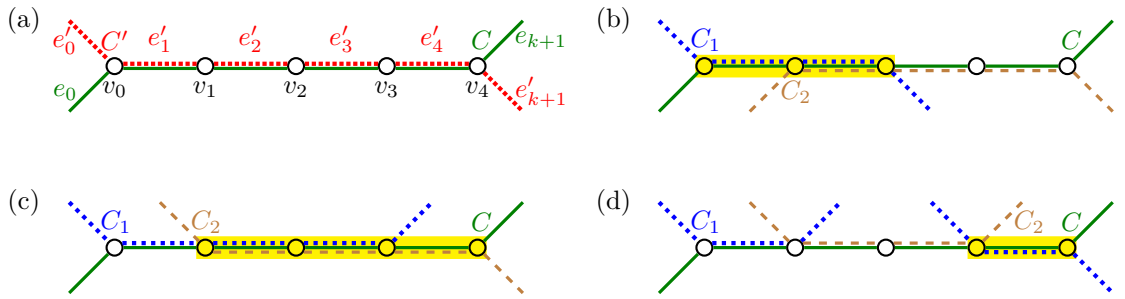



Figure 5.4: For each crossing of $C$ with a new cycle $C' \in \{C'_1, C'_2\}$ at a path $R$ there is a crossing of $C$ with one of the old cycles $C_1$ and $C_2$ at a subpath of $R$. This crossing is marked with yellow shade in the three examples.

Now $e'_0$ belongs to $C_1$ or $C_2$, say $C_1$. If $R$ contains neither $a$ nor $b$, then $e'_0, \ldots, e'_{k+1}$ all belong to $C_1$, and $C_1$ crosses $C$ at $R$. If $R$ contains either $a$ or $b$, say at $v_i$, then $e'_0, \ldots, e'_i$ belong to $C_1$ and $e'_{i+1}, \ldots, e'_{k+1}$ belong to $C_2$. Moreover $C_1$ and $C_2$ cross at a path containing $v_i$, so either $C_1$ crosses $C$ at a subpath of $R$ (Figure 5.4(b)) or $C_2$ crosses $C$ at a subpath of $R$ (Figure 5.4(c)). Finally, if $R$ contains $a$ and $b$, say at $v_i$ and $v_j$ for $0 \leq i < j \leq k$, then $e'_0, \ldots, e'_i$ and $e'_{j+1}, \ldots, e'_{k+1}$ belong to $C_1$ and $e'_{i+1}, \ldots, e'_j$ belong to $C_2$ (Figure 5.4(d)). Again, $C_1$ or $C_2$ crosses $C$ at a subpath of $R$. □

## 5.5 Separating cycles: routing an integral multiflow (Step 3)

Let $\overline{f}$ result from Lemma 5.7, and let $\mathcal{C}_{\text{sep}}$ denote the set of separating cycles in the support of $\overline{f}$. We now consider the case when the separating cycles contribute at least half to the total flow value, i.e., $|\overline{f}(\mathcal{C}_{\text{sep}})| \geq \frac{1}{2}|\overline{f}|$. For simplicity we write $f = \overline{f}(\mathcal{C}_{\text{sep}})$.

This branch of our algorithm consists of two steps:

1. Given $f$, construct a half-integral multiflow $f^{\text{half}}$ of value at least $|f|/2$;
2. Given $f^{\text{half}}$, construct an integral multiflow of value at least $|f^{\text{half}}|/\Theta(\sqrt{g})$.

### 5.5.1 Obtaining a half-integral multiflow

To obtain a half-integral multiflow, we follow the technique used by [61] for the case where $G + H$ is planar. By the Jordan curve theorem, any cycle in a planar graph is separating. As for the plane, the following property is easy to check for higher genus surfaces.

**Proposition 5.8.** *If $C$ and $C'$ are two cycles cycles embedded on a surface, and $C'$ is a separating cycle, then $C$ and $C'$ must cross an even number of times.*

*Proof.* $C'$ is separating the surface into two sides. While walking along $C$ from a vertex $v$, we go from one side to the other each time we cross $C'$. When we return at $v$, we are on the same side where we started so the number of crossing is even. □

Since any two cycles in the support of $\overline{f}$ cross at most once, $\mathcal{C}_{\text{sep}}$ must be a non-crossing family by Proposition 5.8. In particular, we can show that $\mathcal{C}_{\text{sep}}$ have a laminar structure.

We say that a family of subsets of the dual vertex set $V^*$ is laminar if any two members either are disjoint or one contains the other. Let us take any face of $G+H$ that we call $\infty$. For any cycle $C \in \mathcal{C}_{\text{sep}}$ we define $\text{in}(C)$ and $\text{out}(C)$ to be the two connected components of $(G+H)^* \setminus C^*$, such that $\infty \in \text{out}(C)$. We claim that the family $\mathcal{L} := \{\text{in}(C) : C \in \mathcal{C}_{\text{sep}}\}$ is laminar.

Indeed, take any two cycles $C$ and $C'$ in $\mathcal{C}_{\text{sep}}$. Since they do not cross, either (i) $(C' \setminus C)^* \subseteq \text{in}(C)$ or, (ii) $(C' \setminus C)^* \subseteq \text{out}(C)$. In case (i) we must have $\text{in}(C') \subseteq \text{in}(C)$. In case (ii), we have either (ii.a) $\text{in}(C) \subseteq \text{in}(C')$ or (ii.b) $\text{in}(C) \cap \text{in}(C') = \emptyset$, hence laminarity.

Using the terminology in [61], we say that a multiflow $f$ is *laminar* if $\{C^* : C \in \mathcal{C},\ f_C > 0\} = \{\delta(U) : U \in \mathcal{L}\}$ where $\mathcal{L}$ is a laminar family (of subsets of $V^*$). Thus, $f = \overline{f}(\mathcal{C}_{\text{sep}})$ is laminar and we can apply the following result to get $f^{\text{half}}$.

**Theorem 5.9.** *([61]) If $f$ is a laminar multiflow, then there exists a laminar half-integral multiflow $f'$ such that $\mathcal{C}(f') \subseteq \mathcal{C}(f)$ of value $|f'| \geq \frac{1}{2}|f|$. Such a multiflow can be computed in polynomial time.*

### 5.5.2 Obtaining an integral multiflow

In this section we show the following result, which is an extension of a result from [84, 61], who proved it for planar graphs.

**Lemma 5.10.** *Let $(G, H, u)$ be an instance of the maximum multiflow problem such that $G+H$ has genus $g$, and let $f^{half}$ be a laminar half-integral multiflow whose support $\mathcal{C}(f^{\text{half}})$ contains only separating cycles. Then there exists an integral multiflow $f'$ of value $|f'| \geq 2|f^{\text{half}}|/\chi_g$ (such that $\mathcal{C}(f') \subseteq \mathcal{C}(f^{\text{half}})$). Such a multiflow can be found in polynomial time.*

Our proof follows the same outline as the proof of Theorem 1 of Fiorini et al.[51]. Let $\mathcal{C}^{\text{half}} := \mathcal{C}(f^{\text{half}})$ be the set of $D$-cycles $C$ such that $f^{\text{half}}_C > 0$. We first reduce the problem to the case where all cycles in $\mathcal{C}^{\text{half}}$ have flow value $\frac{1}{2}$ and every edge has capacity 1. To do that, we reduce the flow $f^{\text{half}}_C$ by $\lfloor f^{\text{half}}_C \rfloor$ for each cycle $C \in \mathcal{C}^{\text{half}}$, and reduce edge capacities accordingly. Since $f^{\text{half}}$ is small, we can then further reduce demands and capacities to $u'(e) = \min\{u(e), |\mathcal{C}(f^{\text{half}})|\}$ for each $e \in E \cup D$, so that $\sum_{e \in D \dot\cup E} u(e)$ is polynomially bounded. We can then replace each edge $e$ by $u(e)$ parallel edges of unit capacity. Given a cycle $C$ such that $f^{\text{half}}_C = \frac{1}{2}$, we replace each edge $e \in C$ by one of its parallel edges. This can be done while ensuring that the resulting flow is still feasible and laminar. To facilitate the proof, we still denote this graph by $G+H$ and keep all other notations.

Recall that cycles in $\mathcal{C}^{\text{half}} \subseteq \mathcal{C}_{\text{sep}}$ are separating and do not cross each other, so that the family $\{\text{in}(C), C \in \mathcal{C}^{\text{half}}\}$ is laminar. We partially order $\mathcal{C}^{\text{half}}$ with the following relation: $C \prec C'$ if $\text{in}(C) \subset \text{in}(C')$. We have the following property:

**Lemma 5.11.** *If $C_1, C_2, C' \in \mathcal{C}^{\text{half}}$ are such that $C_1 \prec C'$ and $C_2 \not\prec C'$, then $C_1$ and $C_2$ are edge-disjoint.*

*Proof.* Assume, for a contradiction, that $C_1$ and $C_2$ share an edge $e$. Let $e^* = \{u^*_{\text{in}}, u^*_{\text{out}}\}$ denote its dual edge, such that $u^*_{\text{in}} \in \text{in}(C_1)$ and $u^*_{\text{out}} \in \text{out}(C_1)$.

Since $C_2 \not\prec C'$, by laminarity either $C' \prec C_2$ or $\text{in}(C') \cap \text{in}(C_2) = \emptyset$.

In the first case we have $C_1 \prec C' \prec C_2$ and then:

$$u^*_{\text{in}} \in \text{in}(C_1) \subseteq \text{in}(C') \subseteq \text{in}(C_2) \text{ and } u^*_{\text{out}} \in \text{out}(C_2) \subseteq \text{out}(C'),$$

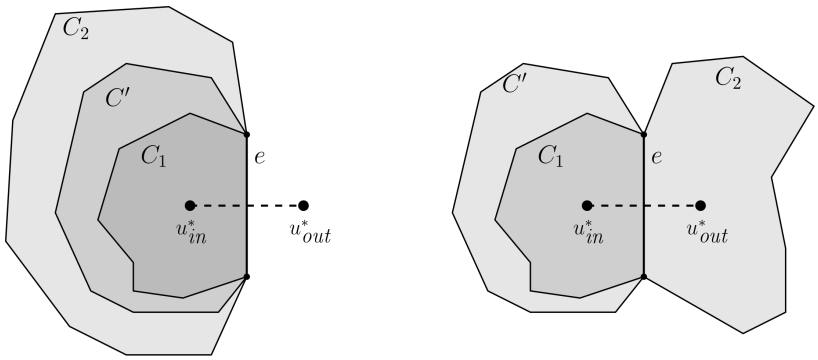
so $e \in C'$.

