Applications and complexity of greedy algorithms in optimisation and mechanism design

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Nan Zhi

June 2019

Abstract

This thesis studies one of the most classical algorithmic optimisation paradigms, which is the greedy paradigm. One of the main motivations for this thesis is to explore further applications of the greedy paradigm and to provide foundation for the mathematical analysis of the resulting greedy algorithms. We present a novel mechanism design model in the area of ontology alignment and study greedy mechanisms for this model. In particular, we provide bounds on the approximation ratios of truthful mechanisms in our model. Then we study the price of anarchy and that of stability of Nash implementations of greedy mechanisms. This study shows that the greedy paradigm provides a fast, practical and efficient mechanism. We then study the computational complexity and approximability properties of the greedy algorithm for the classic optimisation problem of computing maximum size independent sets in bounded degree graphs. We develop a series of novel methods to design advice for the greedy algorithm together with novel mathematical techniques of analysing the approximation performance of the resulting greedy algorithms. These techniques allow us to successfully design and prove the approximation ratios of the best known greedy algorithm for maximum size independent sets on sub-cubic graph, by designing a scheme to use savings to precisely pay for the greedy solution as compared to the optimal independent set. This scheme is highly non-local and it requires a complex inductive argument which we provide. We apply these methods also to the maximum size independent set problem on general bounded degree graphs and obtain near tight results. The maximum size independent set problem and the minimum size vertex cover problem are the two, mutually complementary, generic combinatorial optimisation problems. We apply our techniques also to the minimum size vertex cover problem and obtain improvements compared to previous results. Thus our techniques have a great potential for further applications of analysing greedy on other classes of graphs and for related optimisation problems. Interestingly, our theoretical analysis has been informed and advised by an experimental analysis which is also presented in the thesis. Finally, we also present an experimental analysis of our greedy algorithms.

Acknowledgements

Endeavour to a PhD degree is one of the most wonderful and grateful journeys of my life. This journey would not be culminant and memorable without various people I have known and came cross, who greatly helped and supported me.

First, and foremost, I am indebted to my primary supervisor, Professor Piotr Krysta, one of the most enthusiastic and genius persons I have ever known. Piotr taught me a lot on almost every important aspect of research, about how to correctly think and explore the problem in the most rigorous way, and helped me to learn the way to solve it, and with his useful suggestions on the presentation of the results and writing of papers. His deep insights have been invaluable to me. Above all, the joy and enthusiasm he exemplified in the research was contagious and motivational for me, inspiring me to experience the joy of seeking truth. I would like to thank my second supervisor Terry Payne, who impressed me with his exceptional clarity on interpreting the significance of research problems and his great kindness and patience. I would like to thank Mathieu Mari, one of my collaborators and friends, who is very brilliant and chic person, and always shares his valuable insight and ideas in the discussion to tackle problems.

I wish to thank University of Liverpool for providing such a great academic and living environment. I wish to thank the people in the faculty: Giorgos Christodoulou, Dominik Wojtczak, Martin Gairing, Rahul Savani, Paul Spirakis, John Fearnley, Jinshan Zhang, Argyrios Deligkas, Eleftherios Anastasiadis, Eleni Akrida, Alkmini Sgouritsa, Grammateia Kotsialou, Gregory Palmer, Thomas Spooner, Grzegorz Muszynski, among others, for I learned much from them on academic and other matters.

I wish to thank to all friends of mine in this memorable journey of life, and you are always my indispensable treasures.

Last but not least, I wish to appreciate my family, my parents and grandparents, for all their everlasting love and throughout support for me.

This Thesis is another departure of my journey of seeking Truth.

Preface

The sources of other materials are identified here. We describe here which chapters in this thesis have appeared in which papers as below. The work in Chapter 3 is based on the joint work with Piotr Krysta, Minming Li and Terry Payne, see paper [43] for a preliminary publication of this work. Chapters 4, 5, 6 and 7 are based on joint work with Piotr Krysta and Mathieu Mari, see [44].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this introduction we mention some optimisation problems and complexity classes, however we do not define them formally here, because they are not closely studied in this thesis. For formal definitions of problems which are closely studied in this thesis, and notions related to approximability, complexity and graphs, we refer the reader to Section 1.1 and Chapter 2.

The greedy algorithm paradigm is one of the most important in algorithm design, because of both its simplicity and efficiency. In the domain of algorithm design, greedy paradigms are mostly used in at least three directions: they provide exact algorithms for a variety of problems; they are frequently the best approximation or good enough algorithms for hard optimisation problems. And, due to their simplicity, efficiency and not yet discovered properties, they are frequently used as heuristics for hard optimisation problems. This is often despite the fact that their theoretical analyses are unknown, or far from being tight, or even known to be poor in the worst case. **One of the main motivations for this thesis is to explore further applications of the greedy paradigm and to provide foundation for the mathematical analysis of the resulting greedy algorithms.**

In principle, the greedy paradigm is any algorithm that complies with the following rule: at each step, it takes some element into solution which is regarded as "best" according to a given criterion, and we never alter the solution already found, in the future iterations. Note that the criterion of "best" that the algorithm adopts is not precisely defined as it depends on the concrete problem. The criterion might, for instance, be defined as a minimal or maximal size of a set, or value of a weight function on elements, or even by a sophisticated function on a local structure of the solution and properties of the input data. Therefore, rather than presenting general and abstract

definition of greedy algorithms, we will present some concrete examples.

For the purpose of applying the greedy paradigm to find an exact (optimal) solution, one of the most prominent problems is the *shortest path problem*, and *Dijkstra's* algorithm [16] gives the optimal solution. Note that the Dijkstra's algorithm belongs to the class of greedy algorithms. Another well known example in discrete optimisation is perhaps the optimality of the greedy algorithm for *matroids*, whose discovery dates back to Edmonds [19]. Matroids provide a full characterisation of a structure where one kind of greedy algorithm (in each iteration, the greedy algorithm takes an element e whose weight is maximum, and excludes e from the universal set) can achieve an optimal solution [56]. Various generalisations of Edmonds's approach have been introduced, for instance, generalised polymatroids in [23]. Korte and Lovasz [41] observed that in many cases, even if we relax one condition of matroid, the same greedy algorithm still performs well. In fact, they introduced a notion of *greedoid* to give a generalised such characterisation.

For the purpose of approximation of hard optimisation problems, greedy algorithms have been applied to numerous problems. For instance, for the *minimum set cover* problem, there is a simple greedy algorithm [69], which achieves an approximation ratio of $\ln(k)$, where k is an upper bound on the number of elements in any set. This greedy algorithm applies the following rule: in each iteration, choose the set that contains the largest number of uncovered elements, and remove the newly covered elements from the universe. In [61], Raz and Safra proved a lower bound of $(1 - o(1)) \cdot \ln n$ on the approximation ratio for the minimum set cover problem, under the assumption $P \neq NP$. This inapproximability result shows that the greedy algorithm is essentially the best-possible polynomial time approximation algorithm for the set cover problem. A similar result applies to the complementary problem of set cover: the maximum independent set (MIS) problem. A simple greedy algorithm achieves an n-approximation ratio for the MIS problem, where n is the number of vertices in the input graph. And, this problem cannot be approximated to any factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, in polynomial time, unless $NP \subseteq ZPP$ [35]. This implies that the greedy algorithm is essentially optimal for the MIS problem on general graphs. The best known analysis of greedy by Halldórsson and Radhakrishnan [31] for MIS implies the approximation ratio of $(\Delta + 2)/3$, and better ratios are known for small values of Δ. Here, we assume that the graph has maximum degree Δ. A detailed survey of results for the MIS problem will be presented in Chapter 6. Furthermore, the greedy paradigm applies to many other problems. In [37], Jain et al. provide two greedy algorithms for the metric uncapacitated facility location problem, and the approximation ratio is 1.861 and 1.61. A tight result of the

2-approximation ratio by a greedy algorithm for the k-center problem is presented in [69].

For heuristic purposes, for several combinatorial optimisation problems, the greedy algorithms are also good in the practical sense. This is even despite the fact that their theoretical analysis in terms of the worst case performance is not as good as expected. Examples of such practical applications can be found, for instance, in the following papers: [40], [26], [36] and [63]. These applications show that greedy algorithms usually find out the optimum solution, and for majority of inputs, they generally output relatively good solutions compared to the optimum. And it is quite rare that they output a solution which reaches the worst case approximation ratio or even close to it. The computational experiments show that the greedy algorithm is a popular choice for tour construction heuristics. For instance, greedy methods work at acceptable level for the Euclidean Travelling Salesman Problem (TSP), and present more accurate and faster solutions for the Asymmetric TSP problem [36]. For packing and covering problems, in [26], the authors compare four common algorithms: greedy algorithm, linear programming rounding based algorithm, primal-dual algorithm, and a dual algorithm. All of these algorithms share basically the same approximation ratio in the theoretical worst-case analysis. However, the greedy algorithm turned out to be outstandingly better than the other three approaches when tested on many "typical" instances.

The greedy paradigm also finds useful applications in other areas closely related to optimisation. For example, a recent active research area called algorithmic mechanism design. Algorithmic mechanism design lies in an intersection of the research areas of economic game theory and computer science. It is about the design and analysis of games, where players have unknown and private utilities, and at an equilibrium of the designed game, the mechanism designer's goals are obtained in reasonable time complexity independently of players' utilities.

The motivation for studying greedy paradigm in algorithmic mechanism design is quite strong. Firstly, for many settings in mechanism design, such as combinatorial auction problem [10, 42, 53], greedy algorithms are natural candidates for truthful mechanisms construction. That is because they embody the natural monotonicity properties associated with the concept of truthfulness [50]. Moreover, in the prospect of Nash equilibrium implementation, the greedy algorithm also usually achieves relatively good price of anarchy, or price of stability [48]. Secondly, for many combinatorial auction problems, the greedy algorithms are known to obtain asymptotically tight approximation ratio bounds, even though the mechanisms are simple. For example, a greedy algorithm obtains a tight approximation of $O(\sqrt{m})$ for combinatorial auctions

with single-minded bidders, and it is a truthful mechanism in this setting [53], where m is the number of goods. Finally, which is perhaps the most important reason, many auctions used in practice apply greedy methods, despite the fact that they may not be incentive compatible (truthful) and may not have a good theoretical bound on their approximation guarantee. Importantly, simple greedy mechanisms seem to be a good candidate for auctions and related settings due to other considerations beyond truthfulness, such as being easily understandable to the public and for their perceived fairness.

Although a greedy algorithm is usually easy to understand and to implement, and in general, its approximation ratio is good, it is usually difficult to analyse the performance of such an algorithm precisely. The current approaches are either to focus on the local behaviour of the algorithm or to focus on the global properties that lead to bounds far from the tight bound. Natural questions arise of how to analyse the performance of a greedy algorithm to obtain the upper bound on its approximation ratio, that is close to a lower bound? What ideas and methods can provide a tight analysis, by which we can consider both local and global properties in a unified way without any or much loss?

In this thesis, we focus on the classic combinatorial optimisation problems, namely, the maximum weighted bipartite matching and the maximum independent set problems. Using the first of those problems, we will define a novel setting of mechanism design in context of ontology alignment, a problem which finds many important practical applications. We provide a complete picture of the exact and approximate performance of various truthful mechanisms for this setting. We study the approximate performance of truthful mechanisms in dominant strategies, both with and without payments and both deterministic and randomised mechanisms. Then we study the approximate performance of truthful greedy mechanisms in Nash equilibria implementation. On the one hand, this settles the theoretical analysis of the complexity and approximability of our model in almost all known mechanism design settings. On the other hand, an interesting take-home message from our results is that although dominant strategy mechanisms might not be enough time efficient for the practical applications in ontology alignment, greedy mechanisms provide an excellent, time-efficient, alternative for such applications in Nash equilibria implementation.

For the second problem, the maximum independent set problem, we focus on the classic minimum-degree greedy algorithm. The approximation performance of this algorithm has been analysed in numerous previous papers. Its importance also stems from the fact that because of its simplicity, it has been used as a tool in various proofs in graph theory multiple times. Thus this algorithm is important on its own right and

its efficiency makes it a natural choice in practical implementations where the independent set problem is used. Our main contribution here is a new mathematical theory for the design and analysis of the minimum-degree greedy algorithm with advice (of which minimum degree vertex should greedy choose in case of ties) for the maximum independent set problem on bounded degree graphs. The main highlight of our work here is a very precise mathematical proof of the 4/3-approximation ratio of this algorithm on sub-cubic graphs, i.e., with maximum degree at most 3, which implies the best currently known analysis of greedy in this setting. Moreover, our tools provide a new and simple proof of the approximation ratio of greedy on graphs with any bounded degree. Further applications include faster approximation algorithms for another classic optimisation problem, the minimum vertex cover problem on sub-cubic graphs.

The main novelty that our new analysis techniques are based on is a very precise potential function that allows us to design a scheme to pay for the greedy solution as compared to the optimal independent set. This scheme is highly non-local and it requires a very precise inductive argument for it to be applied. The importance of these new techniques also stems from the fact that it gives a great potential for further applications of analysing greedy on other classes of graphs and for related problems. In conclusion, this study sheds a great light on the essence of greedy algorithms.

1.1 Preliminaries

In this section, we define basic concepts which are used in the thesis.

A graph is a pair $G = (V, E)$ of finite sets such that $E \subseteq V \times V$. The elements of V are *vertices*, the elements of E are 2-element subsets of V, are its *edges*. A graph with vertex set V is said to be a graph on V. The vertex set of a graph G is referred to as $V(G)$, and its edge set as $E(G)$. These conventions are independent of any actual names of these two sets: the vertex set W of a graph $H = (W, F)$ is still refered to as $V(H)$, not as $W(H)$. In this thesis, we shall not always distinguish strictly between a graph and its vertex or edge set. For example, we may speak of a vertex $v \in G$, rather than $v \in V(G)$, and so on.

A vertex v is incident with an edge e if $v \in e$; then e is an edge at v. The two vertices incident with an edge are its end vertices or ends, and an edge joins its ends. An edge $\{x, y\}$ is usually written as (x, y), xy or yx. Two vertices x, y of G are *adjacent*, or *neighbours*, if xy is an edge of G. Two edges $e \neq f$ are adjacent if they have an end in common. If all the vertices are pairwise adjacent, then G is complete. A complete graph on n vertices is a K^n; a K^3 is called a *triangle*.

We set $G \cup G' := (V \cup V', E \cup E')$ and $G \cap G' := (V \cap V', E \cap E')$. If $G \cap G' = \emptyset$, then G and G' are *disjoint*. If $V' \subseteq V$ and $E' \subseteq E$, then G' is a *subgraph* of G, written as $G' \subseteq G$.

If $G' \subseteq G$ and G' contains all the edges $xy \in E$ with $x, y \in V'$, then G' is an *induced subgraph* of G; we say that V' *induces* G' in G, and write $G' =: G[V']$. Thus if $U \subseteq V$ is any set of vertices, then $G[U]$ denotes the graph on U whose edges are precisely the edges of G with both ends in U.

If U is any set of vertices of G, we write $G \setminus U$ for $G[V \setminus U]$. In other words, $G \setminus U$ is obtained from G by *deleting* all the vertices in $U \cap V$ and their incident edges.

Let $G = (V, E)$ be a graph. The set of neighbours of a vertex v in G is denoted by $N_G(v)$, or briefly by $N(v)$. More generally, for $U \subseteq V$, the neighbours in $V \setminus U$ of vertices in U are called neighbours of U, which is denoted by $N(U)$. We also use $N_G(U)$ to denote the neighbours of U according to a given graph G. We also abuse $N_G(e)$, $e = (v_1, v_2) \in E$ to denote the set of endpoint v_1, v_2 of e. The *degree* $d_G(v) = d(v)$ of a vertex v is the number $|E(v)|$ of edges at v according to given graph G. This is equal to the number of neighbours of v. A vertex of degree 0 is *isolated*. If all the vertices of G have the same degree k, then G is *k-regular*. A 3-regular graph is called *cubic*, and a graph with maximum degree at most 3 is called *sub-cubic*.

For a given $r \geq 1$, an algorithm $\mathcal{A}$ is an r-approximation algorithm for the maximization (minimization, respectively) problem Π, if for any instance I of Π, a solution generated by $\mathcal{A}$ with value $A(I)$ respects $r \cdot A(I) \geq OPT_\Pi(I)$ ($A(I) \leq r \cdot OPT_\Pi(I)$, respectively), where $OPT_\Pi(I)$ denotes the value of the optimal solution of Π on instance I. The r is called also an *approximation guarantee, factor* or *ratio*. We will be exclusively interested in polynomial time approximation algorithms, thus calling them just approximation algorithms. If there exists an r-approximation algorithm for an optimisation problem Π, then we also say that problem Π is *approximable to within* (a factor of) r.

1.2 Organisation

This thesis is organised as follows. Chapter 2 provides background information presented in a high-level way, which is necessary to follow this thesis. In Chapter 3, we study the problem of ontological alignment in both dominant strategy and Nash equilibrium implementation and demonstrate that a mechanism based on greedy algorithm achieves good price of anarchy. In the next three chapters, Chapter 4, 5 and 6, we turn our attention to the study a classic combinatorial optimisation problem: the maximum

size independent set problem. In Chapter 4, we provide a series of negative results about limitations of the greedy algorithm for this problem. In Chapter 5, we start our study of the maximum size independent set problem on sub-cubic graphs, including experimental research and explanations of our approach. Then, the culmination of the thesis, in Chapter 6, which is self-contained, is devoted to present the formal result of a simple greedy algorithm which achieves a $\frac{4}{3}$-approximation ratio in $O(n^2)$ time. Finally, in Chapter 7, further applications of the techniques developed in the previous two chapters are presented. Namely, we extend these techniques to higher degree graphs, while we also study the connection between independent set and vertex cover. In Chapter 8 we conduct an experimental study on the heuristic and practical aspects of the greedy algorithm for the maximum independent set problem on sub-cubic graphs. The conclusion Chapter 9 presents possible directions of further study.

1.3 Overview of the main contributions

The main contributions of this dissertation are as follows.

- In Chapter 3, we introduce a novel model for mechanism design in context of ontology alignment. The optimisation problem is an appropriate version of the bipartite matching problem with a natural setting of agent's private information related to ontologies. We first show the impossibility results for this problem for mechanisms in dominant strategy implementation. Then, we design a simple but efficient mechanism and analyze it by the greedy paradigm. Namely, we prove bounds on the price of anarchy and price of stability of the resulting game in the Nash equilibria implementation.
  Highlights: The ontology alignment problem is solved in practise on huge input data, thus running time of the mechanisms is crucial, where even difference between $O(n^2)$ and $O(n)$ matters. The main message of our results in this new mechanism design model is that the dominant strategy mechanisms cannot be time efficient. But when we resort to Nash equilibria implementation we show that the greedy paradigm provides a very fast and practical mechanism.

- In Chapter 4, we study inapproximability of greedy algorithms for the maximum independent set (MIS) problem. We show that the algorithmic problem of identification of graphs on which greedy algorithm obtains any constant approximation ratio is *NP*-hard. A previous result shows that it is *co-NP*-complete [8] by a different proof. The authors also prove that it is *NP*-complete to decide if there

exists a sequence of choices of greedy leading to a fixed approximation. We also present a simple proof to show for general graphs, that to advice greedy to obtain the optimal solution is *NP*-hard. Furthermore, we are able to prove that the same statement is true even for cubic planar graphs, which means that the problem of advising greedy algorithm to obtain its optimum is *NP*-hard.

Highlights: This is the first result of the hardness of greedy on such restricted class of graphs, which significantly strengthens the previously known hardness results. This result potentially implies that it is difficult to algorithmically advise the greedy algorithm to obtain a good solution even compared to the maximum size greedy independent set.

- In the consecutive Chapters 5 and 6, we introduce a novel collection of mathematical tools and techniques to design and analyse greedy approximation algorithms with advice for the MIS problem on sub-cubic graphs. These techniques let us prove the existence in polynomial time of the $\frac{4}{3}$-approximate greedy algorithm, which is the best approximation ratio currently known for greedy algorithms for MIS on sub-cubic graphs.[1] This also gives the fastest known approximation algorithm with this approximation ratio, running in time $O(n^2)$, whereas the best previously known algorithm (based on local search) with the same ratio has running time $O(n^{7.3})$ [12], where n is the number of vertices in the input graph.

  Highlights: The main novelty of our new analysis technique is based on a very precise potential function that allows us to design a scheme to pay for the greedy solution as compared to the optimal independent set. This scheme is highly non-local and it requires a complex inductive argument which we provide. The importance of these new techniques also stems from the fact that it gives a great potential for further applications of analysing greedy on other classes of graphs and for related problems. Moreover, we believe that with further development, these tools will allow us to also analyse algorithms which do not comply with greedy constrains.

- In Chapter 7, we apply our techniques to obtain improved fast greedy algorithms for MIS on Δ-bounded degree graphs for any Δ and for $\Delta = 4$. For any Δ, we obtain a different proof than previous research [31] by using the technique

[1] We have recently managed to finally prove the existence of a $\frac{5}{4}$-approximation greedy algorithm, which shows the strength of our tools and techniques. This proof however is not present in this thesis, see [44].

from the previous chapters, proving that any greedy algorithm provides a $\frac{\Delta+2}{3}$-approximation ratio. This is an alternative and much shorter and simpler proof for the same result in [31]. For $\Delta = 4$, we obtain a 1.8-approximation ratio greedy algorithm, which gives the best known time complexity algorithm with this approximation ratio. Moreover, by strengthening a construction in [31] we prove that *any* greedy algorithm for MIS on Δ-bounded degree graphs has an approximation ratio at least $\frac{\Delta+1}{3} - O(\frac{1}{\Delta})$.

- Also in Chapter 7 we apply our new techniques to obtain improved approximation algorithms for another classic optimisation problem on sub-cubic graphs, the minimum size vertex cover problem. We prove the existence of an $\frac{5}{4}$-approximation algorithm, which is based on the greedy paradigm, but it is not a canonical greedy algorithm. The running time of this algorithm is $O(n^2)$. We also find a better algorithm by conducting a pre-processing step to obtain a $\frac{6}{5}$-approximation ratio and the running time is bounded by complexity of the max-flow computation on the graph. In contrast, the best currently known approximation ratio for the minimum vertex cover problem is 1.166 by a local search approach [5] and the running time of this algorithm is at least $O(n^{50})$, and $O(n^{17.28})$ for a $\frac{6}{5}$-approximation ratio.

  Highlights: This new $\frac{6}{5}$-approximation algorithm also reflects the strength of our techniques, and the potential of extending them to different optimisation problems.

- We also conduct an experimental study in Chapter 8, to illustrate the efficiency of the greedy algorithm for MIS on sub-cubic graphs, and to provide some insights on the average approximation ratio.

Chapter 2

Background

The precise technical definitions of the various notions related to game theory and mechanism design, complexity theory and (approximation) algorithms, used in this thesis are presented in the appropriate respective chapters. This chapter contains informal explanations of these concepts, that are necessary to follow the content of this dissertation. The exceptions are the notions which refer to ontologies and ontology alignment, which will be defined more formally in this chapter and we will not redefine them later on.

2.1 Game theory and mechanism design

On the highest level of game theory, let us first explain the notion of a *(non-cooperative, one-shot) game*. Given a finite set of *selfish agents (players)*, each agent has a finite set of *actions* to choose among. Such a game is played by the agents who simultaneously choose an action from their action sets. For such selection of actions of each agent, every agent gets a specific *payoff*, which is also sometimes called a *utility*. The goal of the agents, supposedly without knowing what actions the other agents choose, is to choose an action from its action set, that maximises their payoff.

As a simple example let us consider the classic *Prisoners' Dilemma*, whose description is taken from the book [55], see Example 1.1 there. Two prisoners are on trial for a crime and each faces a choice to confess to the crime or to remain silent. If they both remain silent then the authorities will not be able to prove charges against them and they will both serve a short prison term, say 2 years, for minor offenses. If only one of them confesses, then its term will be reduced to 1 year and it will be used as a witness against the other prisoner, who will get a sentence of 5 years. Finally if they

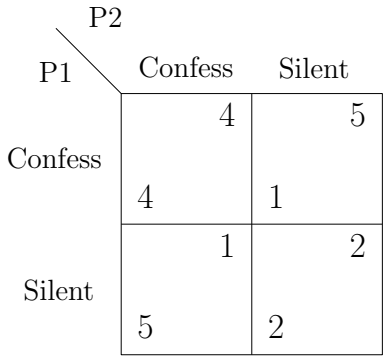


Figure 2.1: Prisoners' Dilemma

both confess they both will get a small break for cooperating with authorities, and will have to serve prison sentences of 4 years each (rather than 5).

In this game the prisoners are the 2 players, P1 and P2, and each player has the same set of actions $\{C, S\}$, where C denotes Confess and S denotes Silent. This game can be modeled as in Figure 2.1 where the numbers in this matrix are the costs that they incur for their choices of actions. Thus their payoffs are the negated costs. For example if P1 chooses Silent and P2 chooses Confess, then P1's cost is 5 (and payoff -5), and P2's cost is 1 (and payoff -1).

Based on this game we can also define basic solution concepts in game theory: namely *equilibria*. A *(pure) Nash equilibrium* of a game is a selection of actions of all players such that no single player can change its action unilaterally to increase its payoff, that is, where the other players stick to their chosen actions.

For instance, observe that the selection P1 = C and P2 = C is a pure Nash equilibrium of the Prisoners' Dilemma. That is because if P1 changes to S, and P2 sticks with C, then P1 decreases its payoff from -4 to -5; the same holds symmetrically for P2 changing its action. This Nash equilibrium is also a very strong kind of equilibrium, called equilibrium, solution or implementation in *dominant strategies*. This means that for a given player, say P1, even if the other player P2 does not stick to its action C, still the best action for P1 is to indeed stick to C. Indeed, it is best for P1 to keep action C when P2 chooses C (because then if P1 switches from C to S, then its payoff decreases from -4 to -5). And, also it is best for P1 to keep action C when P2 chooses S (because then if P1 switches from C to S, then its payoff decreases from -1 to -2). To conclude, each player's dominant strategy in Prisoners' Dilemma is to play Confess.

There are also *mixed* Nash equilibria, where players are allowed to choose probability distribution on the set of their pure strategies. For instance in Prisoners' Dilemma,

player P1 may choose C with probability 2/3 and S with probability 1/3. If, for instance, P2 chooses C with probability 1 (pure strategy), then the *expected payoff* of P1 will then be $(2/3) \cdot (-4) + (1/3) \cdot (-5)$. In a *mixed Nash equilibrium*, no agent can increase its expected payoff, by unilaterally deviating from its mixed strategy, while the other players stick to their chosen mixed strategies. A given game may not possess a pure Nash equilibrium, but if the set of agents is finite and their sets of pure strategies (actions) are also finite, then such a game will always possess a mixed Nash equilibrium (this is the famous Nash's Theorem, see, e.g., [55]).

This thesis contributes to the part of algorithmic game theory, called algorithmic mechanism design (AMD). AMD is the study of optimisation problems where part of the input data is private data of selfish agents. We are interested in the design of truthful mechanisms, where the goal of such mechanism is to incentivise the agents to truthfully report their private data to the mechanism and to optimise the objective function of the problem under consideration.

Intuitively, a *mechanism* is a pair that consists of an *algorithm* and a *payment scheme or rule*. The algorithm outputs a solution to the problem under consideration. Such solution usually contains a subset of the agents. The payment scheme provides monetary payments for the agents present in this solution to incentivise them to be *truthful*. Sometimes however, in quite rare cases, a mechanism *without payment* can also incentivise the agents to be truthful. A mechanism defines a game in the sense defined above by specifying for each agent *valuations* that this agent associates with a solution output by the mechanism. Then, each agent's action set is a *bid*, which is a value that this agent reports to the mechanism about its private data. The mechanism collects these bids and outputs a specific mechanism's solution. Then, the agent's payoff is just its valuation of that solution minus the mechanism's payment for this agent.

As an example, let us consider a simple instantiation of the Vickrey-Clarke-Groves (VCG) mechanism,[1] which is the single item *second-price auction*. We have $n \geq 2$ agents who want to buy a single item. Each agent i has valuation $v_i \geq 0$ for the item. The second-price auction (mechanism) collects bids $b_1, \ldots, b_n$, which are non-negative numbers, from the agents and declares the highest-bid agent a winner (this is a trivial algorithmic part of the mechanism). The payment scheme is to ask the winner to pay

[1] Let Π be a maximisation problem with n agents, where each agent has a (private) valuation function over feasible solutions to problem Π. The Vickrey-Clarke-Groves (VCG) mechanism collects a bid (for the valuation function) from each agent and with these bids computes the optimal solution to Π that maximises the sum of agents' bids over this solution, called a social welfare. There is a way of defining payments to agents (Clarke payments) such that this mechanism is truthful in dominant strategies. This means that each agent's dominant strategy is to report to the mechanism as its bid their true valuation function. For more details, see book [55].

the value of the highest submitted bid among the remaining agents. For instance, if $n = 3$ and $v_1 = 5, v_2 = 7, v_3 = 2$ and the bids are $b_1 = 5, b_2 = 4, b_3 = 2$, then agent 1 wins and pays $b_2 = 4$ and its payoff is $v_1 - b_2 = 5 - 4 = 1$. The payoff of any losing agent is by convention 0. In this example, agents 1 and 3 bid truthfully (i.e., $b_1 = v_1$, $b_3 = v_3$), but agent 2 lies (i.e., $b_2 \neq v_2$). Because agent 2 lost the auction, its payoff is 0. But if 2 bids truthfully $b_2 = v_2 = 7$, and the others bid like before, then 2 wins the auction and its payoff is now $v_2 - b_1 = 7 - 5 = 2$. A simple case analysis shows that in the single item second-price auction truth-telling, that is choosing as action (bid) $b_i = v_i$, is the (weakly) dominant strategy of each agent. This implies that the choices of actions (bids) $b_1 = v_1, \ldots, v_n = b_n$ is a dominant strategy Nash equilibrium of this game.

We will consider general notions of truthfulness in the design of mechanisms. The first is truthfulness in dominant strategies (as explained above), where if an agent lies about its private data, then this agent's payoff is always worse (or not better, for weakly dominant), even if all other agents lie. Mechanisms which are truthful in dominant strategies, which we also call just truthful, are also called to possess a dominant strategy implementation. Here, we will also distinguish two further kinds of truthfulness in dominant strategies for randomised mechanisms, that is, mechanisms that use internal randomisation in their computation. These two kinds are universal truthfulness and truthfulness in expectation. A randomised mechanism is *universally truthful* if it is a probability distribution over a set of deterministic truthful mechanisms. Furthermore, a randomised mechanism is *truthful in expectation*, if any agent maximises its expected payoff by being truthful, independently of the other agents' declarations.

The second notion of truthfulness is truthfulness in Nash equilibria. In this case the mechanism should output a solution and a payment scheme such that the resulting game (as defined above) implies a Nash equilibrium.

As mentioned above, computational problems studied in computer science and in this thesis, usually come with a specific objective function that the mechanism is supposed to optimise (maximise or minimise). Focusing on maximisation problems, we desire to compute a solution that maximises the objective function value.

In context of game theory, the following useful notions of *price of anarchy* and *price of stability* help quantify the quality of Nash equilibria with respect to a given objective function.

For instance let us introduce an objective function called *social cost*, as the sum of the costs (numbers in Figure 2.1) of the two agents in a given solution (where solution is a choice of pure actions of both agents) of the Prisoners' Dilemma. Note that agents

want to minimise their social costs (as opposed to maximising of their payoffs in this game).

The social cost of the solution $P1 = C, P2 = C$ is 8. Observe, however, that the solution $P1 = S, P2 = S$ has the minimum social cost in this game of 4. The solution $P1 = C, P2 = C$ is a Nash equilibrium, whereas the optimal solution $P1 = S, P2 = S$ (i.e., the one that minimises the social cost) is not. Given a game, with specific minimisation objective function, the *price of anarchy* of this game is the worst-case ratio (over all instances of the game) between the objective value of any Nash equilibrium in the game and the objective function value of the optimal solution. For the Prisoners' Dilemma example with the social cost objective the price of anarchy is 2. Note, that to define the price of anarchy for a game with a maximisation objective (when the objective is non-negative) the ratio would be between the value of the objective of the optimum solution and the value of a Nash equilibrium. A related notion of the *price of stability* differs from the price of anarchy in that if on a given instance of the game there are multiple Nash equilibria, we take the one with the objective function value that is closest to the optimal objective value. Note that we define the price of anarchy and that of stability in a way that it is always a number of value at least 1 for both minimisation and maximisation objective.

Intuitively, a large price of anarchy (or stability) says that there is large loss in the objective function value when one resorts to (decentralised or anarchistic) Nash equilibria solutions, compared to that of the optimal (centralised) solution.

As another example let us consider the introduced single item second-price auction. We define the *social welfare* as the objective function value, that is, the (true) valuation of the winner of the auction. We have seen that the truthful solution, that is $b_1 = v_1, \ldots, v_n = b_n$, is a Nash equilibrium of this game, and in this solution the item goes to the highest bid (i.e., highest valuation) agent. This implies that the price of stability of this game is 1. It can be shown by a simple example that the price of anarchy of this game is unbounded.

We finally mention that the notions of the price of anarchy and stability can also naturally be defined for the mixed strategies equilibria. Furthermore, the notions of strategies, equilibria, price of anarchy and stability, etc, which were defined above, naturally extend when we allow for randomisation as part of agents' strategies (mixed strategies) and/or as a part of the mechanism (which translates into universal truthfulness or truthfulness in expectation). Because we use these notions under randomness only for a limited set of results, the precise definitions are in Section 3.4.

2.2 Optimisation, approximation and complexity

This thesis deals with various graph theoretic optimisation problems in context of their approximability and computational complexity.

The very basic notion of *an undirected graph* has been defined in Section 1.1. The following fundamental graph optimisation problems are studied in this thesis:

- the maximum weight bipartite matching problem (Chapter 3),
- the maximum size independent set problem (Chapter 4-8),
- and the minimum size vertex cover problem (Chapter 7).

We will define these problems very briefly here. An undirected graph $G = (V, E)$ is called *bipartite* if its vertex set V can be partitioned into two non-empty sets, U and W, that is $V = U \cup W$ and $U \cap W = \emptyset$, such that $E \subseteq U \times W$. This simply means that in a bipartite graph, edges can only run between these two sets U and W, but no two vertices from U are connected by an edge, and likewise, no two vertices from W are connected by any edge.

Given a graph $G = (V, E)$ we call a subset $E' \subseteq E$ of its edges a *matching* if no two edges in E' share the same end vertex. We also call a subset $V' \subseteq V$ of its vertices an *independent set* if no two vertices in V' are connected by an edge from E. Finally, a subset $V'' \subseteq V$ is called a *vertex cover* of G if for every edge from E at least one of its end vertices belongs to V''.

Given a bipartite graph $G = (U, W, E)$ with $V = U \cup W$, with non-negative weights on its edges, the *maximum-weight bipartite matching* problem is to compute a matching in G with the maximum possible sum of its weights on the edges.

Another fundamental graph optimisation problem is the *maximum size independent set (MIS)* problem, where for a given graph $G = (V, E)$, this problem asks for computing an independent set in G with maximum possible size.

Finally, the *minimum size vertex cover (MVC)* problem is, for a given graph $G = (V, E)$, to compute a vertex cover of G with the smallest possible size.

These three problems find numerous theoretical and practical applications. However, they greatly differ from the point of view of their computational tractability. For more details about relevant complexity theory, see, e.g., the Appendix in the book [69]. We will only explain very brief intuitions here.

For instance, the maximum weight bipartite matching problem is a very prominent member of the complexity class P, of efficiently solvable problems, that is, all problems

(whose decision versions are) solvable in deterministic polynomial time. On the other hand the maximum size independent set and the minimum size vertex cover problems are prominent members of the complexity class *NP*, of all problems (whose decision versions are) solvable in non-deterministic polynomial time. An alternative definition of the class *NP* is that a decision problem belongs to this class if there is a polynomial time algorithm that can *guess* a short certificate that can certify that the problem instance is a "Yes" instance. Once this certificate is guessed, it can be checked efficiently in polynomial time if it indeed certifies the "Yes" instance. A related complexity class is *co-NP*, whose definition is the same as *NP* but the algorithm certifies now a "No" instance of the problem.

In fact both MIS and MVC problems are *NP*-hard, which means that there are no deterministic polynomial time algorithms to solve these problems exactly, unless $P = NP$. This is the very famous P versus NP open problem in theoretical computer science, and it is widely believed that in fact $P \neq NP$. We also say that the decision versions of those problems, MIS and MVC, are *NP*-complete. Intuitively, if a problem is *NP*-hard or *NP*-complete, it means that there is an overwhelming mathematical evidence in the complexity theory that this problem cannot be solved efficiently, that is in polynomial time, to optimality. Similarly, if a problem is *co-NP*-hard, it is also widely believed that it does not possess an efficient exact polynomial time algorithm. For more details on complexity theory, the reader is referred to the book [58].

This brings us naturally to the notion of an *approximation algorithm*, which is an efficient, e.g., polynomial time, algorithm that computes an approximately optimal solution to a given optimisation problem. A formal definition of an r-approximation algorithm for a maximisation and minimisation problem has been given in Section 1.1.

Again, it turns out that the defined *NP*-hard problems, MIS and MVC differ greatly with respect to what approximation guarantees can be achieved for them in polynomial time. For instance, there are known polynomial time 2-approximation algorithms for MVC (or even ones with smaller approximation ratios for bounded-degree graphs). However, in general graphs, MIS cannot be approximated in polynomial time with ratio $n^{1-\epsilon}$ for any fixed $\epsilon > 0$, where n is the number of vertices in the input graph. We say that MIS is *inapproximable* within ratio of $n^{1-\epsilon}$. If, for instance, the given graph has maximum degree at most constant, then MIS can be approximated in polynomial time to within a constant approximation ratios. For more details about approximation algorithms, the reader is referred to the book [69].

2.3 Ontologies and ontology alignment

For agents within an open multi-agent system to successfully communicate in any form of transaction or collaboration, they must first understand and agree on the terminology they use [70]; i.e. any terms or symbols used within the communication should have some agreed meaning, or *semantics*. The notion of describing the meaning of some concept has long been an important subject in the field of Artificial Intelligence, and in particular, the interest in *ontology* as a tool for defining notions within a domain or word theory has seen a resurgence since the emergence of the Semantic Web [7]. Traditionally, such conceptualisations were shared, and assumed an agreed upon representation [65]. However, within open environments ontologies used by different agents for some domain will typically have been developed independently, and thus the agents will need to reach an agreement over the semantics of different terms through the creation of an *ontology alignment*.

Ontologies support the sharing of knowledge across domains and applications by providing a common, ideally machine processable vocabulary. Many different views have emerged on the 'ontological' question of what is an ontology [28, 29, 65], with the most widely cited definition of the meaning of ontologies (in the area of Artificial Intelligence) being that given by Thomas Gruber in 1993 [28]. This was later refined to state that *"an ontology is a formal, explicit specification of a shared conceptualisation"* [65], where, *formal* refers to the requirement for machine-readability and *explicit* means that the meaning of ontological terms is precisely defined. Thus an ontology provides a model, or *"specification of a conceptualisation"* about a domain of interest such as food, or medicine.

