Contracts for Density and Packing Functions

by

Jacob Skitsko

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

All research detailed in this thesis was completed in collaboration with my supervisor, Kanstantsin Pashkovich. In particular, Chapter 3 and Chapter 4 are adapted from a co-authored paper which is anticipated to be published after this thesis.

Abstract

We study contracts for combinatorial problems in multi-agent settings. In this problem, a principal designs a contract with several agents, whose actions the principal is unable to observe. The principal is able to see only the outcome of the agents' collective actions. All agents that decided to exert effort incur costs, and so naturally all agents expect a fraction of the principal's reward as a compensation. The principal needs to decide what fraction of their reward to give to each agent so that the principal's expected utility is maximized. One of our focuses is on the case when the principal's reward function is supermodular and is based on some graph. Recently, [22] showed that for this problem it is impossible to provide any finite multiplicative approximation or additive FPTAS unless $\mathcal{P} = \mathcal{NP}$. On a positive note, [22] provided an additive PTAS for the case when all agents have the same cost. [22] asked whether an additive PTAS can be obtained for the general case, i.e. for the case when agents potentially have different costs. In this thesis, we answer this open question in positive. Additionally, we provide multiplicative approximation algorithms for functions that are based on hypergraphs and encode packing constraints. This family of functions provides a generalization for XOS functions.

Acknowledgements

I would like to thank my supervisor Kanstantsin Pashkovich. His guidance has helped me improve my work immensely, and he has helped show me to many interesting avenues of research. I appreciate his "Eastern European charm", as he calls it.

I also want to thank my readers, Vijay Bhattiprolu and Levent Tuncel for their invaluable feedback on this thesis.

The C&O administrative staff have also been helpful over the last years, and for that I thank them.

I was pulled into the world of approximation algorithms by Zachary Friggstad during my undergraduate degree at the University of Alberta. I am grateful for all the help and inspiration he has given me.

I am forever grateful to all of my friends and family from my time before, during, and after this degree. Thank you to Ian for your persistent friendship and insightful comments. Thanks to Ian and Noah for their time in competitive programming and beyond, and for fostering my interest in algorithms in my undergraduate degree. Thank you to all of those who welcomed me to Waterloo and helped introduce me to some of the (maybe too many) activities I like to do at Waterloo. There are too many friendly faces to name. Thank you to Madison for encouraging my study of coffee and optimization, and for being a great office-mate. Thanks to Sabrina for handling cookie time, and for passing that role on to me. Thank you to Camryn and Joaco for indoctrinating me into the world of climbing. A special thanks to my girlfriend Camryn for all you do.

And of course: thank you to my parents, and to my dog Newton.

Table of Contents

Chapter 1

Introduction

In 2016, the Nobel Prize in economics was awarded to Oliver Hart and Bengt Holmström for their work on contract theory [36, 42]. Indeed, today contracts play one of central roles in economic theory and in our everyday life. "Modern economies are held together by innumerable contracts", *The Royal Swedish Academy of Sciences.* [48].

1.1 Contract Theory

The contract design problem is natural and well motivated. Contracts are ubiquitous enough that the reader has certainly interacted with many contracts throughout their life. The reader has likely been on both the giving and receiving end of several contracts! Contracts are a naturally appearing object when one special agent (called *the principal*) wishes to incentivize other agents (simply called *the agents*) to take some costly actions. A distinguishing feature of the contract theory setting is that the principal observes the outcome of the agents' actions, but cannot observe the actions themselves.

Employer/employee relationships frequently feature carefully designed contracts. Dealing with insurance and incentives is also a common setting for contracts. Indeed, an insurance company must carefully construct their policies so that customers want to use their services, while ensuring the company still makes a profit. Likewise, a city constructing a public good must decide on how best to contract the work to be completed to maximize expected utility. Even from just these examples it should be clear that contracts are prolific, and there are many different settings one may consider.

In our setting each agent takes (or does not take) some costly action to maximize their individual utility (i.e. to maximize their expected reward minus cost). This results in some outcome which depends on the actions taken. The principal derives some reward based on this outcome. As such, the principal wishes to incentivize the agents to take costly actions. Note the agents bear the cost of the actions, but the principal derives reward from the outcome of the actions. This so called *moral hazard* makes designing contracts challenging. Nonetheless, the principal can incentivize the agents to take costly actions by offering some contract. In a contract, the principal promises transfers of reward to each agent based on the outcome the principal will observe. This can incentivize the agents to take actions which result in high reward outcomes, even if the actions are costly for the agents. Note the principal can promise to transfer reward to the agents, but cannot request the agents to transfer reward back to the principal. So, these transfers must be non-negative (this property is called *limited liability* in the literature).

A critical aspect of contract design is that the principal only observes the outcome, and not the actions of the agents. This so called *adverse selection* makes designing contracts challenging. As we will see, in many cases it also makes calculating or even approximating the optimal contract $\mathcal{NP}$-Hard when combined with the aforementioned moral hazard.

Let us consider a few toy examples before we formally introduce contracts and our setting. The first example illustrates the key difference between observable actions and outcomes, and the challenging nature of the problem.

Example 1. Consider a baker working to create and sell a perfect pastry. She receives some reward if she manages to sell her pastry. The baker has discovered she receives more reward from selling her pastry if it tastes exceptionally good. Moreover, certain special ingredients make her pastries taste better compared to using the non-special equivalents.