In the second case we have $C_1 \prec C'$ and $\text{in}(C') \cap \text{in}(C_2) = \emptyset$ and then:

$$u^*_{\text{in}} \in \text{in}(C_1) \subseteq \text{in}(C') \subseteq \text{out}(C_2) \text{ and } u^*_{\text{out}} \in \text{in}(C_2) \subseteq \text{out}(C'),$$

so $e \in C'$. See Figure 5.5.

Thus in both cases $e$ belongs to $C'$ as well as to $C_1$ and $C_2$. Since these three $D$-cycles are in the support of a half-integral multiflow, this implies that the flow along this edge is at least $\frac{3}{2}$, contradicting feasibility. □




**case 1:** $C_1 \prec C' \prec C_2$ **case 2:** $C_1 \prec C'$ and $\text{in}(C') \cap \text{in}(C_2) = \emptyset$

Figure 5.5: Proof of Lemma 5.11.

Our goal is to get a large subset $\mathcal{C}' \subseteq \mathcal{C}^{\text{half}}$ such that any two cycles in $\mathcal{C}'$ are edge-disjoint. This is equivalent to finding a large independent set in a properly defined graph $\text{Int}(\mathcal{C}^{\text{half}})$ with vertex set $\mathcal{C}^{\text{half}}$ and such that two cycles are adjacent if they share at least one edge.

**Lemma 5.12.** *Given a graph embedded in $\mathbb{S}_g$, let $\mathcal{C}^{half}$ be a family of pairwise non-crossing separating cycles that satisfies the condition of Lemma 5.11. Let Int($\mathcal{C}^{half}$) be the graph with vertex set $\mathcal{C}^{half}$ and such that two cycles are adjacent if they share at least one edge. Then Int($\mathcal{C}^{half}$) is a genus-g graph.*

Using Theorem 1.1, this lemma ensures that one can compute in polynomial time a subset $\mathcal{C}' \subseteq \mathcal{C}^{\text{half}}$ of at least $|\mathcal{C}^{\text{half}}|/\chi_g$ pairwise edge-disjoint $D$-cycles. From this set, we define an integral multiflow by setting $f'_C = 1$ for $C \in \mathcal{C}'$ and $f'_C = 0$ for $C \in \mathcal{C} \setminus \mathcal{C}'$. It is easy to check that $f'$ is a multiflow that satisfies the properties of Lemma 5.10.

*Proof.* (Lemma 5.12) We prove the statement by induction on $g + |\mathcal{C}^{\text{half}}|$. When $g + |\mathcal{C}^{\text{half}}| \leq 2$, it is trivial. Otherwise let $G$ be a connected genus-$g$ graph, embedded on $\mathbb{S}_g$, and $\mathcal{C}^{\text{half}}$ the family of cycles as described above.

Suppose first that $\{\text{in}(C) \mid C \in \mathcal{C}^{\text{half}}\}$ are pairwise disjoint. Then, contract in $G^*$ each set $\text{in}(C)$ into a single node. Two cycles $C$ and $C'$ share an edge if and only if in this contracted graph, the nodes corresponding to $\text{in}(C)$ and $\text{in}(C')$ are adjacent. This means

that $\mathrm{Int}(\mathcal{C}^{\mathrm{half}})$ is a minor of $G^*$, and in particular has genus less than or equal to the genus of $G^*$.

The case where there is one cycle $\bar{C}$ such that $C \prec \bar{C}$ for all $C \in \mathcal{C}^{\mathrm{half}} \setminus \bar{C}$ and $\{\mathrm{in}(C) \mid C \in \mathcal{C}^{\mathrm{half}} \setminus \bar{C}\}$ are pairwise disjoint works similarly; here we contract $\mathrm{out}(\bar{C})$.

Otherwise there exists a triple $C_1, C_2, C \in \mathcal{C}^{\mathrm{half}}$ such that $C_1 \prec C$ and $C_2 \not\prec C$. The separating cycle $C$ divides $\mathbb{S}_g$ into two sides. Each side can be closed — by identifying the boundary of a disk with the boundary form by $C$ — so that they are homeomorphic to $\mathbb{S}_{g_{\mathrm{in}}}$ and $\mathbb{S}_{g_{\mathrm{out}}}$, respectively. The connected sum of these two surfaces is homeomorphic to $\mathbb{S}_g$, and in particular we have $g_{\mathrm{in}} + g_{\mathrm{out}} = g$. This equality can easily be checked with Euler's formula.

Let $G_{\mathrm{in}}$ (*resp.* $G_{\mathrm{out}}$) be the subgraph of $G$ induced by the vertices embedded on the side corresponding to $\mathbb{S}_{g_{\mathrm{in}}}$ (*resp.* $\mathbb{S}_{g_{\mathrm{out}}}$), such that both contain $C$. The embedding of $G$ in $\mathbb{S}_g$ induces an embedding of $G_{\mathrm{in}}$ in $\mathbb{S}_{g_{\mathrm{in}}}$ and an embedding of $G_{\mathrm{out}}$ in $\mathbb{S}_{g_{\mathrm{out}}}$. Thus, $\mathrm{genus}(G_{\mathrm{in}}) + \mathrm{genus}(G_{\mathrm{out}}) \leq g$.

Now we define $\mathcal{C}^{\mathrm{half}}_{\preceq C} := \{C' \in \mathcal{C}^{\mathrm{half}} | C' \prec C\} \cup \{C\}$ and $\mathcal{C}^{\mathrm{half}}_{\not\prec C} := \{C' \in \mathcal{C}^{\mathrm{half}} | C' \not\prec C\} \cup \{C\}$. The choice of $C$ implies that these two families are proper subsets of $\mathcal{C}^{\mathrm{half}}$. Since the cycles in $\mathcal{C}^{\mathrm{half}}$ do not cross, we have $\{C \in \mathcal{C}^{\mathrm{half}} : C \subseteq G_{\mathrm{in}}\} = \mathcal{C}^{\mathrm{half}}_{\preceq C}$ and $\{C \in \mathcal{C}^{\mathrm{half}} : C \subseteq G_{\mathrm{out}}\} = \mathcal{C}^{\mathrm{half}}_{\not\prec C}$.

By the induction hypothesis, $\mathrm{Int}(\mathcal{C}^{\mathrm{half}}_{\preceq C})$ and $\mathrm{Int}(\mathcal{C}^{\mathrm{half}}_{\not\prec C})$ can be embedded on $\mathbb{S}_{g_{\mathrm{in}}}$ and $\mathbb{S}_{g_{\mathrm{out}}}$, respectively. By Lemma 5.11, the graph $\mathrm{Int}(\mathcal{C}^{\mathrm{half}})$ arises from $\mathrm{Int}(\mathcal{C}^{\mathrm{half}}_{\preceq C})$ and $\mathrm{Int}(\mathcal{C}^{\mathrm{half}}_{\not\prec C})$ by identifying the two vertices that correspond to $C$.

Finally we prove that $\mathrm{Int}(\mathcal{C}^{\mathrm{half}})$ can be embedded on a surface genus $g_{\mathrm{in}} + g_{\mathrm{out}} \leq g$. To see that, remove small disks $D_{\mathrm{in}}$ and $D_{\mathrm{out}}$ in $\mathbb{S}_{g_{\mathrm{in}}}$ and $\mathbb{S}_{g_{\mathrm{out}}}$, respectively, around the point that corresponds to vertex $C$ and that intersects only edges incident to $C$, and glue them together by identifying boundaries of $D_{\mathrm{in}}$ and $D_{\mathrm{out}}$. The surface obtained is homeomorphic to $\mathbb{S}_{g_{\mathrm{in}}+g_{\mathrm{out}}}$ It is easy to see that $C$, and the edges incident to $C$, can be re-embedded in this surface without intersecting any other edges. This terminates the proof of Lemma 5.12.

□

## 5.6 Non-separating cycles: routing an integral multiflow (Step 4)

If the separating cycles contribute less than half to the total value of the multiflow $\overline{f}$ obtained by Lemma 5.7, we consider the non-separating cycles in the support of $\overline{f}$. Let us call this set $\mathcal{C}_{\text{non-sep}} \subseteq \mathcal{C}(\overline{f})$. We first compute a large subset of pairwise non-crossing curves.

**Lemma 5.13.** *There exists a subset $\mathcal{S} \subseteq \mathcal{C}_{\text{non-sep}}$ of pairwise non-crossing cycles such that the value of the multiflow $\overline{f}(\mathcal{S})$ is at least $|\overline{f}|/O(g^2)$. Such a set can be computed in polynomial time.*

Let $f := \overline{f}(\mathcal{S})$ be the multiflow $\overline{f}$ restricted to cycles in the set $\mathcal{S}$ given by Lemma 5.13. In particular, for a suitable choice of $\beta > 0$, we have $|f| \geq 12g - 12$. Finally, we compute from $f$ an integral multiflow $f'$ with support in $\mathcal{S}$ while losing only a constant fraction of the value.

**Theorem 5.14.** *Let $(G, H, u)$ be an instance of the maximum multiflow problem such that $G + H$ has genus $g$, and let $f$ be a multiflow such that*

*(1) $f$ has value $|f| \geq 12g - 12$,*

*(2) the support of $f$ contains only non-separating cycles,*

*(3) and any two cycles in the support of $f$ do not cross.*

*Then, there exists an integral multiflow $f'$ of value $|f'| \geq |f|/4$. Such a multiflow can be computed in polynomial time.*

Theorem 5.14 holds more generally when the support only contains *non-trivial* cycles. To compute such an integral multiflow, we first partition $\mathcal{S} := \{\mathcal{H}_1, \ldots, \mathcal{H}_s\}$ into *free homotopy classes*, and define a new capacity function for each class in such a way that the union of multiflows for these new capacities is feasible for the original capacities. The algorithmic core of Theorem 5.14 is to construct an integral multiflow for each homotopy class, according to the new capacity function. This is achieved by ordering homotopic cycles in a specific order and then to apply a simple greedy algorithm.

**Lemma 5.15.** *Let $(G, H, u)$ be an instance of the maximum multiflow problem such that $G + H$ is embedded in an orientable surface, and let $f$ be a multiflow such that the support of $f$ contains only non-separating and pairwise freely homotopic and non-crossing cycles. Then, there exists an integral multiflow $f'$ of value $|f'| \geq \frac{|f|}{2}$. Such a multiflow can be computed in polynomial time.*

These three results are proved in the following sections.

### 5.6.1 Finding a set of non-crossing cycles

In this section we prove Lemma 5.13. Our goal is to compute a family of pairwise non-crossing cycles in $\mathcal{C}_{\text{non-sep}}$ that has a large flow value. To do so we reduce our problem to determining the chromatic number of a specific intersection graph.