Due to the subjective nature of ontological design, a domain may be modelled in many ways, resulting in different conceptualisations of the same domain using different names and formalisms. As with their subjective nature, ontologies can represent varying levels of granularity, based on the requirement of the tasks for which they were engineered. For example, ontologies for the medical domain may be targeted at a high and general, or holistic level, as opposed to those that define concepts at the cellular level.

Ontologies are comprised of five main components; *concepts*, *relations*, *individuals*, *functions*, and *axioms*:

- *Concepts*, also known as *classes*, represent objects in a given world. Classes can be subdivided into subclasses which represent an entity that is more specific than

that of its superclass. For example, in the food domain[2], classes include `Food`, `IceCream` and `Pizza`, where the latter two are both subclasses of the first one. As the concept `Pizza` is subsumed by `Food` (i.e. all instances of `Pizza` are also instances of `Food`, but not vice versa), the definition of the `Pizza` concept would also inherit all of the constraints and properties from its parent class `Food`.

- *Relations*, also known as *properties*, are the links between the concepts. Relations can have specific characteristics that define their semantics; e.g. symmetry (e.g. given the property `hasBorder` we could state that `Country hasBorder Country`), or cardinality (e.g. the property `hasChild` may have a minimum cardinality of 1 in the statement `Parent hasChild Child`). The combination of concepts and relations can be represented as a directed graph.

- *Individuals* are the instances or the manifestation of objects of a given world. Whilst it is not a requirement that an ontology has to have instances (the ontology may simply be descriptive), the notion of individuals allows for a division between classes and instances of classes. In the pizza domain, two people may order two pizzas, a *Margherita* pizza and a *Hawaiian* pizza. Whilst both would be instances of `Pizza`, only the *Margherita* pizza would be an instance of a `VegetarianPizza`.

- *Functions* are a particular type of relation, defined on a set of concepts such that they are sub-relations from a given parent quality. These can relate an individual concept to a single value. For example, `Spice` would be a functional relation of a `Topping`, and would draw values from the finite set {`hot`, `medium`, `mild`}.

- *Axioms* are logical formulae that can be used to infer and define specific class restrictions that are always assumed to be true. They are commonly used to verify the correctness of the knowledge represented, through checking, and inference. To facilitate reasoning, the axioms are typically represented as declarative definitions (typically stated using some logical algebra[3]). An example of an axiom is in the definition of the concept `VegetarianPizza`, which is defined as a subclass of `Pizza` that does not have toppings of type `FishTopping` or `MeatTopping`.

[2] A tutorial for a simple pizza ontology is available for the Ontology Editor, *Protege*, and can be found at `https://protegewiki.stanford.edu/wiki/Protege4Pizzas10Minutes`

[3] Often, the use of declarative definitions are not sufficient to constrain completely the meaning of concepts or capture the 'procedural' or decision making aspects of the application business logic. This knowledge is often represented as additional rules that accompany the ontology, that can facilitate the advanced reasoning capabilities necessary for such business logic.

Because we will not present a formal definition of an ontology later in this thesis, let us present it here:

Definition 1. *An* ontology*, $\mathcal{O}$, can be expressed as a tuple $\mathcal{O} = \langle C, \leq_C, R, \leq_R, I, Ax \rangle$, where: C is the set of all concepts; $\leq_C$ is a partial order on C, representing a concept hierarchy; R is a set of properties which is disjoint with C; and $\leq_R$ is a partial order on R, representing a relation hierarchy. I is the set of* individuals*; i.e. instances of concepts in C and relations in R, such that $I = \{x : \exists y \in C \cup R,$ instanceof$(x, y)\}$. Instances of concepts may be interconnected with other elements in I by instances of relations in R. Finally, Ax is the set of axioms used for inferring knowledge or constraining expressions in the language.*

A variety of knowledge representations have been proposed in the past, however there has been a recent convergence on the use of a single syntactic representation for ontological knowledge. By basing a representation on an XML syntax, Web-based machinery can be exploited for publishing, indexing, acquiring and parsing both terminological knowledge (e.g concepts and properties) and assertional knowledge (e.g. instances). The Web Ontology Language, OWL[4] provides a standard, ubiquitous representation for describing ontologies, thus eliminating the need for addressing the problem of representational heterogeneity. Whilst it may be reasonable to assume a standard for ontology languages, it is still impractical to promote the use of a single global ontology within a system of open agents, as agents will often differ in the ontology to which they commit.

Whilst it may be possible to define a single, shared domain ontology for use within well defined communities, it would be impossible to enforce the use of a centralised ontology within a dynamic, open environment, where at runtime, the agents (representing different stake-holders) come together serendipitously, with no prior knowledge of each other. As a consequence, agents may use *different ontologies* to model the same domain, and although these different ontologies may be similar, they may differ in granularity or detail, use different representations, or model the concepts, properties and axioms in different ways.

Ontology alignment. An ontology representing a given domain may be mapped to another ontology representing the same domain through the use of *ontology alignments*. This enables semantic interoperability between the knowledge bases of the respective agents, and thus is an essential component for agent communication. Known as the *Ontology Alignment Problem*, this is crucial in supporting semantic integration. In order

[4]http://www.w3.org/2004/OWL/ - Web Ontology Language.

for two systems to accurately and successfully communicate over their vocabularies, the heterogeneity in ontologies needs to be resolved.

The ontology alignment community has proposed diverse approaches that *align* ontologies in order to find these sets of correspondences [21]. A recent review of ontology matching research [64] highlighted that whilst considerable progress had been made in recent years, the performance of different ontology alignment mechanisms across different tasks is still an issue, and can vary greatly. Thus it can be problematic to determine which approach would produce the best alignment between two ontologies for a specific task [60].

An alignment is a set of mappings (*correspondences*) between the corresponding entities within a pair of ontologies, and can be defined as follows: An *alignment*, $\mathcal{A}_{\mathcal{O},\mathcal{O}'}$ is a set of mappings or correspondences between two ontologies, $\mathcal{O}$ and $\mathcal{O}'$ representing a shared domain. A single correspondence, m, defines a relationship between corresponding entities, e and e' within the two ontologies, and can be defined as follows:

Definition 2 ([21])**.** *A* correspondence*, $m \in \mathcal{A}_{\mathcal{O},\mathcal{O}'}$, is expressed as a tuple: $m = \langle e, e', n, r\rangle$ where: $e \in \mathcal{O}$ and $e' \in \mathcal{O}'$ are the entities (concepts, relations or individuals) between which a relation is asserted by the correspondence; $n \in \mathbb{R}^+$ is a degree of confidence in that correspondence; and $r \in \{\equiv, \sqsubseteq\}$ is the relation*[5] *holding between e and e'.*

The formal definition of correspondences can be specialised when the entity type is constrained: mappings between concepts = $\langle c, c', \_, \_\rangle$, mappings between properties = $\langle p, p', \_, \_\rangle$, and mappings between individuals = $\langle i, i', \_, \_\rangle$, where the entity types correspond to concepts, relations and individuals respectively. A correspondence over which no agreement has yet been reached by the agents is called a *candidate mapping*.

Ontology alignment has traditionally been viewed as a centralised process, whereby a central oracle is invoked in order to identify mappings between corresponding entities belonging to two ontologies provided as input. Such approaches try to maximise the number of correspondences created (*coverage*) given some objective function, often disregarding the reason why the alignment was being generated (i.e. to facilitate some task or queries), or other knowledge possessed by the ontology owner (e.g. an agent). Thus, generic alignment mechanisms do not offer any guarantee that even if an alignment can be found, this will actually support the representation of a joint task.

[5]This set of relations is not meant to constitute an exhaustive set of relationships, as the type of relationships can depend on the type of correspondences generated, and the underlying logical algebra assumed by the ontological model.

Chapter 3

Mechanism design for ontology alignment

3.1 Introduction

To address the problem of agents communicating in a meaningful way within an open, distributed environment, the agents have to align their respective, individual, and typically private ontologies. Although many approaches have been proposed to align ontologies within a centralised setting, few studies have addressed how candidate correspondences should be selected within this setting [38, 46, 60, 68], and even fewer, if any, have considered the problem from a decentralised, one-shot game perspective [43].

Various static [46, 68] and dynamic [60, 38] approaches have explored how agents can propose, and exchange candidate correspondences with the goal of aligning their respective ontologies. In many cases, agents acquire knowledge of different candidate correspondences from a variety of sources, or through negotiation with other agents. These candidate correspondences may have an associated *weight*, which may reflect the utility, significance, or simply the confidence that an agent has in the correspondence. This weight may also be *contextual* as different weights may be associated to a correspondence based on the specific task an agent is trying to achieve. Furthermore, in adversarial scenarios, the agents may not wish to disclose their private weights, and may lie when stating their preferences. An example of such a scenario is the health-care domain where ontologies are currently being used to describe electronic patient records (EPR), but where the vendor *"... may be reluctant to distribute (parts of) the contents of [the ontology], as doing so might allow competitors to plagiarize [it] ... "* [27]

As the composition of different subsets of correspondences can result in different

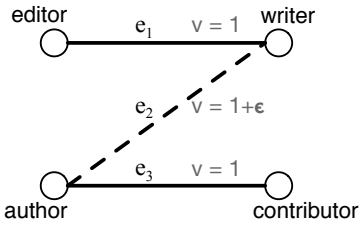


Figure 3.1: Centralised example with two solutions: $\{e_1, e_3\}$ and $\{e_2\}$.

alignments, the challenge in negotiating a mutually acceptable alignment is that of selecting and proposing correspondences that result in a preferred alignment that satisfies the aims of both agents. Furthermore, some correspondences may map a single entity in one ontology to different entities in other ontologies (which can compromise the integrity of the resulting logical model), and therefore the outcome should also be injective, i.e., a matching.

3.2 Background

To date, the ontology alignment community has proposed many diverse approaches that *align* ontologies in order to find sets of correspondences between the ontology pairs.[1] However, most approaches rely on the ontologies being fully shared with some alignment algorithm [21, 64] which attempts to find correspondences between entities. Alignment approaches usually initiate the process of identifying correspondences (mappings) by computing a *similarity matrix* (lexical, structural or a combination of these) between all the entities in the two ontologies that are being aligned [21, 51]. This produces a number of different mappings involving the same entities from which an *injective* (one-to-one) alignment needs to be extracted (i.e. correspondences for which to each entity from the source ontology corresponds only one entity in the target ontology).

Typically, most alignment approaches model the alignment as a bipartite graph, and thus select an injective alignment by finding a *matching* or independent edge set in the graph, such that the set of edges (i.e. correspondences) have no common vertices (i.e no entity in one ontology is mapped to more than one entity in the other ontology, and vice versa). This assumes that each edge (or correspondence) is weighted such that the weight represents the *quality* or *desirability* of the correspondence. The two

[1] For a comprehensive overview of the different approaches, we refer the reader to the Proceedings of the Ontology Matching Workshops that have taken place annually since 2004, as part of the Ontology Alignment Evaluation Initiative - `http://oaei.ontologymatching.org`

most common methods used to compute a *matching* are: 1) to find a global optimal solution (which is equivalent to the *Assignment Problem*) using algorithms such as the Hungarian method [45]; or to find a sub-optimal, but *stable* solution using algorithms such as Gale and Shapley's Stable Marriage algorithm [24]. Solutions to the assignment problem identify correspondences that maximise the sum of the weights (i.e. they assume some *objective function* that maximises *social welfare*), such that the global similarity of the alignment is considered, as opposed to the similarity of each pair of entities. This is illustrated in Figure 3.1, where two correspondences are selected by maximising the weights; in this case the weights associated to the two correspondences $\{e_1, e_3\}$ are $1 + 1 = 2$.

The used approach should also be time-efficient, as ontologies can vary greatly in size, with several in the Bio-Medical domain possessing tens of thousands of entities [39]. Thus, given that the Hungarian method is computationally expensive ($O(n^3)$ for its most efficient implementation), some sub-optimal approximate algorithm such as a greedy matching algorithm [51] or a variant from the family of Stable Marriage algorithms [30] are used that select a sub-optimal set of correspondences in those cases when a *stable* solution is sufficient. This can result in a different alignment that emphasises the weights of individual correspondences; for example a greedy algorithm would generate an alignment with a single correspondence, e_2, as its weight is greater than either e_1 or e_3, resulting in a sub-optimal total weight of $1 + \epsilon$.

A similar problem arises in decentralised settings, where agents negotiate over a set of (partially observable) correspondences to agree upon a mutually acceptable alignment [2, 4, 13, 20, 38, 46, 60, 68], often based on the aims or goals of the agents that may own or utilise them. As no single alignment approach can provide a panacea for all ontology pairs, agents are left with the problem of either: 1) *selecting* a suitable alignment approach from the plethora that exist; or 2) *assembling* alignments from a subset of relevant, candidate correspondences; for example using an ensemble approach. This latter case occurs if agents have access to correspondences from shared repositories [46] or garnered from previous transactions with other agents. Furthermore, alignments with different constituent correspondences may be semantically equivalent with respect to one of the agent's ontologies and aims (due to the logical theory underlying each agent's ontology) but may have a different meaning to another.[2] As the agent may have preferences over the choice of correspondences used (e.g. due to privacy concerns [27, 52]), agents can have a preference order over the resulting alignments within the

[2] A classic example of terminological difference exists with the term *"football"*, which has a different meaning depending on whether the reader is from the US or the UK.

same equivalence class. Hence, for *self-interested* agents, this task becomes one of selecting a mutually acceptable subset of preferred ontological correspondences.

The resulting alignment will typically be dependent on the valuation that each agent associates to each correspondence. Whilst this is uncontroversial in centralised systems, approaches that are decentralised (i.e. where agents may differ in the value they ascribe to a correspondence) are subject to strategic manipulation; i.e. agents may lie about the true valuation of a correspondence to ensure that the final alignment includes their preferred correspondences. The value that each agent assigns to each correspondence (i.e. its *private valuation*) relates to how useful this edge is in resolving a query or achieving a task, and in turn, the potential benefit the agent can obtain from performing a task. Note that this is not the same as the confidence the agent has in the edge (based, for example from some form of linguistic similarity metric over the concept labels). For example, an agent may know of two correspondences in the publishing domain $\{\texttt{writer}, \texttt{editor}\}$ and $\{\texttt{writer}, \texttt{author}\}$. Both are viable correspondences, depending on the task (e.g. for a conference proceedings and monograph respectively), but an agent may assign different valuations to each correspondence based on some preference; for example the agent can increase its benefit by resolving queries or performing tasks (by providing a service to its peers) pertaining to monographs. Conversely, it may have a low valuation for other correspondences for which it has little preference (e.g. $\{\texttt{writer}, \texttt{publisher}\}$). However, within a service landscape where several agents (providing services) may compete to perform a task for a requesting agent, they may not wish to disclose the true valuation. This can potentially lead to agents strategically manipulating the combined value of sets of correspondences, in order to maximise their individual benefits; potentially resulting in semantically compromised correspondences being selected, which may then prevent the query or task from successfully completing. Thus, in an ideal setting, the agents should be incentivised to adopt strategies that result in alignments that benefit both agents; i.e. find solutions that lie within a *Nash Equilibrium* [55]. Our model which relates to two agents who seek to find a matching between their sets of ontologies, assumes an existence of a third party, a *mechanism designer*, who provides an online matching service such as an *Agent Matchmaker* [15, 67], that facilitates the discovery of services (often based on semantically annotated service descriptions[3] [57, 66]). The mechanism designer uses a specific matching mechanism which might require payments from agents. Those payments can be interpreted as the mechanism's charge for the provided matching service.

[3]Note that a discussion of the methods for discovering semantically annotated services [14, 22] is out of scope of this thesis.

3.3 Our contributions

Thus, given two agents, an instance of the ontology alignment problem can be modelled as an edge-weighted bipartite graph $G = (U \cup V, E)$, where the vertices of U and V correspond to named concepts and the edges $e \in E$ represent the candidate correspondences. Each agent assigns an independent value to each edge, which represents a private weight associated to that correspondence. The outcome should be an injective alignment (a *matching*) that maximises social welfare (i.e. the sum of the edge weights in the resulting alignment is maximised).

Remark: Although, optimisation-wise, this problem is just the classic maximum edge-weighted bipartite matching problem, the differences will emerge when we define the very specific mechanism design version suited to our application to the ontology alignment problem. In particular, the main novelty will be a special kind of agents' utilities that will define a special kind of edge weights. This, in turn, will translate in an interesting and new set of results on the approximability of the dominant strategy mechanisms and price of anarchy and stability of the Nash equilibria implementation of the greedy mechanism.

We explore this problem from a mechanism design perspective, and analyse implementations in *Dominant Strategies* and in *Nash Equilibria*. For the implementations in Dominant Strategies, two alternate settings are considered: *with payment*, and *without payment*; where the problem is characterised as a social welfare maximising matching setting, with an additive valuation function. We show that for a deterministic mechanism with payment, the only truthful mechanism is maximal-in-range[4] and any truthful mechanism which is not optimal can do no better than an approximation ratio of 2.

Note here that we can obtain a truthful mechanism by just solving the edge-weighted bipartite matching problem exactly and using the VCG payments to the agents. However, this still requires time for solving optimally the edge-weighted bipartite matching problem, and we ask the question whether it is possible to have a faster truthful mechanism? We answer this question in negative, because our results imply that any truthful mechanism that is not optimal has to be maximal-in-range and at least 2-approximate. This motivates us to study implementation in Nash equilibria by fast greedy algorithms (see below).

[4] A mechanism is maximal-in-range if it computes an optimal, i.e., social welfare maximising, solution on a fixed subset of feasible solutions. See also Definition 4.

For settings without payment, we assume a priori that the individual true valuations of each correspondence are public, and each agent bids a Boolean vector (indicating its selection/rejection of a correspondence). Our polynomial time algorithm determines if a deterministic truthful mechanism exists with a bounded approximation ratio; if so, then the optimal solution is found. However, if such a mechanism does not exist for the bid, then we show there is no truthful mechanism with bounded approximation ratio. We also show that there are *no randomised mechanisms* that are: i) universally truthful with an approximation ratio better than 2; and ii) truthful in expectation with an approximation ratio better than 2. Note that ii) implies i), however, we also mention and explicitly include i), because its proof is different from that of ii).

Given our results on truthful centralised mechanisms, either the problem should be solved optimally (though costly) or strong lower bounds should be found for the approximation ratios of truthful mechanisms. Thus, we have explored an implementation in Nash equilibria to efficiently approximate mechanisms for matching, using the greedy allocation mechanism. We provide a complete picture of the complexity of this mechanism by showing that when coupled with a first-price payment scheme, it implements Nash equilibria which are very close (within a factor of 4) to the optimal matching. The *Price of Anarchy* of this mechanism is characterised completely and shown to be precisely 4 (this bound also holds for Mixed Nash equilibria), and when a pure Nash Equilibrium exists, we show that the *Price of Stability* is at least 2. Thus we can just let the agents play in a decentralised way and reach a Nash equilibrium, which then will give a solution close to optimum within a factor of 4.

3.4 Preliminaries

We consider a setting in which there are two agents $i \in \{L, R\}$ (the *left* agent and *right* agent), where each agent possesses a private ontology $\mathcal{O}_i$, which consists of *named concepts* $\mathsf{N^C}$ and *named relations* $\mathsf{N^R}$; i.e. $\mathcal{O}_i = \mathsf{N}_i^{\mathsf{C}} \cup \mathsf{N}_i^{\mathsf{R}}$. An instance of the alignment is modelled as an edge-weighted bipartite graph $G = (U \cup V, E)$, where the vertices of U and V correspond to named concepts (i.e., *entities*) in the agents' individual ontologies $U = \mathsf{N}_L^{\mathsf{C}}$ and $V = \mathsf{N}_R^{\mathsf{C}}$ respectively, and the edges $e \in E$ correspond to the candidate correspondences. A matching M is a subset of E such that $e \cap e' = \varnothing$ for all $e, e' \in M$ with $e \neq e'$; i.e. no two edges have a common vertex. Each agent $i \in \{L, R\}$ has a non-negative valuation function for different matchings M, denoted $v_i(M)$, where $v_i : M(G) \to \mathbb{R}^+$, which is additive; i.e. $v(S) + v(T) = v(S \cup T)$ such that $S \cap T = \varnothing$ for all $S, T \in M(G)$, and $M(G)$ is the set of all matchings in a graph

G. The agents also have the valuation function $v_i : E \to \mathbb{R}^+$ to represent the value $v_i(e)$ the agent i can get from the edge e. The combined value for an edge e is therefore given as $v(e) = v_L(e) + v_R(e)$ (as described below). Define that $v_i(M) = \sum_{e \in M} v_i(e)$ for every agent $i \in \{L, R\}$. The goal is to establish an alignment which is equivalent to a matching M that maximises $\sum_{e \in M} v(e)$. The valuation function v_i can be regarded as the agent's true valuation, or *type* that it attributes to each matching. Furthermore, we use v to represent the combined type profile for both agents, such that $v = (v_L, v_R)$, where v_i is the type profile for agent i, and similarly, b denotes the combined bid profile for both agents, such that $b = (b_L, b_R)$, where b_i is the bid profile for agent i. We will also introduce the following useful notation: $b_i^e = b_i(e)$ and $v_i^e = v_i(e)$ for any $i \in \{L, R\}$ and $e \in E$.

To determine the outcome given the bids of the two agents, we consider mechanisms with and without payments (see §3.5). We define a direct revelation mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$, which is composed of an allocation rule $\mathcal{A}$ to determine the outcome of the mechanism, and a payment scheme $\mathcal{P}$ which assigns a vector of payments to each declared valuation profile. For the mechanism with payment (§3.5.1), the mechanism proceeds by eliciting a bid profile b_i from each agent i, and then applies the allocation and payment rules to the combined bid profiles to obtain an outcome and payment for each agent. As an agent may not want to reveal its type, we assume that b does not need to be equal to v.

For the mechanism without payment, we consider a restricted model of the declaration, whereby each agent's valuation on an edge e, $v_i(e)$ is public, and thus $\forall e \in E, b_i(e) = v_i(e)$. What is private is a set of desirable edges $E_i \subset E$ that the agent wants in the outcome. Each agent i therefore declares a Boolean value for each edge, denoted $\delta_i(e) \in \{0, 1\}$, such that $\delta_i(e) = 1$ *iff* $e \in E_i$. The payment scheme for this mechanism is simply $\mathcal{P} = 0$, and the allocation rule $\mathcal{A}$ is described in §3.5.2.

The utility $u_i(\cdot)$ for agent i given a bid profile $b = (b_L, b_R)$ and mechanism $\mathcal{M}$ is based on the allocation rule $\mathcal{A}$ and the payment scheme $\mathcal{P}$ over the outcome of $\mathcal{A}(v)$ (i.e. a matching or allocated set M), and can be written as $u_i(\mathcal{A}(b)) = v_i(\mathcal{A}(b)) - \mathcal{P}_i(b)$. For the implementation in Nash Equilibria, we assume a *first-price* payment rule, such that an agent is charged its declared bid $b_i(M)$ for any allocated set M. Our mechanism $\mathcal{M}$ maximises the social welfare given both agents' bids (generating either optimal or approximately optimal solutions), which is defined as $SW(\mathcal{A}(b), v) = \sum_{e \in \mathcal{A}(b)} v(e)$.

A (deterministic) mechanism $\mathcal{M}$ is called *truthful in dominant strategies* or *incentive*

compatible if, for any agent $i \in \{L, R\}$, we have

$$u_i(\mathcal{A}(v_i, b_{-i})) \geq u_i(\mathcal{A}(b_i, b_{-i}))$$

for any bid profile b_i of agent i and any bid profiles b_{-i} of the other agent.

Remark: Given any k-dimensional vector $w = (w_1, \ldots, w_k)$, and given any element w' and any index $i \in \{1, \ldots, k\}$, we denote by w_{-i} the $(k-1)$-dimensional vector that has all coordinates of vector w except the i-coordinate. Then we denote by (w', w_{-i}) the vector $(w_1, \ldots, w_{i-1}, w', w_{i+1}, \ldots, w_k)$. That is, (w', w_{-i}) is vector w that has w' in the i-th coordinate and other coordinates are the same as in vector w.

Thus, given a bid profile $b = (b_L, b_R)$, the notation (v_i, b_{-i}) above for any agent $i \in \{L, R\}$ denotes the new bid profile in which the bid b_i of agent i in b is replaced by v_i and the bid(s) of the agent(s) other than i, b_{-i}, are the same as in profile b. That is, if $i = L$, then we have $(v_i, b_{-i}) = (v_L, b_R)$, and if $i = R$, then we have $(v_i, b_{-i}) = (b_L, v_R)$. Also, to be very formal, we should write $\mathcal{A}((b_i, b_{-i}))$ instead of $\mathcal{A}(b_i, b_{-i})$ above.

If the context is clear, we will omit $\mathcal{A}$ in the utilities u_i, which means: $u_i(\cdot, \cdot) := u_i(\mathcal{A}(\cdot, \cdot))$.

Nash Equilibria. Different types of Nash equilibria may exist, depending on the strategy adopted by the agents. The bid profile b forms a *Pure Nash equilibrium* if, for both agents, there exists no other bid profile b'_i achieving a higher utility, i.e.,

$$\forall b'_i, u_i(b_i, b_{-i}) \geq u_i(b'_i, b_{-i})$$

Thus, no agent can obtain a higher utility by deviating from b.

Remark: Given a bid profile $b = (b_L, b_R)$, the notation (b_i, b_{-i}) and (b'_i, b_{-i}) has been explained in the remark above. And again, to be very formal, we should write $u_i((b_i, b_{-i}))$ instead of $u_i(b_i, b_{-i})$ above.

The bid profile b may also form a *Bayesian Nash equilibrium*. Let $i \in \{L, R\}$, V_i denote the finite set of valuations of agent i. As the set of possible valuation profiles of the agents is $V = V_L \times V_R$, there is a known probability distribution P over the valuation V. We assume that $P = P_L \times P_R$ is the Cartesian product of independent probability distribution P_i. Any valuation profile $v = (v_1, \cdots, v_n)$ occurs with probability $P(v) =$

$\prod_{i=1}^{n} P_i(v_i)$, where $P_i(v_i)$ is the probability that agent i has valuation function v_i.

The strategy function B_i for agent i assigns a bid-vector $b_i = B_i(v_i)$ to every valuation function $v_i \in V_i$. The function $B = (B_1, \cdots, B_n)$ is a *Bayesian Nash Equilibrium* if, for both agents i, and for every valuation function v_i, the bid $B_i(v_i)$ maximizes i's expected utility, given that its valuation function is v_i, and that the bids of other agent j is $B_j(v_j)$, where v_j is drawn from P_j. Thus, for all v_i and b'_i:

$$\mathbb{E}_{v_{-i} \sim P_{-i}}[u_i(b_i, b_{-i}^{v_{-i}})] \geq \mathbb{E}_{v_{-i} \sim P_{-i}}[u_i(b'_i, b_{-i}^{v_{-i}})],$$

where $b_{-i}^{v_{-i}} = (B_1(v_1), \cdots, B_{i-1}(v_{i-1}), B_{i+1}(v_{i+1}), \cdots, B_n(v_n))$, and $\mathbb{E}_{v_{-i} \sim P_{-i}}[X]$ denotes the expectation of the random variable X assuming that vector v_{-i} is sampled at random from the distribution P_{-i}.

We also permit a randomised strategy function which can result in a *Mixed Nash equilibrium*. Given the probability distribution $\omega_1, \cdots, \omega_n$ over the declarations, and any function f over the space of declaration profiles, we can state $\mathbb{E}_{b \sim \omega}[f(b)]$ for the expected value of f over declarations chosen according to the product distribution $\omega = \omega_1 \times \cdots \times \omega_n$. Thus, ω is a *Mixed Nash Equilibrium* if, for any agent and distribution ω'_i, we have: $\mathbb{E}_{b \sim \omega}[u_i(b)] \geq \mathbb{E}_{b \sim (\omega'_i, \omega_{-i})}[u_i(b)]$.

By combining the Bayesian and Mixed strategies, we define a *Mixed Strategy Bayesian Nash Equilibrium*:

$$\mathbb{E}_{v_{-i} \sim P_{-i}, b \sim \omega}[u_i(b_i, b_{-i}^{v_{-i}})] \geq \mathbb{E}_{v_{-i} \sim P_{-i}, b \sim (\omega'_i, \omega_{-i})}[u_i(b_i, b_{-i}^{v_{-i}})]$$

As our aim is to maximise the social welfare, we denote $SW_{opt}(v)$ for $\max_{M \in M(G)} SW(M, v)$, and state that the allocation algorithm $\mathcal{A}$ is a *c-approximation* algorithm if we have $SW(\mathcal{A}(v), v) \geq \frac{1}{c} SW_{opt}(v)$.

The *Price of Anarchy* of mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ in mixed (and pure, respectively) strategies can thus be defined as:

$$PoA_{mixed} = \sup_{v,\omega} \frac{SW_{opt}(v)}{\mathbb{E}_{b \sim \omega}[SW(\mathcal{A}(\omega), v)]}$$

$$PoA_{pure} = \sup_{v,b} \frac{SW_{opt}(v)}{SW(\mathcal{A}(b), v)}$$

where the supremum is over all valuations v, and all mixed Nash equilibria ω (likewise, all pure Nash equilibria b) for v. Here $\mathcal{A}(\omega)$ denotes a random matching with respect to ω.

Given a probability distribution P over valuations (types) of the agents, and an allocation rule $\mathcal{A}$, we denote: $SW(\mathcal{A}(b), P) = \mathbb{E}_{v\sim P}[v_L(\mathcal{A}(b)) + v_R(\mathcal{A}(b))]$. Then, the Bayesian mixed (pure, respectively) price of anarchy of mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is:

$$PoA_{Bayesian} = \sup_{P,\omega} \frac{SW_{opt}(P)}{SW(\mathcal{A}(\omega), P)}$$

where the supremum is over all type distributions P and all mixed Nash equilibria ω (respectively, all pure Nash equilibria ω, assuming that ω assigns probability 0 or 1 to pure strategies) for v.

The price of stability for pure strategy games defined by mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is the ratio between the best objective function value of one of its equilibria and that of optimum:

$$PoS_{pure} = \inf_{v,b} \frac{SW_{opt}(v)}{SW(\mathcal{A}(b), v)}$$

Where the infimum is over all type valuation v, and all pure Nash equilibria b.

3.5 An Implementation in Dominant Strategy

Two truthful mechanism design cases have been investigated: with payment (§3.5.1) and without payment (§3.5.2).

3.5.1 Mechanism design with payment

In this setting, both agents have to pay money to establish a matching. The first observation is that if we are willing to solve the problem optimally (which is possible in polynomial time by simply finding an optimal weighted bipartite matching), then we can use the classic VCG mechanism with Clarke payment (e.g., [55]). The question is: can we have a faster, non-optimal, approximate and truthful mechanism for our problem? We show below that the answer is essentially no. We will need the following well known theorem from the classic mechanism design theory:

Definition 3. *[55] An allocation rule of mechanism $\mathcal{A}$ satisfies weak monotonicity if for all i and all v_{-i}, $\mathcal{A}(v_i, v_{-i}) = a \neq b = \mathcal{A}(v'_i, v_{-i})$ implies that $v_i(a) - v_i(b) \geq v'_i(a) - v'_i(b)$.*

Theorem 1. *[55] If a mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is incentive compatible, then $\mathcal{A}$ satisfies weak monotonicity.*

Theorem 2. *For the alignment problem with payment, any mechanism which does not return an optimal solution, is either non-truthful, or if truthful, the non-optimal solution has an approximation ratio of at least 2.*

Before the proof of Theorem 2, the clarification for it is: if we apply the VCG mechanism, then the mechanism will be truthful and the solution is optimum, thus, the motivation behind this theorem is to seek any mechanism which is not VCG, to examine whether such mechanism can be truthful and what is the quality of its solution. This theorem shows that if any mechanism is not VCG, then it is either not truthful, or it is truthful but cannot achieve a solution whose approximation factor is smaller than 2.

Proof. Let $\mathcal{M}(\mathcal{A}, \mathcal{P})$ be a mechanism, and recall that $\mathcal{A}(v)$ denotes the outcome generated by $\mathcal{M}$, when the input is v (which may not be the true valuation).

We construct an instance of our problem of any size to prove the lemma. For any two positive integers ℓ, k, let the bipartite graph $G = (U \cup V, E)$ have ℓ nodes on the left side of bipartite graph ($|U| = \ell$) and k nodes on the right side ($|V| = k$). Then we have two special edges $e_1, e_2 \in E$ that are disjoint, $e_1 \cap e_2 = \varnothing$, and their true valuations are as in Figure 3.2, $v_L(e_1) = v_L(e_2) = 0$ and $v_R^{e_1} = v_R^{e_2} = \omega$. The valuations of all other edges in G for both agents are zero. In the following discussion, we will only consider the right agent, and thus we omit the agent index when referring to valuations.

Consider any mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ that does not adopt an optimal solution: the outcome contains both edges e_1 and e_2. Thus, the non-optimal solution will contain at most one of these edges, and note if both e_1 and e_2 are not adopted into the solution, the approximation ratio is unbounded. We therefore assume that the mechanism specifically accepts one of these two edges; w.l.o.g., assume that $\mathcal{M}$ will accept $e_1 \in \mathcal{A}(v)$, when the right agent declares its true valuation v.

If the right agent deviates from its valuation v to some other valuation v', the mechanism has two options:

Case-1. The mechanism changes the current outcome $\mathcal{A}(v) \supseteq \{e_1\}$ to $\mathcal{A}(v') \supseteq \{e_1, e_2\}$, adopting both edges. Making the alternative valuation $v'(e_1) = v'(e_2) = 0$, implies that $v'(\mathcal{A}(v')) \leq v'(\mathcal{A}(v))$. From the assumption, we also know that $v(\mathcal{A}(v)) < v(\mathcal{A}(v'))$. Adding the left and right hand sides of these two inequalities, we obtain:

$$v'(\mathcal{A}(v')) + v(\mathcal{A}(v)) < v'(\mathcal{A}(v)) + v(\mathcal{A}(v'))$$

$$v(\mathcal{A}(v)) - v(\mathcal{A}(v')) < v'(\mathcal{A}(v)) - v'(\mathcal{A}(v'))$$

As this violates the weak monotonicity condition in Theorem 1, it follows that $\mathcal{M}$ is not a truthful mechanism.

Case-2. The mechanism does not change the outcome, i.e., $\mathcal{A}(v') = \mathcal{A}(v)$. Then when the agent deviates its valuation to v'. The approximation ratio is at least $\frac{v(e_1)+v(e_2)}{v(e_1)}$. Since we consider the worst case, the ratio is therefore at least 2.

Observe that if the outcome changes from $e_1 \in \mathcal{A}(v)$ to $e_2 \in \mathcal{A}(v')$ and $e_1 \notin \mathcal{A}(v')$, then this case is symmetric to Case 2, and thus will also lead to a ratio of at least 2.

Furthermore, if the right agent has only one non-zero value edge, and the valuation on the remaining edges is 0, then the approximation ratio is also unbounded, and all such cases also lead to the lower bound on the approximation ratio. □

Definition 4. *[17] A mechanism is called maximal in range (MIR) if there exists a fixed subset R of all allocations (the range of the mechanism), such that for every possible input v, the mechanism outputs the allocation that maximizes the social welfare in R with respect to v.*

Theorem 3. *For the alignment problem with payment, any deterministic mechanism that does not return an optimal solution, is either non-truthful, or is a maximal-in-range mechanism.*

The idea behind Theorem 3 is the following. Given the agents' bids, any mechanism, if it does not return the optimal matching as the solution, the agents will declare a bid which is lower than their true valuation. The agents are incentivised to do that, because the mechanism will choose a sub-optimal solution.

Proof. Consider an instance, let the bipartite graph $G = (U \cup V, E)$ have ℓ nodes on the left side of bipartite graph ($|U| = \ell$), and only single node on the right side ($|V| = 1$). For each vertex of the left agent, there is an edge that connects it to the right agent's vertex; thus, any matching for this instance includes only a single edge. Let us name these ℓ edges as $e_1, \ldots, e_\ell$. Fix any deterministic mechanism $\mathcal{A}$ that does not return an optimal solution; and we assume that e_1 is the optimal solution; then the mechanism $\mathcal{A}$ will select one edge in $\{e_2, \cdots e_\ell\}$. When agents deviate from their valuation, according to the solution returned by the mechanism, we have three cases:

Case-1. Mechanism $\mathcal{A}$ does not accept an alternative solution, in particular $\{e_2\}$, whatever the declaration is. Thus, if $\mathcal{A}$ is truthful then it is equivalent to a maximal-in-range mechanism, whose range is $R = \{e_2\}$.

Case-2. Mechanism $\mathcal{A}$ returns an alternative solution, $\{e_1\}$ (i.e. the optimal solution in global) for some declaration v': $\mathcal{A}(v') = \{e_1\}$. However, by Theorem 1, the

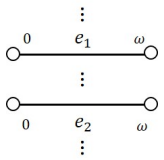


Figure 3.2: Disjoint Edges

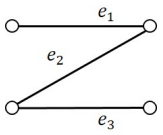


Figure 3.3: Shared vertices

mechanism is not truthful. To show this, suppose w.l.o.g. that the mechanism returns e_2 for a declaration v: $\mathcal{A}(v) = \{e_2\}$, and that one agent deviates from its valuation v to v' with $v'(e_1) < v'(e_2)$. Note we have $v(e_1) > v(e_2)$ by the assumption. Adding the left and right hand sides of these two inequalities, we have $v(e_2) - v(e_1) < v'(e_2) - v'(e_1)$. By assumption, we also have $\mathcal{A}(v) = \{e_2\}$ and $\mathcal{A}(v') = \{e_1\}$. This contradicts the monotonicity condition from Theorem 1, which requires that: $v(e_2) - v(e_1) \geq v'(e_2) - v'(e_1)$.

Case-3. Mechanism $\mathcal{A}$ returns an alternative solution, which is one of the edges from $\{e_3, \cdots, e_l\}$. In such a case, by Theorem 1, the same argument for Case 2 also applies for this, the mechanism is again not truthful, since it violates the monotonicity condition. □

Note that putting together Lemma 2 and Theorem 3, we conclude the following theorem:

Theorem 4. *For the alignment problem and mechanism design with payment, the only truthful mechanisms are ones that are maximal-in-range and have approximation ratio at least* 2*.*

To complement these lower bound results we in fact show below that there is a very simple truthful mechanism which indeed has an approximation ratio of 2, also it does not return optimal solution and it is a maximal in range mechanism.

We introduce some notations here. Let O be the optimal solution, and O_L and O_R be the optimal solution for the left agent and right agent, i.e., $O_L = \arg\max_{M \in M(G)} v_L(M)$, $O_R = \arg\max_{M \in M(G)} v_R(M)$.

Algorithm 1 Larger agent algorithm.

Require: Bipartite graph $G = (V, E)$. b_L, b_R of the left and right agent.
Ensure: A matching M
1: set the left agent bids on all edges to be 0
2: find the optimal solution of the right agent O_R.
3: set the right agent bids on all edges to be 0
4: find the optimal solution of the left agent O_L.
5: $\max(O_L, O_R)$ is the solution.

Theorem 5. *The approximation ratio of Algorithm 1 is at most 2.*

Proof. Since it finds the larger matching in $\{O_L, O_R\}$, then it is higher than $\frac{1}{2}O$. To prove it, we assume w.l.o.g. that $v_L^{O_L} \geq v_R^{O_R}$. Since $v_L^{O_L} \geq v_L^O$, $v_R^{O_L} \geq v_R^O$ then $v_L^O + v_R^O \leq v_L^{O_L} + v_R^{O_R} \leq 2\max\{v_L^{O_L}, v_R^{O_R}\} = 2v_L^{O_L}$. □

3.5.2 Mechanism design without payment

In the mechanism design without payment, if agents can misreport their valuations, no non-trivial truthful mechanism exists [18]. A natural setting commonly used in previous research assumes that an agent can only declare or hide which edge it wants to match. We thus adopt this restricted model of the declaration. We assume that agents cannot lie about their valuations, but they may lie about which edge can be used to establish a matching. An instance of the alignment problem on a private bipartite graph is: the valuations of agents on edge e, $v_i(e)$ are public information or verifiable, and agent i's private information is a set of edges $E_i \subseteq E$ given by $\delta_i(e) \in \{0, 1\}$. An edge e may be accepted in the matching, only if for both agents, $\delta_L(e) = \delta_R(e) = 1$. The agent i will receive value $v_i(e)$ from e if it is matched; otherwise for edge e it receives 0. The goal is to maximise the social welfare via a mechanism without money, such that both agents are incentivised to declare their E_i truthfully.

Theorem 6. *Given an instance of the alignment problem, Algorithm 2 will decide whether there is a deterministic truthful mechanism without payment for this instance. If the answer is yes, then the mechanism also finds an optimal solution. If the answer is no, then there is no deterministic truthful mechanism with a bounded approximation ratio on that instance.*

Algorithm 2

Require: Bipartite graph $G = (V, E)$, the public known valuation of both agents.
Ensure: A matching M or 'NO'
1: Find the optimal solution O_L, given the v_L and G.
2: Find the optimal solution O_R, given the v_R and G.
3: if $O_L = O_R$, return $M = O_L = O_R$. Otherwise, NO.

Proof. If $O_L \neq O_R$, then one agent will hide all edges which are not in its optimal matching, but only declare those edges that are in its optimal matching. Consider the example illustrated in Figure 3.3; in this example, the left agent's valuation is: $v_L(e_1) = v_L(e_3) = 7$, $v_L(e_2) = 0$, and the right agent's valuation is: $v_R(e_1) = v_R(e_3) =$

5, $v_R(e_2) = 12$. The optimal solutions are: $O_L = \{e_1, e_3\}$ and $O_R = \{e_2\}$. If the mechanism does not accept any edges, the ratio is unbounded. Thus if it has a bounded approximation ratio then the mechanism must accept some edges.

Now, suppose that $\{e_2\}$ was accepted with $\delta_L(e_2) = 1$. Note that the left agent can hide the edge (by declaring $\delta_L(e_2) = 0$), so that the outcome would be $\{e_1, e_3\}$ (or one of these two edges), which increases the utility of the left agent compared to that when $\delta_L(e_2) = 1$. On the other hand, if the mechanism accepts edges $\{e_1, e_3\}$, then the right agent would declare $\delta_R(e_1) = \delta_R(e_3) = 0$, and the outcome would be $\{e_2\}$. This again increases the utility of the right agent. In both cases, one agent increases its utility by lying. □

Recall that truthful (deterministic) mechanism has been defined in Section 3.4. We define now notions of truthfulness for randomised mechanisms.

Definition 5. *A randomised mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is truthful in expectation if a bidder always maximises its expected profit by declaring truthfully. The expectation is taken over the internal random coins of the mechanism. Formally, $\mathcal{M}$ is truthful in expectation if for each agent i, every v_i, v_i', and v_{-i}, we have that $\mathbb{E}[u_i(\mathcal{A}(v_i, v_{-i}))] \geq \mathbb{E}[u_i(\mathcal{A}(v_i', v_{-i}))]$.*

Definition 6. *A randomised mechanism $\mathcal{M}(\mathcal{A}, \mathcal{P})$ is a universally truthful if it is a probability distribution over deterministic truthful mechanisms.*

Theorem 7. *There are no randomized mechanisms that are universally truthful and have an expected approximation ratio better than 2 for the setting without payment.*

Proof. Let I and I' be two instances of the graph in Figure 3.3. The agents' valuations are $v_L(e_1) = 5, v_L(e_2) = 10 + \epsilon, v_L(e_3) = 5$ and $v_R(e_1) = 5 + \epsilon, v_R(e_2) = 10, v_R(e_3) = 5 + \epsilon$. In instance I, edge e_2 is not available, whereas in instance I', edges e_1 and e_3 are not available. Each instance occurs with probability $\frac{1}{2}$, and the expected maximum social welfare is $\frac{1}{2} \cdot (5 + 5 + 5 + 5 + 2\epsilon) + \frac{1}{2} \cdot (10 + 10 + \epsilon) = 20 + \epsilon$. Let mechanism $\mathcal{A}$, applied to instance I, accept e_1 and e_3. Since the mechanism is truthful, it must accept the same solution when the instance is I'. The expected social welfare of $\mathcal{A}$ is $\frac{1}{2} \cdot (5 + 5 + 5 + 5 + 2\epsilon) + \frac{1}{2} \cdot 0 = 10 + 2\epsilon$. And the approximation ratio is essentially 2 when we let ϵ tend to 0. If mechanism $\mathcal{A}$ does not accept e_1 and e_3 on instance I', its expected social welfare is at most $10 + \epsilon$. By Yao's principle [71], we obtain the theorem. □

Theorem 8. *There are no randomised mechanisms that are truthful in expectation and have an approximation ratio better than 2 for the setting without payment.*

Proof. Let I and I' be two instances. Again we use the graph structure in Figure 3.3. Agent valuations are $v_L(e_1) = v_L(e_3) = z, v_L(e_2) = k$, $v_R(e_1) = v_R(e_3) = \lambda, v_R(e_2) = 0$. In instance I, all agents truthfully report their types, and in instance I', edges e_1 and e_3 are not available.

Let $\mathcal{A}$ be any randomised mechanism. Assume that the approximation ratio of $\mathcal{A}$ is at most ρ. Suppose that $\mathcal{A}$ will output $\{e_1, e_3\}$ with probability p, and output $\{e_2\}$ with probability $1 - p$ in instance I, as well outputs $\{e_1, e_3\}$ with probability q, and $\{e_2\}$ with probability $1 - q$ in instance I'. We assume that the optimal solution for instance I is $\{e_1, e_3\}$, which then means $2(\lambda + z) > k$. We obtain that:

$$\frac{2(\lambda + z)}{2p(\lambda + z) + (1 - p)k} \leq \rho \tag{3.1}$$

$$\frac{k}{k(1 - q)} \leq \rho \tag{3.2}$$

Moreover, since $\mathcal{A}$ is truthful in expectation, the expected utility of the left agent from $\mathcal{A}$'s allocation for instance I should be at least that of instance I'. Otherwise, this agent would deviate from it. Therefore, we can obtain from inequality (3.2):

$$2zp + k(1 - p) \geq (1 - q)k \geq \frac{k}{\rho} \tag{3.3}$$

From inequality (3.1), we deduce:

$$p(2(\lambda + z) - k) \geq \frac{(2(\lambda + z) - k\rho)}{\rho} \tag{3.4}$$

and from inequality (3.3), we deduce:

$$p(2z - k) \geq \frac{k - \rho k}{\rho} \tag{3.5}$$

Then, we combine the two inequalities, (3.4) and (3.5), and we obtain:

$$\rho \geq \frac{4kz + 4k\lambda - 4z^2 - 4\lambda z - k^2}{2k\lambda}$$

which after dividing by the denominator gives:

$$\rho \geq \frac{2z}{\lambda} + 2 - \frac{2z^2}{k\lambda} - \frac{2z}{k} - \frac{k}{2\lambda}$$

By assigning $z = 1$ and taking limits $\lim k \to \infty$ and $\lim \lambda \to \infty$, finally, we obtain:

$$\rho \geq 2.$$

□

Therefore, randomised mechanisms do not improve the approximation ratios for the alignment problem.

3.6 Nash equilibria implementation

3.6.1 Pure strategy

In the *first price greedy matching* setting, the agents provide their declarations to the mechanism, which computes an outcome. The agents measure their utility by subtracting the payment from their true valuation of this outcome. The mechanism we use is given in Algorithm 3, and the payment scheme is that each agent has to pay its own bid, i.e., $p_i = b_i(\mathcal{A}(b))$.

Theorem 9. *The running time of Algorithm 3 is $O(m \log m)$, where m is the number of edges in the input bipartite graph.*

Proof. This time complexity comes from the fact that the greedy algorithm takes $O(m \log m)$ time to sort the edges by their weights. □

Theorem 10. *The price of anarchy of the first price greedy matching game is precisely* 4.

We prove this theorem by proving Theorems 11 and 13. Recall the notation: $b_i^e = b_i(e)$ and $v_i^e = v_i(e)$ for any $i \in \{L, R\}$ and $e \in E$. For a matching $M = \{e_1, e_2, \cdots, e_j\}$, denote $b_L^M = \sum_{e \in M} b_L^e$, and $v_L^M = \sum_{e \in M} v_L^e$. A matching M is found using the well known *greedy* Algorithm (Alg. 3).

In the next two theorems we prove lower bounds on the price of anarchy and price of stability for a first-price greedy matching game. These proofs present simple instances of this game to give a possible intuition of pure Nash equilibria.

Theorem 11. *The price of anarchy of the first price greedy matching game is at least* 4.

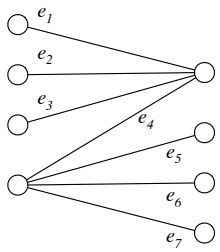


Figure 3.4: Low bound for Price of stability

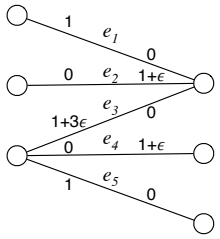


Figure 3.5: Lower bound for Price of anarchy

Proof. The graph structure is illustrated in Figure 3.4. The valuations assignment for both agents are: $v_L^{e_2} = v_L^{e_6} = v_R^{e_2} = v_R^{e_6} = 1, v_L^{e_4} = 1 + \epsilon$, and other unmentioned valuation for remaining is 0. And assume a strategy profile: $b_L^{e_4} = 1 + \epsilon, b_L^{e_3} = b_R^{e_1} = b_L^{e_7} = b_R^{e_5} = 1$, with bids on the remaining edges being 0, and denote strategy profile as b.

The greedy algorithm's outcome under b is $\{e_4\}$. The left agent could not increase its utility, as only one other outcome $\{e_2, e_6\}$ has positive utility. If it bids $\tilde{b}_L^{e_2} > b_R^{e_1}$ and $\tilde{b}_L^{e_6} > b_R^{e_5}$, the outcome changes to $\{e_2, e_6\}$, however, the new outcome leads to a negative utility for the left agent. The left agent also will not decrease its bid on e_4, otherwise, the outcome would be changed to another matching that is not $\{e_2, e_6\}$. The right agent's behaviour is the same as it is symmetric. This leads to a Nash equilibrium, which implies the ratio of 4. □

Algorithm 3 Greedy algorithm

Require: Bipartite graph $G = (V \cup U, E)$, b_L, b_R of the left and right agent.
Ensure: A matching M
  Let $M = \emptyset$
  if $E \neq \emptyset$ **then**
    Find the edge $e \in E$ that maximizes $b_L^e + b_R^e$
    Let $M := M \cup \{e\}$
    Remove from E edge e and edges incident to edge e
  end if
  M is the outcome

Theorem 12. *The price of stability of the first price greedy matching game is at least 2.*

Proof. Consider an instance illustrated in Figure 3.5. The valuations assignment for both agents are: $v_L^{e_1} = v_L^{e_5} = 1$, $v_R^{e_2} = v_R^{e_4} = 1 + \epsilon$, $v_L^{e_3} = 1 + 3\epsilon$. The valuations on the remaining edges are 0 for both agents.

Case-1. Suppose the outcome of the mechanism is $\{e_1, e_5\}$. The current bid cannot be a Nash equilibrium, because the right agent would improve its utility by changing the current outcome to $\{e_2, e_4\}$, when $\tilde{b}_R^{e_2} > b_L^{e_1}$, $\tilde{b}_R^{e_4} > b_L^{e_5}$, where $\tilde{b}_R^{e_2}$ denotes the new bid of agent R on e_2.

Case-2. Suppose the current outcome is $\{e_2, e_4\}$. It also does not admit any Nash equilibrium. If $\max\{b_R^{e_2}, b_R^{e_4}\} < v_L^{e_3}$, then the left agent would improve its utility by changing to e_3, when $\tilde{b}_R^{e_3} > \max\{b_R^{e_2}, b_R^{e_4}\}$. If $\max\{b_R^{e_2}, b_R^{e_4}\} > v_L^{e_3}$, let $b_R^{e_2}$ be a smaller one, the left agent will bid $\tilde{b}_L^{e_1} > b_R^{e_2}$ changing the outcome to $\{e_1, e_4\}$. This is symmetric.

Case-3. Suppose the current outcome is $\{e_1, e_4\}$ (or $\{e_2, e_5\}$). The right agent would bid $\tilde{b}_R^{e_2} > b_L^{e_1}$ to improve its utility, and change the outcome to $\{e_2, e_4\}$.

To complete the proof, we provide a Nash equilibrium: $b_L^{e_1} = b_L^{e_5} = 1$, $b_R^{e_2} = b_R^{e_4} = 1 + \epsilon$, $b_L^{e_3} = 1 + 2\epsilon$. We can see in such a bid profile, the outcome would be e_3, and it is easy to check that no agent can increase its utility. □

Now, we prove the main theorem. We use $\tilde{b}$ to denote an alternative bid to b.

Theorem 13. *The price of anarchy of a first price greedy matching game is at most* 4.

Before the proof of this theorem, we introduce two lemmas.

Lemma 1. *Suppose that the current bid profile (b_L, b_R) produces outcome M by greedy mechanism, then the necessary condition for (b_L, b_R) to be a Nash equilibrium is $b_L^M \leq v_L^M$ and $b_R^M \leq v_R^M$.*

Proof. Assume that for outcome M, some agent's bid satisfies $b_i^M > v_i^M$, then $u_i(b_L, b_R) = v_i^M - b_i^M < 0$, which means its utility is negative. Therefore, agent i will change its bid to a new one which increases its utility to be at least 0. □

Lemma 2. *Suppose that the current bid profile (b_L, b_R) produces outcome M by greedy mechanism and $b_L^M \leq v_L^M$, $b_R^M \leq v_R^M$. Then there exists a bid for one agent, say the left agent, $\tilde{b}_L$, satisfying $\tilde{b}_L^{M'} < 2(v_R^M + v_L^M) + \epsilon$, and $\tilde{b}_L$ can change the outcome to M'.*

Proof. Let $\{e_1, \cdots, e_k\}$ be the set of M indexed by decreasing order with respect to $b_L^e + b_R^e$. Denote e' as an edge in M'. We assign each $\tilde{b}_L^{e'}$ value by the following procedure: $\forall j \in \{1, \ldots, k\}$ (in this order), if e_j's left hand side vertex adjacent edge e' is in M', then let $\tilde{b}_L^{e'} + b_R^{e'}$ take the value slightly higher than $b_R^{e_j} + b_L^{e_j}$, and do the same for the right hand side vertex adjacent edge, i.e., for right side vertex adjacent

edge $e' \in M'$ of e^j, let $\tilde{b}_L^{e'} + b_R^{e'}$ take the value slightly higher than $b_R^{e_j} + b_L^{e_j}$. In any step of this procedure, if we need to reassign to this edge the bid $b_L^{e'}$ of edge e', we keep it as the larger one (actually, we keep the declaration unchanged, since the procedure is conducted by decreasing order of $b_L^e + b_R^e$). Such bids distribution is valid, since we can do it so that $\tilde{b}_L^{M'} > 2(b_R^M + b_L^M)$, which always changes the outcome to M'. We can also easily argue that $\tilde{b}_L^{M'} < 2(v_R^M + v_L^M) + \epsilon$. □

Proof. (of Theorem 13) Let M be any matching whose total valuation is strictly smaller than a quarter of the optimum, i.e., $v_L^M + v_R^M < \frac{1}{4}Opt$. Note that at least one of the following statements holds: $\exists M' v_L^{M'} \geq \frac{1}{2}Opt$, and $\exists M' v_R^{M'} \geq \frac{1}{2}Opt$. If M' is the optimal solution, this results in a contradiction. As they are symmetric, we assume the first statement is true.

Assume $b = (b_L, b_R)$ is a fixed bid profile. If the outcome under b is M, then we either have: $b_L^M \leq v_L^M$ and $b_R^M \leq v_R^M$; or the agents will have negative utilities.

We want to show that the left agent would be incentivised to bid for the outcome M'. Let $\tilde{b}_L^{M'}$ be the bid that can change the outcome M to M'. By Lemma 2 (see above), there exists a bid $\tilde{b}_L$ that will change the outcome to M'. Thus, we want to show: $v_L^{M'} - \tilde{b}_L^{M'} > v_L^M - b_L^M$. By Lemma 1, since $\tilde{b}_L^{M'} < 2(v_R^M + b_L^M) + \epsilon$, then we have:

$$v_L^{M'} - \tilde{b}_L^{M'} \geq v_L^{M'} - 2(v_R^M + b_L^M) - \epsilon$$

Since $v_L^{M'} \geq \frac{1}{2}Opt$ and $v_L^M + v_R^M < \frac{1}{4}Opt$, then

$$v_L^{M'} - 2(v_R^M + b_L^M) - \epsilon \geq v_L^{M'} - (v_R^M + v_L^M) - v_R^M - b_L^M$$

$$v_L^{M'} - (v_R^M + v_L^M) - v_R^M - b_L^M > v_L^M - b_L^M$$

We can remove ϵ, since it can be arbitrarily small. The last inequality shows that the left agent can change its bid from b_L to $\tilde{b}_L$ and get M' with a higher utility. This completes the argument as it shows that b cannot be a Nash equilibrium. □

Theorem 14. *There exists an instance which has no pure Nash-equilibrium.*

Proof. We will prove this claim by showing that in whatever outcome M of mechanism $\mathcal{M}$, this outcome is not stable for agents' bids, which means, there exists a new bid $\hat{b}$, and under this new bid, the new outcome M' would be different with original outcome M.

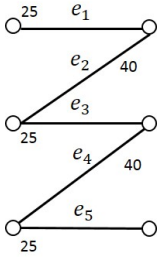


Figure 3.6: The instances where no pure Nash-equilibrium exists

Case-1. Suppose the outcome is $M = \{e_1, e_4\}$. We observe that $b_R^{e_4} \leq 40$, otherwise it leads to negative utility for right agent. And $b_R^{e_1} = 0$, otherwise the right agent can always decrease bid to 0 to get larger utility. Additionally, it also implies that $b_L^{e_1} > b_R^{e_2}$ or $b_L^{e_3} > b_R^{e_2}$, because of the greedy order. The maximum utility the left agent can achieve is at most 25, because the only edge which has positive valuation for the left agent in the outcome M is e_1, and $v_L^{e_1} = 25$. Thus, $b_L^{e_1} \leq 25$, otherwise, the left agent would achieve negative utility. This also implies that $b_R^{e_2} \leq b_L^{e_1} \leq 25$, otherwise, due to the greedy algorithm, e_2 is in M.

We want to show the left agent can declare different bid to change the outcome M to a new one for achieving a higher utility. Let $b_L^{e_1} = b_L^{e_2} = b_L^{e_4} = b_L^{e_5} = 0$ and $b_L^{e_3} > 40$. The new outcome now is $M' = \{e_1, e_3, e_5\}$, because we know that $b_L^{e_3} > b_R^{e_2}$ and $b_L^{e_3} > b_R^{e_4}$. And because

$$v_L^{M'} - b_L^{M'} = 75 - 40 - \epsilon > 25 \geq v_L^M,$$

the utility of left agent is increased.

The arguments can apply to symmetric case where the outcome is $\{e_2, e_5\}$.

Case-2. Suppose the outcome is $M = \{e_1, e_3, e_5\}$. In this case, we observe that the right agent's utility is at most 0, because any edges which have positive valuation for the right agents are not in the outcome M. We also know $b_L^M \leq 75$, otherwise the left agent would achieve a negative utility. Since the right agent's total valuation is 80, if the outcome were $\{e_2, e_4\}$, it is easy to observe that the right agent can always find a distribution of declaration to change the outcome M to $M' = \{e_2, e_4\}$. And under the new outcome, the utility of the right agent is increased.

Case-3. Suppose the outcome is $M = \{e_2, e_4\}$. In this case, we observe that the left agent's utility is at most 0, because any edges which have positive valuation for the left agents are not in the outcome M. We know $b_R^M \leq 80$, otherwise the right agent

would achieve a negative utility. Note that e_2 and e_4 are symmetric with the bid.

Firstly, if we assume that $b_R^{e_2} > 55$ and $b_R^{e_4} < 25$, then the left agent could declare $25 \geq b_L^{e_5} > b_R^{e_4}$, which will change the outcome and increase the utility of the left agent.

Secondly, if we assume that $b_R^{e_2} \leq 55$ and $b_R^{e_4} \leq 55$, then the left agent could declare $b_L^{e_3} > 55 \geq b_R^{e_2}$ We should notice that if $b_R^{e_2} > 55$, $b_R^{e_4} < 25$, then the left agent could bid $25 > b_L^{e_5} > b_R^{e_4}$, which change the outcome and increase the utility. If $b_R^{e_2} > 55 \geq b_R^{e_2}$, and $b_L^{e_1} = b_L^{e_5} = 0$, then the outcome will change to $M' = \{e_1, e_3, e_5\}$, and it increase the utility of the left agent.

This proves the claim that this instance admits no pure Nash-equilibrium. □

Theorem 14 implies that the existence of Nash equilibrium in all possible instances seems to be rare, because the graph structure and agents' types we used to prove the non-existence of pure Nash equilibrium are rather common. Therefore, a natural question is which graph structure and agents' types can form a instance, which guarantee the existence of pure Nash equilibrium? Are those instances truly rare?

3.6.2 Relation to smooth games

Roughgarden [62] has introduced a seminal tool of smoothness as a technique to proving the results on the price of anarchy for various games. One may wonder if we could use his techniques in our context. However, we show here that our game is not smooth, and therefore we cannot apply the smoothness paradigm in our context.

Definition 7 ([62]). *A maximization game is* (λ, μ)-smooth *if*

$$\sum_{i=1}^{k} u_i(s_i^*, \boldsymbol{s}_{-i}) \geq \lambda \cdot V(\boldsymbol{s}^*) - \mu \cdot V(\boldsymbol{s})$$

for all strategy profiles $\boldsymbol{s}$,$\boldsymbol{s}^*$. *Here* $V(\cdot)$ *is an objective function that satisfies* $V(\boldsymbol{s}) \geq \sum_{i=1}^{k} u_i(\boldsymbol{s})$ *for every strategy profiles* $\boldsymbol{s}$.

Claim 1. *The first price greedy matching game is not smooth.*

Proof. Consider an instance where the graph structure is illustrated in Figure 3.3. The valuation of agents are: $v_L^{e_1} = v_L^{e_3} = 0$ and $v_L^{e_2} = 20$, $v_R^{e_1} = 100$ and $v_R^{e_2} = v_R^{e_3} = 0$. We select two strategy profiles $\mathbf{s}^*$, $\mathbf{s}$, where in $\mathbf{s}^*$, the left agent bids is: $b_L^{e_2} = 100 - \epsilon, b_L^{e_1} = b_L^{e_3} = 0$, the right agent bids is: $b_R^{e_1} = 100, b_R^{e_2} = b_R^{e_3} = 0$. And in $\mathbf{s}$, the left agent bids is: $b_L^{e_2} = 20, b_L^{e_1} = b_L^{e_3} = 0$, the right agent bids is: $b_R^{e_1} = 20 - \epsilon, b_R^{e_2} = b_R^{e_3} = 0$.

$u_L(b_L^*, b_R) + u_R(b_R^*, b_L) = v_L(e_2) - b_L^*(e_2) + v_R(e_1 \cup e_3) - b_R^*(e_1 \cup e_3) = -80 + 0 = -80$. $\lambda \cdot V(\mathbf{s}^*) = \lambda \cdot 100$, and $\mu \cdot V(\mathbf{s}^*) = \mu \cdot 20$. Therefore, by this strategy, we can construct any instances to make the inequality to be false. □

3.6.3 Mixed strategy

Let us first recall the definition of mixed Nash equilibrium from Section 3.4. It is usual in the literature to study the price of anarchy even if there might be instances without pure Nash equilibria [49]. Thus, Theorem 10 can be read as: *if there exists pure Nash equilibria, then their social welfare is at least* 25% *of the optimum.* We can also show that mixed Nash equilibria always exist, by transforming the problem into a new one in which each agent only has a finite number of strategies, where a strategy is for bids on edges. We define a small $\epsilon > 0$ as the minimum increment that any two bids can differ by. This leads to a finite number of strategies of any agent i as i will not bid more than $\sum_{e \in E} v_i(e)$. In particular, $b_i^e \in \{0, \epsilon, 2\epsilon, \cdots, \sum_{e \in E} v_i(e)\}$.

Theorem 15. *The mixed Nash equilibrium exists for all instances of the discretised first price greedy matching game.*

This theorem is deduced directly from Nash's theorem [55]. This theorem proves that if agents can use mixed strategies, then every game with a finite number of players in which each player can choose from finitely many pure strategies has at least one mixed Nash equilibrium.

Theorem 16. *The price of anarchy of the discretised first price greedy matching game for mixed strategy is* 4*.*

The key observation for proving this theorem is: for any strategy profile b, if under this profile, the outcome is M, then $b_L^M \leq v_L^M$ and $b_R^M \leq v_L^M$, otherwise, b cannot be part of mixed Nash equilibrium strategy. Therefore, assume that v_L. Another observation is: If new bid b' cannot change entirely M to M', then the mixed new matching M^*'s total valuation is still larger than M.

Proof. Let M be any matching whose total valuation is strictly smaller than a quarter of the optimum, i.e., $v_L^M + v_R^M < \frac{1}{4}Opt$. Note that at least one of the following statements holds: $\exists M' v_L^{M'} \geq \frac{1}{2}Opt$, and $\exists M' v_R^{M'} \geq \frac{1}{2}Opt$. If M' is the optimal solution, this results in a contradiction. As they are symmetric, we assume the first statement is true.

Now, let $b = (b_L, b_R)$ be a fixed bid profile from a mixed strategy profile ω. If the outcome under b is M, then we have the following statements: $b_L^M \leq v_L^M$ and $b_R^M \leq v_L^M$.

Assume that given a b, where $b_R^M > v_L^M$ and $b\omega_R$, which means $P(b) > 0$. Under the assumption that M is the outcome, $u_R(b) = v_R^M - b_R^M < 0$, then $P(b)(u_R(b)) < 0$, thus, there always exists a mixed strategy ω'_R, where $P(0) = 0$, satisfy $\mathbf{E}_{b\sim\omega'_R}[u_R^{M(b)}] > \mathbf{E}_{b\sim\omega_R}[u_R^{M(b)}]$. Therefore, given a outcome M, for all strategy $b \in \omega$, where $P(b) > 0$, $b_L^M \leq v_L^M$ and $b_R^M \leq v_L^M$. Note that this claim ensures the applicability of the lemma.

We want to show that the left agent would be incentivised to bid for the outcome M' with respect to corresponding strategy $\tilde{b}$. Let $\tilde{b}_L^{M'} \subseteq \tilde{b}$ be the pure strategy (bid) that can change the outcome M to M', precisely, $\tilde{b} \in \omega_L$ and $P(\tilde{b}) = 1$. By Lemma 2, there exists a bid $\tilde{b}_L$ that will change the outcome to M'. Thus, we want to show:

$$\mathbf{E}_{\tilde{b}\sim\omega'_L} u_R^M(\tilde{b}) > \mathbf{E}_{b\sim\omega_L} u_R^M(b)$$

which is:

$$\mathbf{E}_{\tilde{b}\sim(\omega'_L,\omega_R)}[v_L^{M(\tilde{b})} - \tilde{b}_L^{M(\tilde{b})}] > \mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_L^{M(b)} - b_L^{M(b)}]$$

By Lemma 2, since $\tilde{b}_L^{M(\tilde{b})} < 2\mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_R^{M(b)} + v_L^{M(b)}] + \epsilon$, then

$$\mathbf{E}_{\tilde{b}\sim(\omega'_L,\omega_R)}[v_L^{M(\tilde{b})} - \tilde{b}_L^{M(\tilde{b})}] \geq \mathbf{E}_{\tilde{b}\sim(\omega'_L,\omega_R)}[v_L^{M(\tilde{b})}] - 2\mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_R^{M(b)} + v_L^{M(b)}] + \epsilon$$

Since $\mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_L^{M(\tilde{b})}] \geq \frac{1}{2}OPT$ and $\mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_L^{M(b)} + v_R^{M(b)}] < \frac{1}{4}OPT$, then

$$\begin{aligned}
&\mathbf{E}_{\tilde{b}\sim(\omega'_L,\omega_R)}[v_L^{M(\tilde{b})}] - 2\mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_R^{M(b)} + v_L^{M(b)}] + \epsilon \\
&\geq \mathbf{E}_{\tilde{b}\sim(\omega'_L,\omega_R)}[v_L^{M(\tilde{b})}] - \mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_R^{M(b)} + v_L^{M(b)} - v_R^{M(b)}] - b_L^{M(b)} \\
&> \mathbf{E}_{b\sim(\omega_L,\omega_R)}[v_L^{M(b)}] - b_L^{M(b)}
\end{aligned}$$

We can remove ϵ, since it can be arbitrarily small. The last inequality shows that the left agent can change its bid from ω_L to ω'_L and get $M(\tilde{b})$ with a higher utility. This completes the argument as it shows that b cannot be a Nash equilibrium. □

3.7 Conclusion

In this chapter, we have presented the ontology alignment problem, and we studied this problem algorithmically as a mechanism design problem, modeled as a social welfare maximising bipartite matching setting, where the valuation function is additive. Firstly, we proved the impossibility results for this problem, that dominant strategy

mechanisms cannot be time efficient. Secondly, we have provided a complete picture of the complexity of this mechanism design problem by showing that when coupled with a first-price payment scheme and a greedy method, it implements Nash equilibria which are very close to the optimal matching. This has been achieved by completely characterising the Price of Anarchy of this mechanism that has been shown to be precisely 4; this bound also holds for Mixed Nash equilibria.

Chapter 4

Negative results for greedy maximum independent set

4.1 Introduction

In this chapter, we will present negative results related to finding the maximum independent set problem by the greedy algorithm. These results prove two statements. Firstly, the identification of graphs on which the greedy algorithm can obtain optimal solution in general graphs and planar cubic graphs is computationally hard. Secondly, and most importantly, we prove that it is computationally hard to find an appropriate advice (of which minimum degree vertex to choose in case if there are more than one) for the greedy algorithm that can lead to good approximation. This suggests that the task of finding such advice and proving that they lead to good approximation can itself be a difficult task. Indeed, our analysis in the following Chapter 6 is inherently complex.

An independent set in a graph G is a set of vertices in which every pair of vertices are not adjacent. An independent set is *maximal* if it is not a proper subset of other independent sets, and is *maximum* if it has maximum cardinality, i.e., size, among all independent sets. We denote by $\alpha(G)$ the cardinality of a maximum independent set, and let I be any particular independent set. Then, the *maximum independent set* problem or simply *MIS* is defined as the following decision problem: Given a graph G and an integer k, whether there is an independent set I in G with cardinality $|I| \geq k$?

The algorithm that is described in Algorithm 4 is called a *Greedy algorithm.* We call an outcome S of the greedy algorithm a greedy set and its elements are called chosen or selected vertices. It is easy to check that a greedy set is a maximal independent set.

Noting that the minimum degree in the remaining graph might not be unique, then the greedy algorithm varies according to different greedy rule for finding the minimum degree vertex. For example, a greedy algorithm with an *oracle* advice will generate the optimal greedy set (maximum greedy set), and a greedy algorithm with no particular advice will in each iteration arbitrarily choose one of the minimum degree vertices. We present a precise definition here.

Definition 8. *The* greedy order *or* advice of greedy *of a greedy algorithm is a fixed deterministic rule or algorithm which advises the greedy algorithm which minimum degree vertex to be chosen, if the choice of minimum degree vertices is not unique.*

We note here that such randomised rules were also studied in context of finding large independent sets in random graphs. However, in this thesis we only study such deterministic rules.

For example, *oracle* advice is an algorithm that in each iteration, advises the greedy algorithm to choose the vertex which maximise the size of the greedy set. Specifically, oracle advice might compute the optimum maximum independent set of given graph G, and then advise greedy algorithm how to choose its minimum degree vertices according to this solution. Note however that it requires exponential time to compute.

Algorithm 4 General greedy algorithm

Input: a graph $G = (V, E)$

$U \leftarrow V$
$S \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  Find a vertex $v \in U$ with minimum degree in $G[U]$, according to given greedy order.
  $U \leftarrow U \setminus N_G(v)$
  $S \leftarrow S \cup \{v\}$
end while
return S

4.2 Inapproximability

In this section, we will address several questions about limitations of the greedy algorithm for the maximum independent set problem. The following definitions are useful.

Definition 9. *The* MaxGreedy *problem is given a graph G and an integer k, and to decide if there is a greedy set S in G with cardinality $|S| \geq k$.*

Definition 10. *The* MinGreedy *problem is given a graph G and an integer k, and to decide if it is true that for all possible greedy sets S in G their cardinality is such that $|S| \geq k$.*

It is easy to see that definition of *MinGreedy* is equivalent with the following statement:

Definition 11. *The* MinGreedy *problem is given a graph G and an integer k, and to decide if there is a greedy set S in G with cardinality $|S| \leq k$.*

We denote $\alpha^+(G)$ and $\alpha^-(G)$ as the size of a maximum and minimum greedy set in G respectively.

We want to measure given a graph G, whether there exists one greedy set S in G with cardinality $r \cdot |S| \geq \alpha(G)$, where $\alpha(G)$ is the size of the maximum independent set of G. Additionally, we also want to measure a stronger version of such kind of measurement, i.e., given a graph G, whether for all possible greedy sets S in G their cardinality is $r \cdot |S| \geq \alpha(G)$.

Definition 12. *The $GreedyOPT_r^{\exists}$ problem is given a graph G, to decide whether it holds that $r \cdot \alpha^+(G) \geq \alpha(G)$.*

Definition 13. *The $GreedyOPT_r^{\forall}$ problem is given a graph G, to decide whether it holds that $r \cdot \alpha^-(G) \geq \alpha(G)$.*

Theorem 17. *The $GreedyOPT_r^{\forall}$ problem is* NP*-hard, for any $r \in \mathcal{Q}$ and $0 < r \leq 1$.*

Proof. To prove hardness for *NP*, we present a reduction from maximum independent set problem to the $GreedyOPT_r^{\forall}$ problem. According to the statement, r is a rational, so we can use two integers s, t to represent it: $r = \frac{s}{t}$ and $0 < r \leq 1$. Let $G = (V, E)$ be the given graph where we want to know whether the size of maximum independent set of G is larger than or equal to a given integer k.

We construct the graph G'_k as follows:

Let G^c be a clique with $t \cdot |V|$ vertices. And G^* is a graph which contains $s \cdot k - 1$ vertices without any edges between these vertices. And $\mathbf{G}$ be a set of graphs, which contains t times duplicated graph G. Let G' be the graph such that:

$V(G') = V(\mathbf{G}) \cup V(G^c) \cup V(G^*)$,

$E(G') = \{(u, v) | u \in V(G^c), v \in V(\mathbf{G})\} \cup \{(u, v) | u \in V(\mathbf{G}), v \in V(G^*)\} \cup E(\mathbf{G}) \cup E(G^c)$

Figures 4.1 and 4.2 illustrate the construction of graph G'. An observation about the degree of different vertices in these graphs is helpful to understand the following part of the proof. For each vertex v in the graph G^*, the degree $d(v)$ of v is $d(v) = t \cdot |V|$. For each vertex v in each graph G, $d(v) \geq t \cdot |V| + s - 1$, because besides the edge it already has, each of its vertices connects to $t \cdot |V|$ vertices in the clique G^c. And for each vertex v in G^c, $d(v) \geq 2t \cdot |V|$.

Now, we run greedy algorithm on the graph G', we observe that whatever greedy algorithm it is, it will always select all $v \in G^*$ and one of the vertices in G^c as the solution, because at the beginning, vertices' degree in G^* are smallest. After the execution of this vertex, all vertices in $\mathcal{G}$ are removed, thus the remaining $s-2$ vertices in G^* will be selected as solution, and for clique G^c, one of its vertices will be selected as solution. Observe that finally the size of the greedy solution is s.

Let us define an oracle $O(G, r)$ for $GreedyOPT_r^{\forall}$ that given a graph and a rational number r, will provide the answer of whether $r \cdot \alpha^+(G) \geq \alpha(G)$ or not. Then, we run the oracle on graph G' and given rational number $r = 1$. If the answer from the oracle is Yes, then we know that the independent number of the set of graphs $\mathcal{G}$ will not be larger than s. Thus, the independent number of each graph $G \in \mathbf{G}$ will be no larger than $\frac{s}{t} = r$, and if $1 \leq r < 2$, then the independent number of G will be 1. If the answer is No, then we repeatedly execute the following procedure until the answer turns to Yes.

In each iteration, we add s number of vertices $\mathbf{v}$ into G^*, and add edges into G' with $\{(u, \mathbf{v}) | u \in G, \forall G \in \mathbf{G}\}$, assuming now G^* has $z = k \cdot s - 1$ vertices, $k \in \mathcal{N}^+$. We run the oracle again, and if the answer is Yes, then we know the independent number of graph G will not be larger than $\frac{z}{s}$. This procedure only requires at most $|V(G)|$ iterations, and after that we can know the exact independent number of G.

This completes the argument for proving *NP*-hardness of the $GreedyOPT_r^{\forall}$ problem. □

For a comparison to previous work, in paper [8], the authors show the following theorem:

Theorem 18. *[8] $GreedyOPT_r^{\forall}$ problem is* co-NP*-complete, for any $r \in \mathcal{Q}$ and $0 < r \leq 1$.*

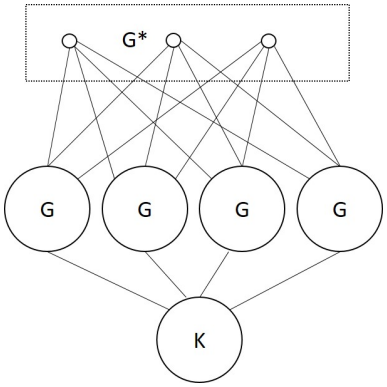


Figure 4.1: Step 1 of constructing graph G^*

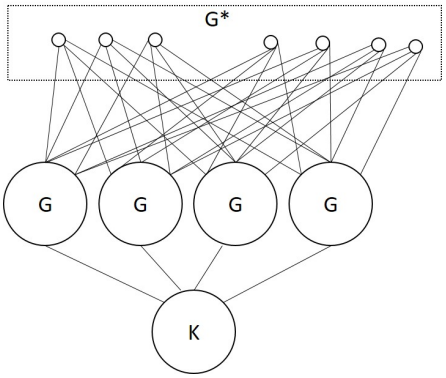


Figure 4.2: Step 2 of constructing graph G^*

4.2.1 Planar graphs

Since the main goal of the thesis is to study the greedy algorithm on sub-cubic graph, we want to know whether the advice for greedy algorithm on sub-cubic graph is also computationally hard. The answer is positive by the following theorem.