Let $\Gamma$ be a set of non-separating curves on $\mathbb{S}_g$ such that any pair of cycles cross at most once. Then we define $G(\Gamma)$ to be the graph with vertex set $\Gamma$ where two cycles are adjacent if they cross. Let $\chi(G(\Gamma))$ denote the coloring number of $G(\Gamma)$. Then, for $\Gamma = \mathcal{C}_{\text{non-sep}}$, there is a color class, i.e., a set $\mathcal{S}$ of pairwise non-crossing cycles such that $|\overline{f}(\mathcal{S})| \geq |\overline{f}|/\chi(G(\mathcal{C}_{\text{non-sep}}))$. We show that $\chi(G(\Gamma)) = O(g^2)$ and that such a coloring can be computed in polynomial time via the classic greedy algorithm.

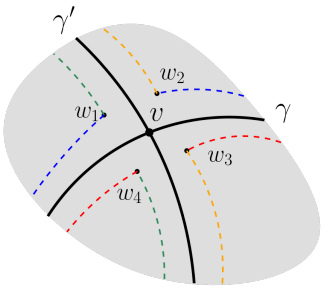
We first prove that without loss of generality, we can assume that any two cycles in $\Gamma$ are not freely homotopic.

**Lemma 5.16.** *If two cycles cross exactly once, then they are not freely homotopic.*

This fact needs the assumption that the surface is orientable. On a projective plane for instance, two freely homotopic cycles may cross exactly once.

*Proof.* Let us take two topological cycles $\gamma$ and $\gamma'$ such that $\gamma$ and $\gamma'$ that only cross at a single point $v$ of the surface. We show that $\gamma \cup \gamma'$ do not disconnect the orientable surface. By Fact 5.4 this implies that $\gamma$ and $\gamma'$ are not freely homotopic.

To see that $\gamma \cup \gamma'$ do not disconnect the surface, pick four points $w_1, w_2, w_3, w_4$ in a small neighborhood of $v$, each one of them being on a different of the four sections of this neighborhood delimited by $\gamma \cup \gamma'$. If $(w_i)_{1 \leq i \leq 4}$ are in clockwise order around $v$, then $w_i$ and $w_{i+1}$ are still connected for $i = 1, \ldots, 4$ (where $w_5 := w_1$), because we can walk all along $\gamma$ (or $\gamma'$). Notice that here we use the property that the surface is orientable (otherwise, $w_i$ might be connected to $w_{i+2}$ instead of $w_{i+1}$). By transitivity, we conclude that $\gamma \cup \gamma'$ do not disconnect the surface.




□

Moreover, by Fact 5.4, if $C$ and $C'$ are freely homotopic they form a cut, and thus for any cycle $C''$ the total number of crossing $cr(C'', C) + cr(C'', C')$ must be an even number. Since cycles in $\mathcal{C}_{\text{non-sep}}$ cross at most once, when $C$ and $C'$ in $\mathcal{C}_{\text{non-sep}}$ are freely homotopic, then for any cycle $C'' \in \mathcal{C}_{\text{non-sep}}$, $(C'', C)$ is an edge of $G(\mathcal{C}_{\text{non-sep}})$ if and only if $(C'', C')$ is also an edge. Thus, the coloring number of $G(\mathcal{C}_{\text{non-sep}})$ is the same as the coloring number of the graph obtained from $G(\mathcal{C}_{\text{non-sep}})$ after contracting each free homotopy class into a single node.

We now use the following result initially contained in the proof of ([128], Theorem 4), and later generalized by [64].

**Theorem 5.17** ([128, 64])**.** *There is a universal constant $\beta' > 0$ such that the following is true. Let $\Gamma$ be a family of simple curves on $\mathbb{S}_g$ such that any two of them are not freely homotopic and cross at most once. Then, we have the following upper bound on the maximum degree of the intersection graph of $\Gamma$.*

$$\Delta(G(\Gamma)) \leq \beta' \cdot g^2$$

Using this structural result, we can give a polynomial time algorithm that compute $\mathcal{S} \subseteq \mathcal{C}_{\text{non-sep}}$ such that $|\overline{f}(\mathcal{S})| \geq |\overline{f}|/O(g^2)$. We can now define the constant $\beta$ used in step 1 of the overall algorithm as $\beta := 48\beta'$.

To determine free homotopy classes, take pairs of cycles in the support of $\mathcal{C}_{\text{non-sep}}$ and check whether they are freely homotopic, for example as in [104, 48]. Keep only one curve per class and compute the intersection graph as defined above and apply the classic greedy algorithm for coloring. Return the set of cycles $\mathcal{S}$ that corresponds to the color class that maximizes $|\overline{f}(\mathcal{S})|$.

### 5.6.2 Proof of Theorem 5.14

Let $\mathcal{S}$ be the set of non-separating and pairwise non-crossing cycles in the support of $\overline{f}$ obtained in the previous section, and call $f := \overline{f}(\mathcal{S})$ the restriction of $\overline{f}$ to these cycles. Its value is $|f| \geq |\overline{f}|/O(g^2) = \Omega(g)$. We partition $\mathcal{S} = \mathcal{H}_1 \cup \cdots \cup \mathcal{H}_s$ into free homotopy classes. With the following well-known bound, we deduce that the number of classes is $s \leq 3g - 3$.

**Lemma 5.18.** *Let $\mathcal{S}$ be a family of disjoint simple curves on $\mathbb{S}_g$, such that any two of them are not freely homotopic. If $g \geq 2$, then $|\mathcal{S}| \leq 3g - 3$.*

When $\mathbb{S}_g$ is the torus, it is easy to see that $\mathcal{S}$ has exactly one free homotopy class.

*Proof.* Without loss of generality, we can assume that $\mathcal{S}$ is a maximal set with this property. We show that its size is exactly $3g-3$. By maximality, cutting along cycles in $\mathcal{S}$ separates $\mathbb{S}_g$ into $p$ *pairs of pants*, i.e., $p$ spheres with exactly three boundaries each, where each boundary corresponds to a cycle in $\mathcal{S}$. In particular, each cycle is incident to exactly two pair of pants. Thus, $|\mathcal{S}| = \frac{3}{2}p$. The Euler Characteristic of a pair of pants is $-1$ and since $\mathbb{S}_g$ is obtained by identifying boundaries of $p$ pairs of pants, we get $2-2g = (-1)\cdot p$. We proved the upper bound as expected. □

Now, we define a new capacity function $u_i : E \cup D \to \mathbb{Z}_{\geq 0}$ for each free homotopy class $\mathcal{H}_i$. To do this, we first need to identify the *extreme cycles* of each class.

**Lemma 5.19.** *Each free homotopy class $\mathcal{H}_i$ in $\mathcal{S}$ has two* extreme cycles *$C_i^+$ and $C_i^-$ such that any cycle $C \in \mathcal{S} \setminus \mathcal{H}_i$ that shares an edge with a cycle in $\mathcal{H}_i$ also shares an edge with $C_i^+$ or $C_i^-$. The set of extreme cycles can be computed in polynomial time.*

Eventually, when a class is reduced to a unique cycle $C$, we have $C_i^+ = C_i^- = C$. We know that any two disjoint free homotopic cycles delimit an annulus on the surface. Intuitively, for each class, the extreme cycles correspond to the pair of cycles that delimits the maximal annulus among all pairs in this class.

*Proof.* We can assume that $g \geq 2$, otherwise $\mathcal{S}$ has at most one free homotopy class and the statement if trivially true. Let $\mathcal{H}_i$ a free homotopy class in $\mathcal{S}$ of size at least two. Cutting along cycles in $\mathcal{H}_i$ separates the surface into several components that are homeorphic to annuli or disks except one component $K$ that has genus at least one. Its boundary is contained in the union of two cycles, that we call $C_i^+$ and $C_i^-$. All other cycles in $\mathcal{S}$ are contained in $K$. Thus, if a cycle shares an edge $e$ with a cycle in $\mathcal{H}_i$, this edge must be on $K$'s boundary, and in particular $e \in C_i^+ \cup C_i^-$. □

For each free homotopy class $\mathcal{H}_i \subseteq \mathcal{S}$, we can now define a new capacity function $u_i$. Let $E_i := \bigcup_{C\in\mathcal{H}_i} C$ be the set of edges used by at least one cycle in $\mathcal{H}_i$. We set $u_i(e) = \lfloor f(\mathcal{H}_i)(e) \rfloor$ if $e \in C_i^+ \cup C_i^-$; $u_i(e) = u(e)$ if $e \in E_i \setminus (C_i^+ \cup C_i^-)$ and $u_i(e) = 0$ otherwise.

By reducing the flow $f_C$ by at most one for $C \in \{C_i^+, C_i^-\}$, we can see that there exists a multiflow of value at least $|f(\mathcal{H}_i)| - 2$ for capacities $u_i$. By Lemma 5.15, we can compute in polynomial time, for each class $\mathcal{H}_i$ an integral multiflow $f_i'$, feasible for capacities $u_i$ with value at least $\frac{|f(\mathcal{H}_i)|-2}{2}$. We finally return the union $f' := \sum f_i'$ of these integral multiflows. It is clear that $f'$ is also integral. We now check that it is feasible for the original capacity function $u$.

Take any edge $e$ that belongs to an extreme cycle. By Lemma 5.19, we know that this edge is only shared by extreme cycles, and thus

$$f'(e) = \sum_i f_i'(e) \leq \sum_i \lfloor f(\mathcal{H}_i)(e) \rfloor \leq \sum_i f(\mathcal{H}_i)(e) = f(e) \leq u(e)$$

Now if $e$ is not an edge of an extreme cycle, then $e$ may be contained by $D$-cycles from at most one free homotopy class, say $\mathcal{H}_j$. Then, $f'(e) = f_j'(e) \leq u(e)$. Thus, $f'$ is an integral

multiflow. We conclude by lower bounding its value. Using Lemma 5.18, and hypothesis (1) of Theorem 5.14 we obtain

$$|f'| = \sum_{i=1}^{s} |f'_i| \geq \sum_{i=1}^{s} \frac{|f(\mathcal{H}_i)| - 2}{2} = \frac{|f|}{2} - s \geq \frac{|f|}{2} - (3g-3) \geq |f|/4$$

which concludes the proof of Theorem 5.14.

### 5.6.3 Greedy selection of freely homotopic cycles

Let $\mathcal{H}$ be a free homotopy class of non-separating cycles and pairwise non-crossing and let $f$ be a multiflow whose support is $\mathcal{H}$. We will run the following simple greedy algorithm (Algorithm 13) on $\mathcal{H}$ to get an integral multiflow.

**Input:** a sequence $C_1, \ldots, C_h$ of $D$-cycles.
**Output:** an integral multiflow $f'$.
$f' \leftarrow$ the all-zero multiflow;
**for** $i = 1$ *to* $h$ **do**
  Set $f'_{C_i}$ to be the greatest integer such that $f'$ remains feasible.