Theorem 19. *MaxGreedy problem is* NP*-hard even for planar cubic graphs.*

Proof. To prove *NP*-hardness, we present a reduction from the maximum independent set problem to the MaxGreedy problem.

Let $G = (V, E)$ be a cubic planar graph with $|E| = m$. We construct a graph $G' = (V', E')$ by replacing each edge $(u, v) \in E$ by a graph structure described in Figure 4.3; we call this gadget as $\mathcal{H}_e$. When we refer to this gadget, we will use the same notations. The gadget $\mathcal{H}_e$ contains vertices $\{a, b, c, d, g\}$ and three substructures $\mathcal{A}, \mathcal{D}, \mathcal{C}$, and the edges between them as defined in the figure. The edge point vertices u and v in the original graph are connected to vertices a and b respectively as illustrated in the figure. It is easy to see that the construction of G' can be done in polynomial time from graph G. Denote by $I(G)$ a maximum independent set on a graph G.

To prove the theorem, we use an equivalent optimisation version of the maximum independent set problem.

Let S be a greedy set for the constructed graph G', and $S = (S \cap V) \cup (S \cap V')$. We say a set of vertices $S \in V'$ of the constructed graph G' is independent with respect to the original graph G, if its corresponding set of vertices in V of G is independent. For convenience, let $f : V \to V^*$ be a $1 - 1$ mapping, where v^* is the set of vertices of G' which are not in any gadget.

Some observations about $\mathcal{H}_e$ are needed. For any greedy algorithm, denote its solution on graph G by $SOL(G)$. Consider a $\mathcal{H}_e$, and suppose that:

1. u and v of $\mathcal{H}_e$ is in $SOL(G)$, then $|(SOL(G) \cap V(\mathcal{H}_e))| = 8$. Assume that u is chosen by the greedy algorithm first, then the algorithm will successively choose vertices b, c in $\mathcal{H}_e$ and the remaining vertices in $\mathcal{A}, \mathcal{D}$ and $\mathcal{C}$. That is because these vertices will be the minimum degree vertices in each step of the execution. Thus, we see that we have $|(SOL(G) \cap V(\mathcal{H}_e))| = 8$. Note that after these executions, v can be chosen by the greedy algorithm because now its degree is minimum.

2. only one of u or v of $\mathcal{H}_e$ is in $SOL(G)$, then $|(SOL(G) \cap V(\mathcal{H}_e))| = 9$. Assume that u is chosen by the greedy algorithm, then the algorithm will successively choose vertices b, g in $\mathcal{H}_e$ and the remaining vertices in $\mathcal{A}, \mathcal{D}$ and $\mathcal{C}$ (the reason for this is as above). Then by a calculation, we have $|(SOL(G) \cap V(\mathcal{H}_e))| = 9$. Note that after the execution of vertex g, v is removed by the greedy algorithm, thus v will never be chosen.

3. none of u or v of $\mathcal{H}_e$ is in $SOL(G)$, then $|(SOL(G) \cap V(\mathcal{H}_e))| = 9$. Assume that u is removed by greedy algorithm previously, then the algorithm will successively choose vertices a, g in $\mathcal{H}_e$ and remaining vertices in $\mathcal{A}, \mathcal{D}$ and $\mathcal{C}$. By a calculation, we have $|(SOL(G) \cap V(\mathcal{H}_e))| = 9$. Note again that after the execution of vertex g, v is removed by the greedy algorithm, thus v will never be chosen.

Then, we prove the following claim.

Claim 2. *There is a greedy set S for the constructed graph G', which satisfies:*

1. *$S \cap V'$ is independent with respect to G.*

2. *$|S| = z + 9m$, where $z = |S \cap V'|$.*

3. *S is a maximum greedy set on G'*

Proof. Firstly, we show that a particular set of vertices is the maximum greedy set on G'. Let $I(G)$ be the maximum independent set of graph G, and let $V(I(G))$ be the set of vertices $\{f(u) \in V' | u \in I(G)\}$. There exists a greedy set S^*, such that $V(S^*) \setminus \bigcup_{e \in E} V(\mathcal{H}_e) = V(I(G))$ by the greedy algorithm. Then, we want to show that for any other greedy set S' on G', $|S^*| \geq |S'|$.

Note $|S^*| = z + 9m$, where $z = |I(G)|$. Observe that for any $\mathcal{H}_e$, at most one of u or v is in S^*. Thus, for every $\mathcal{H}_e$, we have $|V(\mathcal{H}_e) \cup S^*| = 9$ by the above observation.

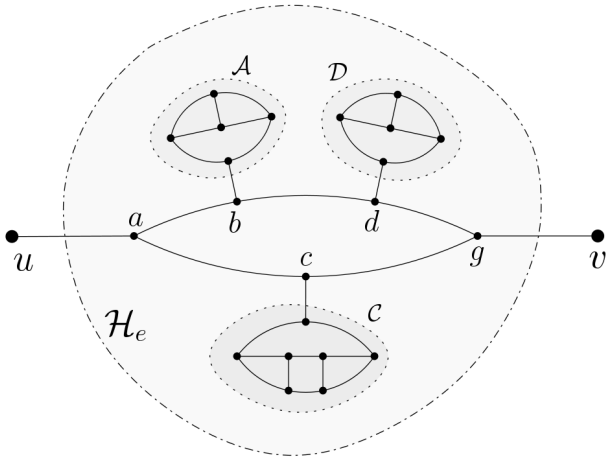


Figure 4.3: Gadget $\mathcal{H}_e$ for edge of planar graph

For any different set of vertices I' on G that is an independent set, any set of vertices S', with $V(S') \setminus \bigcup_{e \in E'} V(\mathcal{H}_e) = V(I')$, it is obvious that $|S^*| \geq |S'|$.

It completes the proof of the claim. □

Now, we will complete the proof of the theorem. By the assumption of *MaxGreedy*, it provides a greedy set which is of maximum size, and let S^* be the greedy set such that it satisfies the conditions in Claim 2. Then $f^{-1}(V(S^*) \setminus \bigcup_{e \in E} \mathcal{H}_e)$ is the maximum independent set of G. This completes the reduction from the maximum independent set problem to the *MaxGreedy* problem. □

4.3 Conclusion

In this chapter, we have studied the negative aspects of finding MIS by the greedy algorithm. The results show that even in planar cubic graphs, the problem of computing the optimal greedy set is an *NP*-hard problem. This suggests that the task of finding the nice advice to guide greedy and proving that they lead to good approximation can itself be a difficult task. Indeed, our design of good advice and its analysis in the next chapters turns out to be quite complex.

Chapter 5

Instance study for maximum independent set problem

5.1 Introduction: Approximability of MIS

The problem of finding an independent set of maximum size in a graph, the Maximum Independent Set problem (MIS), is one of the fundamental *NP*-complete combinatorial optimisation problems. Because of its hardness of exact computation, we are interested in approximation algorithms for the maximum independent set problem. However, the MIS problem is also notoriously known for its approximation hardness. The best known algorithm for the general MIS problem performs slightly better than trivial, whose approximation ratio is $O(n/\log^2 n)$ [9]. Hastad [35] provided a strong lower bound of $n^{1-\epsilon}$ for any $\epsilon > 0$ for general MIS problem, under a reasonable assumption that $NP \subseteq ZPP$. Furthermore, even for MIS with bounded degree Δ (MIS-Δ) problem, it is still *NP*-complete, and it belongs to the class *MAX SNP*-complete, a subclass of *NP* optimisation problems consisting solely of constant factor approximate problems, shown by Papadimitriou and Yannakakis [59]. Later, for *MAX SNP*-hard problems it has been proved that no polynomial time approximation scheme (PTAS) is possible unless $NP = P$ [3]. Actually, even MIS-3 is known to be *MAX SNP*-complete [1].

The first known nontrivial approximation ratio for MIS on graphs with maximum degree Δ is Δ acquired by Lovasz's algorithmic proof [47] of Brooks's coloring theorem. Hochbaum [34] using the coloring technique accompanied with a method of Nemhauser and Trotter [54] obtained an algorithm with ratio $\Delta/2$. This approach can also be applied to the case of weighted graphs. Halldorsson and Radhakrishnan [31] showed that the greedy algorithm actually delivers a better ratio, $(\Delta+2)/3$. Berman and Furer

[6] designed a new algorithm whose performance ratios are arbitrarily close to $(\Delta+3)/5$ for even Δ and $(\Delta+3.25)/5$ for odd Δ.

Halldorsson and Radhakrishnan [32] afterwards, via subgraph removal techniques, obtained asymptotically better ratios, $\Delta/6+o(1)$ for relative small Δ, and $O(\frac{\Delta}{\log\log\Delta})$. After that, Berman and Fujito [5] obtained a better ratio arbitrarily close to $\frac{\Delta+3}{5}$. Finally, the latest results from Chlebík and Chlebíková [12], 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. Note, that the symbol $k!!$ denotes a product of all integers in $\{1,2,\ldots,k\}$ that have the same parity (odd or even) as k.

Since in this thesis we are particularly interested in the case of subcubic graphs, we shall give more attention to it. Firstly, the negative results show that it is *NP*-hard to approximate MIS for low degree graphs to within $\frac{95}{94}$ for $\Delta=3$, $\frac{48}{47}$ for $\Delta=4$ and $\frac{46}{45}$ for $\Delta=5$ see [11]. We will see that these lower bound results are far from the best currently known upper bounds analyses. There are numerous research papers on the upper bounds. Hochbaum [34] presented an algorithm with 1.5 ratio, whose running time is proportional to the time complexity of the bipartite matching problem, $O(n^{1.5})$. Berman and Fujito [5] obtained a $\frac{6}{5}$ ratio, however, their running time is huge, and even a tighter analysis from [32] shows that the complexity appears to be no less than n^{50}. Chlebík and Chlebíkova [12] showed 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^{16}, and if the ratio is fixed to obtain $\frac{4}{3}$, then the running time is still $n^{7.3}$. The authors of [6], [5] and [12] used basically the same technique: local search. Halldorsson and Radhakrishnan [32] provided another local search approach based on [6] and obtained a ratio of $\frac{7}{5}$ in linear time, and a $\frac{4}{3}+\epsilon$ ratio in time $O(ne^{1/\epsilon})$. Halldórsson and Yoshihara [33] *wrongly* claimed a $\frac{9}{7}$ approximation ratio of a greedy algorithm, and we will show why it is wrong in this chapter.

Since there is a complementary relation between the maximum independent set and minimum vertex cover, it is useful to see what is the previous research about the minimum vertex cover problem. For the minimum vertex cover problem in general, Garey and Johnson [25] presented a simple approximation algorithm based on maximal matching and gave an approximation ratio of 2 for general graphs. For the minimum vertex cover problem on sub-cubic graphs, Hochbaum [34] provided a $\frac{4}{3}$-approximation ratio, by using the method of Nemhauser and Trotter [54]. Berman [6] gave a $\frac{7}{6}$ ratio by the same approach. And the authors of [12] showed that a ratio which is slightly better than $\frac{7}{6}$ can be obtained. The time complexity of these algorithms is the same as

the previously cited ones and it is huge.

The ultimate goal of this study is to understand the full power of the greedy paradigm for the maximum independent set problem on sub-cubic and more general bounded degree graphs. This means, we aim to obtain the best possible approximation ratio of the greedy algorithms with best possible time complexity. By the above literature review, we see that the advantage of the greedy paradigm is not only its simplicity and efficiency, but also the potential for theoretical analysis. This chapter is not necessary for the understanding of Chapter 6.

5.2 Instances study for Greedy MIS

In this section, we study the characterisation of graph structures which prevent any greedy algorithm from achieving a relatively good solution on them. The motivation of such kind of study is clear, we want to prevent some graph structure which is the barrier to a good solution to appear in the remaining graph created after some execution of the greedy algorithm. This is potentially feasible, because given a graph G, the vertex which has minimum degree might not be unique. Actually, in most of the cases, the minimum degree vertices are hardly unique, and thus the "good" greedy algorithm should choose the "right" one. The different choice of the minimum degree vertex affects the quality of the solution significantly. Therefore, to characterise potentially problematic graph structures is a crucial study for the greedy algorithms for MIS. We begin from the primitive greedy algorithm of Algorithm 5. Note that this algorithm differs from the previous Algorithm 4 in that it does not use any fixed order of choosing minimum degree vertices.

Algorithm 5 Primitive greedy algorithm

Input: a graph $G = (V, E)$
  $U \leftarrow V$
  $S \leftarrow \emptyset$
  while $U \neq \emptyset$ **do**
    Arbitrarily select a vertex $v \in U$ with minimum degree in $G[U]$
    $U \leftarrow U \setminus (N_G(v) \cup v)$
    $S \leftarrow S \cup \{v\}$
  end while
  return S

We begin our study from cubic graphs, and in the following Chapters 5 and 6, when we mention graph G, then G is a sub-cubic graph. During the study, several important

observations are found. We first present some notation. The important concept in the following is that of a *reduction*, inspired by Halldorsson and Radhakrishnan [31]. We will present their precise definition in the next chapter and provide further elaboration. However, for the purpose of the current explanation of the observations and ideas, in here, we provide a less complete but consistent version of the definition of a *reduction*.

Observe that for a graph with its maximum independent set I, it is natural to define a *reduction* with respect to I. Initially, a reduction can be understood as vertices and edges removed by one iteration of the greedy algorithm. That is, a reduction consists of a vertex which is added into the current solution and its neighbouring vertices along with their incident edges. An *(i,j)-reduction* refers to the case of such reduction, where $i + 1$ vertices and j edges are deleted from the current graph when the reduction is executed.

We will have the following reductions: *(0,0)*, *(1,1)*, *(1,2) (1,3)*, *(2,3)*, *(2,4)-1*, *(2,4)-2*, *(2,5)-1*, *(2,5)-2*, *(2,6)*-reduction, which are the same as those in Halldorsson and Radhakrishnan [31], and are illustrated in Figure 5.1. Note, that the number $x \in \{1, 2\}$ in the notation *(2,4)-x* is used to just differentiate between two different reductions of type *(2,4)*, and the same applies to *(2,5)-x*. For instance, if a *(2,6)*-reduction is executed, see the last reduction in Figure 5.1, then its root vertex is chosen as part of the independent set (the current solution of the greedy), and the two vertices in the middle are also deleted from the current graph. Also, 6 edges in total are deleted.

Note that these reductions are all possible reductions in sub-cubic graph G, where the minimum degree of G is at most 2. For any reduction, there is a unique vertex v^* that we call a root vertex, and such that the greedy algorithm will take it into the solution. The set of vertices V_R of reduction R includes v^* and its neighbour vertices $N(v^*)$. We use a term *contact vertices* to refer to the vertices which are neighbours to $N(v^*)$. Moreover, the edges of the reduction include edges which are removed by its execution. We call *contact edges* the edges which are incident to the contact vertices. If the context is clear, we abbreviate *(2,4)-2* and *(2,5)-2* to just *(2,4)* and *(2,5)*, respectively.

The execution of the greedy algorithm naturally defines a sequence of reductions S as an ordered set of reductions, $S = \{R_1, \ldots, R_k\}$, and for every pair of i, j, if $i < j$, then we say the reduction R_i is executed before R_j.

An example illustrated in Figure 5.2, shows how greedy algorithm executes reductions. In the first step, the greedy algorithm execute reduction 1, as a *(2,6)*-reduction. In the second step, it executes a *(2,3)*-reduction. Subsequently, it executes the remaining reductions. Note that reduction 4 is a *(1,3)*-reduction, reduction 7 is a *(1,2)*-

reduction, and reduction 8 is a *(1,1)*-reduction.

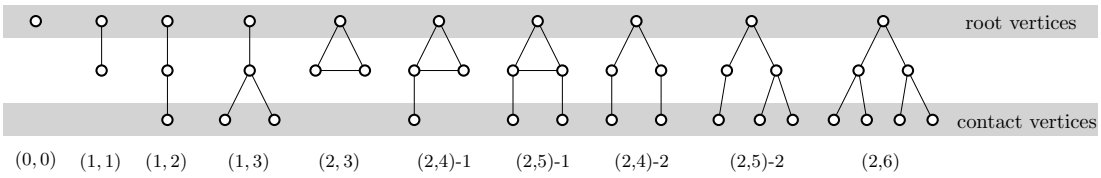


Figure 5.1: Collection of reductions with root of degree at most 2.

A natural question is what we can say about the reductions such as $(3, x)$, whose root vertex has degree 3? The answer is that for any connected graph, only the first reduction is possible to be such reduction in the entire sequence of reductions. Therefore, with the increasing of number of vertices, the loss of solution will be asymptotically small. Actually, in the next chapter, we will provide an approach to even eliminate this small loss.

Given a graph G, let *OPT(G)* be the maximum independent set of G, we also use *OPT(S)* to denote the maximum independent set in sequence of reductions S. And let *SOL(G)* be a greedy set output by the greedy algorithm we considered on graph G, and use *SOL(S)* to denote the set of the solution in sequence of reductions S, note $|SOL(S)| = |S|$.

The following definition is crucial for the following analysis.

Definition 14. *A* 1*-good reduction is a reduction R executed by greedy algorithm on graph G with maximum independent set, where $|OPT(G) \cap V_R| = 1$. A* 0*-good reduction is a reduction R, where $|OPT(G) \cap V_R| = 0$. Finally, a* bad *reduction is reduction R, where $|OPT(G) \cap V_R| = 2$. Denote the number of* 1*-good reductions as g_1, and the number of* 0*-good reduction as g_0, and the number of bad reduction as b.*

Let us note here that we only consider here reductions R such that $|OPT(G) \cap V_R| \leq 2$, that is, assuming that the root of R has degree at most 2. We do that because if the input graph has minimum degree 3, only the first executed reduction can have $|OPT(G) \cap V_R| = 3$, and all the following executed reductions will have $|OPT(G) \cap V_R| \leq 2$. We will treat the very first such reduction with $|OPT(G) \cap V_R| = 3$ separately in a different way later.

We will also refine the concept of *good* and *bad* reduction later, but the current version of these concepts is sufficient to follow the text here.

When a reduction R is executed, depending of which type R is, 0-good, 1-good or bad, we can locally compute the approximation ratio: let k be the current execution

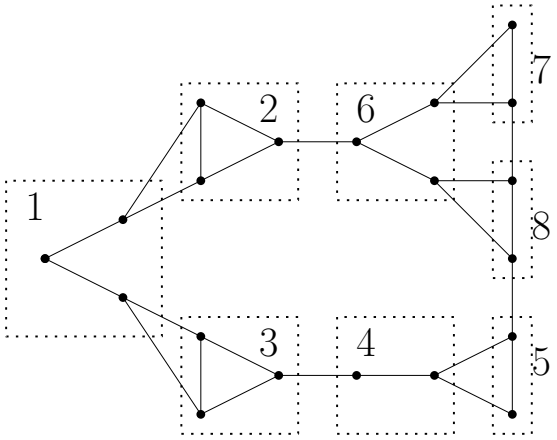


Figure 5.2: Example of reductions during an execution of the greedy algorithm.

step, then, the partial approximation ratio is defined as $\frac{\sum_{i=1}^{k} OPT(R_i)}{\sum_{i=1}^{k} SOL(R_i)}$. Note that if the executed reduction is bad, then 2 will be added to the numerator of this expression and one will be added to the denominator. Therefore, this will potentially increase approximation ratio. However, if the executed reduction is a 1-good reduction, then to both numerator and denominator 1 is added, which implies that the approximation ratio decreases to 1. The 0-good reduction decreases the approximation ratio even further. By this observation, we can introduce notions of *saving* and *payment*. A saving comes from a good reduction, and if there exists a bad reduction, we can say that we can use savings to pay for a bad reduction. These notions will be refined and made more precise in the next chapter.

Claim 3. *For any sequence of reductions S executed by any greedy algorithm on G, if S does not contains any* (2,5)*,* (2,4) *or* (2,6)*-reductions, then* $\frac{|OPT(S)|}{|SOL(S)|} = 1$.

Proof. As we observed, only *(2,5)*, *(2,4)* or *(2,6)*-reduction can be a bad reduction, which means two of the vertices in the reduction belong to the independent set. Thus, if there is no such reduction, $\frac{|OPT(S)|}{|SOL(S)|} = 1$. □

This observation is simple but useful, because it implies that given a degree at most 3 graph G and any greedy algorithm, if the greedy algorithm executes a sequence of reductions $S' = \{R_1, \ldots, R_\ell, R_{\ell+1}\} \subseteq S$, where $\forall R \in \{R_1, \ldots, R_\ell\}$, R is not *(2,5)*, *(2,4)* or *(2,6)* reduction, and $R_{\ell+1}$ is one of them, then, the approximation ratio of the greedy algorithm on graph G is not larger than the approximation ratio of this algorithm on subgraph $G' \subseteq G$, where $G' = (V', E')$, $V' = V \setminus V(\{R_1, \ldots, R_\ell\})$, $E' = E \setminus E(\{R_1, \ldots, R_\ell\})$. This implies that we can always assume that the first

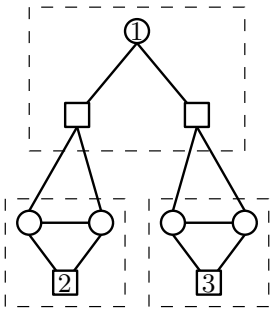


Figure 5.3: Example for Claim 4

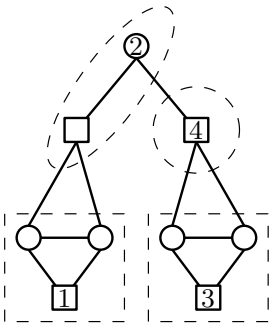


Figure 5.4: Example for Claim 5

reduction in the sequence of reductions executed by the greedy algorithm is one of *(2,5)*, *(2,4)* or *(2,6)*-reduction without any loss in the approximation ratio.

The following observation shows that there exist a universal graph structure, for any greedy algorithm, on which it cannot achieve optimum solution.

Claim 4. *Consider several connected components $C_1, \ldots, C_k$, and a reduction structure R as* (2,6)*, such that the contact edges of R are connected to $C_1, \ldots, C_k$. If the degree of any vertex in $C_1, \ldots, C_k$ is 3, then there exists an instance that any greedy algorithm cannot achieve optimum solution on this graph.*

Proof. Let the independent set vertices of OPT in R be two vertices adjacent to the root vertex. Because the degrees of all other vertices are 3, R must be executed before any other reductions. Therefore, $\frac{|OPT|}{|SOL|} = \frac{2+\sum_{i=1}^{k}|\alpha(G)\cap C_i|}{|SOL(G)|} \geq \frac{2+\sum_{i=1}^{k}|\alpha(G)\cap C_i|}{1+\sum_{i=1}^{k}|\alpha(G)\cap C_i|} > 1$.

Figure 5.3 illustrates such case, where in the first step, the greedy algorithm chooses vertex 1 into the solution and 2, 3 subsequently. Note the size of the maximum independent set in this graph is 4, but the size of solution is 3. □

An important observation from Claim 4 is the following:

Claim 5. *Given a graph G described in Claim 4, where not all vertices in the connected components $C_1, \ldots, C_k$ have degree* 3*, there exists a greedy order to prevent the execution of reduction R.*

Proof. Let v be the vertex in $C_1, \ldots, C_k$ whose degree is not 3, then the existence of such vertex forms a different reduction R'. If $d(v) = 1$, then it will be executed before the execution of R, because whatever greedy order we design, reduction whose root vertex's degree is 1 always has higher priority than reductions whose root degree is 2. If $d(v) = 2$, then greedy algorithm can choose R' rather than R to execute, because

the greedy algorithm can freely choose one of two reductions because their degrees are equal. If greedy algorithm executed R', then after the execution of R', either it removed some vertices of R (then R will never be executed), or it removed some edges of the new connected component C' which contains R. In this new connected component C', because not all vertices have degree 3, it implies an existence of vertex with degree at most 2, which forms a reduction R''. Thus, the same argument can be applied to each iteration of the greedy algorithm. Therefore, we have proved that by the right greedy algorithm, R will never be executed.

Figure 5.4 illustrates such case, where in the first step, the greedy algorithm takes vertex 1, and 2,3,4 subsequently. Note in this execution order, the size of the solution is 4. The key is that in the first step, the greedy chooses vertex 1 rather than 2, thus no *(2,6)* reduction is executed.

□

The importance of Claim 5 is that it implies that if the greedy algorithm is carefully designed, the problematic reductions such as *(2,6)* will never be actually executed in some worst cases. However, since reduction R' might be another bad reduction, such approach is not necessarily useful for solving the problem unless we can find a method to show that the execution of R' rather than R is actually good.

This observation actually implies the order of greedy algorithm. Since for the reduction R which is not *(2,4)*, *(2,5)*, *(2,6)*, we have: $|OPT(G) \cap V(R)| \leq 1$, then the greedy algorithm we expect should prioritise R rather than reductions *(2,4)*, *(2,5)*, *(2,6)*. Therefore, the updated greedy algorithm is:

Algorithm 6 Updated greedy algorithm

Input: Graph $G = (V, E)$
  $U \leftarrow V$
  $S \leftarrow \emptyset$
  while $U \neq \emptyset$ **do**
    choose a reduction R according to the following order in $G_E[U]$:
    1: *(0,0)*, *(1,1)*, *(1,2)*, *(1,3)*, *(2,3)*, *(2,4)-1*, *(2,5)-1*-reduction
    2: *(2,6)*-reduction
    3: *(2,5)-2*-reduction
    4: *(2,4)-2*-reduction
    $U \leftarrow U \setminus (N_G(v^*) \cup v^*)$, where v^* is the root vertex of R.
    $S \leftarrow S \cup \{v^*\}$
  end while
  return S

If there is no further designation, then in this chapter, the default greedy algorithm is updated greedy algorithm of Algorithm 6.

Since Claim 4 implies that there exists a worst case instance to prevent any greedy algorithm from archiving a good solution, the concrete examples are useful for further study. Before we go to the next subsection to study such examples, we present a characterisation of graphs on which any greedy algorithm obtains optimum solution, for arbitrary degree.

Lemma 3. *Given any graph G with arbitrary degree, any greedy algorithm can achieve an optimal solution on G, if for each iteration of the execution, $\forall v \in \arg\min_{u \in V} deg(u)$ $G[N(v) \cup \{v\}]$ forms a clique.*

Proof. For each iteration, let v be the vertex which is chosen, and R is the reduction defined by v. Since $G[N(V) \cup \{v\}]$ forms a clique, the size of the solution of the greedy algorithm on R is 1, and the size of the optimum is at most 1. This implies that $\frac{|OPT(R)|}{|SOL(R)|} \leq 1$. Therefore, $r = \frac{|OPT(G)|}{|SOL(G)|} = \frac{\sum_{R \in S} |OPT(R)|}{\sum_{R \in S} |SOL(R)|} \leq 1$. □

This claim is not restricted to degree at most 3 graphs. We can use Lemma 3 to prove that in the following classes of graphs: complete graphs, trees, co-graphs, and split graphs, the greedy algorithm is optimal.

5.3 Towards computer assisted guide for proof of greedy MIS

As we have shown in Claim 4, in the general case, the greedy algorithm cannot obtain the optimum. Thus, a natural question is what is the best ratio that greedy algorithm can obtain? We note here that Halldorsson [31] has proved that $\frac{5}{4}$ is the best possible approximation ratio achievable by any greedy algorithm for MIS on sub-cubic graphs. Our main goal in the next chapter will be to indeed design such best possible greedy algorithm and prove that it indeed obtains the $\frac{5}{4}$-approximation.

In our study of greedy algorithms for maximum independent set problem on sub-cubic graphs, he following conjecture is crucial for achieving the $\frac{5}{4}$ approximation ratio for the greedy algorithm.

Conjecture 1. *Consider a graph G such that there exists a greedy algorithm with specific greedy order, which will execute a bad (2,6)-reduction firstly. Then the following inequality holds: $5g_0 + g_1 \geq 4$. And if the first executed reduction is a bad (2,5)-reduction, then $5g_0 + g_1 \geq 3$.*

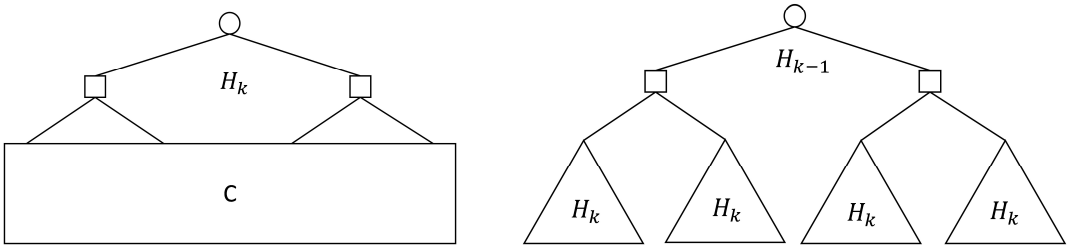


Figure 5.5: Construction of H_k

Observe that if Conjecture 1 is true, then if the first executed reduction is a *(2,6)*-reduction, then the approximation ratio $r = \frac{|OPT|}{|SOL|} = \frac{2+g_1}{1+g_1+g_0}$,

$$\frac{2+g_1}{1+g_1+g_0} = \frac{10+5g_1}{5+5g_1+5g_0} \leq \frac{10+5g_1}{5+5g_1+4-g_1} \leq \frac{10+5g_1}{9+4g_1} \leq \frac{5}{4}. \tag{5.1}$$

Therefore, the approximation ratio of the greedy algorithm on G is at most $\frac{5}{4}$. Then, we will show that Conjecture 1 is necessary for achieving a $\frac{5}{4}$-approximation ratio for the greedy algorithm.

Claim 6. *If Conjecture* 1 *is not true, then there exists a counter-example, that no greedy algorithm can obtain* $\frac{5}{4}$ *approximation ratio for MIS problem on sub-cubic graphs.*

Proof. Consider the graph G illustrated in Figure 5.5. Graph G is constructed in the following way: at the top of the graph is a *(2,6)*-reduction, and we say that it is in layer 1, and its four contact edges are connected to another four *(2,6)*-reductions. We say that these four new *(2,6)*-reductions are in layer 2. For each *(2,6)*-reduction, we do the same. Thus, for layer i, there will be 4^i *(2,6)*-reductions. In the end, say, k-th layer, for each *(2,6)*-reduction R, its four contact edges are connected to some structure C, where after the execution of R, the greedy algorithm can only execute at most 3 1-good reductions.

Suppose that there are k layers of *(2,6)*-reductions. Thus, the number of *(2,6)*-reductions in the last layer is 4^{k-1}, and the total number of *(2,6)*-reductions is $\frac{4^k-1}{3}$. Assume that Conjecture 1 is not true. Observe that if $g_0 \leq 1$, then it is always true, then we can assume that $g_0 = 0$, and $g_1 \leq 3$ for each *(2,6)*-reduction. Denote by $r_{2,6}$ the number of *(2,6)*-reductions in the graph. For any *(2,6)*-reduction R in the last layer S, let g_1^R be the number of 1-good reductions which are executed after R. Therefore, we have that: for any $k \geq 2$,

$$r = \frac{|OPT|}{|SOL|} = \frac{2 \cdot r_{(2,6)} + \sum_{R \in S} g_1^R}{r_{(2,6)} + \sum_{R \in S} g_1^R} \geq \frac{2 \cdot r_{(2,6)} + 3|S|}{r_{(2,6)} + 3|S|} \tag{5.2}$$

$$= \frac{2 \cdot \frac{4^k - 1}{3} + 3 \cdot 4^{k-1}}{\frac{4^k - 1}{3} + 3 \cdot 4^{k-1}} = \frac{8(4^{k-1}) - 2 + 9 \cdot 4^{k-1}}{4(4^{k-1}) - 1 + 9 \cdot 4^{k-1}} > \frac{5}{4}. \tag{5.3}$$

It proves that for any greedy algorithm, the statement of Conjecture 1 must be true, if the approximation ratio of such an algorithm is as good as $\frac{5}{4}$. □

A natural question arises to ask is which greedy algorithm satisfies Conjecture 1? Does the updated greedy algorithm satisfy it?

Conjecture 2. *Consider a graph G such that the updated greedy algorithm of Algorithm 6 will execute a bad* (2,6)*-reduction firstly. Then, the following inequality holds:* $5g_0 + g_1 \geq 4$. *And if the first executed reduction is a bad* (2,5)*-reduction, then* $5g_0 + g_1 \geq 3$.

We have implemented a computer program to examine whether Conjecture 1 is true. If it is not true, then the program will identify what the potential problematic structure looks like. The identification of potential problematic structures played a important role in finding a proof for greedy on MIS problem. As we have shown, if such structure has the property that every vertex has degree exactly 3, then we could find a higher lower bound and show that $\frac{5}{4}$-approximation ratio is impossible to obtain. Or, if not all degrees of the vertices are 3, then we want to know the structure in order to design a specific greedy advice to resolve such problematic cases. Or if the program cannot find any such structure, then it would be promising to theoretically prove that the Conjecture 1 is true.

Some notation is useful to elaborate the detail of the program. Let us say an edge e of a sequence of reductions S is *type* L, if $(N_G(e) \cap V(S)) \in I \wedge (V(S) \setminus N_G(e)) \neq \emptyset$. It means that one endpoint of such edge is not included in $\bigcup_{R \in S} V(R)$, and another endpoint which is included in $\bigcup_{R \in S} V(R)$ are in the independent set.

The idea of how the program works is the following: because of the observation that the savings from the 1-good-reduction or 0-good-reduction have to compensate bad *(2,6)* or bad *(2,5)* reductions, the program will construct a small graph G which consists of limited number of reductions. Essentially, a sequence of reductions $S = \{R_1, \ldots, R_\ell\}$, such that R_1 is bad *(2,6)*-reduction or *(2,5)*-reduction, and the type of reduction $R \in S \setminus R_1$ is not these two kind reductions. However, good *(2,6)*-reduction or good *(2,5)*-reduction is allowed to be present in S. The proposition we want to

examine is that following a bad *(2,6)*-reduction, before the execution of the next bad reduction, the total number of 1-good-reductions g_1 and the number of edges of type L, is equal or greater than 4, and is equal or greater than 3, if they are followed by a bad *(2,5)*-reduction. If there is no feasible combination of $\{R_1, R_2, R_3, R_4\}$, then the feasible combination must contain at least 5 reductions. Thus the sequence of reductions $S = \{R_1, \ldots, R_\ell\}$ we want to examine with $\ell \leq 3$ is enough for our purpose. The program enumerates all possible combinations of $S = \{R_2, \ldots, R_\ell\}$, for $\ell = 1, 2, 3$.

Let us consider the possible combinations S. Now, for each reduction in S, the independent vertex has been assigned, and this determines the type of edges which are connected to vertices. The program executes an operation of Cartesian product to connect vertices between different reductions.

A full example of one possible combination is given to explain how it works: given three reductions R_2, R_3 and R_4, assume that R_2 and R_3 are *(2,4)-1*-reductions and R_4 is a *(2,3)*-reduction. Then for R_1 as a *(2,6)*-reduction, it has four edges which can be connected to other reductions. Similarly, for R_2 and R_3, each of them has one edge which can be connected to R_2, and each of them can be connected to another vertex which belongs to different reduction, in this case, R_2. Therefore, after connecting vertices, the program can compute how many edges with type L remains, and then the program can show whether the proposition we want to examine is true or false.

Algorithm 7 Conjecture testing algorithm

1: Define all possible reductions $\mathcal{S} = (R_1, R_2, R_3, R_4)$, where R_1 is *(2,5)* or *(2,6)* reduction.
2: **for** all combinations of four reductions (R_1, R_2, R_3, R_4) **do**
3: Enumerate all possible combinations between the contact edges and the vertices which can receive such edges. The forbidden situation would be that after connection, both endpoints are in the independent set.
4: Check whether the combination is feasible by examining the independent set property. The feasible combination demands the first reduction executed is *(2,6)* or *(2,5)*.
5: **if** it is feasible **then**
6: Return False.
7: **end if**
8: **end for**
9: Return True.

Surprisingly, the final result after running the program was positive and negative. The negative part was that Conjecture 2 is false. The program examined thousands of combinations and actually generated several instances which violate the claim of

the conjecture. We observed however that all those instances share a single unified property, which will profoundly be reflected in the proof of the greedy algorithm in the next chapter. This observation is: after the execution of R_1 as a bad reduction, $\{R_2, \ldots, R_\ell\}$ forms an isolated odd cycle. The positive part was that Conjecture 1 is not proved false, because in the above situation, there exists alternative choice of degree 2 vertex, and if we choose this alternative vertex, then Conjecture 1 holds. Now, we turn back to consider this problematic structure for the updated greedy algorithm.

Definition 15. *Say a sequence of reductions $S = \{R_1, \ldots, R_k\}$ forms an isolated odd cycle, if R_1 is a* (2,4)*-2 reduction, and R_k is a* (1,1)*-reduction and reductions in $\{R_2, \ldots, R_{k-1}\}$ are* (1,2) *reductions. For simplicity, we call S an isolated odd cycle reduction R. An isolated odd cycle reduction with k vertices R is bad or problematic, if $|OPT(G) \cup V(R)| = \frac{k-1}{2}$.*

Based on this observation, we found another general observation that, informally states, that if any isolated odd cycles are created at last, then in the worst case, they are not enough alone to compensate for the bad reductions. Peculiarly, each of the isolated odd cycle would need one more saving, in order to pay for the bad reductions.

The existence of such examples also implies a mistake in the previous research in [31]. The authors of the paper [31] presented a greedy algorithm for MIS on sub-cubic graphs for which they claimed a 9/7-approximation ratio. We have found an example of instances on which this algorithm does not provide this approximation ratio.

Claim 7. *The example given in Claim 6 shows that the $\frac{9}{7}$-approximation greedy algorithm for MIS in subcubic graphs presented in [31] is incorrect.*

Proof. As we calculated in Claim 6, in the worst case, the algorithm from [31] will have the same sequence of reductions, and will achieve a $\frac{17}{13}$ approximation ratio, which is worse than $\frac{9}{7}$. □

However, if we exclude the existence of any such problematic odd cycle reduction, then the conjecture will finally be proved true.

Corollary 1. *Consider a graph G such that Greedy will execute* (2,6) *reduction firstly. Then the following inequality holds: $5g_0 + g_1 \geq 4$. And if the first executed reduction is* (2,5)*, then $5g_0 + g_1 \geq 3$, if there is no problematic odd-cycle reduction which is executed.*

We do not prove Corollary 1 here, because in the next chapter, we will prove a more general Lemma 4, which will imply this corollary.

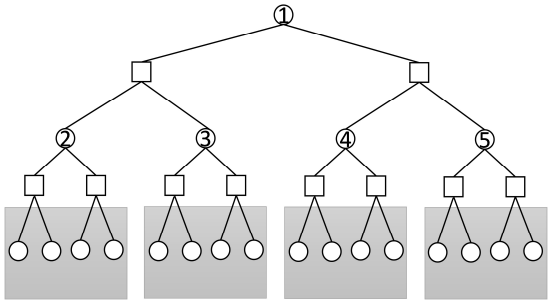


Figure 5.6: Example of bad cycle-reductions.

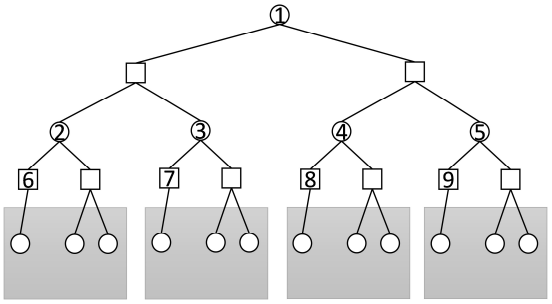


Figure 5.7: Example for bad *(2,5)*-reductions.

5.4 Observation for problematic graph structures

In this section, we present the basic observation for two problematic graph structures identified in the previous section: bad odd cycle and bad *(2,5)*-reduction.

5.4.1 Isolated odd cycle reduction

Mentioned in the previous section, the experimental study provides a counter-example to disprove Conjecture 1: there exist instances such that after the execution of bad *(2,6)* or *(2,5)*-reduction, the following reductions executed by the greedy algorithm are not enough to compensate the loss from the first bad *(2,6)* or *(2,5)*-reduction, to achieve desired approximation ratio. From the point of view of algorithm design, for preventing such situation to occur, this implies that we have to either:

(1) find enough savings from the previously executed reductions which have extra saving to pay for such odd problematic cycle, and that they can uniquely pay for it, or

(2) modify the order of the greedy algorithm, to make the occurrence of the odd problematic cycle impossible due to the execution order of the greedy algorithm.

The savings in (1) must be found in the previously executed reductions. This is because our experimental study shows that the example found implies that if the problematic case happens, then it might has no further reductions (that could pay for it in the future).

Let us consider two examples to show that both methods described in (1) and (2) are necessary for solving the problem.

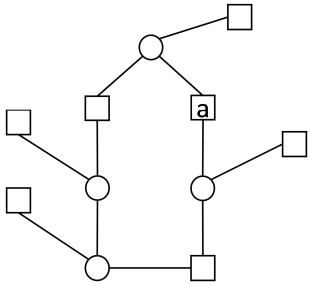


Figure 5.8: Example of problematic cycle reduction

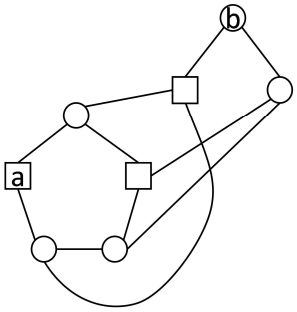


Figure 5.9: Example for odd-problematic cycle

Consider the first example in Figure 5.6, combined with Figure 5.8. We assume there that in each component C (represented as grey rectangles in the figure), there is a problematic structure that we found in our experiment, *i.e.* the four edges which run from independent into non-independent vertices are connected to four non-independent vertices in Figure 5.8. Assume the greedy algorithm chooses vertex 1 and creates four connected components, and then the algorithm chooses vertices $2, 3, 4, 5$ and creates four isolated cycles. Finally, the greedy inevitably executes the following cycle reductions and there is no extra saving from the previous reductions. By this sequence of reductions, the approximation ratio in this case is strictly greater than $\frac{5}{4}$. However, we also observe that when the greedy algorithm decides which vertex, 2 or a, should be chosen, and if the greedy algorithm chooses a, then this is feasible, because b is one of the minimum degree vertices at that moment. Then the sequence of reductions under the new execution order contains no bad reduction. Thus, the total approximation ratio of this example is at most $\frac{5}{4}$. This implies that approach (2) is necessary.

Observe that in the second example illustrated in Figure 5.9, whichever the choice of a or b by the greedy algorithm is, it will create a problematic odd cycle left alone. Therefore, the method in (1) alone might not work anymore, because the odd problematic cycle will be inevitably created. However, we can also observe that the reduction executed first is actually a good reduction, and it can be used to compensate the loss in this problematic odd cycle. Again, it implies that the total approximation ratio of this example is at most $\frac{5}{4}$. This implies on the other hand that the approach (2) is necessary.

Therefore, the above two examples illustrate that it seems quite difficult to tell how the algorithm should proceed, because sometimes the greedy algorithm has to face two different choices. However, it might still be possible to design a sophisticated

greedy algorithm to choose the right minimum degree vertex, which achieves a good enough approximation ratio. In fact, in the next chapter, we prove the existence of such a sophisticated greedy algorithm, and such algorithm is capable of finding the right minimum degree vertex in polynomial times.

Surprisingly, the special rule for advising greedy algorithm to find the right choice is simple. However, we have found this rule after testing many ideas.

5.4.2 Bad *(2,5)*-reduction

A *(2,5)*-reduction there is a further problem. Observe that even the following sequence of reductions does not contain any problematic odd cycles, the inequality $5g_0 + g_1 \geq 3$ is not sufficient to achieve the $\frac{5}{4}$ approximation ratio. See the example illustrated in Figure 5.7. We can assume that each vertex in every components C represented by the grey boxes has degree exactly 3. This means that no alternative choice is possible. Note that if the greedy algorithm firstly chooses vertex 1 as a *(2,6)*-reduction, and subsequently chooses vertices $2, 3, 4, 5$ as *(2,5)*-reductions, then it satisfies the formula, but it is still not enough.

This example shows that we need some ways to avoid the choice of $2, 3, 4, 5$ as *(2,5)*-reductions, but rather choose $6, 7, 8, 9$. Actually, there exists two possible ways to address this situation, either:

(1) the greedy chooses one of the vertices $6, 7, 8, 9$ as *(2,6)*-reduction first, rather than choosing vertex 1 as the first reduction.

(2) after the execution of the *(2,6)*-reduction whose root vertex is a, the greedy chooses $6, 7, 8, 9$ as a *(2,5)*-reduction, rather than choosing one of $2, 3, 4, 5$ as a *(2,5)*-reduction firstly.

Only one of these two solutions suffices for our purpose. But whichever way we decide, it seems that it is not easy to design a proper greedy order such that the greedy algorithm will choose the right reduction. That is because if we adopt the first approach, the greedy algorithm has to decide which *(2,6)*-reduction is the right one, under the circumstance that there might be many different *(2,6)*-reductions in the current graph. And if we adopt the second approach, the greedy algorithm has to decide which of the two alternative *(2,5)*-reductions to choose as the right one. This seems also difficult, because the algorithm might be required to decide this also for the reductions in the future.

However, as we will show in the next chapter, both ways can potentially lead to greedy algorithms whose approximation ratio is $\frac{5}{4}$.

5.4.3 Non-locality of payment

As we observed in the previous two subsections, in some cases, after the execution of a *(2,6)*-reduction, there might be only isolated problematic cycle alone. Or after the execution of a *(2,5)*-reduction, only three 1-good reductions may be left. In this case it would not be enough to get a $\frac{5}{4}$-approximation ratio. We also observed that in some cases, we cannot prevent it to happen by only modifying the greedy algorithm for avoiding such graph structures to occurs. This implies that in some situations we have to show that in the previous reductions, there exist extra saving which can pay. We observe that it is possible that the "location" of such reduction with an extra saving may be very far from the location of the problematic reduction that needs a payment. Such example is difficult to find without the analytic tools that we develop in the next chapter. Thus, we will present this in next Chapter 6. However, this strongly suggests that if we are able to obtain a $\frac{5}{4}$-approximation ratio, then there must exist some non-locality of payment in the analysis of the greedy algorithm.

5.4.4 Conclusion

In this chapter, we introduced basic notions and presented a number of basic observations about how to design a good greedy advice. We identified two kinds of problematic cases, and gave a basic idea of how to resolve them. In the next chapter, we will see that these observations will actually imply the graph structures which are "problematic", and will help us to essentially characterise graphs which are short of savings.

Chapter 6

Towards Ultimate Greedy for MIS in sub-cubic graphs

We present the result of our study on the greedy algorithm for the maximum independent set problem on sub-cubic graphs in this chapter. We achieve a greedy algorithm with an approximation ratio of $\frac{4}{3}$ on sub-cubic graphs in $O(n^2)$ time.

This chapter is devoted to present the entire proof. We begin in Section 6.1 to present formal arguments for the payment scheme, that is, for how to "pay" for bad reductions by using the reductions with savings (1- and 0-good reductions). Next, we introduce an useful graph computation during the execution of the greedy algorithm, namely by using an extended graph and extended reductions. Finally, we give an argument for dealing with problematic structures and obtain the main result in Section 6.3. After that, we discuss a possibility to obtain an analysis for the *ultimate greedy algorithm* in Section 6.3.4, to achieve an approximation ratio of $\frac{5}{4}$ in linear time complexity.

6.1 Payment scheme

6.1.1 Definitions

At the beginning, we start from the updated greedy algorithm (Algorithm 5), and finally present the ultimate greedy algorithm. Given a graph with degree at most 3, observe that when the greedy algorithm is working, in each iteration i, one of the vertices in the remaining graph is taken into the solution set S, and its adjacent vertices and edges are removed from the remaining graph. Finally, after the k-th iteration, the graph will

be empty and the solution of greedy will be $S = \{v_1, \cdots, v_k\}$.

Let G_i be the remaining graph after the execution of step i of the greedy algorithm (in step i execution the vertex v_i and its neighboring vertices are removed). Note for the k-th iteration of the greedy algorithm, G_{k-1} is the graph in which v_k is still present, and G_k is the graph in which vertex v_k is removed. Therefore, we talk about a sequence of graphs created by an execution of the greedy algorithm: $G \supset G_1 \supset G_2 \supset \ldots \supset G_k = \emptyset$.

For convenience, given an independent set I in G, we call a vertex v *black* vertex if $v \in I$ and a *white* vertex otherwise. Also, given a subset V' of vertices in the graph G, V' is called black (white, respectively) if all vertices in V' are black (white, respectively).

As we said in the previous chapter, in each iteration, the greedy algorithm executes a *reduction*, which was explained intuitively in the previous chapter. Now, we begin to present the precise definition of a reduction. We define a reduction R_i executed by the greedy algorithm in iteration i as a 2-tuple (we will omit subscript i of R_i if the context is clear), $R = \{V_R, E_R\}$, where $V_R = V(G_{k-1}) \setminus V(G_k)$. There exists a vertex $v^* \in V_R$, which we call a *root* vertex. The root vertex v^* is the vertex which is taken into the solution of the algorithm when reduction R_i is executed. The set of edges of reduction R, is $E_R = E^R_{past} \cup E^R_{self} \cup E^R_{contact}$, where $E^R_{self} = E(\{v^*\} \cup N_{(G_{k-1} \setminus G_k)}(v^*))$, $E^R_{contact} = (E(G_{k-1}) \setminus E(G_k)) \setminus E_{self}$, and $E^R_{past} = E(N_G(V_R)) \setminus E(G_{k-1})$. Whenever R is clear from the context, we omit R from $E^R_{past}, E^R_{self}, E^R_{contact}$ and write just $E_{past}, E_{self}, E_{contact}$. We also call *contact vertices* the vertices which are incident to $E_{contact}$, but are not in V_R. These notions will be illustrated in Figure 6.1.

Furthermore, we define two types of edges: *loan* and *debt* edges. The loan edge e of a reduction R is the edge such that $e \in E_{contact}$ and $N(e) \setminus V_R$ is white; and a *debt* edge is the edge such that $e \in E_{past}$ and $N(e) \cap V_R$ is white. Let us use $E_{loan}(R)$ to denote the total number of loan edges of reduction R, and $E_{debt}(R)$ to denote the total number of debt edges. Note for any loan edge from a reduction, it is identical to one debt edge from some other reduction. Therefore, the total number of loan edges of all reductions is equal to the total number of debts edges of all reductions.

In Figure 6.1, we illustrate two reductions giving two examples. Let a *box* vertex in the figures represent the vertex which is black. V_R are vertices which are in the dotted rectangle. The contact vertices are vertices which are in layer L_c. $E_{contact}$ are edges between the vertices of layer L_m and L_c, and E_{past} are edges between the vertices of V_R and vertices of layer L_p. And E_{self} are edges which are completely contained in the dotted rectangle. The loan edges in the left reduction are $\{(2,4),(3,5)\}$, because vertices 4,5 in layer L_e are white. The same argument applies to the right reduction, showing that $\{(2,4),(2,5),(3,6),(3,7)\}$ are loan edges. The debt edge in the left reduction is

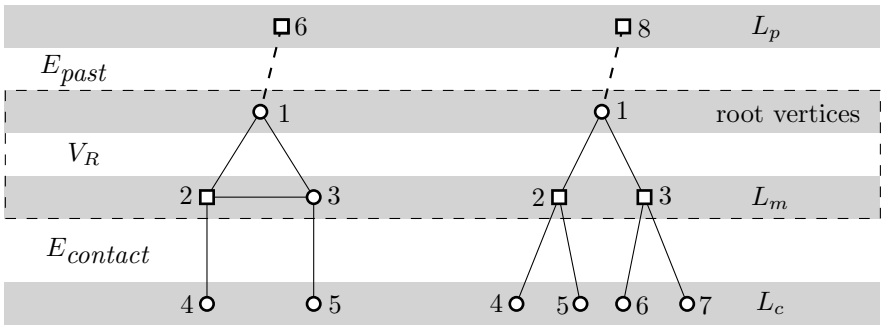


Figure 6.1: example of reductions

$(6, 1)$, because the vertex in V_R is white, and debt edge in the right reduction is $(8, 1)$.

The crucial concept throughout the following thesis is a potential function of a reduction. The potential function helps us to clearly capture the complex relation between the locality of saving and losing of single reduction. It also helps to capture the global properties if such a reduction is executed, what will be the affects it creates in the future execution of the greedy algorithm. It also reflects the way of how the already executed reductions in the past influence the future.

We define a potential function $\Phi(R)$ of a reduction R as

$$\Phi(R) = 5|SOL(R)| - 4|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|, \tag{6.1}$$

where $SOL(R) = \{v \in V_R | v \text{ is root vertex}\}$, thus $|SOL(R)| = 1$ always and $OPT(R) = \{v \in V_R | v \text{ is black}\}$. $E_{loan}(R)$ and $E_{debt}(R)$ is the set of loan edges and debt edges of R.

For instance, for the left hand side reduction in Figure 6.1, namely a good *(2,5)-1* reduction, R, its potential function value is $\Phi(R) = 5-4+2-1 = 2$, while for the right hand side reduction, namely a bad *(2,6)*-reduction, R', its potential function value is $\Phi(R') = 5 - 8 + 4 - 1 = 0$.

6.1.2 Ideas

Let us first explain the intuition behind the arguments of a payment scheme that uses the potential function defined above. Let us assume I is a fixed maximum independent set of graph G. When the greedy algorithm executes a reduction, it will take one of the vertices into the solution and remove adjacent vertices to it. At this moment, the size of the solution is increased by 1, and depending on how many vertices in $V_R \cap I$ there are, the size of optimum in this reduction or particular "sub-graph" is increased by 1,

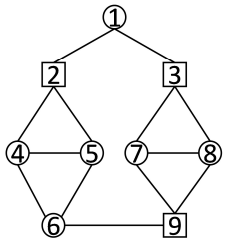


Figure 6.2: Structure with bad *(2,6)*-reduction

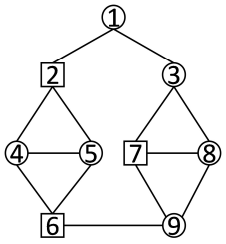


Figure 6.3: Structure with good *(2,6)*-reduction

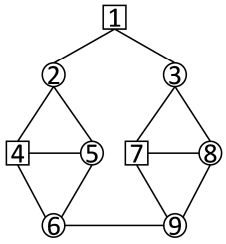


Figure 6.4: Structure with good *(2,6)*-reduction

by 2 or by 3. It is easy to see that given a reduction R, if the increase in the size of its solution and optimum are the same, then this reduction is fine from the point of view of the local approximation. Such a reduction is also better than other reductions whose size of the solution is strictly smaller than that of the optimum, because potentially it will lead to an approximation ratio of 2 or worse. Unfortunately, for any greedy algorithm, the size of the solution is only depending on the number of executions of reductions, particularly, on the number of root vertices of the reductions. This means that, each reduction can only contribute 1 to the size of the solution, but it is possible to contribute 2 (or 3) to the size of the optimum. For instance, consider a bad *(2,6)*-reduction (illustrated in the Figure 6.2). Because two of its vertices belong to I, thus, locally, the approximation ratio is 2 for this reduction. Even though it is possible, it would be extremely difficult to argue that the greedy algorithm will never execute such bad *(2,6)*-reduction. Therefore, we need some method to address such kind of reductions.

The crucial observation is that whenever such kind of bad *(2,6)*-reduction occurs, the following type of reductions in some sense are determined, due to the fact that both vertices $2, 3$ are in the independent set I. An example will illustrate it.

Observe that in Figure 6.2, vertices $\{2, 3, 9\}$ belong to the provided maximum independent set I. The greedy algorithm will execute the bad *(2,6)*-reduction R_1 (formed by vertices $\{1, 2, 3\}$), and subsequently two *(2,3)*-reductions (the first is formed by vertices $\{4, 5, 6\}$, and the second is formed by $\{6, 7, 8\}$). Now, we can see that the first *(2,3)*-reduction R_2 formed by $\{4, 5, 6\}$ contains no vertex in I, and this implies that the size of the solution of R_1 is increased by 1, while the size of the optimum does not increase. Moreover, because we assume that the vertices $\{2, 3\}$ belong to I, then whatever is the assignment of independent set vertices among the vertices $\{4, 5, 6, 7, 8, 9\}$, there is only one vertex which is feasible to belong to I. This implies that only one of the two reductions R_2 and R_3 contains one independent set vertex. In contrast, illustrated

in Figure 6.3, because R_1 is a *(2,6)*-reduction which has single black vertex, then it is possible that two vertices in $\{4, 5, 6, 7, 8, 9\}$ belong to I. In the example, we assume that they are vertices 6 and 7. Thus, though both R_2 and R_3 contain an independent set vertex and consequently both increase the size of the solution and optimum by 1, R_1 also increases both sizes by 1. Therefore, it achieves in this example the optimal solution. Furthermore, we can observe that in Figure 6.4, though the independent vertex in R_1 is 1 rather than 2, the same property is preserved.

What we can learn from these observations is that: the type of reduction determines the type of the following reductions, and the reason for that is that the type of vertices determines the type of the adjacent vertices. More precisely, given a reduction, through the type of its contact edges, it determines the type of the following reduction, and this is why we introduce *loan* and *debt* edges. For instance, if the end vertex of a contact edge is white then such an edge is called a loan edge. Let us explain this in a more detail now with an example.

We observe that each loan edge "predicts" the existence of a particular kind of reduction in the future, while each debt edge "recalls" the existence of another particular kind of reduction in the past. For example, in Figure 6.3, because R_1 has two loan edges $(2, 4), (2, 5)$, it implies that in the worst case, in the future, there are two good reductions which will be executed by the greedy algorithm. Even though these two reductions are not executed immediately after the execution of R_1, there will need to be executed "somewhere". Moreover, reduction R_2 also has one loan edge $(6, 9)$, so it also implies that there will be one good reduction in the future, and, actually, R_3 is this reduction.

In fact, the story of "prediction" and "recall" is even more complicated, and we will explain it later. Let us recall the potential function $\Phi(R)$ defined above.

Observe that for any sequence of reductions S executed by the greedy on graph G,

$$\Phi(S) = \sum_{R \in S} \Phi(R) = \sum_{R \in S} (5|SOL(R)| - 4|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|) \quad (6.2)$$

$$= \sum_{R \in S} (5|SOL(R)| - 4|OPT(R)|), \quad (6.3)$$

where the last equality is true because $|E_{loan}(S)| = |E_{debt}(S)|$.

Therefore, if we can prove for any S executed on a graph G such that $\Phi(S) \geq 0$, $(5|SOL(S)| - 4|OPT(S)|) = \Phi(S) \geq 0$, then $\frac{OPT(S)}{SOL(S)} \leq \frac{5}{4}$. This proves the $\frac{5}{4}$-approximation ratio.

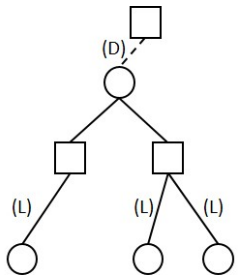


Figure 6.5: Bad *(2,5)*-reduction

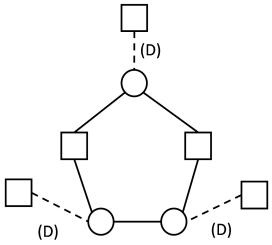


Figure 6.6: Bad 5-cycle-reduction

Let us try to compute the potential function on the previous example illustrated in Figure 6.2: the execution of greedy algorithm leads to the sequence of reductions $S = \{R_1, R_2, R_3\}$. $\Phi(R_1) = 5 - 8 + 4 - 0 = 1$, $\Phi(R_2) = 5 - 0 + 0 - 3 = 2$ and $\Phi(R_3) = 5 - 4 + 0 - 2 = -1$, then $\Phi(S) = 2$. And we know in this case, the greedy algorithm achieves an optimal solution.

Therefore, the crucial part is to show that $\Phi(S) \geq 0$. Fortunately, for almost every reduction $R \in S$, $\Phi(R) \geq 0$, and this fact will be formally proved in the next section. Thus, the sum of the potential values in the sequence of reductions is also larger than 0. Unfortunately, not every reduction R has non-negative value. The following two reductions violate this property. It is easy to check that both reductions R_1 and R_2 in Figure 6.5 and 6.6 have $\Phi(R_1) = \Phi(R_2) = -1$. To deal with such reductions is the main difficulty, and we will address it in Section 6.3.

6.1.3 Value of potential function of payment scheme

In this subsection, we will compute the potential value for all kinds of reductions. We only consider the reductions whose root vertex's degree is at most 2. Firstly, we classify reductions depending on their graph structures. Initially, we have: *(0,0)*, *(1,1)*,*(1,2)*, *(1,3)*,*(2,3)*,*(2,4)-1*,*(2,4)-2*,*(2,5)-1*,*(2,5)-2*,*(2,6)*. However, because the algorithm used here is the updated greedy algorithm of Algorithm 6, then *(2,4)-2* reduction is executed only if there exists a collection of isolated cycles. Then we replace the *(2,4)-2* reduction by an isolated cycle reduction – see Definition 15.

A *bad (2,5)-reduction* R is defined as a *(2,5)*-reduction with each vertex v adjacent to the root vertex of R to be black, and such that $\exists e \in E_{past}$ – e is a debt edge. A *bad (2,3)-reduction* R is defined as a *(2,3)*-reduction such that $\exists v \in V_R$, v is black and $\forall v' \in N_R(v)$ v' is white, and $\forall e \in E^R_{past}$ – if $(N_R(e) \cup V_R)$ is white, then $(N_R(e) \setminus V_R)$ is black. Similarly, a *bad isolated cycle reduction* with length of k is an isolated cycle reduction with length k such that k is odd and there are exactly $\frac{k-1}{2}$ black vertices,

and $\forall e \in E^R_{past}$, if $(N_R(e) \cup V_R)$ is white, then $(N_R(e) \setminus V_R)$ is black. And, finally, a *bad (1,1)-reduction* R is defined as a *(1,1)*-reduction such that $\exists v \in V_R$, v is black and $\forall e \in E^R_{past}$, if $(N_R(e) \cup V_R)$ is white, then $(N_R(e) \setminus V_R)$ is black.

Claim 8. *For any reduction R, if R is a bad* (2,5)*-reduction, bad isolated cycle reduction, bad* (2,3)*-reduction of bad* (1,1)*-reduction, then* $\Phi(R) = -1$.

Proof. Observe that a *(2,5)*-reduction R has three loan edges and one debt edge, and two black vertices in $V(R)$. Then the potential is $\Phi(R) = 5 - 8 + 3 - 1 = -1$. A bad *(2,3)*-reduction R has two debt edges, and one black vertex in V_R. Then the potential is $\Phi(R) = 5 - 4 - 2 = -1$. Finally, a bad *(1,1)*-reduction R has two debt edges, and one black vertex in V_R. Ans so the potential is $\Phi(R) = 5 - 4 - 2 = -1$. This proves the claim. □

Observe that for all the remaining reductions R, we have $|E_{contact}| > 0$, which means that there must exist at least one contact edge.

Lemma 4. *For any reduction R, if R is not of of the above four types of reductions, then* $\Phi(R) \geq 0$.

Proof. To prove this claim in a simple way, we only consider reductions' structure on V_R and E_{self}. Thus they can be further classified into four groups: isolated vertex reduction, single edge reduction, triangle reduction and branching reduction. Denote by v^* the root vertex of the reduction we consider.

1. For an isolated vertex reduction R, $V_R = \{v^*\}$ and $E_{self} = \emptyset$. Note $|E_{past}| \leq 3$. If v^* is black, then $|\{e \in E_{past} | e \text{ is debt edge}\}| = 0$, then $\Phi(R) = 5 - 4 + 0 - 0 = 1$. If v is white, then $|\{e \in E_{past} | e \text{ is debt edge}\}| \leq 3$, and then $\Phi(R) \geq 5 - 0 + 3 = 2$.

2. For a single edge reduction R, $V_R = \{v^*, v_2\}$ and $E_{self} = \{(v^*, v_2)\}$. Note $|E_{past}| + |E_{contact}| \leq 4$ and $|E_{past}| \leq 3$. If v^* is black, then $|E_{debt}| \leq 1$, because two of them cannot be debt edges. Thus, $\Phi(R) \geq 5 - 4 + 0 - 1 = 0$. If $N(v)$ is black, then $|E_{debt}| \leq 2$ and $|E_{loan}| \geq 1$, and thus, $\Phi(R) \geq 5 - 4 + 1 - 2 = 0$. If v and $N(v)$ are white, it is obvious that $\Phi(R) \geq 0$.

3. For a triangle reduction R, $V_R = \{v^*, v_2, v_3\}$ and $E_{self} = \{(v^*, v_2), (v^*, v_3), (v_2, v_3)\}$. Note $|E_{past}| + |E_{contact}| \leq 3$ and $E_{past} \leq 2$. If there is one of the vertices v such that v is black, then $|E_{debt}| \leq 2$. If $|E_{debt}| = 2$, then $|E_{loan} = 1|$, and in such a case, $\Phi(R) = 5 - 4 - 2 + 1 = 0$. If $|E_{debt}| \leq 1$, then also $\Phi(R) \geq 5 - 4 - 0 + 1 = 0$. If none of the vertices v is black, then $\Phi(R) \geq 5 - 0 - 0 + 3 = 2$.

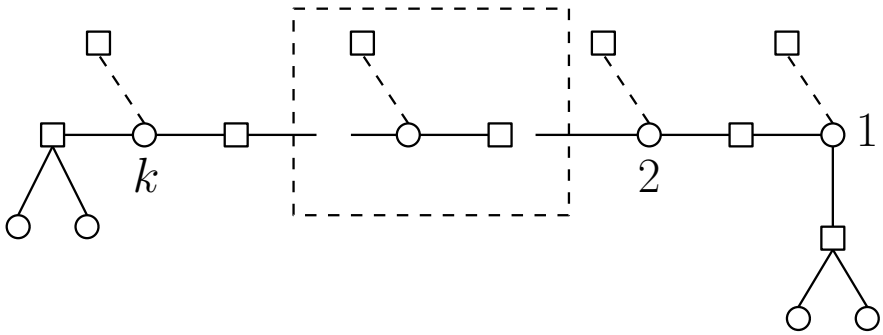


Figure 6.7: Example of an extended reduction.

4. For a branching reduction R, $V_R = \{v^*, v_2, v_3\}$ and $E_{self} = \{(v^*, v_2), (v^*, v_3)\}$. Note $|E_{past}| + |E_{contact}| \leq 5$ and $|E_{past}| \leq 2$. If v^* is black, then $|E_{debt}| \leq 1$, and then we have $\Phi(R) \geq 5 - 4 + 0 - 1 = 0$. If v^* is white, and at least one of two adjacent vertices is black, then we have $|E_{loan}| \geq 1$, and $|E_{debt}| \leq 2$, thus $\Phi(R) \geq 5 - 4 + 1 - 2 = 0$.

□

Therefore, by Claim 4, we proved Corollary 1 from the previous chapter. This completely reflects the observations that we have made in our experimental study.

Remark. *The fact that the potential value of a bad* 5*-odd cycle reduction is* -1 *also reflects the fact that in our experimental study in the previous chapter, it is possible to construct an instance where the approximation ratio is strictly greater than* $\frac{5}{4}$.

6.2 Extended reductions

In this section, we consider an useful graph transformation during the execution of the Greedy: basically, in each iteration, we locally modify the graph into an *extended* graph, and the Greedy can execute *extended* reductions of the extended graph.

The motivation for that is derived from the following observation. We consider a part of the remaining graph, which is illustrated in Figure 6.7. We assume that greedy will subsequently execute reductions $R_1, R_2, \ldots, R_k$, (in the example, the dotted box are repetitions of a *(1,1)*-reduction). Note that R_1 is a bad *(2,5)*-reduction, and as we show in the previous section, $\Phi(R_1) = -1$. Observe that for this particular sub-sequence of reductions $S = \{R_1, R_2, \ldots, R_k\}$, $\Phi(S) = 0$, because even though $\Phi(R_1) = -1$, $\Phi(R_i) = 0, \forall i, 1 < i < k$ and $\Phi(R_k) = 1$. Besides that, if we look at these reductions in

a unified way, we can see that:

$$E_{contact}(S) = \bigcup_{i=1}^{k} E_{contact}(R_i) \setminus \bigcup_{i=1}^{k} E_{past}(R_i) = \bigcup_{i=1}^{k} E_{loan}(R_i) \setminus \bigcup_{i=1}^{k} E_{debt}(R_i) = E_{loan}(S) \tag{6.4}$$

This implies that the number of contact edges in S is equal to the number of loan edges in S. Moreover, we have: $|E_{loan}(S)| = 4$.

Let us look at the potential function of S more closely.

$$\begin{aligned}
|E_{debt}(S)| &= |E_{loan}(S)| + \sum_{i=1}^{k}(5|SOL(R_i)| - 4|OPT(R_i)|) \\
&= 4 + (\sum_{i=2}^{k} 5|SOL(R_i)| - \sum_{i=2}^{k} 4|OPT(R_i)|) + 5|SOL(R_1)| - 4|OPT(R_1)| \\
&= (\sum_{i=2}^{k} 5|SOL(R_i)| - \sum_{i=2}^{k} 4|OPT(R_i)|) + 1 \\
|E_{debt}(S)| &- (\sum_{i=2}^{k} 5|SOL(R_i)| - \sum_{i=2}^{k} 4|OPT(R_i)|) = 1
\end{aligned}$$

We can see that there might be k debt edges in S, however, according to the interpretation of debt edges, only one of them is "real" (or "unpaid"), which means that the other $k-1$ debt edges is paid by reductions from S itself.

Therefore, by $|E'_{debt}(S)| = 1$ and $|E_{loan}(S)| = 4$, we are able to say that for the sequence of reductions S, its performance according to the potential function is equivalent to that of a bad *(2,6)*-reduction. That is because there are four contact edges and all of them are loan edges, and only one of debt edges really affects the potential function. This implies that in some sense, we might contract S into a single reduction.

6.2.1 Definition of extended reductions

Define vertices of the extended graph G_E as extended-vertices or nodes, and its edges as extended-edges. For the original input graph G these are just referred to as vertices and edges, respectively.

Let $G = (V, E)$ be the (current) original graph, then the extended graph $G_E = (V_E, E_E, L)$ is defined informally as follows:

$$V_E = \{v \in V : d_G(v) \in \{0, 1, 3\}\}$$

$E_E = \{(u, v) : u, v \in V_E$ such that there exists a $u - v -$ path in G with consecutive degree-2 vertices in $G\}$.

We now have the following formal definition of the extended graph.

Definition 16. *An extended graph $G_E = (V_E, E_E, L)$ is an undirected multi-graph with labeled edges defined for a given input graph $G = (V, E)$ as follows. Let the following two functions $\{f, g\}$ be bijective, such that:*
$f : V' \to V_E$, where $V' = \{v \in V \ | d_G(v) \in \{0, 1, 3\}\}$,
$g : P \to E_E$, where P is the set of path of degree-2 vertices in G, and p is the element of P, that is,
$p = \{(v_0, \ldots, v_k) \in V^{k+1} | v_0 \in V' \wedge v_k \in V', \forall i = 0, 1, \ldots, k-2 : (v_i, v_{i+1}) \in E, d_G(v_{i+1}) = 2, (v_{k-1}, v_k) \in E\}$,
$L : E_E \to \{0, 1\}$, where $L(e) = 1$, if the length of path $g^{-1}(e)$ in G is odd and $L(e) = 0$, if the length of path $g^{-1}(e)$ in G is even.

By using functions $\{f, g, L\}$ defined above, given any graph G, we can convert it into its corresponding extended graph G_E. Therefore, after each iteration i of the execution, we convert graph G_i into its extended graph.

Claim 9. *In each iteration of the greedy execution, we can update the extended graph from the previous extended graph in $O(1)$ time.*

Proof. Because in each iteration of the greedy algorithm, only one original reduction is removed, then only a constant number of edges and vertices are removed, and therefore, we can update the extended graph in constant time. □

Then, based on the extended graph, we define extended reductions. The illustration is present in the Appendix in Figure A.1. We formulate now a new greedy algorithm, see Algorithm 8, on the extended graph, by defining a specific order of executing extended reductions. The detailed explanation of how this algorithm executes extended reductions is described just below Definition 18.

Definition 17. *An execution of the updated greedy algorithm of Algorithm 8 on the extended graph takes* single edge non-branching*,* single edge branching*,* loop*,* even-backbone *and* odd-backbone *reduction, where these reductions are defined below.*

1. *Single edge non-branching reduction R_E: $R_E = \{V_R, E_R\}$, where $V_R = \{u\}$, $d(u) = 1$ in the extended graph G_E^{k-1}, $E_R = \{(u, N(u))\}$, and $L((u, N(u))) = 0$.*

2. *Single edge branching reduction R:* $V_R = \{u_1, u_2\}$, *where* $d_{G_E^{k-1}}(u_1) = 1$ *and* $d(u_2) = 3$ *in* G_E^{k-1}, $E_R = \{(u_1, u_2)\}$ *and* $L((u_1, u_2)) = 1$.

3. *Loop reduction R:* $V_R = \{u\}$, *where* $d_{G_E^{k-1}}(u) = 3$, *and* $E_R = \{(u, u), N(u)\}$.

4. *Even-backbone reduction R:* $V_R = \{u_1, u_2\}$, *where* $d_{G_E^{k-1}}(u_1) = d_{G_E^{k-1}}(u_2) = 3$, $E_R = \{(u_1, u_2)\}$, *and* $L((u_1, u_2)) = 0$.

5. *Odd-backbone reduction R:* $V_R = \{u\}$, *where* $d_{G_E^{k-1}}(u) = 3$, $E_R = \{(u_1, u_2)\}$, *and* $L((u_1, u_2)) = 0$.

Remark. *We will use the terms "basic" and "extended" to distinguish the reductions in the original graph and in the extended graph, respectively.*

Now, we need to explore the connection between a reduction and an extended reduction. For convenience, assume that an extended reduction is executed at k-th iteration. We have $V_R = V(G_E^{k-1}) \setminus V(G_E^k)$, $E_R = E(G_E^{k-1}) \setminus E(G_E^k)$.

Definition 18. *We say that the execution of an extended reduction R in G_E is equivalent to the execution of a sequence of (basic) reductions S, if*

$$f^{-1}[V(G_E^1) \setminus V(G_E^2)] \cup [V(g^{-1}(E(G_E^1))) \setminus V(g^{-1}(E(G_E^1)))] = V(G^1) \setminus V(G^2),$$

Where G_1 and G_2 is the graph before and after the execution of the sequence of reductions S, respectively, and G_E^1 and G_E^2 is the extended graph before and after the execution of R, respectively.

How extended reductions are executed: Given a graph G and its extended graph G_E, for an extended edge $e \in E_E$, we say that the extended edge e is removed by an extended reduction R, if after the execution of R, the path $g^{-1}(e)$ in G is removed by a sequence of reductions $S_{\mathcal{R}}$.

The following are extended reductions R and their sequences of (basic) reductions S_R.

1. Single edge non-branching reduction, $R_E = \{R_1, \ldots, R_k\}$, $\forall R \in R_E$, R is a *(1,2)*-reduction. Moreover, $E_{contact}^{R_i} \cap E_{past}^{R_j} \neq \emptyset$ for every $i = j - 1$.

2. Single edge branching reduction consists of a series of reductions $\{R_1, \ldots, R_k\}$ in which R_1 to R_{k-1} are *(1,2)*-reductions and R_k is a *(1,3)*-reduction. Moreover $E_{contact}^{R_i} \cap E_{past}^{R_j} \neq \emptyset$ for every $i = j - 1$.

3. Loop reduction consists of a series of reductions $R_1, \ldots, R_k$, in which R_1 is a *(2,5)* reduction, R_2 to R_{k-1} are *(1,2)*-reductions, and R_k is a *(1,1)*-reduction, where $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$, and $E^{R_1}_{contact} \cap E^{R_k}_{past} \neq \emptyset$.

4. Even-backbone reduction either is a *(2,6)*-reduction, or it consists of a series of reductions $R_1, \ldots, R_k$, where R_1 is a *(2,5)*-reduction, R_1 to R_{k-1} are *(1,2)*-reductions, and R_k is a *(1,3)*-reduction. Moreover, $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

5. Odd-backbone reduction consists of a series of reductions $R_1, \ldots, R_k$, where R_1 is a *(2,5)*-reduction, R_1 to R_k are *(1,2)*-reductions. Moreover, $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

We see that every extended reduction, say R, consists of a sequence of basic reductions, say $\{R_1, \ldots R_k\}$. The notions of E_{past}, E_{self} and $E_{contact}$ defined previously for basic reductions naturally extend to extended reductions. Namely, we define $E^R_{past} = \bigcup_{i=1}^k E^{R_i}_{past}$, and analogously for E^R_{self} and $E^R_{contact}$.

Then, we need to compute the extended graph's potential value. Because of the following analysis, the value of the potential needs to be very accurate, we will classify the reductions into two categories: good and bad.

Also, we will use following notation.

Definition 19. *Single edge non-branching, single edges branching, loop and even-backbone reductions are called* high-priority reductions.

Furthermore, we categorise reductions into two groups, *mixed* or *non-mixed* reductions. Firstly, we present them in a basic reduction form (that is, those in Figure 5.1), then we present them in an extended reduction form. The interpretation of a non-mixed and mixed reduction is that all contact edges of the former have the same type of endpoints (black or white), and such claim is false if the reduction is mixed.

Definition 20. *A reduction R is non-mixed, if $\forall v \in N(E^R_{\text{contact}}) \setminus V(R)$, v is black or $\forall v \in N(E^R_{\text{contact}}) \setminus V(R)$, v is white. And a reduction R is mixed, if it is not non-mixed.*

Definition 21. *The set of contact edges of an extended reduction $R_E = \{R_1, \ldots, R_k\}$ is defined as $E^{R_E}_{\text{contact}} = (E^{R_1}_{\text{contact}} \cup E^{R_k}_{\text{contact}}) \setminus \bigcup_{i=1}^k E^{R_i}_{\text{past}}$. An extended reduction R_E is non-mixed, if $\forall v \in (N(E^{R_E}_{\text{contact}}) \setminus \bigcup_{i=1}^k V(R_i))$, v is black or $\forall v \in (N(E^{R_E}_{\text{contact}}) \setminus \bigcup_{i=1}^k V(R_i))$, v is white. And a reduction R is mixed, if it is not non-mixed.*

6.2.2 Value of potential function of extended reduction

In this subsection, we compute the precise value of the potential function of the extended reductions. The crucial property for an extended reduction which is mixed is that its potential value is strictly larger than 0. It implies that they are the reductions which have extra saving to pay for other reductions which lack payment, *i.e.* their potential value is negative.