**Algorithm 13:** Greedy selection algorithm.

The value of the integral multiflow returned by this algorithm depends on the order of the $D$-cycles in the input. If it is ordered according to the following definition, then we show that we lose only a constant fraction of the flow value.

**Definition 5.20.** A family of cycles $\{C_1, C_2, \ldots, C_k\}$ is *cyclically ordered*, or has a *cyclic order* if, whenever two cycles $C_a$ and $C_b$ share an edge, where $a < b$, then this edge is:

1. shared by all cycles $C_a, C_{a+1}, \ldots, C_{b-1}, C_b$,
2. or shared by all cycles $C_b, C_{b+1}, \ldots, C_k, C_1, \cdots, C_{a-1}, C_a$.

The following lemma establishes the approximation ratio of Algorithm 13 on cyclically ordered inputs.

**Lemma 5.21.** *Let $f$ be a multiflow and $\mathcal{H} = \{C_1, C_2, \ldots, C_h\}$ a cyclically ordered family of $\mathcal{C}(f)$. Then Algorithm 13 returns in polynomial time an integral multiflow of value at least $|f(\{C_1, \ldots, C_h\})|/2$.*

To conclude, we prove that $\mathcal{H}$ can be cyclically ordered in polynomial time.

**Lemma 5.22.** *A family of non-separating, pairwise non-crossing and freely homotopic cycles of a graph embedded on an orientable surface can be cyclically ordered. Such a cyclic order can be found in polynomial time.*

This result holds more generally for a family of *non-trivial*[3], pairwise non-crossing and freely homotopic cycles. For simplicity, we only consider the special case of non-separating cycles.

[3] all cycles that are not freely homotopic to a point on the surface.

#### 5.6.3.1 Finding a cyclic order

In this section we prove Lemma 5.22. Let $\mathcal{H}$ be a set of non-separating, pairwise freely homotopic and non-crossing cycles. One key ingredient in the proof is that cycles in $\mathcal{H}$ are pairwise non-crossing. Recall that this fact uses the assumption that the surface is orientable. In a non-orientable surface, two freely homotopic cycles may cross exactly once.

We first order the cycles in $\mathcal{H}$ and then prove that this is a cyclic order. We assume that $|\mathcal{H}| \geq 3$, otherwise any order on $\mathcal{H}$ is a cyclic order.

In topology it is usually more convenient to work with disjoint cycles. If two (graph) cycles do not cross, but may share common edges, it is possible to continuously deform by free homotopy one of them, into an arbitrarily small open neighborhood so that the two resulting (topological) cycles are now disjoint.

We describe here a new graph $Q$ to translate this idea in the context of graph cycles. Initially, $Q = G + H$.

**Step 1:** If an edge is shared by $s$ cycles, replace it $s$ parallel edges. Each of these edges corresponds to a different cycle so that the resulting set of cycles is still pairwise non-crossing. Now the cycles are pairwise edge-disjoint but may still share some vertices.

**Step 2:** Let $v$ be a vertex shared by two cycles $C$ and $C'$. Edges incident to $v$ are embedded around $v$ in the cyclic order $e_1, a_1, \ldots, a_i, e_2, b_1, \ldots, b_j$ where $C \cap \delta(v) = \{e_1, e_2\}$. Since $C$ and $C'$ do not cross, we have $C' \cap \delta(v) \subseteq \{a_1, \ldots, a_i\}$ or $C' \cap \delta(v) \subseteq \{b_1, \ldots, b_j\}$. Then replace $v$ by two adjacent vertices $v', v''$ and distribute the incident edges so that $\delta(v') = (e_1, a_1, \ldots, a_i, e_2, \{v', v''\})$ and $\delta(v'') = (\{v', v''\}, b_1, \ldots, b_j)$. Repeat step 2 until all cycles are vertex-disjoint.

If it easy to see that this graph is connected and can be embedded on the same surface $\mathbb{S}_g$. Figure 5.6 illustrates the construction of $Q$. For simplicity, let us also call $\mathcal{H}$ the family of cycles in $Q$.

Let $\mathcal{K}$ denote the set of connected components of $Q^* \setminus (\bigcup_{C \in \mathcal{H}} C)$. We say that a cycle $C \in \mathcal{H}$ is *incident* to a connected component $K \in \mathcal{K}$ if there is an edge in $C^*$ with one endpoint in $K$. Consider the bipartite graph $B$ that has a vertex for each cycle in $\mathcal{H}$ and a vertex for each element of $\mathcal{K}$, and whose edges represent the incidence relation. Next we show that the graph $B$ is a cycle, and we order the $D$-cycles in $\mathcal{H}$ according to the cyclic order induced by $B$.

**Claim 5.23.** *$B$ is a cycle.*

The connectivity of $B$ follows by construction from the connectivity of $G + H$. Then it is enough to prove that $B$ is 2-regular.

There is a bijection between $\mathcal{K}$ and the components of $\mathbb{S}_g \setminus (\bigcup_{C \in \mathcal{H}} C)$. Since the cycles in $\mathcal{H}$ are disjoint, each cycle $C$ has one component on its left, and one on its right, when we walk along the cycle. Notice that in general these two sides can eventually correspond to the same component. Since cycles in $\mathcal{H}$ are non-separating, each component is incident to at least two cycles. Assume that a cycle $C$ is incident to only one component of $\mathcal{K}$. This cycle is also incident to only one component of $\mathbb{S}_g \setminus (C \cup C')$ where $C'$ is any other cycle. By Fact 5.4, $\mathbb{S}_g \setminus (C \cup C')$ has two connected components, which means each cycle in $B$ must have degree two.

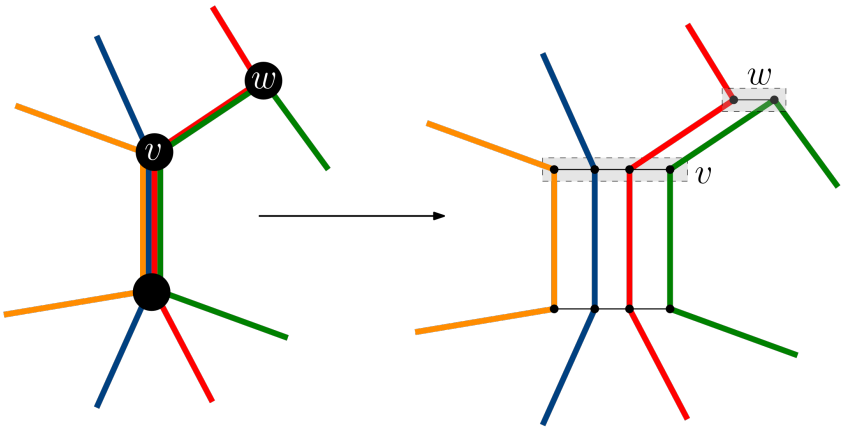



Figure 5.6: Construction of $Q$.

Now, assume that an element of $\mathcal{K}$ is incident to three cycles $C, C', C''$ or more. Then one component of $Q^* \setminus (C \cup C' \cup C'')$ is also incident to $C, C'$ and $C''$, and $Q^* \setminus (C \cup C' \cup C')$ has two or three components in total. If it has three components, then one of the other two components would be incident to exactly one cycle, which contradicts what precedes. If $Q^* \setminus (C \cup C' \cup C'')$ has exactly two connected components, then $Q^* \setminus (C \cup C')$ must be connected, which would contradict Fact 5.4. Thus, each component is incident to exactly two cycles. We have proved that $B$ is a cycle.

It remains to show that the order induced by $B$ satisfies the property of Definition 5.20. If an edge $e = \{u, v\}$ of $G + H$ is shared by some cycles $C'_1, \dots, C'_\ell$, then the vertex $v$ can be mapped to a path $P = (v_1, \dots, v_\ell)$ in $Q$, so that $C'_i \cap P = \{v_i\}, 1 \leq i \leq \ell$. See Figure 5.6. It follows that for all $1 \leq i \leq k-1$, $C'_i$ and $C'_{i+1}$ are both incident to the same connected component of $Q^* \setminus (\bigcup_{C \in \mathcal{H}} C)$ that contains the edge $\{v_i, v_{i+1}\}^*$. In particular, $C'_i$ and $C'_{i+1}$ are consecutive in the order induced by $B$.

#### 5.6.3.2 Analysis of the Greedy selection Algorithm for cyclically ordered inputs

Let $f$ be a multiflow and $\mathcal{H} = \{C_1, C_2, \dots, C_h\}$ a cyclically ordered family of $\mathcal{C}(f)$. It is clear that Algorithm 13 runs in polynomial time and returns a integral multiflow. Let $f'$ be this multiflow. We show that its value is at least $|f(\mathcal{H})|/2$.

Let us define $\mathcal{H}_{a,b} = \{C_a, C_{a+1}, \dots, C_{b-1}\}$ and $\mathcal{H}_{b,a} = \{C_b, C_{b+1}, \dots, C_h, C_1, \dots, C_{a-1}\}$ for all $1 \leq a < b \leq h$. We call $M$ the greatest index $1 \leq i \leq h$ such that $f'_{C_i} \geq 1$. We first show by induction that for all $1 \leq i < M$ : $|f'(\mathcal{H}_{1,i+1})| \geq |f(\mathcal{H}_{1,i+1})|$.

For $i = 1$, we have $|f'(\mathcal{H}_{1,i+1})| = |f'(\mathcal{H}_{1,2})| = f'_{C_1} = \min\{u(e) | e \in C_1\} \geq f_{C_1} \geq |f(\mathcal{H}_{1,2})|$.