Claim 10. *For an even-backbone reduction R, if R is a mixed reduction, then $\Phi(R) \geq 2$.*

Proof. Because an even-backbone reduction R has an even number of edges, let us say k, in its backbone $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$, there are $k+1$ vertices in backbone p. And because R is a mixed reduction, then only one vertex of the two endpoints of p is in I by definition. Assume the v_1 is black, and v_{k+1} is white, then v_2 is white. Now, we consider the sub-backbone $p' = \{(v_3, v_4), \ldots, (v_{k-1}, v_k)\}$ of p, and denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of sub-backbone p' is odd, and by the property of an independent set in odd length path, we have: $j \leq \ell$. By adding v_1, v_2 and v_{k+1}, we have: $j + \ell = k - 2$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j+1$, $|E^R_{loan}| = 2$ and $|E^R_{debt}| = \ell+1$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4(j+1) + 2 - (\ell+1) = \frac{5(j+\ell+2)}{2} - 4j - 4 + 2 - \ell - 1 = \frac{3}{2}\ell - \frac{3}{2}j + 5 - 4 + 2 - 1 \geq 2$. □

Claim 11. *For an odd-backbone reduction R, if R is a mixed reduction, then $\Phi(R) \geq 1$.*

Proof. Because an odd-backbone reduction R has an even number of edges (it should not be confusing with the fact that an odd-backbone reduction's backbone is an odd length path), let us say k, in its backbone $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$, there are $k+1$ vertices in backbone p. And because R is a mixed reduction, then only one vertex of the two endpoints of p is in I by definition. Let us assume that $d_R(v_1) = 2$ and $d_R(v_{k+1}) = 3$, then we have two cases.

Case 1, let us assume v_1 is black, and v_{k+1} is white, then v_2 is white. Now, we consider the sub-backbone $p' = \{(v_3, v_4), \ldots, (v_{k-1}, v_k)\}$ of p, and denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of sub-backbone p' is odd, and by the property of the independent set in an odd length path, we have: $j \leq \ell$. By adding v_1, v_2 and v_{k+1}, we have: $j + \ell = k - 2$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j+1$, $|E^R_{loan}| = 1$ and $|E^R_{debt}| = \ell+1$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4(j+1) + 1 - \ell - 1 = \frac{5(j+\ell+2)}{2} - 4j - 4 + 1 - \ell - 1 = \frac{3}{2}\ell - \frac{3}{2}j + 5 - 4 + 1 - 1 \geq 1$.

Case 2, let us assume v_1 is white, and v_{k+1} is black, then v_k is white. We consider the sub-backbone $p' = \{(v_2, v_3), \ldots, (v_{k-2}, v_{k-1})\}$ of p. Note that the length of p' is odd, and by the property of independent set in odd length path, we have: $j \leq \ell$. By adding v_1, v_2 and v_{k+1}, we have: $j + \ell = k - 2$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j+1$, $|E^R_{loan}| = 2$ and $|E^R_{debt}| = \ell+2$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4(j+1) + 2 - (\ell+2) = \frac{5(j+\ell+2)}{2} - 4j - 4 + 2 - \ell - 2 = \frac{3}{2}\ell - \frac{3}{2}j + 5 - 4 + 2 - 2 \geq 1$. □

Claim 12. *For an even-backbone reduction R, if R is a non-mixed reduction, and v_1 and v_{k+1} are black, then $\Phi(R) \geq 0$.*

Proof. Let us p be the backbone of R, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider the sub-backbone of p, $p' = \{(v_3, v_4), \ldots, (v_{k-2}, v_{k-1})\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of p' is even, and by the property of the independent set in an even length path, we have: $j \leq \ell + 1$. By adding v_1, v_2, v_{k-2} and v_{k-1}, we have: $j + \ell = k - 3$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j + 2$, $|E^R_{loan}| = 4$ and $|E^R_{debt}| = \ell + 2$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4(j+2) + 4 - (\ell+2) = \frac{5(j+\ell+3)}{2} - 4j - 8 + 4 - \ell - 2 = \frac{3}{2}\ell - \frac{3}{2}j + \frac{3}{2} \geq \frac{3}{2}\ell - \frac{3}{2}(\ell+1) + \frac{3}{2} = 0$. □

Claim 13. *For an even-backbone reduction R, if R is a non-mixed reduction, and v_1 and v_{k+1} are white, then $\Phi(R) \geq 1$.*

Proof. Let us p be the backbone of R, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider the sub-backbone of p, $p' = \{(v_2, v_3), \ldots, (v_{k-1}, v_k)\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of p' is even, and by the property of an independent set in an odd length path, we have: $j \leq \ell + 1$. By adding v_1 and v_k, we have: $j + \ell = k - 1$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j$, $|E^R_{loan}| = 0$ and $|E^R_{debt}| = \ell$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4j - (\ell+2) = \frac{5(j+\ell+1)}{2} - 4j - \ell = \frac{3}{2}\ell - \frac{3}{2}j + \frac{5}{2} \geq \frac{3}{2}\ell - \frac{3}{2}(\ell+1) + \frac{5}{2} = 1$. □

Claim 14. *For an odd-backbone reduction R, if R is a non-mixed reduction, and v_1 and v_{k+1} are black, then $\Phi(R) \geq -1$.*

Proof. Let us p be the backbone of R, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider the sub-backbone of p, $p' = \{(v_3, v_4), \ldots, (v_{k-2}, v_{k-1})\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of p' is even, and by the property of an independent set in an even length path, we have: $j \leq \ell + 1$. By adding v_1, v_2, v_{k-2} and v_{k-1}, we have: $j + \ell = k - 3$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j + 2$,

$|E^R_{loan}| = 3$ by the fact that $d_R(v_1) + d_R(v_{k+1}) - 2 = 3$ and $|E^R_{debt}| = \ell + 2$. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4(j+2) + 4 - (\ell+2) = \frac{5(j+\ell+3)}{2} - 4j - 8 + 3 - \ell - 2 = \frac{3}{2}\ell - \frac{3}{2}j + \frac{3}{2} \geq \frac{3}{2}\ell - \frac{3}{2}(\ell+1) - \frac{1}{2} = -1$. □

Claim 15. *For an odd-backbone reduction R, if R is a non-mixed reduction, and v_1 and v_{k+1} are black, then $\Phi(R) \geq 0$.*

Proof. Let us p be the backbone of R, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider the sub-backbone of p, $p' = \{(v_2, v_3), \ldots, (v_{k-1}, v_k)\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. Note that the length of p' is even, and by the property of an independent set in an odd length path, we have: $j \leq \ell + 1$. By adding v_1 and v_k, we have: $j + \ell = k - 1$. We compute the potential value: $SOL(R) = \frac{k}{2}$, $OPT(R) = j$, $|E^R_{loan}| = 0$ and $|E^R_{debt}| = \ell + 1$ by the fact that one of v_1 and v_{k+1} might has a debt edge. Then, $\Phi(R) \geq \frac{5 \cdot k}{2} - 4j - (\ell+2) = \frac{5(j+\ell+1)}{2} - 4j - \ell - 1 = \frac{3}{2}\ell - \frac{3}{2}j + \frac{3}{2} \geq \frac{3}{2}\ell - \frac{3}{2}(\ell+1) + \frac{3}{2} = 0$. □

Now, we can update our greedy algorithm into the extended reduction form of Algorithm 8.

Algorithm 8 Updated Greedy algorithm in extended reduction version

Input: a graph $G = (V, E)$
  G_E = extended graph of G
  $U \leftarrow V_E$
  $S \leftarrow \emptyset$
  while $U \neq \emptyset$ **do**
    choose a reduction R according to the following order in $G_E[U]$:
    1: single edge non-branching reduction
    2: single edge branching reduction.
    3: loop reduction
    4: even-backbone reduction
    5: odd-backbone reduction
    $U \leftarrow U \setminus V_E(R)$
    $S \leftarrow S \cup$ root vertices in S
    Update the extended graph
  end while
  return S

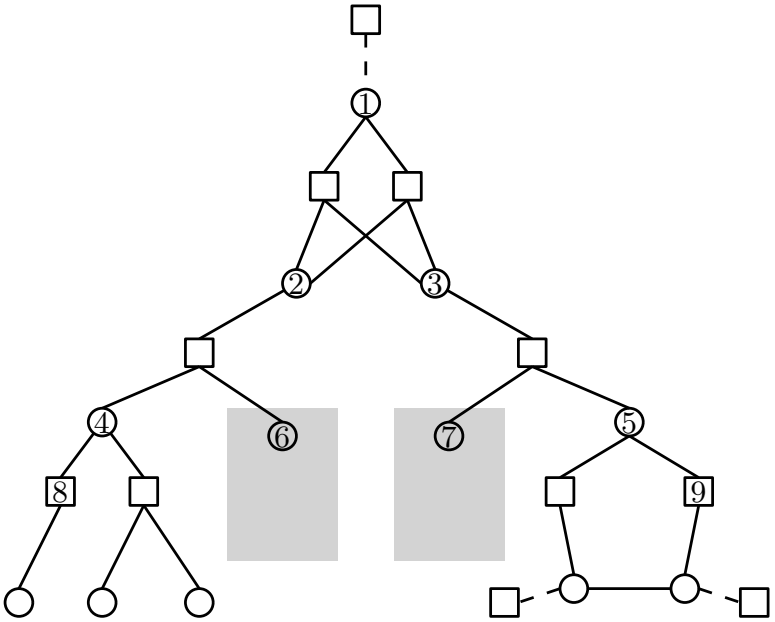


Figure 6.8: Instance example with an arbitrarily negative potential value.

6.3 The final proof

In this section, we will present the final proof. From the previous section, we know that for any reduction $R \in S$, only bad odd-backbone reductions and isolated cycle reductions' potential value is negative. Additionally, we have observed that the difficulties of addressing these two kind of reductions.

6.3.1 Observations and ideas

For some inspiration, let's review the example from the previous section and extend it in Figure 6.8.

The sequence of reductions in this example is: $S' = \{R_1, R_2, R_3, R_4, R_5\}$, where R_1 is a bad even-backbone reduction with vertex 1 as its root vertex. R_2 and R_3 are two single edge branching reductions with 2 and 3 as their root vertices. R_4 is a bad odd-backbone reduction with 4 as its root vertex and R_5 is an isolated cycle reduction. The dash edges in the example are edges which are already removed by the previous execution of the greedy. The two grey boxes containing vertex 6 and 7 are gadgets which are other isolated cycle reductions or odd-backbone reductions. It is easy to check that $\Phi(S') = -2$, because at the end, a bad odd-backbone reduction and an isolated 5-odd cycle reduction are created. This structure can be extended to

an arbitrarily large structure by including more isolated cycle and bad even-backbone reductions. Thus the potential value can also be arbitrarily large.

We observe that at the moment before R_1 is executed, potentially, there are at least three different vertices 1,8,9 whose degree is 2 and no vertices' degree is 1, and because the reductions formed by 8 and 9 are an even-backbone reduction and a *loop*-reduction, they are the candidates for the root vertex of the next execution. If the greedy algorithm chooses either vertex 8 or 9 as a first reduction, then for the following sequence of reductions executed, S'', it is easy to check that $\Phi(S'') \geq 0$. That is because no bad reduction is created, thus no bad reductions are in S''. Nevertheless, if the first reduction executed by the greedy algorithm is R_1 by taking vertex 1, then the degree of vertices 2 and 3 would become 1. Then R_2 and R_3 formed by vertices 2 and 3 are inevitably executed, and then, the isolated cycle reduction is created. This implies that the greedy algorithm has to determine which reduction it needs to execute in an early iteration of the execution in order to obtain the desired approximation ratio.

Furthermore, although the greedy algorithm is unable to do anything if an isolated bad cycle reduction has been created, the algorithm is able to do something for an odd-backbone reduction. Note that an even *(1,3)*-reduction of vertex 2 has been executed, and a *(2,5)*-reduction is left. At this moment, the greedy algorithm still has a chance to choose the right vertex, by choosing the odd-backbone reduction with root vertex 8. However, in the following analysis in this section, we will not adopt such an approach; the discussion of this approach is relocated to Section 6.4.

A crucial observation has been found in the study of bad isolated cycle reductions. Let us call it R, and, without loss of generality, assume that R is an 5-odd-cycle. We know that if $\Phi(R) = -1$, which means that it is a bad reduction, it must have all 3 debt edges. Now, we consider the previous reduction R', and assume that the execution of R' will create R. Then if either all edges from R' are connected to vertices which are in the independent set or are not in the independent set, formally, $\forall e \in E^{R'}_{contact}$, $(N(e) \setminus V_{R'})$ are black or $\forall e \in E^{R'}_{contact}$, $N(e) \setminus V_{R'}$ are white, then there exists a high-priority reduction in $R \cup R'$. This reduction is not R itself, if R itself is a high-priority reduction. In the next section, we can generalise this observation by induction for all sequences of reductions S, where $\Phi(S) = -1$.

Now, let us consider a reduction R which creates two such structures, which means that after the execution of R, two disjoint such structures are created. Then before this execution, if the edges from R which are connected to these two structures satisfy above property, then in both structures, if the assumption is correct, then there should be two high-priority reductions in there. Also because of the disjointedness of the

two structures, R and together with these two structures form a tree-like graph. This observation of such tree-like structure implies that the greedy algorithm should be advised to choose the leaf reduction of this tree rather than the reduction that is non-leaf. We discuss the role of leaf reductions in next subsection 6.3.2. This argument would be crucial to the final proof.

At this moment, we present the *Ultimate Greedy algorithm* of Algorithm 9. We will prove that the ultimate greedy algorithm 9 obtains a $\frac{4}{3}$-approximation ratio. For the reminder of this Section, when we mention greedy algorithm in the proof, this phase refers to the ultimate greedy algorithm.

Algorithm 9 Ultimate Greedy algorithm in extended reduction version

Input: a graph $G = (V, E)$

G_E = extended graph of G
$U \leftarrow V_E$
$S \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  choose a reduction R according to the following order in $G_E[U]$:
  1: single edge non-branching reduction
  2: single edge branching reduction.
  3: loop reduction
  4: even-backbone reduction in the leaf
  5: odd-backbone reduction
  $U \leftarrow U \setminus V_E(R)$
  $S \leftarrow S \cup$ root vertices in R
end while
return S

6.3.2 The leaf reduction

In this subsection, we discuss the definition of even-backbone reductions in the leaf and how the algorithm finds such a reduction.

Definition 22. *We say that a graph contains or that there exists a* high-priority *reduction in the graph, if there exists a vertex with degree at most* 2*, where the greedy algorithm can choose it as a reduction whose priority is at least the priority of the even-backbone reduction.*

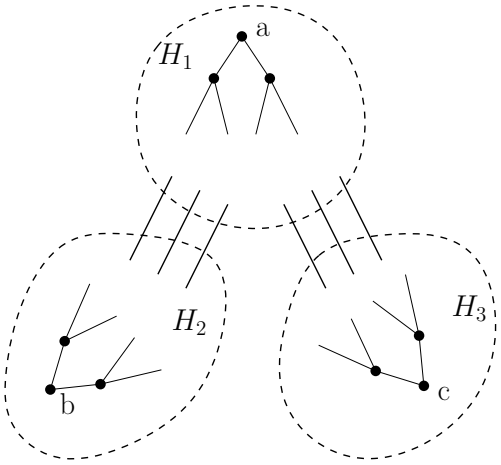


Figure 6.9: Example of an even-backbone reduction in the leaf.

Given a graph G, we consider a spanning tree of G, and a leaf path of a spanning tree is a path whose one endpoint has degree 1.

Definition 23. *Given a graph G, an even-backbone reduction R is in the leaf, if for a spanning tree of G, at least one vertex of V_R belongs to a leaf path of this spanning tree.*

In Figure 6.9, we present an example to show which even-backbone reduction is the leaf one. In the example, there are three different even-backbone reductions, and their root vertices are respectively a, b, c. Consider a spanning tree T of G, where the spanning tree will cover all vertices in the graph. Let us assume that both b and c are degree one vertices of T, and a and the adjacent vertices of a are not in any leaf path. Then the even-backbone reductions in H_2 and H_3 are in the leaf. Therefore, the greedy algorithm of Algorithm 9 will execute either even-backbone reductions in H_2 or H_3, but not the reductions in H_1.

Lemma 5. *There exists an algorithmic way to find a leaf high-priority reduction in $O(n)$ time complexity.*

Proof. Assume that the current graph is G, and the execution of R will create at least two disjoint connected components C_1 and C_2. If as assumed, in both C_1 and C_2, there exist high-priority reductions R_a and R_b, then, there must exists a path p from R_a to R_b, and one of the vertices of R belongs to p, *i.e.* $\exists v \in V(R), v \in p$. Therefore, a spanning tree can be implemented in $O(n)$ time. □

6.3.3 Proof for existence of $\frac{4}{3}$-approximation greedy algorithm

We will prove a less tight result for MIS problem with $\Delta \leq 3$, to show the existence of a $\frac{4}{3}$-approximation greedy algorithm with $O(n^2)$ time complexity. Though this result is a progressive achievement to the ultimate greedy algorithm, it is interesting in its own right, because it improves the previously known analysis of the greedy. The previous best known ratio of the greedy on sub-cubic graphs was $\frac{3}{2}$ [31]. As we show in Claim 7, the $\frac{9}{7}$-approximation greedy algorithm in [31] is incorrect.

Since here we prove less tight ratio, we can reformulate the potential function as:

$$\Phi(R)^{\frac{4}{3}} = 4|SOL(R)| - 3|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|$$

In this section, we will only consider $\Phi^{\frac{4}{3}}(R)$.

Lemma 6. *For any reduction R, $\Phi^{\frac{4}{3}}(R) = \Phi^{\frac{5}{4}}(R)$, except a bad (2,6)-reduction, for which we have $\Phi^{\frac{4}{3}}(R) = 1$ and a bad (2,5)-reduction, for which $\Phi^{\frac{4}{3}}(R) = 0$.*

Proof.

$$\begin{aligned}
\mathcal{D}(R) =& \Phi^{\frac{5}{4}}(R) - \Phi^{\frac{4}{3}}(R) \\
=& 5|SOL(R)| - 4|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)| \\
& - (4|SOL(R)| - 3|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|) \\
=& |SOL(R)| - |OPT(R)|
\end{aligned}$$

Observe that if $|SOL(R)| = |OPT(R)|$, then $\mathcal{D}(R) = 0$. It is easy to check for every reduction R, that only a bad even-backbone reduction and a bad odd-backbone reduction R satisfy $|SOL(R)| = |OPT(R)| - 1$. Then in this case, we have: $\mathcal{D}(R) = -1$. Therefore, we have: $\Phi^{\frac{5}{4}}(R) + 1 = \Phi^{\frac{4}{3}}(R)$, and thus obtain the lemma. □

Remark. *For the consistence of notation, when we talk about a bad odd-backbone reduction, $\Phi(R)$ is an abbreviation of $\Phi^{\frac{4}{3}}(R)$.*

Now, we introduce some concepts to characterise the property we described in the previous subsection. For an isolated cycle reduction R, observe that if the previous reduction R' provides only single type of edges to connect R, *i.e.* all edges that are connected only to $v \in V_R$ have black v or all the edges that are connected to $v \in V_R$ have white v, then there exists a high-priority reduction in graph $G(R' \cup R)$. We extend this observation to every graph G such that $\Phi(G) = -1$. To formally capture this observation, we introduce the *dummy* graph as follows. Before that, given a connected

graph G and an independent set I in G, we call any path p in G *alternating* if the black and white vertices on p alternate, i.e., for any two consecutive vertices u, v joined by an edge on path p, we have that $|\{u, v\} \cap I| = 1$.

Definition 24. *Let G be a connected graph with minimum degree ≤ 2 and I an independent set in G. We construct a* black dummy graph *(*resp. white dummy graph*), denoted $\widetilde{G}$, as follows. First, add a* dummy vertex $\omega \notin V(G)$ *and choose a non-empty subset of* dummy black edges *(*resp. dummy white edges*) : $\widetilde{E} \subseteq \{(\omega, v), v \in V(G), d_G(v) \leq 2, v \in X\}$ where $X = I$ (*resp. $X = V(G) \setminus I$*), and define $\widetilde{G} = (V(G) \cup \{\omega\}, E(G) \cup \widetilde{E})$. A dummy graph $\widetilde{G}$ might contain parallel edges (ω, v) but must satisfy $d_{\widetilde{G}}(v) \leq 3, \forall v \in V(G)$.*

We say that an (extended) reduction R in G is of white type *or* white (black type *or* black, *respectively) if the root of R is white (black, respectively).*

Then, we say that G is potentially problematic *if, for any black (white, respectively) dummy graph $\widetilde{G}$, there exists a non-odd-backbone reduction R of white type (black type, respectively), such that $V(R) \subseteq V(G)$.*

Lemma 7. *Given a problematic odd isolated cycle reduction or an isolated single edge reduction R, its any black (white, respectively) dummy graph contains a non-odd-backbone reduction R of white type (black type, respectively), such that $V(R) \subseteq V(G)$.*

Proof. Observe that for a bad isolated cycle reduction R, if the length of R is k, then there are $\frac{k-1}{2}$ vertices v, and for each of them, v is black. We have: $\exists v_1, v_2 \in V_R$, and v_1 and v_2 are white, and $v_1 \in N_R(v_2)$. Moreover, let p be a path of R, then for every pair of distinct vertices v_s, if v_s and v_e are black, then there exists a path p with v_s and v_e as its two endpoints, and $|p|$ is an even number. Therefore, if we consider the black dummy graph $\tilde{G}$ of R, then either there exists an extended edge e_e with label 0, which means there exists an even-backbone reduction, or there exists a loop reduction. In both cases, we have a high-priority reduction.

The argument is the same for the white dummy graph. For every pair of vertices v_s and v_e with $v_s \neq v_e$, if v_s and v_e are white, then there exists a path p with v_s and v_e as its two endpoints, and $|p|$ with even. Then either there exists an extended edge e_e with label 0, which means there exists an even-backbone reduction, or there exists a loop reduction. Again, in both cases, we have a high-priority reduction.

For a single edge reduction R, the arguments for both white and black dummy graph are the same. Thus, there are high-priority reductions of each corresponding type within it. □

The strategy of the next lemma is: for every connected component C, we first prove $\Phi(C) \geq -1$. Then, if $\Phi(C) = 1$, then let us consider a reduction R, such that after the execution of R C is created. Then there is a high-priority reduction R' in $R \cup C$ and $R' \neq R$ if all contact edges of R are of the same type. Therefore, if R_1 is a bad odd-backbone reduction, then R_1 will not be executed, since it would violate the greedy order. We consider all kind of reductions and we will prove that either it will not be executed because of the greedy order, or their potential value is enough to pay for these connected components.

Lemma 8. *Let G be a graph, and R be a non-mixed reduction in G and $V_R \subset V(G)$ and $E^R_{\text{contact}} \subset E(G)$, and $\Phi(R) \leq 0$. If the connected component $C = G(V(G) \setminus V(R))$ is potentially problematic, then G is potentially problematic.*

Proof. We consider each kind of a reduction.

1. Suppose R is an even-backbone reduction, the by the assumption, if $\Phi(R) \leq 0$, then $\Phi(R) = 0$. Let p be the backbone of R with length k, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider a sub-backbone of p, $p' = \{(v_2, v_3), \ldots, (v_{k-1}, v_k)\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. We have: $j + \ell = k - 3$, $SOL(R) = \frac{k}{2}$, $OPT(R) = j + 2$, $|E^R_{loan}| = 4$ and $|E^R_{debt}| = \ell + 2$, and $\Phi(R) = 0$. Then we have: $\Phi(R) = \frac{5 \cdot k}{2} - 4(j + 2) + 4 - (\ell + 2) = \frac{3}{2}(\ell - j + 1)$. In the even path p', by the property of an independent set, $j \leq \ell + 1$. Then, note that $j = \ell + 1$, $\Phi(R) = 0$. Therefore, in the even backbone p', $\ell + 1 + \ell = k - 3$, and then $\ell = \frac{k}{2} - 2$ and $j = \frac{k}{2} - 1$. Therefore, in the backbone p, the number of vertices v such that v is black is $\frac{k}{2}$. And, the number of vertices v such that v is black is $\frac{k}{2} + 1$, and by the property of an independent set in an even path, it is an alternating backbone.

   Because C is potentially problematic, then by the definition, if $E_{contact} \subseteq \tilde{E}$ of dummy white edges (dummy black edges, resp.), then there exists a high-priority reduction R of black type (white type, resp.). Therefore, we obtain the lemma.

2. Suppose R is an odd-backbone reduction, then by the assumption, if $\Phi(R) \leq 0$, then $\Phi(R) = 0$. Let p be the backbone of R with length k, $p = \{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$. Let us consider a sub-backbone of p, $p' = \{(v_3, v_4), \ldots, (v_{k-2}, v_{k-1})\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. We have: $j + \ell = k - 3$, $SOL(R) = \frac{k}{2}$, $OPT(R) = j + 2$, $|E^R_{loan}| = 3$ and $|E^R_{debt}| = \ell + 2$, and $\Phi(R) = 0$ by assumption, then we have: $\Phi(R) = \frac{4 \cdot k}{2} - 3(j + 2) + 3 - (\ell + 2) = \ell - j + 1$ (Recall that $\Phi(R)$

in here is an abbreviation of $\Phi^{\frac{4}{3}}(R)$). In the even path p', by the property of an independent set, $j \leq \ell + 1$. Then, note that $j = \ell + 1$, $\Phi(R) = 0$. Therefore, in the even backbone p', $\ell + 1 + \ell = k - 3$, and then $\ell = \frac{k}{2} - 2$ and $j = \frac{k}{2} - 1$. Therefore, in backbone p, the number of vertices v such that v is black is $\frac{k}{2}$, and the number of vertices v such that v is black is $\frac{k}{2} + 1$. By the property of an independent set in an even path, it is an alternating backbone.

Because C is potentially problematic, then by the definition, if $E_{contact} \subseteq \tilde{E}$ of the dummy white edges (dummy black edges, resp.), then there exists a high-priority reduction R of black type (white type, resp.). Therefore, we obtain the lemma.

□

Lemma 9. *Let G be a graph, and R be a single edge branching or a single edge non-branching reduction in G. Let $V_R \subset V(G)$ and $E^R_{\text{contact}} \subset E(G)$, and $\Phi(R) \leq 1$. If graph $G' = G(G(V) \setminus V_R)$ forms at most 2 connected components C_1 and C_2, such that $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = G(V(G) \setminus V(R))$ and C_1 and C_2 are potentially problematic, then G is potentially problematic.*

Proof. We consider two cases.

1. Suppose R is a single edge non-branching reduction and by the assumption, $\Phi(R) = 0$. Let p be the path of R with length k, $\{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$, where k is odd. Firstly, suppose v_{k+1} is black. Let us consider the sub-backbone of p, $p' = \{(v_1, v_2), \ldots, (v_{k-2}, v_{k-1})\}$. Denote by j the number of vertices v such that v is black in p', and by ℓ the number of vertices v such that v is white in p'. We have: $j + \ell = k - 1$, $SOL(R) = \frac{k+1}{2}$, $OPT(R) = j + 1$, $|E^R_{loan}| = 1$ and $|E^R_{debt}| \leq \ell + 2$. In an odd path p', we have: $j \leq \ell$. Then, $\Phi(R) = \frac{5\cdot(k+1)}{2} - 4(j+1) + 1 - (\ell + 2) = \frac{3}{2}(\ell - j) = 0$, and thus, $\ell = j$. And by the property of an independent set in an odd path, this implies that the path is alternating.

   Secondly, v_{k+1} is black. Let us consider sub-backbone of p, $p' = \{(v_1, v_2), \ldots, (v_{k-1}, v_k)\}$. Denote j as the number of vertices v such that v is black in p' and ℓ as the number of vertices v such that v is white in p'. We have: $j + \ell = k + 1$, $SOL(R) = \frac{k+1}{2}$, $OPT(R) = j$, $|E^R_{loan}| = 0$ and $|E^R_{debt}| \leq \ell + 1$. Then $\Phi(R) = \frac{5\cdot(k+1)}{2} - 4j + 0 - (\ell + 1) = \frac{3}{2}(\ell - j) - 1 = 0$. This is contradiction, then $E^R_{debt} \leq \ell$, then if $j = \ell$, $\Phi(R) = 0$. And by the property of independent set in odd path, this implies the alternating of the path.

2. Suppose R is a single edge branching reduction and by assumption, $\Phi(R) = 0$. Let p be the path of R with length k, $\{(v_1, v_2), \ldots, (v_k, v_{k+1})\}$, and note k is odd. Firstly, suppose v_{k+1} is black. Let us consider a sub-backbone of p, $p' = \{(v_1, v_2), \ldots, (v_{k-2}, v_{k-1})\}$. Denote by j the number of vertices v such that v is black in p' and by ℓ the number of vertices v such that v is white in p'. We have: $j + \ell = k - 1$, $SOL(R) = \frac{k+1}{2}$, $OPT(R) = j + 1$, $|E^R_{loan}| = 1$ and $|E^R_{debt}| \leq \ell + 2$. In an odd path p', we have: $j \leq \ell$. Then, $\Phi(R) = \frac{5\cdot(k+1)}{2} - 4(j+1) + 1 - (\ell+2) = \frac{3}{2}(\ell - j) = 0$, and thus, $\ell = j$. And by the property of an independent set in an odd path, this implies that the path is alternating.

   Secondly, let v_{k+1} be black. Let us consider a sub-backbone of p, $p' = \{(v_1, v_2), \ldots, (v_{k-1}, v_k)\}$. Denote by j the number of vertices v such that v is black in p', and by ℓ the number of vertices v such that v is white in p'. We have: $j+\ell = k+1$, $SOL(R) = \frac{k+1}{2}$, $OPT(R) = j$, $E^R_{loan} = 0$ and $E^R_{debt} \leq \ell+1$. Then $\Phi(R) = \frac{5\cdot(k+1)}{2} - 4(j) + 0 - (\ell+1) = \frac{3}{2}(\ell - j) - 1 = 0$. This is a contradiction because $E^R_{debt} \leq \ell$, and if $j = \ell$, $\Phi(R) = 0$. And by the property of an independent set in an odd path, this implies that the path is alternating.

□

We present now the main lemma for the proof of an $\frac{4}{3}$-approximation ratio the greedy algorithm.

Lemma 10. *Let G be a connected graph with minimum degree ≤ 2, I an independent set in G and the sequence of reduction $S = \{R_1, \cdots, R_k\}$ executed by greedy algorithm on G, then:*

1. $\Phi(S) \geq -1$.
2. *if $\Phi(S) = -1$, then G is potentially problematic.*

Proof. We prove this result by induction on the number k of the executed reductions.

Firstly, we consider the base case. If $k = 1$, it implies that the reduction R_1 has no contact edges, *i.e.* it would be one of an odd problematic cycle, isolated single edge reduction or an isolated vertex reduction. From Claim 8, we know that the potential value of each of those reductions is at least -1. Among them, the odd problematic cycle and bad isolated single edge reduction's potential value is exact -1. Then due to Lemma 7, $G(R_1)$ is potentially problematic. The inductive base is true.

Then, we prove the inductive step. Suppose that S contains $k \geq 2$ reductions. Now we consider the following cases depending on how reduction R_1 can look like. In all

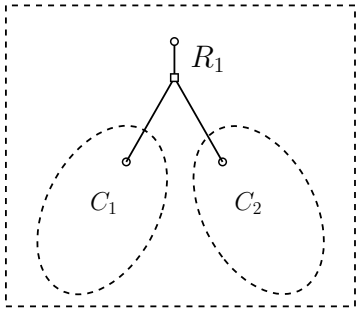


Figure 6.10: Single edge branching reduction in Lemma 10

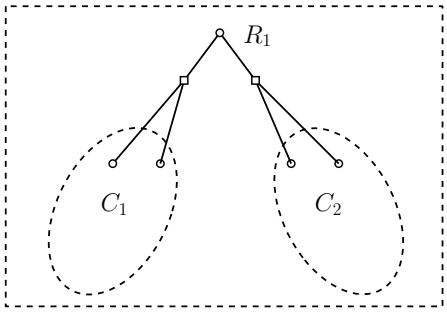


Figure 6.11: Bad even-backbone reduction in Lemma 10

these cases the induction hypothesis will be applied to each connected component of the graph after executing reduction R_1.

We will frequently refer to Claim 10, Claim 15, and to Lemmas 7, 8, 9.

1. Let R_1 be a single edge branching reduction. Note that $\Phi(R) = 1$. If the execution of R_1 creates only one connected component C, then due to the inductive hypothesis, $\Phi(C) \geq -1$. Then $\Phi(R_1) + \Phi(C) \geq 0$. If the execution of R_1 creates 2 connected components, see Figure 6.10, C_1 and C_2, the due to the inductive hypothesis, $\Phi(C_1) \geq -1$ and $\Phi(C_1) \geq -1$. Then $\Phi(R_1) + \sum_{i=1}^{2} C_i \geq -1$. Due to Lemma 8, $G(R_1 \cup \bigcup_{i=1}^{2} C_i)$ is potentially problematic, thus the inductive hypothesis is preserved.

2. Let R_1 be a bad even-backbone reduction. Note that $\Phi(R_1) = 0$. If the execution of R_1 creates only one connected component C, the due to the inductive hypothesis, $\Phi(C) \geq -1$. Then $\Phi(R_1) + \Phi(C) \geq -1$. Due to Lemma 8, $G(R_1 \cup C)$ is potential problematic.

   If the execution of R_1 creates at least two connected components, see Figure 6.11, say C_i where $i \in \{1, \ldots, k\}$ and $k \leq 4$, then due to the inductive hypothesis, $\Phi(C_i) \geq -1$, $\forall i \in \{1, \ldots, k\}$. Assume that there are at least two connected components C_1 and C_2 with $\Phi(C_1) = \Phi(C_2) = -1$. Because for each C_i, $E^{R_1}_{contact} \subseteq \tilde{E}$ of dummy white edges of C_i, there exists a high-priority reduction R' of black type because C_i is potentially problematic. Due to the leaf order of the greedy, R_1 will not be executed which leads to a contradiction. Thus, there is at most one connected component with $\Phi(R_1) = -1$. Therefore, the inductive hypothesis is again preserved.

3. Let R_1 be a bad odd-backbone reduction. Note that $\Phi(R_1) = 0$. If the execution

of R_1 creates k connected components C_i, where $i \in \{1, \ldots, k\}$ and $k \leq 3$, the due to the inductive hypothesis $\Phi(C_i) \geq -1$ for $i \in \{1, \ldots, k\}$. Assume that there is at least one connected component C with $\Phi(C) = -1$. Because for C there exists a high-priority reduction R' of black type, because C_k is potentially problematic, by the order of greedy, R_1 will not be executed. This leads to a contradiction. Thus, no connected component C_i has its potential value equal to -1. Thus, $\forall i \in \{1, \ldots, k\}$, $\Phi(C_i) = 0$. Then $\Phi(R_1) + \sum_{i=1}^{k} \Phi(C_i) \geq 0$, and therefore, the inductive hypothesis is preserved.

4. Let R_1 be a good mixed odd-backbone reduction. Note that $\Phi(R_1) = 1$. If the execution of R_1 creates only one connected component C, then due to the inductive hypothesis, $\Phi(C) \geq -1$ and $\Phi(R_1) + \Phi(C) \geq 0$.

   If the execution of R_1 creates at least two connected components, say C_i for $i \in \{1, \ldots, k\}$ and $k \leq 3$, then due to the inductive hypothesis, $\Phi(C_i) \geq -1$, $\forall i \in \{1, \ldots, k\}$. Assume that there are at least two connected components C_1 and C_2 with $\Phi(C_1) = \Phi(C_2) = -1$. Since R_1 provides 2 edges of one type and 1 edge of another type, at least one connected component, say C_1, receives only one type of edges. Due to the inductive hypothesis, C_1 is potentially problematic, and then there exists a high-priority reduction R' of black type. Then R_1 will not be executed by the order of the greedy which leads to a contradiction.

   Thus, at most one connected component created by R_1 has potential value -1, and then $\Phi(R_1) + \sum_{i=1}^{k} \Phi(C_i) \geq 0$. Therefore, the inductive hypothesis is preserved.

5. Let R_1 be a good mixed even-backbone reduction. Note that $\Phi(R_1) = 2$. If the execution of R_1 creates at most two connected component C_1 and C_2, then due to the inductive hypothesis, $\Phi(C_1) \geq -1$ and $\Phi(C_2) \geq -1$. Then $\Phi(R_1) + \sum_{i=1}^{2} \Phi(C_i) \geq 0$.

   If the execution of R_1 creates at least 2 connected components, say C_i for $i \in \{1, \ldots, k\}$ and $k \leq 4$, then due to the inductive hypothesis, $\Phi(C_i) \geq -1$, $\forall i \in \{1, \ldots, k\}$. Assume that there are at least three connected components C_i, $i = 1, 2, 3$, with $\Phi(C_i) = -1$ for $i = 1, 2, 3$. Because R_1 provides only two edges of one type and two edge of another type, at least two of the connected components receive only one type of edges. Let us say that C_1 and C_2 are those two connected components. Due to the inductive hypothesis, C_1 and C_2 are potentially problematic, and then there exists a high-priority reduction R' of black type in both

C_1 and C_2. Thus, R_1 will not be executed by the leaf order of the greedy which leads to a contradiction. Thus, at most two of the connected components created by R_1 have potential value -1. Then $\Phi(R_1) + \sum_{i=1}^{k} \Phi(C_i) \geq 0$. Therefore, the inductive hypothesis is again preserved.

6. Let R_1 be a good non-mixed even-backbone reduction. Note that $\Phi(R_1) = 1$. If the execution of R_1 creates only one connected component C, then due to the inductive hypothesis, $\Phi(C) \geq -1$, and $\Phi(R_1) + \Phi(C) \geq 0$.

   If the execution of R_1 creates at least 2 connected components C_i, where $i \in \{1, \ldots, k\}$ and $k \leq 4$, then due to the inductive hypothesis, $\Phi(C_i) \geq -1$, $\forall i \in \{1, \ldots, k\}$. Assume that there are at least two connected components, say C_1 and C_2, with $\Phi(C_1) = \Phi(C_2) = -1$. Since for each C_i, $E^{R_1}_{contact} \subseteq \tilde{E}$ of dummy black edges of C_i, there exists a high-priority reduction R' of white type, because C_i is potentially problematic. Due to the leaf order of the greedy, R_1 will not be executed which leads to a contradiction. Thus, there is at most one connected component with $\Phi(R_1) = -1$. Therefore, the inductive hypothesis is preserved.