Assume now that at some iteration $i \leq M$ of the algorithm we set $f'_{C_i} = x$. By the choice of $x$, we know that there is an edge $e \in C_i$ such that $u(e) = f'(\mathcal{H}_{1,i+1})(e)$. By Lemma 5.22, this edge is such that $\{C \in \mathcal{H} | e \in C\} = \mathcal{H}_{a,b}$ for some distinct indexes $1 \leq a, b \leq h$, and obviously $C_i \in \mathcal{H}_{a,b}$. In particular, $|f'(\mathcal{H}_{a,b} \cap \mathcal{H}_{1,i+1})| = f'(\mathcal{H}_{1,i+1})(e) = u(e)$. Moreover, by feasibility of $f$, we have $|f(\mathcal{H}_{a,b})| = f(\mathcal{H}_{a,b})(e) \leq f(e) \leq u(e)$. Altogether this gives

$$|f'(\mathcal{H}_{a,b} \cap \mathcal{H}_{1,i+1})| \geq |f(\mathcal{H}_{a,b})| \tag{5.6}$$

If $a < b$, then we have $a \leq i < b$ and $\mathcal{H}_{a,b} \cap \mathcal{H}_{1,i+1} = \mathcal{H}_{a,i+1}$. Therefore, $|f'(\mathcal{H}_{a,i+1})| \geq |f(\mathcal{H}_{a,b})| \geq |f(\mathcal{H}_{a,i+1})|$. By the induction hypothesis, we have $|f'(\mathcal{H}_{1,a})| \geq |f(\mathcal{H}_{1,a})|$ and then

$$|f'(\mathcal{H}_{1,i+1})| = |f'(\mathcal{H}_{1,a})| + |f'(\mathcal{H}_{a,i+1})| \geq |f(\mathcal{H}_{1,a})| + |f(\mathcal{H}_{a,i+1})| = |f(\mathcal{H}_{1,i+1})|$$

Otherwise $a > b$. This means that $i \in \{a, a+1, \ldots, h, 1, \ldots, b-1\}$ and in particular, $e \in C_j$ for all indexes $j$ in $\{i, \ldots, h\}$, which implies that no further $D$-cycles can be added in the support of $f'$, *i.e.,* $i = M$. In particular we have established the induction. Moreover the support of $f'$ is $\mathcal{H}_{1,M+1}$ so that we can re-writeEquation 5.6 as $|f'| \geq |f'(\mathcal{H}_{a,b})| \geq |f(\mathcal{H}_{a,b})| \geq |f(\mathcal{H}_{M,1})|$. Finally,

$$|f(\mathcal{H})| = |f(\mathcal{H}_{1,M})| + |f(\mathcal{H}_{M,1})| \leq |f(\mathcal{H}_{1,M})| + |f'| = 2|f'|,$$

which finishes the proof of Lemma 5.21.

**Remark.** The analysis of Algorithm 13 for cyclically ordered inputs is tight. To see this, imagine that $|\mathcal{H}| = 2k-1$, and there are two edges $e_1, e_2$, both of capacity $k$, that are shared by the first $k$ cycles and the last $D$-cycles, respectively. Moreover assume that $C_1$ contains both edges. Then Algorithm 13 may only set $f'_{C_1} = k$ while $f$ could be such that $f_C = 1$ for all $C \in \mathcal{H}$ for a total value $2k-1$.

## 5.7 Analysis of the overall algorithm (Proof of Theorem 5.1)

By construction, the output of the algorithm is a feasible solution. We now analyze the value of the output. Since (5.1) is a relaxation of the maximum integral multiflow problem, $|f^*| \geq \mathrm{OPT}$. When $|f^*| < \beta g^3$, then we compute an maximum integral multiflow.

Otherwise, by Lemma 5.7, $|\overline{f}| \geq (1-\epsilon)|f^*|$. For $\epsilon = \frac{1}{2}$ we have $|\overline{f}| \geq \frac{1}{2}|f^*|$.

Consider the multiflow restricted to separating cycles, $\overline{f}_{\mathrm{sep}}$. If $|\overline{f}_{\mathrm{sep}}| \geq \frac{1}{2}|\overline{f}|$, then by Theorem 5.9, Lemma 5.10, and Theorem 1.1 we obtain an integral multiflow of value at least $|\overline{f}_{\mathrm{sep}}|/\Theta(\sqrt{g})$. Otherwise, by Theorem 5.14 we obtain an integral multiflow of value $|\overline{f}|/O(g^2)$.

Finally, we analyze the running time. As observed in Section 5.3, an optimum fractional multiflow $f^*$ can be found in polynomial time.

If $|f^*| < \beta \cdot g^3$, then we compute an optimal solution in time $h(g)|D|^{O(g^3)}$ where $h$ is a computable function inherited from [132, 85]. Indeed, for all integer $1 \leq k < \beta \cdot g^3$, and for all $k$-multiset $D' \subseteq D$, we use the algorithm in [85] to check in time $h'(k) \cdot n^2$ whether there exists a multiflow with demand $D'$.

Otherwise, (discretizing and) uncrossing is done in time polynomial in $|E||D|$ by Lemma 5.7. Finally, the operations of Theorem 5.9, Theorem 1.1, Lemma 5.10, Lemma 5.21 and Lemma 5.22 can all be done in polynomial time, hence polynomial running time overall.

## 5.8 Max-Multiflow Min-Multicut gap

In this section, we prove that the multiflow-multicut gap is bounded by a constant when $G + H$ has bounded genus.

### 5.8.1 Proof of Theorem 5.2

To obtain an solution an integral multiflow whose value is at most a factor $O(g^2 \log g)$ smaller than the value of any *fractional* multiflow, we apply the following changes to the algorithm:

1. If in step 1, we go directly to step 2 without computing the maximum integral multiflow.
2. Steps 2 and 3 are unchanged.
3. we consider the non-separating cycles in the support of $\overline{f}$. We first partition them into *free homotopy classes*. The next theorem gives an upper bound on the number of such classes.

**Theorem 5.24.** *([64]) Let $\mathbb{S}_g$ be an orientable surface of genus $g$. Then there are at most $O(g^2 \log g)$ topological cycles such that any two of them are in different free homotopy classes and cross each other at most once.*

**Corollary 5.25.** *The D-cycles in the support of $\overline{f}$ can be partitioned into $O(g^2 \log g)$ free homotopy classes in polynomial time.*

*Proof.* Take pairs of cycles in the support of $\overline{f}$ and check whether they are freely homotopic, for example as in [104, 48]. □

4. We pick a free homotopy class $\mathcal{H}$ with flow value $|\overline{f}(\mathcal{H})| \geq |f^*|/O(g^2 \log g)$. We cyclically order this class (Lemma 5.22) and run the Greedy selection algorithm, with the original capacities, to obtain an integral multiflow with value at least $|\overline{f}(\mathcal{H})|/2 = |f^*|/O(g^2 \log g)$.

Following the analysis in Section 5.7, the running time of this algorithm is $n^{O(1)}$. In particular, it is independent from the genus $g$ of the input graph. Thus, we proved Theorem 5.2, and in particular we established that the integrality gap of the maximum multiflow LP is $O(g^2 \log g)$ when $G + H$ has genus $g$.

### 5.8.2 Proof of Corollary 5.3

In this section, we observe how Corollary 5.3 follows from Theorem 5.1 and the following result.

**Theorem 5.26.** *[141] Let $(G, H, u)$ be a multiflow instance and $\gamma > 1$ such that the supply graph $G$ does not have a $K_{\gamma,\gamma}$ minor. Then the minimum capacity of a multicut is $O(\gamma^3)$ times the maximum value of a (fractional) multiflow.*

The following is well known.

**Claim 5.27.** *If a graph $G$ has genus at most $g$, where $g \geq 1$, then it has no $K_{\gamma,\gamma}$ minor for any $\gamma > 2(\sqrt{g} + 1)$.*

*Proof.* Suppose that such a minor $K_{\gamma,\gamma}$ exists in $G$. As the three operations for obtaining a minor (deleting edges/vertices and contracting edges) does not increase the genus, $K_{\gamma,\gamma}$ has genus at most $g$. Furthermore, $K$ has $2\gamma$ vertices, $\gamma^2$ edges, and at most $\frac{\gamma^2}{2}$ faces (since there is no odd cycle in a bipartite graph). By Euler's formula, $2 - 2g \leq 2\gamma - \gamma^2 + \frac{\gamma^2}{2}$, which implies $\gamma \leq 2(\sqrt{g} + 1)$. □

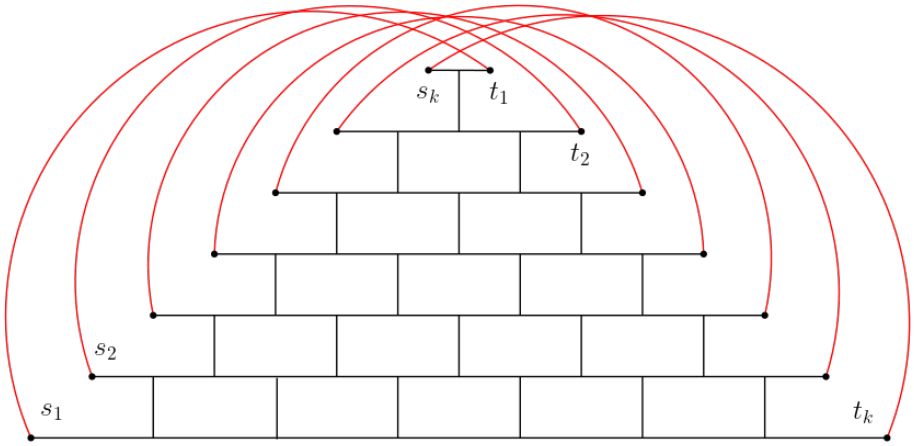



Figure 5.7: An example where the classic linear program for EDGEDISJOINTPATHS has an integrality gap equal to $k/2$. The optimal fractional multiflow consists of routing a flow of value $1/2$ through each edge, while it is easy to see any integral solution can satisfy at most one demand edge. In terms of number of vertices $n$ of the graph, the integrality gap is $\Theta(\sqrt{n})$. This graph has genus $\Theta(k)$, thus the integrality gap when $G+H$ has genus $g$ is $\Omega(g)$. This example was originally described in [63].

.

By Claim 5.27 and Theorem 5.26, the ratio between the minimum capacity of a multicut and the maximum value of a (fractional) multiflow is $O(g^{1.5})$. This, combined with Theorem 5.1, proves Corollary 5.3.

## 5.9 Conclusion

In this chapter, we have presented an $O(g^2)$-approximation algorithm for the Maximum Integral Multiflow problem when the input graph $G+H$, demand and supply edges together can be embedded in an orientable surface of genus $g$.

**Improving the approximation ratio.** Can we improve the approximation ratio ? First we can see in Figure 5.7 that if one uses only the optimal fractional multiflow as an upper bound on **OPT**, then the approximation ratio of such an algorithm when $G+H$ has genus $g$ is at least $\Omega(g)$.

The bottleneck in the approximation ratio of our algorithm lies in Theorem 5.17, where we lose a factor $O(g^2)$ in the approximation ratio. Notice that in the case where at least half of the multiflow value goes through separating cycles, then we only lose a factor $O(\sqrt{g})$ in the approximation. Thus, any improvement for the following topological question would immediately implies the same improvement in the approximation ratio of the Maximum Integral Multiflow when $G+H$ has genus $g$.

**Open Question.** *Let $\Gamma$ be a set of simple curves on an orientable surface of genus $g$, such that any pair of curves in $\Gamma$ are non freely homotopic and cross at most once. Let $G(\Gamma)$ be the graph with vertex set $\Gamma$ and where two curves are adjacent if they cross. What is the best upper bound on the chromatic number of $G(\Gamma)$ ?*

Such a family of curves is called a *1-system* in the literature. There has been a long line of work in trying to capture the cardinality of 1-systems, a question initially raised by

Farb and Leininger. Juvan, Malnic and Mohar proved that such a set is finite [88], then an exponential upper bound on the genus was given [113]. Later, Przytycki solved the question for arcs, proving in the meantime a cubic upper bound for curves. Recently, the bound was improved to $|\Gamma| \leq O(g^2 \log g)$. On the other hand, constructions of 1-system with cardinality $\Theta(g^2)$ are known [113].

One tool used to prove these bounds was to bound the maximum degree $\Delta(G(\Gamma)) = O(g^2)$. In particular, this implies a quadratic upper bound on the chromatic number of $G(\Gamma)$, which is the best know upper bound as far as we know. It was also proved that the maximum clique in $G(\Gamma)$ has size at most $2g + 1$ [113], implying the same lower bound on the coloring number.

**Running time.** The most costly step in the running time is Step 1 where we compute an optimal solution when the optimal fraction multiflow is at most $O(g^3)$. Using algorithms in [132] or [85], we obtain an overall running time of $h(g)n^{O(g^3)}$ where $h$ is a computable function inherited from [132, 85]. In the case where the supply graph is planar, very recently it was shown that one can decide in time $2^{O(k^2)}n^{O(1)}$ whether an instance with $k$ demand edges is positive or not [107].

If the genus of the surface is no longer bounded by a constant, then our algorithm might not run in polynomial time. For such settings, one could use the algorithm of Theorem 5.2 with a slightly worse approximation guarantee of $O(g^2 \log g)$ but with the running time $n^{O(1)}$ where the exponent is now independent from the genus. For more details see [83].

**Non-orientable surfaces.** We discuss here the Maximum Integral Multiflow problem when $G + H$ is embedded on *non-orientable* surfaces.

A graph has a *non-orientable genus* $\widetilde{g}$ if it can be embedded on a connected sum of $\widetilde{g}$ projective planes. It is well-known that a genus-$g$-graph has non-orientable genus at most $2g + 1$ (see *e.g.* Prop. 4.4.1 in [119]). However, there are no upper bounds on the (orientable) genus as a function of the non-orientable genus. For our problem, the graph on Figure 5.7 has non-orientable genus one because it can be embedded on a projective plane but has (orientable) genus $\Omega(k)$. In particular, while using the optimal fractional multiflow as an upper bound on **OPT** we cannot hope for a constant approximation algorithm for the Maximum Integral Multiflow problem when the non-orientable genus of $G + H$ is bounded. As far as we know, the best approximation ratio for the Maximum Integral Multiflow problem when $G + H$ has bounded non-orientable genus is $O(\sqrt{|V(G)|})$ as for the general case. We leave open the question of improving the approximation ratio in that special case.

# Bibliography


[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice-Hall, 1993.

[2] P. Alimonti and V. Kann, *Some APX-completeness results for cubic graphs*, Theor. Comput. Sci., 237 (2000), pp. 123–134.

[3] M. Andrews, J. Chuzhoy, S. Khanna, and L. Zhang, *Hardness of the undirected edge-disjoint paths problem with congestion*, in Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2015, pp. 226–244.

[4] S. Arora, *Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems*, J. ACM, 45 (1998), pp. 753–782.

[5] S. Arora, M. Grigni, D. R. Karger, P. N. Klein, and A. Woloszyn, *A polynomial-time approximation scheme for weighted planar graph TSP*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA, H. J. Karloff, ed., ACM/SIAM, 1998, pp. 33–41.

[6] Y. Aumann and Y. Rabani, *Improved bounds for all optical routing*, in Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 1995, pp. 567–576.

[7] P. Austrin, S. Khot, and M. Safra, *Inapproximability of vertex cover and independent set in bounded degree graphs*, Theory of Computing, 7 (2011), pp. 27–43.

[8] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. ACM, 41 (1994), pp. 153–180.

[9] N. Bansal, *Approximating independent sets in sparse graphs*, in Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, 2015, pp. 1–8.

[10] N. Bansal, A. Gupta, and G. Guruganesh, *On the lovász theta function for independent sets in sparse graphs*, in Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing, STOC '15, New York, NY, USA, 2015, ACM, pp. 193–200.

[11] M. Bateni, M. T. Hajiaghayi, and D. Marx, *Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth*, J. ACM, 58 (2011), pp. 21:1–21:37.

[12] C. Bentz, *Résolution exacte et approchée de problèmes de multiflot entier et de multicoupe : algorithmes et complexité*, Ph.D. thesis, Conservatoire National des Arts et Métiers, 2006.

[13] C. Berge, *Graphs and hypergraphs*, North-Holland, Ansterdam, 1973.

[14] P. Berman and T. Fujito, *On approximation properties of the independent set problem for low degree graphs*, Theory of Computing Systems, 32 (1999), pp. 115–132.

[15] P. Berman and M. Fürer, *Approximating maximum independent set in bounded degree graphs*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '94, Philadelphia, PA, USA, 1994, Society for Industrial and Applied Mathematics, pp. 365–371.

[16] H. L. Bodlaender, D. M. Thilikos, and K. Yamazaki, *It is hard to know when greedy is good for finding independent sets*, Information Processing Letters, 61 (1997), pp. 101–106.

[17] R. Boppana and M. M. Halldórsson, *Approximating maximum independent sets by excluding subgraphs*, BIT Numerical Mathematics, 32 (1992), pp. 180–196.

[18] A. Borodin, M. N. Nielsen, and C. Rackoff, *(incremental) priority algorithms*, in Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '02, USA, 2002, Society for Industrial and Applied Mathematics, p. 752–761.

[19] G. Borradaile, P. N. Klein, and C. Mathieu, *An* O*(*n *log* n*) approximation scheme for steiner tree in planar graphs*, ACM Trans. Algorithms, 5 (2009), pp. 31:1–31:31.

[20] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz, *A tight linear time (1/2)-approximation for unconstrained submodular maximization*, in 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS2012, New Brunswick, NJ, USA, October 20-23, 2012, IEEE Computer Society, 2012, pp. 649–658.

[21] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák, *Maximizing a monotone submodular function subject to a matroid constraint*, SIAM Journal on Computing, 40 (2011), pp. 1740–1766.

[22] Y. Caro, A. Sebő, and M. Tarsi, *Recognizing greedy structures*, Journal of Algorithms, 20 (1996), pp. 137 – 156.

[23] E. W. Chambers, S. P. Fekete, H.-F. Hoffmann, D. Marinakis, J. S. Mitchell, V. Srinivasan, U. Stege, and S. Whitesides, *Connecting a set of circles with minimum sum of radii*, Computational Geometry, 68 (1991), pp. 62–76. special issue in memory of Ferran Hurtado.

[24] S. O. Chan, *Approximation resistance from pairwise-independent subgroups*, J. ACM, 63 (2016), pp. 27:1–27:32.

[25] S. Chaplick, M. De, A. Ravsky, and J. Spoerhase, *Approximation schemes for geometric coverage problems*, in ESA, vol. 112 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, pp. 17:1–17:15.

[26] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar, *On the hardness of approximating multicut and sparsest-cut*, Computational Complexity, 15 (2006), pp. 94–114.

[27] C. Chekuri, S. Khanna, and F. B. Shepherd, *An* $O(\sqrt{n})$ *approximation and integrality gap for disjoint paths and unsplittable flow*, Theory of Computing, 2 (2006), pp. 137–146.

[28] C. Chekuri, F. B. Shepherd, and C. Weibel, *Flow-cut gaps for integer and fractional multiflows*, Journal of Combinatorial Theory, Series B, 103 (2013), pp. 248 – 273.

[29] J. Cheriyan, H. Karloff, R. Khandekar, and J. Könemann, *On the integrality ratio for tree augmentation*, Operations Research Letters, 36 (2008), pp. 399–401.

[30] M. Chlebík and J. Chlebíková, *Inapproximability results for bounded variants of optimization problems*, in Fundamentals of Computation Theory, 14th International Symposium, FCT 2003, Malmö, Sweden, August 12-15, 2003, Proceedings, A. Lingas and B. J. Nilsson, eds., vol. 2751 of Lecture Notes in Computer Science, Springer, 2003, pp. 27–38.

[31] M. Chlebík and J. Chlebíková, *On approximability of the independent set problem for low degree graphs*, in Structural Information and Communication Complexity, R. Královic and O. Sýkora, eds., Berlin, Heidelberg, 2004, Springer Berlin Heidelberg, pp. 47–56.

[32] J. Chuzhoy, *Routing in undirected graphs with constant congestion*, in Proceedings of the 44th Annual ACM Symposium on Theory of Computing Conference (STOC), 2012, pp. 855–874.

[33] J. Chuzhoy and D. H. K. Kim, *On approximating node-disjoint paths in grids*, in Proceedings of the Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX), N. Garg, K. Jansen, A. Rao, and J. D. P. Rolim, eds., vol. 40 of LIPIcs, 2015, pp. 187–211.

[34] J. Chuzhoy, D. H. K. Kim, and S. Li, *Improved approximation for node-disjoint paths in planar graphs*, in Proceedings of the 48th Annual ACM Symposium on Theory of Computing Conference (STOC), D. Wichs and Y. Mansour, eds., 2016, pp. 556–569.

[35] J. Chuzhoy, D. H. K. Kim, and R. Nimavat, *New hardness results for routing on disjoint paths*, in Proceedings of the 49th Annual ACM Symposium on Theory of Computing Conference (STOC), H. Hatami, P. McKenzie, and V. King, eds., 2017, pp. 86–99.

[36] ——, *Almost polynomial hardness of node-disjoint paths in grids*, in Proceedings of the 50th Annual ACM Symposium on Theory of Computing Conference (STOC), 2018, pp. 1220–1233.

[37] V. Chvátal, *A greedy heuristic for the set-covering problem*, Math. Oper. Res., 4 (1979), pp. 233–235.

[38] V. Cohen-Addad, E. Colin de Verdière, and A. de Mesmay, *A near-linear approximation scheme for multicuts of embedded graphs with a fixed number of terminals*, in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2018, pp. 1439–1458.

[39] E. Colin de Verdière, *Topological algorithms for graphs on surfaces*, in Handbook of Discrete and Computational Geometry, J. Goodman and J. O'Rourke, eds., CRC Press, 2017. Chapter 23.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*, MIT Press, 2009.

[41] D. Cornaz, *Max-multiflow/min-multicut for G+H series-parallel*, Discrete Mathematics, 311 (2011), pp. 1957–1967.

[42] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, *The complexity of multiterminal cuts*, SIAM Journal on Computing, 23 (1994), p. 864–894.

[43] A. Darmann and J. Döcker, *Monotone 3-SAT-4 is NP-complete*, CoRR, abs/1603.07881 (2016).

[44] M. Demange and V. Paschos, *Improved approximations for maximum independent set via approximation chains*, Applied Mathematics Letters, 10 (1997), pp. 105 – 110.

[45] J. Edmonds and E. L. Johnson, *Matching, Euler tours and the Chinese postman*, Mathematical Programming, 5 (1973), pp. 88–124.