7. Finally, let R_1 be a good non-mixed odd-backbone reduction. Note that $\Phi(R_1) = 0$. If the execution of R_1 creates k connected components C_i, where $i \in \{1, \ldots, k\}$ and $k \leq 3$, then due to the inductive hypothesis, $\Phi(C_i) \geq -1$ for each i. Assume that there is at least one connected component, say C_1 with $\Phi(C_1) = -1$. For C_1 there exists a high-priority reduction R' of white type by the fact that C_1 is potentially problematic. Due to the order of the greedy, R_1 will not be executed thus we have a contradiction. Consequently, no connected component C_i has -1 potential value, thus, $\forall i \in \{1, \ldots, k\}$, $\Phi(C_i) = 0$. Then $\Phi(R_1) + \sum_{i=1}^{k} \Phi(C_i) \geq 0$. Therefore, the inductive hypothesis is preserved.

   This concludes the proof of the lemma.

□

Then we will show that the first reduction in the entire sequence of reductions executed by the greedy has a saving of 1 to pay for the cases in which the potential value of the execution is -1.

Corollary 2. *Let $S = \{R_1, \ldots, R_\ell\}$ be a sequence of reductions executed by the greedy algorithm on G, then $\Phi(S) \geq 0$.*

Proof. By Lemma 10 we have that $\Phi(\{R_2, \ldots, R_\ell\}) \geq -1$ and we now consider the very first reduction, R_1, in the sequence, depending on how it may look like.

1. Let R_1 be any good-reduction, including: good even-backbone reduction, good mixed odd-backbone reduction, good mixed even-backbone reduction and good odd-backbone reduction. If any of those is the first executed reduction, then $\Phi(R_1) \geq 1$ as we know, and therefore, we proved that $\Phi(S) \geq 0$.

2. Let R_1 be a bad odd-backbone reduction and let $E_{past} = \emptyset$. Then $\Phi(R_1) = 1$, therefore, we have $\Phi(S) \geq 0$.

3. Let R_1 be a bad even-backbone reduction and $E_{past} = \emptyset$. Then $\Phi(R_1) = 2$, therefore, we again have $\Phi(S) \geq 0$.

□

Therefore, we have proved the following theorem.

Theorem 20. *The ultimate greedy algorithm of Algorithm 9 for MIS on sub-cubic graphs, obtains a $\frac{4}{3}$-approximation ratio and its time complexity is $O(n^2)$, where n is the number of vertices of the input graph.*

6.3.4 Towards a proof of existence of the ultimate greedy algorithm

We will show now that the previous proof does not work when we apply it to prove a $\frac{5}{4}$-approximation for the greedy algorithm.

To prove the $\frac{5}{4}$-approximation we would need to apply following potential function:

$$\Phi(R) = 5|SOL(R)| - 4|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|.$$

Note that $\Phi(R) = -1$, if R is an odd-backbone reduction. Then in the proof of Lemma 10, we cannot preserve the inductive hypothesis in the case of a bad odd-backbone reduction R_1. Note that in such a case, if the potential value of one of the connected components created by R_1 is -1, then we are able to show that such a case violates the greedy order, then it leads to a contradiction. However, if the potential values of all of these connected components are 0, then the sequence of reductions $\{R_1, \ldots, R_k\}$'s potential value is -1. But we cannot prove that the inductive hypothesis is preserved, *i.e.* that $G(\{R_1, \ldots, R_k\})$ is potentially problematic. Observe that this problem does not happen in the proof for the $\frac{4}{3}$-approximation ratio, where we use a different potential function (with 4 and 3 instead of 5 and 4, respectively).

Payment delaying approach: The crucial observation for solving this problem is that the analysis fails only under the circumstances where in a connected component

there exists a special cycle structure. At this moment, the inductive hypothesis cannot be preserved.

We construct an example to show the problematic case. Consider the sub-sequence of reductions $S = \{R_1 \cdots, R_4\}$, and let R_4 be a bad isolated cycle reduction with 9 vertices. Let us call the vertices in R_4 as follows $V(R_4) = \{v_1, v_2, \ldots, v_9\}$, where v_1 and v_2 are two adjacent vertices, v_1 and v_2 are white, and v_k is the anti-clockwise vertex adjacent to v_{k-1}. Note that in a bad isolated cycle reduction, for $i = 0 \mod 2$ and 1, v_i is white, and for $i = 1 \mod 2$, v_i is black. R_3 is a good non-mixed *(2,5)*-reduction such that $\Phi(R_3) = 1$ and let v_{10} and v_{11} be two vertices adjacent to the root vertex of R_3. Assume also that v_{10} is white and v_{11} is black, and before the execution of R_3, in current graph, there exists the set of edges $\{(v_{10}, v_5), (v_{10}, v_7), (v_{11}, v_6)\}$. R_2 is a bad *(2,5)*-reduction such that $\Phi(R_2) = -1$, and let us denote two non-root vertices as v_{12} and v_{13}. Before the execution of R_3, there exists the set of edges $\{(v_{12}, v_4), (v_{13}, v_8)\}$. Finally, R_1 is a good non-mixed *(2,6)*-reduction, and assume that before the execution of R_1, v_{14} is the vertex connected to v_3 and v_9. Note that for the sub-sequence of reductions $S' = \{R_2, R_3, R_4\}$, we have $\Phi(S') = -1$. However, even if v_9 and v_3 are black, satisfying the inductive hypothesis in Lemma 10, there is still no high-priority reduction in S. Therefore, the inductive hypothesis cannot be preserved.

6.4 Technique of Super-Advice

The content of this section is independent from the other sections.

We will explain here an approach that has a potential to prove that our ultimate greedy algorithm has a $\frac{5}{4}$-approximation ratio, and it even has a potential to help improve the approximation ratio beyond this ratio using a non-greedy approach. We explain this last point in the conclusion section of this thesis.

To start we will make the following interesting observation in the study of the greedy algorithm about the properties of *(2,5)* and *(2,6)*-reductions. We observe that one of the two choices of the roots of an *(2,5)*-reduction must be a good reduction, depending of the type of the root vertex that the greedy chooses. It is also true for a *(2,6)*-reduction, however, the difference between it and the *(2,5)*-reduction is that the choice of the vertex adjacent to the root vertex of the *(2,6)*-reduction violates the greedy order. Thus, we cannot modify any greedy algorithm to accommodate for such a rule. However, it seems that this provides a potential implication for the further study of the maximum independent set problem. Particularly, for the design of an algorithm with a better approximation ratio, which is not necessary greedy. Nevertheless, we also

expect to adopt such technique to obtain a better greedy algorithm. Therefore, the following section will present the technique based on such observation and general idea of how to use it to improve the performance of the algorithm, even though the analysis here is not complete.

Firstly, we consider a simple case to illustrate the general idea of *Super-Advice*. In this section, the default potential function $\Phi(R)$ is $\Phi^{\frac{5}{4}}(R)$. Let us consider a sequence of reductions $S^a = \{R_1^a, \ldots, R_k^a\}$ executed by the ultimate greedy algorithm of Algorithm 9, where R_1^a is a bad odd-backbone reduction. We know that for an odd-backbone reduction, there exists an alternative choice by choosing a vertex b adjacent to the root vertex a of reduction R_1^a, which we denote by R_1^b. And if the greedy algorithm chooses R_1^b and executes the subsequent reductions, then we denote this sequence of reductions as $S^b = \{R_1^a, \ldots, R_m^a\}$. Now, assume that for any reduction $R \in S^a$ and $R' \in S^b$, except R_1^a and R_1^b, both R and R' are not odd-backbone reductions. Then for one of the choices, say a, $\Phi(S^a) \geq 0$. Note that it is the moment when the analysis for the $\frac{5}{4}$-approximation ratio fails, because the inductive hypothesis cannot be preserved. Moreover, at this moment, the greedy algorithm in the worst case will choose the wrong vertex as a bad *(2,5)*-reduction. But by the super-advice, at this moment, the greedy algorithm is able to choose the right vertex as good a *(2,5)*-reduction.

Lemma 11 (Super-advice)**.** *Consider a sequence of reductions $S^a = \{R_1^a, \ldots, R_k^a\}$, where R_1 is a bad odd-backbone reduction. Now consider the alternative choice of such a bad odd-backbone reduction, which means, that greedy algorithm now chooses the alternative degree 2 vertex b. This implies a new sequence of reductions $S^b = \{R_1^b, \ldots, R_m^b\}$ than after the execution of R_k^a or R_m^b. Moreover, for any reduction $R \in S^a$ and $R' \in S^b$, except R_1^a and R_1^b, both R and R' are not odd-backbone reductions. Then for one of the choices of a,b, say a is the right choice, we have $\Phi(S^a) \geq 0$.*

Proof. Note that the choice of the greedy algorithm will be $\arg\max_{i \in \{a,b\}} |S^i|$, which means that the sequence of reductions will be S^a or S^b depending of which maximises the size of the solution. Firstly, let us assume that the size of the solution by choosing a is larger than when b is chosen. Thus a is the right choice, which means $\Phi(R_1^a) = 0$. Thus, by the contact edge lemma, $\Phi(S^a) \geq 0$, because there is no reduction R in S^a with $\Phi(R) = -1$. Secondly, assume that the size of the solution by choosing a is larger than b, but b is the right choice. This means that $\Phi(R_1^b) = 0$. In this case, note that R_1^a is possible to be a bad odd-backbone reduction, which means that $\Phi(R_1^a) = -1$. However, the sizes of the optimum in both sequences of reductions are equal, because the optimum is fixed. Thus, we have that $|OPT(S^a)| = |OPT(S^b)|$. Also, we assume

that the size of S^a is equal to S^b. By the property of the potential function and the fact that $\Phi(S^b) = \Phi(R_1^b) + \Phi(S_1^b \setminus R_1^a) \geq 0$, we obtain that $\frac{|OPT(S^b)|}{|SOL(S^b)|} \leq \frac{5}{4}$. Then, we have:

$$\frac{|OPT(S^a)|}{|SOL(S^a)|} = \frac{|OPT(S^b)|}{|SOL(S^b)|} \leq \frac{5}{4}.$$

This implies that $\Phi(S^a) \geq 0$. If the size of S^a is strictly larger than S^b, then we have

$$\frac{|OPT(S^a)|}{|SOL(S^a)|} \leq \frac{|OPT(S^b)|}{|SOL(S^b)|} \leq \frac{5}{4}.$$

This also implies that $\Phi(S^a) \geq 0$, which concludes the proof of the lemma. □

Inspired by the above lemma, we can generalise it: we can not only apply it to the case where for every reduction R in $(S^a \setminus R_1^a) \cup (S^b \setminus R_1^b)$, its potential value of $\Phi(R)$ is non-negative, but also to the case where $\Phi(S^a \setminus R_1^a) \geq 0$ and $\Phi(S^b \setminus R_1^b) \geq 0$.

Lemma 12. *For S^a and S^b, if $\Phi(S^a \setminus R_1^a) \geq 0$ and $\Phi(S^b \setminus R_1^b) \geq 0$, then one of the choices of a and b is the right choice, and if a is the right choice, then $\Phi(S^a) \geq 0$.*

Remark. *The technique of* Super-Advice *is aimed to obtain a greedy algorithm with $\frac{5}{4}$-approximation ratio, however, it does not work alone. But, this technique can be used to address the problem when the input graph is cubic.*

Chapter 7

Further applications

We study further applications of the techniques we developed in the previous chapter. The chapter is organised as follows. In Section 7.1.1, we extend our method to MIS on general degree bounded graphs, to obtain an alternative proof that the primitive greedy algorithm is a $\frac{\Delta+2}{3}$- approximation algorithm for any Δ bounded degree graph. Next, in Section 7.2, we present a 1.8-approximation ratio for the greedy algorithm for MIS on degree at most 4 graphs. In the next section, we address a closely related optimisation problem: minimum vertex cover, and we present a complementary greedy algorithm for this problem on degree at most 3 graphs.

7.1 Greedy algorithm for MIS on bounded degree graph

In this section, we study what performance the greedy algorithm can obtain in bounded degree graphs. We first present an alternative and simpler proof for a $\frac{\Delta+2}{3}$-ratio for primitive greedy algorithm on bounded degree graphs. Secondly, we present a lower bound of $\frac{\Delta+1}{3}$ for any greedy algorithm on bounded degree graphs.

7.1.1 Alternative proof for $\frac{\Delta+2}{3}$-ratio greedy algorithm on Δ-degree graphs

Halldorsson and Radhakrishnan [31] proved that for any bounded degree graph, the primitive greedy algorithm obtains $\frac{\Delta+2}{3}$-approximation ratio. In here, we present an alternative proof for the same result, but using our method of a payment scheme. Our proof will be incredibly short as compared to the proof in [31].

Let us define the following potential function:

$$\Phi(R) = (\Delta+2)\cdot\frac{\Delta+k}{3}\cdot|SOL(R)| - 3\cdot\frac{\Delta+k}{3}\cdot|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|,$$

where $k \in \{0,1\}$ is fixed. If S is the sequence of all reductions, note that $|E_{loan}(S)| - |E_{debt}(S)| = \sum_{R\in S}(|E_{loan}(R)| - |E_{debt}(R)|) = 0$. If we prove that $\Phi(R) \geq 0$ for any reduction R, then we obtain that $\Phi(S) = (\Delta+2)\cdot\frac{\Delta+k}{3}\cdot|SOL(S)| - 3\cdot\frac{\Delta+k}{3}\cdot|OPT(S)| \geq 0$, and $\frac{|OPT(S)|}{|SOL(S)|} \leq \frac{\Delta+2}{3}$. Therefore, we want to show for all reductions $R \in S$, $\Phi(R) \geq 0$.

We note that although there are many types of reductions, 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 minimise the potential, $R^* = \arg\min_{R\in\mathcal{R}}\Phi(R)$, then such reduction needs more debt edges and vertices in OPT and less loan edges. Also, for each $v \in V(R)\setminus v^*$, if $d_R(v^*) = k$, $d_R(v) \geq k$, by the greedy rule. For any reduction R, let j be the number of vertices in OPT and ℓ be the number of vertices not in OPT. We have following formulas:

$$|E_{loan}| \geq (j+\ell-1-\ell)\cdot j,$$

$$|E_{debt}| \leq (\Delta-j-\ell+1)\cdot\ell.$$

We will justify these bounds now. Given any reduction, the degree of its root vertex is $j+\ell-1$. The lower bound on E_{loan} depends on the vertices in OPT, by the definition. By the greedy order, in the current graph G', for each of vertex $v \in OPT$, $|N_{G'}(v)| \geq j+\ell-1$. There are at most ℓ vertices not in OPT which can be connected to v, thus, the total number of loan edges of v is at least $(j+\ell-1-\ell)$, and we have j such vertices. The upper bound on E_{debt} depends on Δ, the degree of root vertex and the number vertices not in OPT. The number of debt edges is at most $\Delta-j-\ell+1$, as otherwise it violates greedy order, and we have ℓ vertices not in OPT.

$$\Phi(R) = \frac{\Delta+k}{3}\cdot(\Delta+2)|SOL(R)| - (\Delta+k)|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)| \tag{7.1}$$

$$\geq \frac{\Delta+k}{3}(\Delta+2) - (\Delta+k)j + (j-1)j - (\Delta-j-\ell+1)\ell \tag{7.2}$$

$$= \ell^2 - (\Delta-j+1)\ell + \frac{\Delta+k}{3}(\Delta+2) - (\Delta+k)j + (j-1)j \tag{7.3}$$

Let $F(\Delta,j,\ell) = \ell^2 - (\Delta-j+1)\ell + \frac{\Delta+k}{3}(\Delta+2) - (\Delta+k)j + (j-1)j$. Then, the question

now is to find the minimum value of $F(\Delta, j, \ell)$ with constrains $\Delta, j, \ell \in \mathcal{Z}^+ \cup \{0\}$.

We will first prove that $F(\Delta, j, \ell) \geq k/3 - k^2/3 - 1/3$ for any $\Delta, j, \ell \in \mathcal{R}^+ \cup \{0\}$. For any fixed Δ and j let us first treat the function $F(\Delta, j, \ell)$ as a function of ℓ. We know that it is a parabola with the global minimum at point ℓ such that $\frac{\partial F}{\partial \ell} = 0$, which gives us that $\ell = (\Delta - j + 1)/2$.

Plugging $\ell = (\Delta - j + 1)/2$ in $F(\Delta, j, \ell)$, we obtain the following function:

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

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

Similarly as above, for any fixed Δ, we see that the function $F(\Delta, j) = \frac{3}{4}j^2 - (\Delta/2 + 1/2 + k)j + \frac{\Delta+k}{3}(\Delta + 2) - \frac{1}{4}\Delta^2 - \frac{1}{2}\Delta - \frac{1}{4}$ as a function of j is a parabola with the global minimum for j such that $\frac{\partial F}{\partial j} = 0$, which gives us that $j = \frac{2}{3}(\Delta/2 + 1/2 + k)$.

Plugging $j = \frac{2}{3}(\Delta/2 + 1/2 + k)$ in $F(\Delta, j)$ we obtain the following:

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

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

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

This proves the following theorem.

Theorem 21. *For MIS on any graph with maximum degree Δ, any greedy algorithm achieves $\frac{\Delta+2}{3}$-approximation ratio.*

7.1.2 Limitations of greedy algorithm on Δ-degree graphs

In this subsection, we present a result showing that the approximation ratio of any greedy algorithm cannot be improved for graphs with degree at most Δ. This result is an extension of Theorem 6 in [31]. In this result Halldorsson and Radhakrishnan present examples where the ratio between the worst execution of the primitive greedy of Algorithm 5 and the optimal independent set is $\frac{\Delta + 2}{3} - O(\Delta^2/n)$. However, on

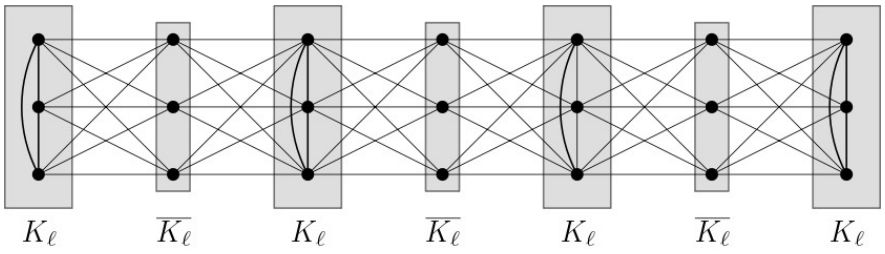


Figure 7.1: An example when $\ell = 3$. K_ℓ and $\overline{K_\ell}$ respectively denotes a clique and an independent set of size ℓ.

these examples there exists several vertices with minimum degree and picking the right minimum degree vertex could lead greedy to an optimal solution. Our extension of these examples consists in increasing the degree of some vertices by one in these graphs so that any greedy algorithm outputs a solution of the same size where the corresponding ratio is $\frac{\Delta+1}{3} - O(1/\Delta)$.

Theorem 22. *The approximation ratio of any greedy algorithm in form of Algorithm 4 for MIS on graphs with degree at most Δ is at least $\frac{\Delta+1}{3} - O(1/\Delta)$.*

Proof. We show this construction for the case $\Delta \equiv 2 \pmod 3$. See Figure 7.1. Let ℓ be the integer such that $3\ell - 1 = \Delta$. The graph consists in a chain of subgraphs, alternating with a clique on ℓ vertices and an independent set of size ℓ. Each subgraph is completely connected with the adjacent subgraphs in the chain. This structure ends with a complete graph on ℓ vertices. The degree of the vertices in the extreme clique is $2\ell - 1$, while the degree of vertices of the other cliques and the independent set are respectively $\Delta = 3\ell - 1$ and 2ℓ. Any greedy like algorithm will pick 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 output solution is

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

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

For the case $\Delta \equiv 1 \pmod 3$, we need a more complicated graph that can be described as a chain of groups of six subgraphs. Consider the integer ℓ such that $3\ell - 2 = \Delta$. Each group is formed by a chain of subgraphs of size ℓ or $\ell - 1$ that are alternately a clique and an independent set. The complete graph consists of a chain of

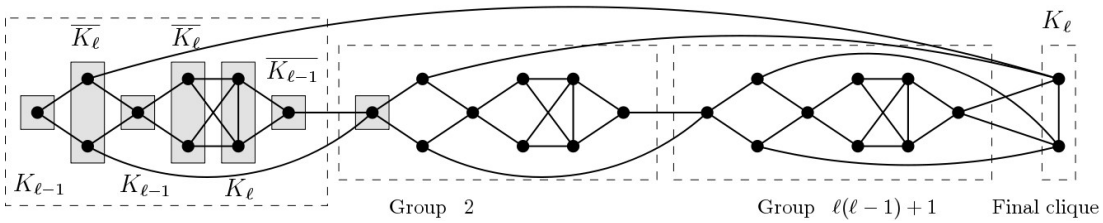


Figure 7.2: The construction when $\Delta = 3\ell - 2$.

$\ell(\ell-1)+1$ 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 ℓ fully connected with the last independent set of the last group. Additionally, we add a matching of size $\ell - 1$ between the first independent set of each group to the first clique of the next group. Because these independent sets have size ℓ, there is one unmatched vertex per each such independent set. Finally, we 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 7.2. We can see that on this graph, the maximum degree is $D = 3\ell - 2$, 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 algorithm will pick one vertex in 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 therefore

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

The case $\Delta \equiv 0 \pmod 3$ is treated similarly to 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 the 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. □

7.2 MIS on degree at most 4 graphs

We can use the same technique for degree at most 4 graph. The greedy algorithm for $\Delta = 4$ graph is the primitive greedy algorithm, but it only avoids the execution of a

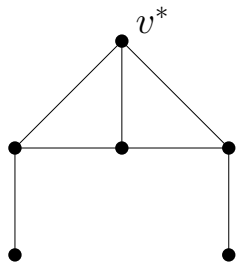


Figure 7.3: *(3,7)*-reduction

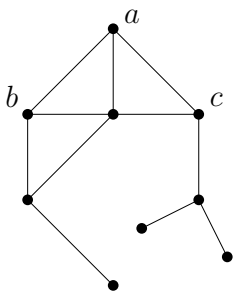


Figure 7.4: Avoiding the *(3,7)*-reduction

(3,7)-reduction.

Algorithm 10 Greedy algorithm for $\Delta = 4$ graph

Input: a graph $G = (V, E)$

$U \leftarrow V$
$S \leftarrow \emptyset$
while $U \neq \emptyset$ **do**
  Choose $v \in U$ with minimum degree in $G[U]$.
  (3,8)-reduction has higher priority than *(3,7)*-reduction.
  $U \leftarrow U \setminus V(R)$
  $S \leftarrow S \cup$ root vertices in S
end while
return S

A important observation of about the greedy algorithm of 10 is stated as follows:

Lemma 13. *The reduction* (3,7) *will not be executed by Algorithm 10.*

Proof. When the algorithm meets a *(3,7)*-reduction, it will execute the adjacent vertex with degree 3. Observe that in such a case, it will be a *(3,8)*-reduction, and it is impossible that a *(3,7)*-reduction occurs again. See Figure 7.4. Note that it is impossible that all three vertices a, b, c are root vertices of the *(3,7)*-reduction. Therefore, if there exists a *(3,7)*-reduction in the graph, then there must exists a reduction with higher priority than that of the *(3,7)*-reduction. □

Therefore, we obtain the following theorem.

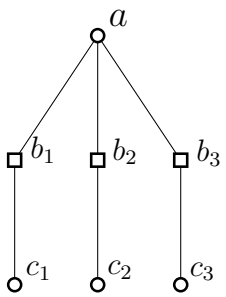


Figure 7.5: Example for greedy algorithm for minimum vertex cover

Theorem 23. *The greedy algorithm of Algorithm 10 for MIS on degree at most* 4 *graphs achieves a* $\frac{9}{5}$*-approximation ratio.*

Proof. We define the potential function as $\Phi(R) = 9|SOL(R)| - 5|OPT(R)| + |E_{loan}(R)| - |E_{debt}(R)|$. Then we can show for all reductions R, except a *(3,7)*-reduction illustrated in Figure 7.3, $\Phi(R) \geq 0$. Therefore, by the same argument as before, it proves the theorem. □

Remark. *Observe that for a graph with degree at most* $\Delta = 4$*, the ratio given in Theorem 23 is* $\frac{9}{5}$*, which is better than the ratio of the primitive greedy algorithm. The lesson learn here is that if we are able to design a sophisticated greedy algorithm by advising it to avoid choosing particular graph structures, then it might give a better approximation ratio. That is because the lower and upper bound on the number of loan and debt edges in the proof of Theorem 21 might be relaxed.*

7.3 Study for vertex cover

In the vertex cover problem, we are given a graph $G = (V, E)$. A vertex cover S is a subset of V such that each edge has at least one end vertex in S. A minimum vertex cover of G is a vertex cover of G in which the number of vertices is minimised. The goal is to find such minimum vertex cover of G. The vertex cover problem is a special case of the set cover problem.

The approach is basically same as we had for the maximum independent set problem. We want to identify which graph structures are problematic.

Theorem 24. *For any greedy algorithm for the vertex cover problem on degree at most* 3 *graphs, the approximation ratio* $r \geq \frac{4}{3}$.

Proof. See Figure 7.5. It is easy to see that the greedy algorithm will take a as the solution and then c_1, c_2, c_3. The optimum is $\{b_1, b_2, b_3\}$, which covers all the edges. Therefore, the ratio of the greedy algorithm is $r = \frac{|OPT|}{|SOL|} = \frac{4}{3}$. □

Conjecture 3. *The greedy algorithm achieves the approximation ratio $r = \frac{4}{3}$ for the vertex cover problem on degree at most 3 graphs.*

We believe that this conjecture is true and as such it would imply a fundamental fact about the greedy algorithm for the vertex cover problem. However, in the next subsection, we will present a different but closely related algorithm whose approximation ratio is strictly better than $\frac{4}{3}$, and the time complexity is as good as that of the greedy algorithm.

Remark. *In the reminder of this section, we assume that the ultimate greedy algorithm of Algorithm 9 for MIS on sub-cubic graphs obtains a $\frac{5}{4}$-approximation ratio. Therefore, some of the following results will be proved conditionally on this assumption. Our current proof shows only a $\frac{4}{3}$-approximation ratio of the ultimate greedy. But we will also present unconditional results, that use the $\frac{4}{3}$-approximation ratio of the ultimate greedy for MIS and apply it to the vertex cover problem.*

7.3.1 Complementary Greedy algorithm for vertex cover problem

In this section, we present direct applications of the greedy algorithm with the previously developed techniques. Given a graph G with degree at most 3, observe that if I is the maximum independent set of G, then $C = V(G) \setminus I$ is the minimum vertex cover of G. Also, for any maximal independent set I', $C' = V(G) \setminus I'$ is a vertex cover of G. Therefore, given any algorithm which computes a maximal independent set, if we take the complement of this set, we will obtain a vertex cover.

This is easy to prove, because a set I of vertices is an independent set if and only if every edge in the graph is adjacent to at most one member of I, and also if and only if every edge in the graph is adjacent to at least one member not in I, and thus also if and only if the complement of I is a vertex cover. Therefore, if I is the maximum independent set in G, then $C = V(G) \setminus I$ is the minimum vertex cover in G.

A natural idea is to run the ultimate greedy algorithm of Algorithm 9, and take the complement of the output solution, which will form a feasible vertex cover.

In the following subsections, we present a series of analyses to obtain algorithms for the minimum vertex cover problem with improved running time for a given approximation ratio. In subsection 7.3.3, the analysis shows that the complementary greedy

Algorithm 11 Complementary Greedy algorithm:
Require: G
Ensure: Vertex cover S
1: Run ultimate greedy algorithm of Algorithm 9 on G, get independent set I.
2: Let $S = V(G) \setminus I$.
3: S is the solution.

algorithm of Algorithm 11 obtains a $\frac{4}{3}$-approximation ratio with running time $O(n^2)$. And in subsection 7.3.4, we present an algorithm based on the complementary greedy algorithm which obtains o ration of $\frac{5}{4}$ and has running time $O(n^2)$. For a comparison, let us recall the known results for the minimum vertex cover problem on sub-cubic graphs. In [34], the author provides a $\frac{4}{3}$-approximation ratio algorithm with running time $O(n^{\frac{3}{2}})$; and in [12], they provide a $\frac{5}{4}$-approximation ratio algorithm with running time $O(n^{7.3})$ for the minimum vertex cover problem on sub-cubic graphs. Here, n denotes the number of vertices of the input graph.

Firstly, we study limitations of the complementary greedy algorithm of Algorithm 11.

Theorem 25. *No kind of complementary greedy algorithm for the vertex cover problem can achieve an approximation ratio of $\frac{8}{7}$ on sub-cubic graphs.*

Proof. Consider the example illustrated in Figure 5.5. The complementary greedy algorithm has an unique sequence of reductions S. The argument is basically the same as in the Claim 6, but the difference is now that we consider the vertex cover rather than an independent set. Let $r_{(2,6)}$ be the number of *(2,6)*-reductions in the graph, and g_2 be the number of good reductions in the graph, thus:

$$r = \frac{|SOL|}{|OPT|} = \frac{2 \cdot r_{(2,6)} + \sum_{R \in S} g_2^R}{r_{(2,6)} + \sum_{R \in S} g_2^R} \geq \frac{2 \cdot r_{(2,6)} + 8|S|}{r_{(2,6)} + 8|S|} \tag{7.4}$$

$$= \frac{2 \cdot \frac{4^k - 1}{3} + 8 \cdot 4^{k-1}}{\frac{4^k - 1}{3} + 8 \cdot 4^{k-1}} = \frac{8(4^{k-1}) - 2 + 24 \cdot 4^{k-1}}{4(4^{k-1}) - 1 + 24 \cdot 4^{k-1}} \leq \frac{8}{7}. \tag{7.5}$$

□

Theorem 26. *The complementary greedy algorithm of Algorithm 11 cannot achieve a $\frac{6}{5}$-approximation ratio for the minimum vertex cover problem on sub cubic graphs.*

Proof. Consider a graph which contains a seven-cycle, and a *(2,6)*-reduction that connects to that cycle as illustrated in Figure 5.8. If the algorithm executes the top *(2,6)*-reduction, then the solution of the algorithm has size 5. However, the optimum size is 6. □

7.3.2 Naive analysis for $\frac{7}{5}$-approximation ratio

In this section, we present a simple analysis for proving that complementary algorithm has an $\frac{7}{5}$ approximation ratio under the assumption that the ultimate greedy algorithm for MIS obtains a $\frac{5}{4}$-approximation ratio. This results by itself is less significant, but it provides the idea of how to use the complementary property between the maximal independent set and the minimal vertex cover.

Theorem 27. *Under the assumption that the ultimate greedy algorithm of Algorithm 9 obtains a $\frac{5}{4}$-approximation ratio for MIS, the complementary greedy algorithm of Algorithm 11 achieves an $\frac{7}{5}$-approximation ratio with running time $O(n^2)$ for the minimum vertex cover problem on sub-cubic graphs.*

Proof. Let $G = (V, E)$ and G be sub-cubic. Denote $|E| = m$. Let I be an independent set in G found by the ultimate greedy algorithm, which we assume is a $\frac{5}{4}$-approximation to MIS on G. Let I^* be the maximum independent set in G. Thus, we have $|I| \geq \frac{4}{5} \cdot |I^*|$. Note that I is a maximal independent set, because the greedy algorithm always outputs a maximal independent set. Therefore, $S = V(G) \setminus I$ is a vertex cover in G. Then, we have:

$$|S| = n - |I| \leq n - \frac{4}{5}|I^*| = n - \frac{4}{5}(n - |S^*|) = n - \frac{4}{5} \cdot n + \frac{4}{5} \cdot |S^*| = \frac{n}{5} + \frac{4}{5} \cdot |S^*|$$

If $|S^*| \geq \frac{m}{3}$, then $|S^*| \geq \frac{n}{3}$, the claim that we prove below. Therefore,

$$S = \frac{n}{5} + \frac{4}{5} \cdot |S^*| \leq \frac{3}{5} \cdot |S^*| + \frac{4}{5} \cdot |S^*| = \frac{7}{5} \cdot |S^*|$$

This proves that the complementary greedy algorithm achieves a $\frac{7}{5}$-approximation ratio for the vertex cover problem on sub-cubic graphs.

Now, we will prove the claim $|S^*| \geq \frac{m}{3}$. We write the linear programming relaxation of the vertex cover problem and its dual linear program.

$$\begin{aligned} \min \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_u + x_v \geq 1, && \forall e = (u, v) \in E \\ & x_v \geq 0, && \forall v \in V \end{aligned}$$

$$\begin{aligned} \max \quad & \sum_{e \in E} y_e \\ \text{s.t.} \quad & \sum_{u \in N(v)} y_{(v,u)} \leq 1, && \forall v \in V \\ & y_{(v,u)} \geq 0, && \forall e \in E \end{aligned}$$

Note that $\sum_{v \in V} x_v$ will be a fractional vertex cover. And for the dual linear program, assigning each variable $y_{(u,v)} = \frac{1}{3}$ is a feasible dual solution. Under such an assignment, $\sum_{e \in E} y_e = \frac{m}{3}$. By the weak duality theorem of linear programming, we have

$$|S^*| \geq \min \sum_{v \in V} x_v \geq \max \sum_{e \in E} y_e \geq \frac{m}{3}$$

This proves the claim we needed. □

7.3.3 Sophisticated analysis for $\frac{4}{3}$-approximation ratio

In this section, we present a more sophisticated analysis for the complementary greedy algorithm which achieves an even better approximation ratio.

Theorem 28. *Under the assumption that the ultimate greedy algorithm of Algorithm 9 obtains $\frac{5}{4}$-approximation ratio for MIS, the complementary greedy algorithm of Algorithm 11 achieves a* 1.25*-approximation ratio with running time $O(n^2)$, for the minimum vertex cover problem on sub-cubic graphs.*

For the vertex cover problem, we define a potential function $\Psi(R)$ in analogy to what we did in Chapter 6 for the MIS problem.

Firstly, we present the definition of reductions in the vertex cover problem. A reduction R executed by the complementary greedy algorithm on graph G is defined as $R = \{V_R, E_R\}$ and it is exactly the same as the reduction in MIS problem. Note

however that the difference is that the vertex which is in C is black, and the vertex which is not in C is white.

We present a definition of a *loan* and *debt* edges for the vertex cover problem. The loan edge e of a reduction R is an edge such that $e \in E_{contact}$ and $N(e) \cap V_R$ is white; and a debt edge of R is an edge e such that $e \in E_{past}$ and $N_G(e) \setminus V_R$ is white. The situation is the same as in the MIS problem, that is for a sequence of reductions S executed by the complementary algorithm on graph G, we have: $\sum_{R\in S} \sum_{e\in E_{loan}(R)} 1 = \sum_{R\in S} \sum_{e\in E_{debt}(R)} 1$ or $\sum_{R\in S} E_{loan}(R) = \sum_{R\in S} E_{debt}(R)$.

Then, we define the potential function $\Psi(R)$ for the vertex cover problem:

$$\Psi(R) = 4|SOL(R)| - 5|OPT(R)| - |E_{loan}(R)| + |E_{debt}(R)| \tag{7.6}$$

By an argument analogous to that for the MIS problem, we have:

$$\Psi(S) = \sum_{R\in S} \Psi(R) = \sum_{R\in S} (4|SOL(R)| - 5|OPT(R)|) - |E_{loan}(S)| + |E_{debt}(S)| \tag{7.7}$$

$$= \sum_{R\in S} (4|SOL(R)| - 5|OPT(R)|) \tag{7.8}$$

Therefore, if we can prove for any S executed on a graph G such that $\Phi(S) \leq 0$, then

$$\sum_{R\in S} (4|SOL(R)| - 5|OPT(R)|) \leq 0$$

$$SOL(S) \leq \frac{5}{4} OPT(S)$$

This proves the $\frac{5}{4}$-approximation ratio.

To finalise our proof, we need to prove the following lemma. For clarity, we use an upper index to refer to the different terms in $\Phi(R)$ and $\Psi(R)$.

Lemma 14 (The Duality Lemma)**.** *For all reductions R except an isolated vertex reduction, $-\Phi(R) \geq \Psi(R)$.*

Proof. Recall the definition of the potential functions:

$$\Phi(R) = 5|SOL^{\Phi}(R)| - 4|OPT^{\Phi}(R)| + |E^{\Phi}_{loan}(R)| - |E^{\Phi}_{debt}(R)|,$$

and

$$\Psi(R) = 4|SOL^{\Psi}(R)| - 5|OPT^{\Psi}(R)|) - |E^{\Psi}_{loan}(R)| + |E^{\Psi}_{debt}(R)|.$$

For any reduction R, it is easy to check the following statement by the definition of loan edges and debt edges: $|E^{\Phi}_{loan}(R)| = |E^{\Psi}_{loan}(R)|$ and $|E^{\Phi}_{debt}(R)| = |E^{\Psi}_{debt}(R)|$.

Then, we have:

$$-|E^{\Psi}_{loan}(R)| + |E^{\Psi}_{debt}(R)| = -(+|E^{\Phi}_{loan}(R)| - |E^{\Phi}_{debt}(R)|)$$

Therefore, if we can show for any reduction R, that we have:

$$5|SOL^{\Phi}(R)| - 4|OPT^{\Phi}(R)| \leq -(4|SOL^{\Psi}(R)| - 5|OPT^{\Psi}(R)|),$$

then we prove what we desire.

Observe that following claims: $|SOL^{\Phi}(R)| = |V_R| - |SOL^{\Psi}(R)|$ and $|OPT^{\Phi}(R)| = |V_R| - |OPT^{\Psi}(R)|$. Thus:

$$5|SOL^{\Phi}(R)| - 4|OPT^{\Phi}(R)| \leq -(4|SOL^{\Psi}(R)| - 5|OPT^{\Psi}(R)|)$$

$$5|SOL^{\Phi}(R)| - 4|OPT^{\Phi}(R)| \leq -(4(|V_R| - |SOL^{\Phi}(R)|) - 5(|V_R| - |OPT^{\Phi}(R)|))$$

$$|SOL^{\Phi}(R)| + |OPT^{\Phi}(R)| \leq |V_R| \tag{7.9}$$

Observe that $|V_R| \leq 3$, $|SOL^{\Phi}(R)| = 1$ and $|OPT^{\Phi}(R)| \leq 2$ for any reduction R. These hold by the following checks:

1. for a single edge reduction R, $|V_R| = 2$, and $|SOL^{\Phi}(R)| + |OPT^{\Phi}(R)| \leq 2$.
2. for a triangle reduction R, $|V_R| = 3$, and $|SOL^{\Phi}(R)| + |OPT^{\Phi}(R)| \leq 2$.
3. for a branching reduction R, $|V_R| = 3$, and $|SOL^{\Phi}(R)| + |OPT^{\Phi}(R)| \leq 3$.

Thus, the inequality (7.9) holds, which concludes the proof of the lemma. □

To complete the proof, we need to show that the isolated vertex reduction does not affect our argument.

Claim 16. *For an isolated vertex reduction R, we have $\Psi(R) \leq 0$.*

Proof. If v^* is black, then $\Psi(R) \leq -2$. If v^* is white, then $\Psi(R) = 0$. □

Note that for the case of v^* being white, $\Psi(R) = 0 < 1 = \Phi(R)$. This explains the reason why the dual lemma does not apply to all reductions.