[46] D. B. A. Epstein, *Curves on 2-manifolds and isotopies*, Acta Math., 115 (1966), pp. 83–107.

[47] P. Erdős, *On the graph theorem of Turán*, Mat. Lapok, 21 (1970), pp. 249–251.

[48] J. Erickson and K. Whittlesey, *Transforming curves on surfaces redux*, in Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2013, pp. 1646–1655.

[49] U. Feige, *A threshold of ln* n *for approximating set cover*, J. ACM, 45 (1998), pp. 634–652.

[50] ——, *Approximating maximum clique by removing subgraphs*, SIAM J. Discrete Math., 18 (2004), pp. 219–225.

[51] S. Fiorini, N. Hardy, B. Reed, and A. Vetta, *Approximate min–max relations for odd cycles in planar graphs*, Mathematical Programming, 110 (2007), pp. 71—-91.

[52] L. Ford and D. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[53] S. Fortune, J. Hopcroft, and J. Wyllie, *The directed subgraph homeomorphism problem*, Theoretical Computer Science, 10 (1980), pp. 111–121.

[54] A. Frank, *Packing paths, circuits and cuts – a survey*, in Paths, Flows, and VLSI-Layout, B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds., Springer, 1990, pp. 47–100.

[55] ——, *A survey on T-joins, T-cuts, and conservative weightings*, in Combinatorics, Paul Erdős is Eighty, D. Miklós, V. T. Sós, and T. Szőnyi, eds., vol. 2, Springer, 1996, pp. 213–252.

[56] A. M. Frieze and S. Suen, *On the independence number of random cubic graphs*, Random Struct. Algorithms, 5 (1994), pp. 649–664.

[57] S. Funke, A. Kesselman, F. Kuhn, Z. Lotker, and M. Segal, *Improved approximation algorithms for connected sensor cover*, Wireless Networks, 13 (2007), pp. 153–164.

[58] M. M. Garey. and D. S. Johnson, *The rectilinear steiner tree problem is NP-complete*, SIAM Journal on Applied Mathematics, 32 (1977), pp. 826–834.

[59] M. R. Garey, D. S. Johnson, and L. Stockmeyer, *Some simplified np-complete problems*, in Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, STOC '74, New York, NY, USA, 1974, ACM, pp. 47–63.

[60] M. R. Garey, D. S. Johnson, and L. Stockmeyer, *Some simplified NP-complete graph problems*, Theoretical Computer Science, 1 (1976), pp. 237–267.

[61] N. Garg, N. Kumar, and A. Sebő, *Integer plane multiflow maximisation: Flow-cut gap and one-quarter-approximation*, 2020. arXiv:2002.10927.

[62] N. Garg, V. Vazirani, and M. Yannakakis, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM Journal on Computing, 25 (1996), pp. 235–251.

[63] ——, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica, 18 (1997), pp. 3–20.

[64] J. E. Greene, *On curves intersecting at most once*, 2018. arXiv:1807.05658.

[65] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica, 1 (1981), pp. 169–197.

[66] M. Guan, *Graphic programming using odd and even points*, Chinese Journal of Mathematics, 1 (1962), pp. 273–277.

[67] H. Gupta, Z. Zhou, S. R. Das, and Q. Gu, *Connected sensor cover: Self-organization of sensor networks for efficient query execution*, IEEE/ACM Transactions on Networks, 14 (2006), pp. 55–67.

[68] A. Hajnal and E. Szemerédi, *Proof of a conjecture of P. Erdős*, in Combinatorial Theory and its Applications, P. Erdős, A. Rényi, and V. Sós, eds., North-Holland, Amsterdam, 1970, pp. 601–623.

[69] M. M. Halldórsson, *Private communication. Erratum: https://www.ru.is/∼mmh/papers/hy95-errata.html*, March, 2019.

[70] M. M. Halldórsson and J. Radhakrishnan, *Greed is good: approximating independent sets in sparse and bounded-degree graphs*, in Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC 1994, Montréal, Québec, Canada, 23-25 May 1994, 1994, pp. 439–448.

[71] ——, *Improved approximations of independent sets in bounded-degree graphs*, in Algorithm Theory - SWAT '94, 4th Scandinavian Workshop on Algorithm Theory, Aarhus, Denmark, July 6-8, 1994, Proceedings, 1994, pp. 195–206.

[72] ——, *Improved approximations of independent sets in bounded-degree graphs via subgraph removal.*, Nord. J. Comput., 1 (1994), pp. 475–492.

[73] M. M. Halldórsson and J. Radhakrishnan, *Greed is good: Approximating independent sets in sparse and bounded-degree graphs*, Algorithmica, 18 (1997), pp. 145–163.

[74] M. M. Halldórsson and K. Yoshihara, *Greedy approximations of independent sets in low degree graphs*, in Algorithms and Computations, J. Staples, P. Eades, N. Katoh, and A. Moffat, eds., Berlin, Heidelberg, 1995, Springer Berlin Heidelberg. Full version in: http://www.ru.is/∼mmh/raunvis/Papers/yoshi.pdf, pp. 152–161.

[75] E. Halperin, *Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs*, SIAM J. Comput., 31 (2002), pp. 1608–1623.

[76] S. Har-Peled, *Being fat and friendly is not enough*, CoRR, abs/0908.2369 (2009).

[77] P. J. Heawood, *Map colour theorem*, Quart. J. Math., (1890), pp. 332–338.

[78] D. S. Hochbaum, *Efficient bounds for the stable set, vertex cover and set packing problems*, Discrete Applied Mathematics, 6 (1983), pp. 243 – 254.

[79] D. S. Hochbaum and A. Pathria, *Node-optimal connected k-subgraphs*, 1994.

[80] J. Håstad, *Clique is hard to approximate within* $n^{1-\epsilon}$, Acta Math., 182 (1999), pp. 105–142.

[81] W.-L. Hsu and G. L. Nemhauser, *Easy and hard bottleneck location problems*, Discrete Applied Mathematics, 1 (1979), pp. 209 – 215.

[82] C. Huang, M. Mari, C. Mathieu, J. S. B. Mitchell, and N. H. Mustafa, *Maximizing covered area in the euclidean plane with connectivity constraint*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA, D. Achlioptas and L. A. Végh, eds., vol. 145 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, pp. 32:1–32:21.

[83] C. Huang, M. Mari, C. Mathieu, and J. Vygen, *Approximating maximum integral multiflows on bounded genus graphs*, CoRR, abs/2005.00575 (2020).

[84] C.-C. Huang, M. Mari, C. Mathieu, K. Schewior, and J. Vygen, *An approximation algorithm for fully planar edge-disjoint paths*, 2020. arXiv:2001.01715.

[85] K. ichi Kawarabayashi, Y. Kobayashi, and B. Reed, *The disjoint paths problem in quadratic time*, Journal of Combinatorial Theory, Series B, 102 (2012), pp. 424 – 435.

[86] K. Jain, *A factor 2 approximation algorithm for the generalized Steiner network problem*, Combinatorica, 21 (2001), pp. 39–60.

[87] D. S. Johnson, *Worst case behavior of graph coloring algorithms*, in Proceedings of the 5th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantium X, 1974, pp. 513—-527.

[88] M. Juvan, A. Malnic, and B. Mohar, *Systems of curves on surfaces*, J. Comb. Theory, Ser. B, 68 (1996), pp. 7–22.

[89] K. Kar and S. Banerjee, *Node placement for connected coverage in sensor networks*, 2003.

[90] R. M. Karp, *On the computational complexity of combinatorial problems*, Networks, 5 (1975), pp. 45–68.

[91] K. Kawarabayashi and Y. Kobayashi, *An* $O(\log n)$*-approximation algorithm for the edge-disjoint paths problem in Eulerian planar graphs*, ACM Transactions on Algorithms, 9 (2013), pp. 16:1–16:13.

[92] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani, *On syntactic versus computational views of approximability*, SIAM Journal on Computing, 28 (1998), pp. 164–191.

[93] S. Khot and O. Regev, *Vertex cover might be hard to approximate to within 2-$\epsilon$*, Journal of Computer and System Sciences, 74 (2008), pp. 335 – 349. Computational Complexity 2003.

[94] S. Khuller, M. Purohit, and K. K. Sarpatwar, *Analyzing the optimal neighborhood: Algorithms for budgeted and partial connected dominating set problems*, in Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '14, Philadelphia, PA, USA, 2014, Society for Industrial and Applied Mathematics, pp. 1702–1713.

[95] P. N. Klein, C. Mathieu, and H. Zhou, *Correlation clustering and two-edge-connected augmentation for planar graphs*, in Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS), 2014, pp. 554–567.

[96] J. Kleinberg, *Approximation algorithms for disjoint paths problems*, Ph.D. thesis, MIT, 1996.

[97] J. M. Kleinberg, *An approximation algorithm for the disjoint paths problem in even-degree planar graphs*, in 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), 23-25 October 2005, Pittsburgh, PA, USA, Proceedings, IEEE Computer Society, 2005, pp. 627–636.

[98] E. Korach and M. Penn, *Tight integral duality gap in the Chinese postman problem*, Mathematical Programming, 55 (1992), pp. 183–191.

[99] E. Koutsoupias and C. H. Papadimitriou, *On the greedy algorithm for satisfiability*, Information Processing Letters, 43 (1992), pp. 53 – 55.

[100] D. Král' and H.-J. Voss, *Edge-disjoint odd cycles in planar graphs*, Journal of Combinatorial Theory, Series B, 90 (2004), pp. 107–120.

[101] P. Krysta, M. Mari, and N. Zhi, *Ultimate greedy approximation of independent sets in subcubic graphs*, in Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020, S. Chawla, ed., SIAM, 2020, pp. 1436–1455.

[102] A. Kulik, H. Shachnai, and T. Tamir, *Maximizing submodular set functions subject to multiple linear constraints*, in SODA, SIAM, 2009, pp. 545–554.

[103] T.-W. Kuo, K. C.-J. Lin, and M.-J. Tsai, *Maximizing submodular set function with connectivity constraint: Theory and application to networks*, IEEE/ACM Transactions on Networks, 23 (2015), pp. 533–546.

[104] F. Lazarus and J. Rivaud, *On the homotopy test on surfaces*, in proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS), IEEE Computer Society, 2012, pp. 440–449.

[105] J. Lee, M. Sviridenko, and J. Vondrák, *Submodular maximization over multiple matroids via generalized exchange properties*, Math. Oper. Res., 35 (2010), pp. 795–806.

[106] R. J. Lipton and R. E. Tarjan, *Applications of a planar separator theorem*, SIAM J. Comput., 9 (1980), pp. 615–627.

[107] D. Lokshtanov, P. Misra, M. Pilipczuk, S. Saurabh, and M. Zehavi, *An exponential time parameterized algorithm for planar disjoint paths*, in Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds., ACM, 2020, pp. 1307–1316.