Proof. (of Theorem 28) For any reduction R, if $\Psi(R) = k > 0$, then $\Phi(R) = j < 0$ and $-j \geq k$. And for every reduction such that $\Phi(R) < 0$, if such a reduction is executed by the complementary greedy algorithm, then we have proved there exist unique savings to pay for it. And thus $\Phi(S) \geq 0$ when we consider the MIS problem. And by the Duality Lemma 14, the reductions which have these savings in the MIS problem also have enough savings for the corresponding reductions in the vertex cover problem. Therefore, $\Psi(S) \leq 0$. This proves Theorem 28. □

If we remove the assumption about the 5/4-approximation of the ultimate greedy for MIS, and we apply the approximation ratio of $\frac{4}{3}$ rather than $\frac{5}{4}$ of Algorithm 9 from Theorem 20, then we are able to obtain following theorem.

Theorem 29. *The complementary greedy algorithm of Algorithm 11 achieves a $\frac{4}{3}$-approximation ratio with running time $O(n^2)$, for the minimum vertex cover problem on sub-cubic graphs.*

Although the analysis in here shows a worse running time compared to $O(n^{\frac{3}{2}})$ by [34], however, we strongly believe that we are able to obtain a refined ultimate greedy algorithm of Algorithm 9, whose the running time will be reduced to $O(n)$. Therefore, in such a case, we could improve the running time from $O(n^{\frac{3}{2}})$ to linear time.

7.3.4 Further analysis for $\frac{5}{4}$-approximation ratio

If the ultimate greedy algorithm of 9 obtains a $\frac{5}{4}$-approximation ratio for the MIS problem, then we are able to use a Nemhauser-Trotter technique to obtain a $\frac{6}{5}$-approximation ratio algorithm for the vertex cover problem. In [5], they claim that there is a $\frac{7}{6}$-approximation algorithm for the minimum vertex cover problem in sub-cubic graphs by using a $\frac{6}{5}$ approximation ratio algorithm for MIS for sub-cubic graph. They only outline a proof of the this fact but do not provide the full proof. We provide here a complete proof of the essentially same claim but with different $\frac{6}{5}$ ratio, under the assumption that the ultimate greedy for MIS achieves a $\frac{5}{4}$ approximation ratio.

Firstly, we present the Nemhauser-Trotter technique.

Theorem 30 (Nemhauser-Trotter [54])**.** *For any graph $G = (V, E)$, there is a way to compute a partition $\{V_1, V_2, V_3\}$ of V with time complexity of the bipartite matching problem, such that:*

1. *there is a maximum independent set I containing all of the nodes of V_1 but none of V_2,* i.e. *$I \cap V_1 = V_1$ and $I \cap V_2 = \emptyset$,*

2. *there is no edge between V_1 and V_3,* i.e. *$N(V_1) \subseteq V_2$,*

3. *$\alpha(G(V_3)) \leq \frac{1}{2}|V_3|$.*

Note that if graph G is sub-cubic, then the running time of computing such a partition is $O(n^{\frac{3}{2}})$.

Therefore, the algorithm would first execute the Nemhauser-Trotter reduction on the original graph G, and then run the complementary greedy algorithm on $G(V_3)$ and obtain a solution of S. The final solution would be the set $V_2 \cup S$.

Theorem 31. *Under the assumption of the ultimate greedy algorithm of Algorithm 9 obtains a $\frac{5}{4}$-approximation ratio for MIS, the algorithm which combines the Nemhauser-Trotter reduction with the complementary greedy algorithm of Algorithm 11 achieves a $\frac{6}{5}$-approximation ratio with running time $O(n^2)$, for the minimum vertex cover problem on sub-cubic graphs.*

Proof. We apply the Nemhauser-Trotter reduction from Theorem 30 to $G = (V, E)$, and V is partitioned into V_1, V_2, V_3. Then we run the $\frac{5}{4}$-approximation greedy algorithm on $G_3 = G(V_3)$, and we choose the complement of the independent set J output by the algorithm. Then let $C_3 = V_3 \setminus J$ denote the resulting vertex cover in G_3. Let also C^* denote a minimum vertex cover of G, and I^* be a maximum independent set in G.

Observe that for (any) graph G_3, we have that $V(G_3) \setminus I^*$ is the complement of a maximum independent set on G_3, so its size is equal to the size of the minimum size of a vertex cover in G_3. Analogously, let I_3^* be a maximum independent set in G_3 and $C_3^* = V_3 \setminus I_3^*$ be a minimum vertex cover in G_3.

Thus, we have: $|C^*| = |V_2| + |C_3^*| = |V_2| + (|V_3| - \alpha(G_3))$ by Theorem 30. The vertex cover of G_3 computed by the algorithm is $C_3 = V_3 \setminus J$. By Theorem 20, we have: $|J| \geq \frac{4}{5}\alpha(G_3)$. By Theorem 30, $\alpha(G_3) \leq \frac{|V_3|}{2}$, thus, $0 \leq \frac{|V_3|}{5} - \frac{2}{5}\alpha(G_3)$, and

$$|C_3| \leq |V_3| - \frac{5}{4}\alpha(G_3) \leq |V_3| - \frac{4}{5}\alpha(G_3) + \frac{|V_3|}{5} - \frac{2\alpha(G_3)}{5} = \frac{6}{5} \cdot (|V_3| - \alpha(G_3)).$$

Our algorithm outputs $C = V_2 \cup C_3$ as the vertex cover in G, and we have that

$$|C| = |V_2| + |C_3| \leq \frac{6}{5}|V_2| + \frac{6}{5}|C_3^*| \leq \frac{6}{5}|C^*|,$$

and this concludes the proof of the approximation guarantee. The running time bound follows from Theorems 29 and 30. □

Without the assumption that the ultimate greedy algorithm obtains a $\frac{5}{4}$-approximation ratio for MIS, but only a $\frac{4}{3}$-approximation, we obtain the following theorem.

Theorem 32. *The complementary greedy algorithm of Algorithm 11 with Nembauser-Trotter reduction achieves a $\frac{5}{4}$-approximation ratio with running time $O(n^2)$, for the minimum vertex cover problem on sub-cubic graphs.*

Proof. The proof is the same as that of Theorem 31, but in this proof, we only apply the approximation ratio of $\frac{4}{3}$ rather than $\frac{5}{4}$ of the ultimate greedy algorithm of Algorithm 9 from Theorem 20. □

Remark. *Theorem 32 shows the existence of an algorithm which obtains the best currently known running time of $O(n^2)$ with an approximation ratio of $\frac{5}{4}$ for the minimum vertex cover problem on sub-cubic graphs.*

7.4 Conclusion

In this chapter, we have obtained a series results. We extended our method to MIS on general bounded degree graphs, and have given an alternative and simple proof of the known $\frac{\Delta+2}{3}$-approximation ratio for MIS on graphs with maximum degree Δ. Also, we have presented an 1.8-approximation ratio greedy algorithm for MIS on degree at most 4 graphs. Furthermore, we presented an interesting algorithm which is based on our greedy algorithm for MIS, which obtains a $\frac{6}{5}$-approximation ratio for the minimum vertex cover problem on sub-cubic graphs.

Chapter 8

Heuristic and experimental study for MIS

We first note that the experimental analysis presented in this section is preliminary and it only supplements the theoretical results that are proved in the previous chapters.

Although we have proved in the previous chapter better results for the greedy algorithm for the maximum independent set problem on degree at most 3 graphs, such theoretical study only applies to the worst case analysis. It provides very few insights in terms of practical use of such algorithms. Thus an experimental study of how the greedy algorithm performs in practice is conducted here to reveal some connections to the theoretical study. The experimental study shows that even for the primitive greedy algorithm 5, which in each iteration, arbitrarily chooses one of the minimum degree vertices v and removes it and all neighbour vertices of v from the graph, without any specific advice of order, in almost every "realistic" input, its approximation ratio is much better than 1.25. For the updated greedy algorithm of Algorithm 8, the experiments show that it obtains significantly better ratios as we have expected.

This phenomenon is not surprising, because the tight lower bound examples illustrated in Figure 5.5 and others which are not presented, are highly restricted by theirs peculiar graph structure. More essentially, they are restricted by the specific sequence of reductions which consists of particular type of reductions after an execution of a bad reduction. Any slight disturbance of such graph structure will change the order of greedy execution significantly, and basically, such disturbance rather improves the performance than attenuates it. As shown in Chapter 6, each bad reduction *(2,6)* will receive four units of savings from somewhere in the sequence of reductions. That im-

plies that the good reductions which offer these savings in the sequence of reductions must exactly satisfy this correspondence to the bad reductions. Intuitively, the number of instances where such a precise situation occurs should be rare within the set of all possible instances. To characterise such phenomenon, we need to develop some different kind of arguments, but it will depend on the techniques that we developed in the previous chapter.

Observe that for any sequence of reductions S executed by a given greedy algorithm on G, the problematic type of reductions are only bad odd-backbone, even-backbone and isolated odd cycle reductions. The remaining type of reductions, including an isolated vertex reduction, single edge and single edge branching, good odd-backbone, even-backbone, and an isolated even cycle reduction, are good. As we have already shown, in S, if there only exist these good reductions $R \in S$, which means that there are no debt edge which really requires a payment, then the greedy will provide an optimal solution. This implies that if the number of good reductions is much larger than the number of bad reductions, then if the difference between these two numbers increases, the approximation ratio should decrease and it should tend to 1.

Therefore, we are curious about the following question:

Question 1. *What is the distribution of each type of reduction in the sequence of reductions executed by a given greedy algorithm?*

For an initial study, we implement an experimental approach in order to answer this question. The experiment is conducted by the following steps:

1. **Instance generation:** The designed program randomly generates a graph G with degree at most 3. The generation of this graph is as follows: the program determines the number of vertices of an current instance by sampling a number from a uniform distribution from $U[50, 150]$. For each pair of vertices (v, v') in V^2 chosen uniformly at random, the program assigns an edge between v and v' with probability p. If one of the vertices v and v' in the pair already has degree 3, then the program will not assign an edge between these two vertices. Note that if the probability p is set to be higher, than the graph will be denser.

2. **Optimum computation:** The program will compute an optimum solution OPT on an instance G by using integer linear programming (ILP). An integer linear programming formulation of the maximum independent set problem is the standard one. An ILP solver does not only provide the size of OPT, but also provides which vertices in G belong to the set OPT. Thus, after the program obtains the

optimum solution, it will label G: for each vertex v, if $v \in OPT$, a label will be given to v.

3. **Execution of the greedy:** The program will execute the given greedy algorithm on such labeled graph G. It will record the number of each type of reductions according to the vertices' labels. The experiment considers bad, 1-good and 0-good reductions among all kinds of reductions.

4. **Experimental analysis:** The program repeat this process from 1 until it collects enough data. Then, we will compute a ratio of each type of reductions.

We consider two kinds of greedy algorithms, the primitive greedy algorithm (Algorithm 5) and the updated greedy algorithm (Algorithm 8). For each type of reductions $R \in \mathcal{R}$, let $N(R)$ be the number of occurrences of this reduction in the experiment. Thus, each cell in Table 8.1 and 8.2 contains the percentage of $N(R)/\sum_{R\in\mathcal{R}} N(R)$.

We run both greedy algorithms for different choices of the probability p. We consider five cases, where $p_i = (2 + 0.2i)/|V|$, where $i \in \{1, 2, 3, 4, 5\}$. Note that if i is smaller, then the graph would be sparser, and denser otherwise. The expected average degree of such a generated graph is $|V| \cdot (1 + 0.5i)/|V| = 1 + 0.5i$. Because, if the maximum of degree of G is 2, then the greedy algorithm always finds the optimum, for our experiment, we start with the expected average degree from at least 2. Also, because the maximum degree of G is 3, it is reasonable to set the expected average degree to be 3. For each value of the probability p, we generate and test 10000 instances and 50000 instances in total for each of the greedy algorithms.

8.1 Results and discussion

The results are presented in Table 8.1 and 8.2, and illustrated in the bar chart 8.1 and 8.1. The average approximation ratio is 1.0227 for the primitive greedy and 1.0083 for the updated greedy algorithm.

We can observe a presence of a significant number of *(1,2)* and *(1,3)*-reductions, for both algorithms, as their proportion is about 67% of the total number of reductions. This result is not very surprising, and it can be roughly deduced from the following observation.

Firstly, triangle reductions such as *(2,3)*, *(2,4)-1* and *(2,5)-1*-reduction are rare, because if such a reduction is executed, it implies that in the original graph, such triangle is already present, and the probability to form a 3-clique is low.

Secondly, observe that for each bad reduction, we can use a 0-good reduction to balance its demand for payment. In general, a 0-good reduction contributes 1 to the size of the solution and 0 to the size of the optimum. And a bad reduction contributes 1 to the size of the solution and 2 to the size of the optimum. Then, together, they contribute 2 to the size of the solution and 2 to the size of the optimum, which is equivalent to two 1-good reductions. Therefore, let R_b be a bad reduction, R_0 be 0 a good reduction and R_1 be a 1-good reduction found in an experiment. Then $2(N(R_b) - N(R_0) + N(R_1))$ would be equivalent to the total number of 1-good reductions. By the experiment, the ratio between the number of 1-good and bad reductions is: for the primitive greedy, $r = 2.56\%$, and for the updated greedy, $r = 1.2\%$.

Although, the approximation ratio of the updated greedy algorithm is better as expected, the main difference between the primitive greedy algorithm and the updated greedy algorithm seems to be that for the former, the number of bad reductions is smaller, and the number of 0-good reductions is also smaller. For the latter algorithm, the number of bad reductions is not only larger but it is significantly larger.

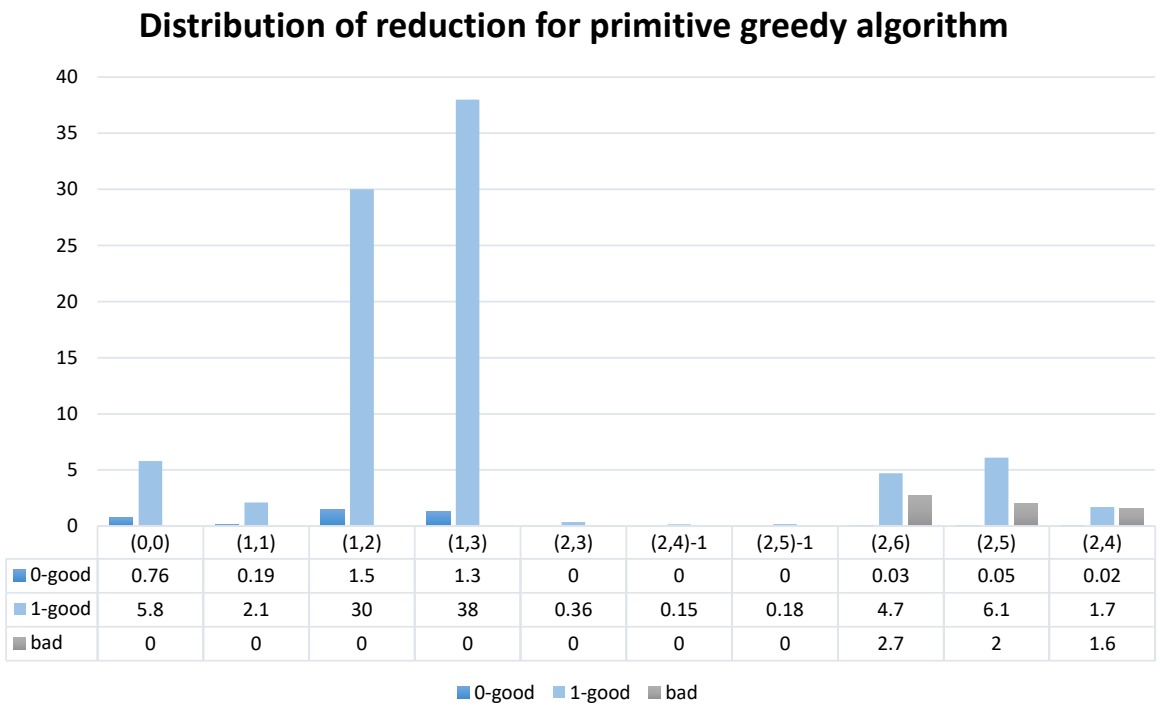
Thirdly, for the primitive greedy algorithm, the number of *(1,3)* reductions of both 1-good and 0-good type is approximately 8.2% larger than that of *(1,2)*-reductions. But for the updated greedy algorithm, this relation is opposite, that is, the number of *(1,2)*-reductions is about 8.4% lager than that of *(1,3)*-reductions.

	0-good	1-good	bad
(0,0)	0.76	5.8	$\emptyset$
(1,1)	0.19	2.1	$\emptyset$
(1,2)	1.5	30	$\emptyset$
(1,3)	1.3	38	$\emptyset$
(2,3)	0.00	0.36	$\emptyset$
(2,4)-1	0.00	0.15	$\emptyset$
(2,5)-1	0.00	0.18	$\emptyset$
(2,6)	0.03	4.7	2.7
(2,5)	0.05	6.1	2.0
(2,4)	0.02	1.7	1.6

Table 8.1: Results of experiments for the primitive greedy algorithm.

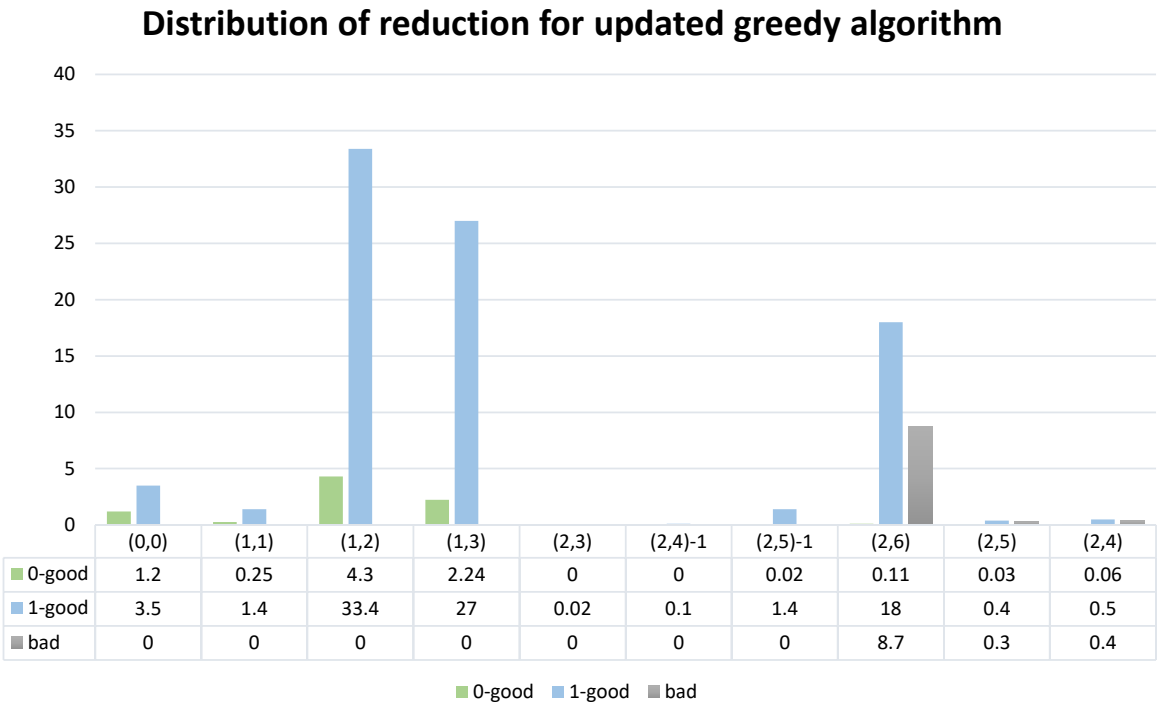
	0-good	1-good	bad
(0,0)	1.2	3.5	$\emptyset$
(1,1)	0.25	1.4	$\emptyset$
(1,2)	4.3	33.4	$\emptyset$
(1,3)	2.24	27.0	$\emptyset$
(2,3)	0.00	0.02	$\emptyset$
(2,4)-1	0.00	0.1	$\emptyset$
(2,5)-1	0.02	1.4	$\emptyset$
(2,6)	0.11	18	8.7
(2,5)	0.03	0.4	0.3
(2,4)	0.06	0.5	0.4

Table 8.2: Results of experiments for the updated greedy algorithm.



	(0,0)	(1,1)	(1,2)	(1,3)	(2,3)	(2,4)-1	(2,5)-1	(2,6)	(2,5)	(2,4)
0-good	0.76	0.19	1.5	1.3	0	0	0	0.03	0.05	0.02
1-good	5.8	2.1	30	38	0.36	0.15	0.18	4.7	6.1	1.7
bad	0	0	0	0	0	0	0	2.7	2	1.6

Figure 8.1: The performance of primitive greedy algorithm.



	(0,0)	(1,1)	(1,2)	(1,3)	(2,3)	(2,4)-1	(2,5)-1	(2,6)	(2,5)	(2,4)
0-good	1.2	0.25	4.3	2.24	0	0	0.02	0.11	0.03	0.06
1-good	3.5	1.4	33.4	27	0.02	0.1	1.4	18	0.4	0.5
bad	0	0	0	0	0	0	0	8.7	0.3	0.4

Figure 8.2: The performance of the updated greedy algorithm.

Chapter 9

Conclusions and further study

9.1 Conclusions

In this thesis, we studied the new model of mechanism design in context of ontologies. We showed negative and positive results on the approximation ratios of truthful mechanisms in this setting. We moreover showed upper and lower bounds on the price of anarchy and stability of the Nash implementation of a greedy mechanism in this setting.

Moreover, we studied the inapproximability of the greedy algorithms for the maximum size independent set problem (MIS) on general graphs and in particular on planar cubic graphs. The main and strongest contribution of the thesis is a development a series of effective techniques to prove the approximation ratios of greedy algorithms for the MIS problem. The specific achievement is the proof of existence of a $\frac{4}{3}$-approximation greedy algorithm for MIS on sub-cubic graphs. This result does not only improve on the current best known ratio of greedy algorithms, but it provides a very precise analysis of the greedy. This new methodology holds a promise to help us obtain even better approximation ratios, such as $\frac{5}{4}$-ratio of the greedy algorithm. Based on the techniques we developed, some further applications have also been studied. We proved a $\frac{\Delta+2}{3}$-approximation ratio for any greedy algorithm for MIS on bounded degree Δ graphs. We showed an improved $\frac{9}{5}$-approximation ratio for the $\Delta = 4$ case. We have also used a complement property between the maximal independent set and the minimum vertex cover to show a $\frac{5}{4}$-approximation ratio for the minimum vertex cover problem on sub-cubic graphs. Finally, We conducted an experimental study to explore the average approximation ratio of the greedy algorithm on sub-cubic graphs for both practical and theoretical purposes.

9.2 Further study

9.2.1 Study for ontology mechanism design

The main goal in the future about the ontologies matching game is to prove that the price of stability is 2.

Question 2. *Is it possible to prove that the ontologies matching game has the price of stability of* 2*?*

Since we already observed that there are only few instances which admit pure Nash equilibrium, the initial idea is to characterise in which instances pure Nash equilibrium exists. Then, among these instances, we can start to characterise a sub-class of these instances, whose structure and properties are nice enough for us to be able to prove the price of stability of 2. Then, we might try to extend these results from such sub-class of instances to general cases.

9.2.2 Study for maximum independent set and minimum vertex cover problems

Although the techniques developed in Chapter 6: *Payment scheme with potential function* and *Backward inductive method* already have proven their usefulness with addressing the maximum independent set (MIS) and the minimum vertex cover (MVC) problems, their potential is far from being fully explored. There are numerous intriguing directions.

The most promising work for the future is to prove that the ultimate greedy algorithm of Algorithm 9 is really ultimate. As we observed in Section 6.3.4, it very likely to be possible to obtain the analysis of $\frac{5}{4}$-approximation ratio for the ultimate greedy algorithm of Algorithm 9. The only obstacle is how to address a problem that occurs in the analysis of the odd-backbone reductions in our inductive proof. We have two directions to explore here. The first promising approach is to extend the potential function to $\Psi(R) = \Phi(R) + \mathcal{L}(R)$, where $\mathcal{L}$ is an extra parameter to measure the reduction. Informally, if a reduction R contains a "blocking" structure, then $\mathcal{L}(R) = 1$. And for any reduction R, where its contact edges "remove" such "blocking" structures, if the number of "blocking" structures it "removes" is k, then $\mathcal{L}(R) = -k$. Note that then for a sequence of reductions S on graph G, $\sum_{R \in S} \mathcal{L}(R) = 0$. The second promising approach is to explore the power of greedy advice further. It might be possible to design a new greedy advice or find out a new way to use the current advice to preserve the

inducting hypothesis that is required. For example, we do not use the greedy advice to deal with odd-backbone reduction, which might be helpful.

Question 3. *Does the ultimate greedy algorithm obtain a $\frac{5}{4}$-approximation ratio with running time $O(n)$ for the MIS problem on sub-cubic graphs?*

The second direction for the future is to study the performance of greedy algorithms for MIS on bounded degree graphs. Although Theorem 21 implies that the primitive greedy algorithm cannot be improved to obtain a better approximation ratio than $\frac{\Delta+2}{3}$, Theorem 22 implies that the lower bound for any greedy algorithm is only $\frac{\Delta+1}{3} - O(\frac{1}{\Delta})$. Thus, there exists a chance to improve the approximation ratio, if the greedy algorithm is advised properly. Therefore, we aim to answer the following question.

Question 4. *Is there an ultimate greedy algorithm that obtains a $\frac{\Delta+1}{3}$-approximation ratio for MIS problem on Δ-bounded degree graphs?*

Moreover, observe in Theorem 25, that for any complementary greedy algorithm, the approximation ratio cannot be improved to be better than $\frac{8}{7}$. And even if we are able to prove that the ultimate greedy algorithm obtains a $\frac{5}{4}$-approximation ratio and thus the complementary greedy algorithm of Algorithm 11 obtains a $\frac{6}{5}$ ratio, there still exists a huge gap between the upper and lower bounds.

Also observe that in the analysis of the complementary greedy algorithm, it seems that a tighter analysis might be feasible to obtain a $\frac{8}{7}$-approximation ratio for the MVC problem on sub-cubic graphs. Note that the potential value in the vertex cover problem is in general better than that for the MIS problem. Therefore, with a carefully designed advice and greedy order, a tighter approximation ratio might be possible. Therefore, we aim to answer the following question.

Question 5. *Is there a complementary greedy algorithm that obtains a $\frac{8}{7}$-approximation ratio for the MVC problem on sub-cubic graphs? Or is it possible to find a counter-example to show a larger lower bound for the complementary greedy algorithm?*

Furthermore, in chapter 8, we conducted an experimental study on the power of two kinds of greedy algorithms on various instances. The results show that for both primitive and updated greedy algorithm, the observed average approximation ratios are extremely good. The interpretation of such phenomenon is as follows. Let $R^b_G = \{R \in S_G | R \text{ be a bad reduction}\}$, and $\mathcal{G}$ be the set of all sub-cubic graphs. Then for all G in $\mathcal{G}$ with $|V(G)| \leq k$, consider the sequence of reductions S_G executed by the greedy algorithm on G. Then the total number of bad reductions in $\bigcup_{G\in\mathcal{G}} S_G$ should

be much smaller than the number of good reductions in $\bigcup_{G\in\mathcal{G}} S_G$. Therefore, we aim to answer the following question.

Question 6. *How to theoretically prove that* $\sum_{G\in\mathcal{G}} |R_G^b| \leq \alpha \cdot \sum_{G\in\mathcal{G}} |S_G|$, *for any* $\alpha \leq 1$*?*

Finally, the ultimate goal of further study is to design a non-greedy algorithm which obtains a better approximation ratio than $\frac{5}{4}$.

Question 7. *Is there a non-greedy algorithm that obtains a* $\frac{6}{5}$ *or* $\frac{7}{6}$*-approximation ratio with running time* $O(n^2)$ *for the MIS problem on sub-cubic graphs?*

We believe that our technique of super-advice will be very useful towards reaching this goal.

Appendix A

A.1 Graph structure of extended reductions

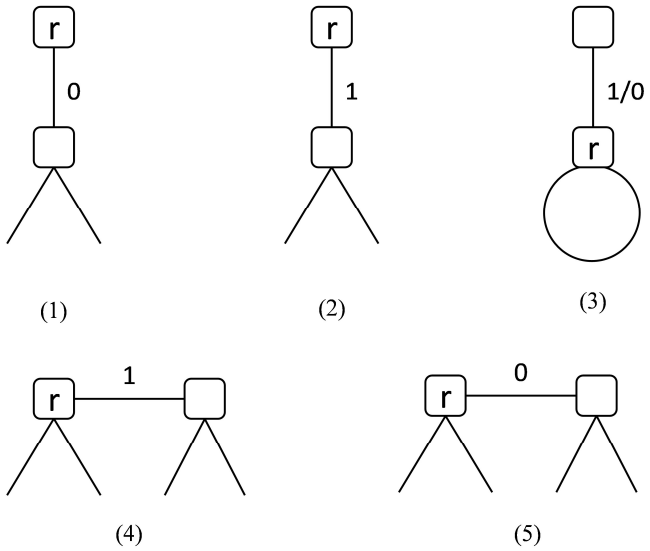


Figure A.1: Collections of extended reductions

We give an interpretation of extended reductions in Figure A.1. For details see Definition 17.

(1) Single edge non-branching reduction consist of a series of *(1,2)*-reductions $R_1, \ldots, R_k$, where $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

(2) Single edge branching reduction consist of a series of reductions $R_1, \ldots, R_k$ in which R_1 to R_{k-1} are *(1,2)*-reductions and R_k is a *(1,3)*-reduction. Moreover $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

(3) Loop reduction consist of a series of reductions $R_1, \ldots, R_k$, in which R_1 is a *(2,5)*-reduction, R_2 to R_{k-1} are *(1,2)*-reductions, and R_k is a *(1,1)*-reduction, where $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$, and $E^{R_1}_{contact} \cap E^{R_k}_{past} \neq \emptyset$.

(4) Even-backbone reduction is either a *(2,6)*-reduction, or it consists of a series of reductions $R_1, \ldots, R_k$, where R_1 is a *(2,5)*-reduction, R_1 to R_{k-1} are *(1,2)*-reductions, and R_k is a *(1,3)*-reduction. Moreover, $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

(5) Odd-backbone reduction consists of a series of reductions $R_1, \ldots, R_k$, where R_1 is a *(2,5)*-reduction, R_1 to R_k are *(1,2)*-reductions. Moreover, $E^{R_i}_{contact} \cap E^{R_j}_{past} \neq \emptyset$ for every $i = j - 1$.

Bibliography

[1] Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.*, 237(1-2):123–134, 2000.

[2] Michael Anslow and Michael Rovatsos. Aligning experientially grounded ontologies using language games. In *Graph Structures for Knowledge Representation and Reasoning - 4th International Workshop, GKR 2015, Buenos Aires, Argentina, July 25, 2015, Revised Selected Papers*, pages 15–31, 2015.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *Proceedings., 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, Oct 1992.

[4] Manuel Atencia and W. Marco Schorlemmer. An interaction-based approach to semantic alignment. *J. Web Semant.*, 12:131–147, 2012.

[5] P. Berman and T. Fujito. On approximation properties of the independent set problem for low degree graphs. *Theory of Computing Systems*, 32(2):115–132, Apr 1999.

[6] Piotr Berman and Martin Fürer. Approximating maximum independent set in bounded degree graphs. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, pages 365–371, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[7] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(May):35–43, 201.

[8] Hans L Bodlaender, Dimitrios M Thilikos, and Koichi Yamazaki. It is hard to know when greedy is good for finding independent sets. *Information Processing Letters*, 61(2):101–106, 1997.

[9] Ravi Boppana and Magnús M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT Numerical Mathematics*, 32(2):180–196, Jun 1992.

[10] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 39–48, New York, NY, USA, 2005. ACM.

[11] Miroslav Chlebik and Janka Chlebikova. Inapproximability results for bounded variants of optimization problems. pages 27–38, 12 2003.

[12] Miroslav Chlebík and Janka Chlebíková. On approximability of the independent set problem for low degree graphs. In Ratislav Královic and Ondrej Sýkora, editors, *Structural Information and Communication Complexity*, pages 47–56, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[13] Paula Chocron and Marco Schorlemmer. Vocabulary alignment in openly specified interactions. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1064–1072, 2017.

[14] DAML-S Coalition:, A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. DAML-S: Web Service Description for the Semantic Web. In *First International Semantic Web Conference (ISWC) Proceedings*, pages 348–363, 2002.

[15] Keith Decker, Katia Sycara, and Mike Williamson. Middle-Agents for the Internet. In *IJCAI97*, 1997.

[16] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, Dec 1959.

[17] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 610–618. ACM, 2005.

[18] Shaddin Dughmi and Arpita Ghosh. Truthful assignment without money. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 325–334. ACM, 2010.

[19] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, Dec 1971.

[20] Jérôme Euzenat. Interaction-based ontology alignment repair with expansion and relaxation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 185–191, 2017.

[21] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013.

[22] Christina Feier, Axel Polleres, Roman Dumitru, John Domingue, Michael Stollberg, and Dieter Fensel. Towards intelligent web services: the web service modeling ontology (wsmo). In *2005 International Conference on Intelligent Computing (ICIC'05)*, 2005.

[23] András Frank and Éva Tardos. Generalized polymatroids and submodular flows. *Mathematical Programming*, 42(1):489–563, Apr 1988.

[24] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[25] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified np-complete graph problems. *Theoretical Computer Science*, 1(3):237 – 267, 1976.

[26] Fernando C. Gomes, Cludio N. Meneses, Panos M. Pardalos, and Gerardo Valdisio R. Viana. Experimental analysis of approximation algorithms for the vertex cover and set covering problems. *Computers & Operations Research*, 33(12):3520 – 3534, 2006. Part Special Issue: Recent Algorithmic Advances for Arc Routing Problems.

[27] Bernardo Cuenca Grau and Boris Motik. Reasoning over ontologies with hidden content: The import-by-query approach. *Journal of Artificial Intelligence Research (JAIR)*, 45:197–255, October 2012.

[28] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[29] N. Guarino. Formal ontologies and information systems. In N. Guarino, editor, *Proceedings of FOIS'98*, Amsterdam, 1998. IOS Press.

[30] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989.

[31] M. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica*, 18(1):145–163, May 1997.

[32] Magnús Halldórsson and Jaikumar Radhakrishnan. Improved approximations of independent sets in bounded-degree graphs via subgraph removal. *Nord. J. Comput.*, 1:475–492, 01 1994.

[33] Magnús M. Halldórsson and Kiyohito Yoshihara. Greedy approximations of independent sets in low degree graphs. In John Staples, Peter Eades, Naoki Katoh, and Alistair Moffat, editors, *Algorithms and Computations*, pages 152–161, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[34] Dorit S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics*, 6(3):243 – 254, 1983.

[35] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.

[36] G. Jäger and B. Goldengorin. How to make a greedy heuristic for the asymmetric traveling salesman problem competitive. *University of Groningen, Research Institute SOM (Systems, Organisations and Management), Research Report*, 01 2005.

[37] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, November 2003.

[38] Ernesto Jiménez-Ruiz, Terry R Payne, Alessandro Solimando, and Valentina Tamma. Limiting consistency and conservativity violations through negotiation. In *The 15th International Conference on Principles of Knowledge Representation and Reasoning (KR 2016)*, pages 217–226, 2016.

[39] Ernesto Jimnez-Ruiz, Christian Meilicke, Bernardo Cuenca Grau, and Ian Horrocks. Evaluating mapping repair systems with large biomedical ontologies. In *26th International Workshop on Description Logics*, July 2013.

[40] Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. Average performance of greedy heuristics for the integer knapsack problem. *European Journal of Operational Research*, 154(1):36 – 45, 2004.

[41] B. Korte and L. Lovász. Mathematical structures underlying greedy algorithms. In Ferenc Gécseg, editor, *Fundamentals of Computation Theory*, pages 205–209, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

[42] Piotr Krysta. Greedy approximation via duality for packing, combinatorial auctions and routing. In Joanna Jdrzejowicz and Andrzej Szepietowski, editors, *Mathematical Foundations of Computer Science 2005*, pages 615–627, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[43] Piotr Krysta, Minming Li, Terry R. Payne, and Nan Zhi. Mechanism design for ontology alignment. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 1587–1588, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems. *Journal version in submission.* Estimated no. pages: 27.

[44] Piotr Krysta, Mathieu Mari, and Nan Zhi. Ultimate greedy approximation of independent sets in subcubic graphs. *Manuscript in preparation.*, Estimated no. pages: 45, April, 2019.

[45] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[46] L. Laera, I. Blacoe, V. Tamma, T.R. Payne, J. Euzenat, and T.J.M. Bench-Capon. Argumentation over ontology correspondences in MAS. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1285–1292, 2007.

[47] L. Lovász. Three short proofs in graph theory. *Journal of Combinatorial Theory, Series B*, 19(3):269 – 271, 1975.

[48] B. Lucier and A. Borodin. Price of anarchy for greedy auctions. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 537–553, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[49] Brendan Lucier and Allan Borodin. Price of anarchy for greedy auctions. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 537–553. Society for Industrial and Applied Mathematics, 2010.

[50] Brendan Lucier and Vasilis Syrgkanis. Greedy algorithms make efficient mechanisms. 03 2015.

[51] C. Meilicke and H. Stuckenschmidt. Analyzing mapping extraction approaches. In *Proceedings of the 2Nd International Conference on Ontology Matching - Volume 304*, OM'07, pages 25–36, 2007.

[52] Prasenjit Mitra, Peng Lin, and Chi Chun Pan. *Privacy-preserving ontology matching*, volume WS-05-01, pages 88–91. 2005.

[53] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, 64(2):612 – 631, 2008. Special Issue in Honor of Michael B. Maschler.

[54] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, Dec 1975.

[55] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[56] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

[57] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In Ian Horrocks and James Hendler, editors, *The Semantic Web — ISWC 2002*, pages 333–347, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[58] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[59] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425 – 440, 1991.

[60] Terry R. Payne and Valentina Tamma. Negotiating over ontological correspondences with asymmetric and incomplete knowledge. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 517–524, 2014.

[61] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 475–484, New York, NY, USA, 1997. ACM.

[62] Tim Roughgarden. Intrinsic robustness of the price of anarchy. *J. ACM*, 62(5):32:1–32:42, 2015.

[63] Peter W. Shor. The average-case analysis of some on-line algorithms for bin packing. *Combinatorica*, 6(2):179–200, Jun 1986.

[64] Pavel Shvaiko and Jérôme Euzenat. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1):158–176, 2013.

[65] R. Studer, V.R. Benjamins, and D. Fensel. Knowledge engineering, principles and methods. *Data and Knowledge Engineering*, 25(1-2):161–197, 1998.

[66] K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information systems. *ACM SIGMOD Record. Special Issue on semantic interoperability in global information systems*, 1998.

[67] Katia Sycara and Mattheus Klusch. Brokering and matchmaking for coordination of agent societies: A survey. In Omicini et al, editor, *Coordination of Internet Agents*. Springer, 2001.

[68] C. Trojahn dos Santos, P. Quaresma, and R. Vieira. Conjunctive queries for ontology based agent communication in MAS. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 829–836, 2008.

[69] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

[70] M. Wooldridge. *An Introduction to Multiagent Systems*. John Wiley and Sons, 2002.

[71] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 222–227. IEEE, 1977.