[108] L. Lovász, *2-matchings and 2-covers of hypergraphs*, Acta Mathematica Academiae Scientiarum Hungaricae, 26 (1975), pp. 433–444.

[109] L. Lovász, *Three short proofs in graph theory*, Journal of Combinatorial Theory, Series B, 19 (1975), pp. 269 – 271.

[110] L. Lovász, *On two minmax theorems in graph*, Journal of Combinatorial Theory B, 21 (1976), pp. 96–103.

[111] L. Lovász, *On the ratio of optimal integral and fractional covers*, Discrete Mathematics, 13 (1975), pp. 383 – 390.

[112] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, J. ACM, 41 (1994), p. 960–981.

[113] J. Malestein, I. Rivin, and L. Theran, *Topological designs*, Geometriae Dedicata, 168 (2010).

[114] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz, *Simple heuristics for unit disk graphs*, Networks, 25 (1995), pp. 59–68.

[115] T. Matsui, *Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs*, in Discrete and Computational Geometry, J. Akiyama, M. Kano, and M. Urabe, eds., Berlin, Heidelberg, 2000, Springer Berlin Heidelberg, pp. 194–200.

[116] C. McDiarmid, *Colouring random graphs*, Annals of Operations Research, 1 (1984), pp. 183–200.

[117] M. Middendorf and F. Pfeiffer, *On the complexity of the disjoint paths problem*, Combinatorica, 13 (1993), pp. 97–107.

[118] J. S. B. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems*, SIAM Journal on Computing, 28 (1999), pp. 1298–1309.

[119] B. Mohar and C. Thomassen, *Graphs on Surfaces*, Johns Hopkins series in the mathematical sciences, Johns Hopkins University Press, 2001.

[120] N. H. Mustafa and S. Ray, *Improved results on geometric hitting set problems*, Discrete & Computational Geometry, 44 (2010), pp. 883–895.

[121] G. Narasimhan and M. Smid, *Geometric Spanner Networks*, Cambridge University Press, New York, NY, USA, 2007.

[122] G. Naves and A. Sebő, *Multiflow feasibility: an annotated tableau*, in Research Trends in Combinatorial Optimization, W. Cook, L. Lovász, and J. Vygen, eds., Springer, 2009, pp. 261–283.

[123] G. L. Nemhauser and L. E. Trotter, *Vertex packings: Structural properties and algorithms*, Mathematical Programming, 8 (1975), pp. 232–248.

[124] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, *An analysis of approximations for maximizing submodular set functions—I*, Mathematical programming, 14 (1978), pp. 265–294.

[125] C. H. Papadimitriou, *The euclidean travelling salesman problem is NP-complete*, Theoretical Computer Science, 4 (1977), pp. 237 – 244.

[126] M. D. Plummer, *Some covering concepts in graphs*, Journal of Combinatorial Theory, 8(1) (1970), pp. 91–98.

[127] ——, *Well-covered graphs: a survey*, Quaestiones Mathematicae, 16(3) (1993), pp. 253–287.

[128] P. Przytycki, *Arcs intersecting at most once*, Geometric and Functional Analysis, 25 (2015), p. 658–670.

[129] Y. Rabani and G. Scalosub, *Bicriteria approximation tradeoff for the node-cost budget problem*, ACM Transactions on Algorithms, 5 (2009), pp. 19:1–19:14.

[130] P. Raghavan, *Probabilistic construction of deterministic algorithms: Approximating packing integer programs*, Journal of Computer and System Sciences, 37 (1988), pp. 130 – 143.

[131] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, *The four-colour theorem*, Journal of Combinatorial Theory, Series B, 70 (1997), pp. 2–44.

[132] N. Robertson and P. Seymour, *Graph minors. XIII. The disjoint paths problem*, Journal of Combinatorial Theory, Series B, 63 (1995), pp. 65–110.

[133] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, Springer, 2003.

[134] A. Sebő, *Potentials in undirected graphs and planar multiflows*, SIAM Journal of Computing, 26 (1997), pp. 582–603.

[135] A. Sebö, *Integer plane multiflows with a fixed number of demands*, J. Comb. Theory, Ser. B, 59 (1993), pp. 163–171.

[136] L. Seguin-Charbonneau and F. B. Shepherd, *Maximum edge-disjoint paths in planar graphs with congestion 2*, in Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS), R. Ostrovsky, ed., 2011, pp. 200–209.

[137] P. D. Seymour, *On odd cuts and plane multicommodity flows*, Proceedings of the London Mathematical Society, s3-42 (1981), pp. 178–192.

[138] J. B. Shearer, *A note on the independence number of triangle-free graphs*, Discrete Mathematics, 46 (1983), pp. 83 – 87.

[139] H. U. Simon, *On approximate solutions for combinatorial optimization problems*, SIAM J. Discrete Math., 3 (1990), pp. 294–310.

[140] P. Slavík, *A tight analysis of the greedy algorithm for set cover*, in Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, New York, NY, USA, 1996, Association for Computing Machinery, p. 435–441.

[141] É. Tardos and V. V. Vazirani, *Improved bounds for the max-flow min-multicut ratio for planar and $K_{r,r}$-free graphs*, Information Processing Letters, 47 (1993), pp. 77–80.

[142] P. Turán, *An extremal problem in graph theory*, Mat. Fiz. Lapok, 48 (1941), pp. 436–452.

[143] F. Vandin, E. Upfal, and B. J. Raphael, *Algorithms for detecting significantly mutated pathways in cancer*, Journal of Computational Biology, 18 (2011), pp. 507–522.

[144] V. V. Vazirani, *Approximation Algorithms*, Springer, 2001.

[145] V. V. Vazirani, *Euclidean TSP*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 84–89.

[146] V. Wei, *A lower bound on the stability number of a simple graph*, Bell Laboratories Technical Memorandum, 81-11217-9 (Murray Hill, NJ, 1981).

[147] L. A. Wolsey, *Maximising real-valued submodular functions: primal and dual heuristics for location problems*, Mathematics of Operations Research, 7 (1982), pp. 410–425.

[148] D. Zuckerman, *Linear degree extractors and the inapproximability of max clique and chromatic number*, in Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing, STOC '06, New York, NY, USA, 2006, ACM, pp. 681–690.

## RÉSUMÉ

L'approche gloutonne est naturelle pour concevoir un algorithme. Elle permet la conception d'algorithmes rapides, faciles à implémenter, qui produisent en moyenne des solutions de qualité, pour de nombreux problèmes d'optimisation. Pour ces raisons, c'est une heuristique importante en pratique. Dans cette thèse, nous présentons des algorithmes gloutons pour trois problèmes d'optimisation NP-difficiles. Nous discutons les relations entre ces algorithmes, leurs techniques de preuve et la structure des problèmes étudiés.
Dans la permière partie de cette thèse, on se concentre sur le problème de la couverture maximale avec une contrainte de connexité, contrainte que l'on rencontre lors de la conception de réseaux sans fil. Nous montrons que le problème est NP-difficile, même quand la couverture et la connexité proviennent d'un ensemble de disques unité dans le plan. Pour ce cas particulier, nous montrons que choisir de manière gloutonne deux disques qui maximisent le gain marginal tout en maintenant la solution connexe réalise une 2-approximation. Nous améliorons ce ratio d'approximation en donnant un schéma d'approximation en temps polynomial avec une légère augmentation de ressource, basé sur la technique d'Arora pour le voyageur de commerce euclidien.
Dans la deuxième partie de cette thèse, on se concentre sur le problème du stable maximum. Une approche naturelle est de successivement choisir un sommet de degré minimum, le placer dans la solution en construction, puis le retirer avec ses voisins du graphe. Nous présentons une nouvelle technique pour analyser la performance de cette approche gloutonne dans différentes classes de graphes et abordons la question suivante : s'il y a plusieurs sommets de degré minimum, lequel l'algorithme devrait-il choisir pour maximiser la taille de la solution finale ? Avec cet outil, nous concevons une règle pour briser les cas d'égalité, qui conduit à la meilleure approximation possible dans les graphes sous-cubiques et pour ce type d'algorithmes gloutons. Nous complémentons ces résultats par des résultats négatifs qui suggèrent que la conception de bonnes règles brisant les cas d'égalité est une tâche difficile.
La troisième et dernière partie de la thèse est consacrée au problème du multiflot entier maximum. Ce problème est difficile et a été très étudié. Par exemple, une approximation de facteur constant est probablement impossible même quand le graphe d'offre est planaire et cubique. Dans le cas particulier où le graphe d'offre et les arêtes de demande forment ensemble un graphe de genre borné, nous présentons un algorithme avec un facteur d'approximation constant. L'algorithme consiste en une succession de procédures gloutonnes qui exploitent les propriétés topologiques des graphes et des lacets sur des surfaces.



## MOTS CLÉS

optimisation combinatoire, algorithmes, approximation, glouton



## ABSTRACT

The greedy approach is natural for the design of algorithms. It is fast, easy to implement, has a good performance on average, and is applicable to many optimization problems in various settings. For those reasons, it is an important heuristic in practice. In this thesis, we present greedy algorithms for three different NP-hard optimization problems. We discuss the relation between those algorithms, their proof techniques and the structure of the problems under study.
In the first part of this thesis, we focus on the Maximum Coverage problem with a connectivity constraint, which comes up for the design of wireless networks. We show that the problem is NP-hard, even when the coverage and the connectivity are induced by a set of unit disks in the plane. For that special case, we show that greedily picking two disks so as to maximize the marginal area covered while correctness achieves a 2-approximation. We further improve the approximation ratio by providing a PTAS for this problem with a small resource augmentation, using Arora's shifted grid dissection technique.
In the second part of this thesis, we focus on the Maximum Independent Set problem. A natural approach is to repeatedly pick a vertex of minimum degree, and remove it and its neighbours from the graph. We present a new technique to analyze the performance of this greedy approach in various classes of graphs and address the following question: if there are several minimun-degree vertices, what vertex should the greedy algorithm pick in order to maximise the size of the final solution ? With this tool, we design an "ultimate tie-breaking" rule that leads to the best possible approximation ratio for sub-cubic graphs and for this type of greedy algorithms. We complement this by lower bound results that show that designing good tie-breaking rules is a difficult task.
The third and last part of the thesis is devoted to the Maximum Integral Multiflows problem. The problem is difficult and well-studied. For instance, a constant factor approximation is unlikely to exist even when the supply graph is planar and cubic. In the special case where the supply graph together with the demand edges form a bounded genus graph, we present a constant factor approximation algorithm. The algorithm consists of a succession of greedy procedures that exploit topological properties of graphs and curves on surfaces.



## KEYWORDS

combinatorial optimization, algorithms, approximation, greedy