She works with different suppliers for each of these special ingredients $i \in \{1, \ldots, 6\} = [6]$. Each supplier i also has non-special versions of the ingredient i, which have no additional cost to the supplier but add no value to the pastry. Each special ingredient $i \in [6]$ has some additional cost c_i to the supplier i and adds value v_i to the baker's pastry. A set of special ingredients S adds value $\sum_{i \in S} v_i$ to the pastry, and costs each supplier $i \in S$ the amount c_i. For simplicity we will say the baker is very reliant on the special ingredients, and her pastry has 0 value when it has no special ingredients. Suppose Table 1.1 gives the values v_i, c_i for each special ingredient.

Observe that the baker would be happy to pay for all of the special ingredients herself, if given the choice. This gives her pastry a value of 10, and she pays a cost of $\sum_{i \in [6]} c_i = 2$. This gives her utility $10 - 2 = 8$. Note it is worth it to pay the cost and obtain added value for each individual ingredient.

special ingredient i	1	2	3	4	5	6	$\sum$
added value, v_i	1	1	1	2	2	3	10
cost to supplier i, c_i	1/10	1/10	1/10	4/10	4/10	9/10	2

Table 1.1: Special ingredient values and costs for the baker and suppliers.

However, the suppliers have different motives from the baker. Since it is more expensive for the suppliers to send the special version of an ingredient they may be motivated to send the non-special version instead. Even worse, the baker only knows the value her pastry has, and is unable to test if each supplier sent her the special or non-special version of an ingredient. For example, this could be because it is unwise to eat raw eggs, and it is unclear how the special ingredients differ from non-special ingredients outside of the final pastry. (Alternatively, one could consider this reward as being the price she is able to sell the pastry for.) We note that being able to only observe the total value prevents the baker from simply promising to pay the additional costs for a supplier if that supplier sends the special ingredient. This is because the final outcome is affected by all the suppliers, and the baker has no way of verifying if any given supplier did in fact send their special ingredient.

Therefore, the baker should promise to send the suppliers a percentage of the value added to the final product. This should align the motives of the suppliers to the motives of the baker (i.e. everyone should want to increase the value of the pastry). But what portion of the added value should be transferred from the baker to each supplier? If the baker wishes to keep some of the added value then clearly she should not transfer all of the added value to the suppliers. Moreover, the baker needs to ensure she promises a high enough percentage to each supplier if she wants that supplier to send their special ingredient. Clearly, some suppliers will be more critical than others. She will need to decide on both which suppliers to incentivize and on how much to give them.

Say S is the the set of suppliers the baker wants to recieve special ingredients from. It can be shown that if the baker promises a c_i/v_i portion of the total added value $\sum_{i \in S} v_i$ to supplier i then supplier i will be motivated to send the special ingredient rather than the non-special ingredient. This is because it benefits the supplier to send the special ingredient, even if they pay the associated cost c_i. However, observe that the baker promising such a percentage is more expensive than simply paying the cost c_i. Indeed, if the baker decides to incentivize exactly the suppliers $i \in S$ for some $S \subseteq [6]$ with a c_i/v_i fraction of the total added value then each supplier $i \in S$ would receive

$$\frac{c_i}{v_i} \cdot \left(\sum_{j \in S} v_j \right) ,$$

and the baker would receive

$$\left(\sum_{i\in S} v_i\right) - \sum_{i\in S}\left(\frac{c_i}{v_i}\cdot\left(\sum_{j\in S} v_j\right)\right) = \left(1-\sum_{i\in S}\frac{c_i}{v_i}\right)\left(\sum_{i\in S} v_i\right).$$

This leads us to observe that the baker's utility from motivating all suppliers is

$$\left(1-\sum_{i\in[6]}\frac{c_i}{v_i}\right)\left(\sum_{i\in[6]} v_i\right) = 0\cdot 10 = 0\,,$$

but the utility from motivating $S := \{5,6\}$ is

$$\left(1-\sum_{i\in S}\frac{c_i}{v_i}\right)\left(\sum_{i\in S} v_i\right) = \left(1-\frac{5}{10}\right)\cdot 5 = \frac{5}{2}\,.$$

In fact, it can be shown that this is the optimal utility for the baker in this example. Here we come across the fact that the baker's utility in the contract setting can be much less than in the setting where she could directly observe and pay costs. Indeed, the baker is now receiving utility 5/2 instead of utility 8. The difference between only having observable outcomes and having observable actions is a key feature and challenge of contract theory.

Remark 1.1.1. In fact, Example 1 also suggests the contract theory setting leads to more formal hardness. This example can be extended to show the optimal contract is $\mathcal{NP}$-Hard on additive reward functions using a reduction from the PARTITION problem [26]. Given a list of values, the PARTITION problem asks if there is a partition of the values into two equal sums.

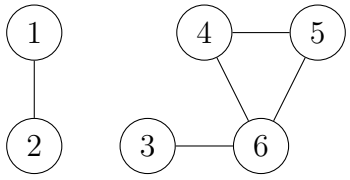

Figure 1.1: Reward function graph.

Example 2. Let us briefly consider a harder version of Example 1, using a different reward function for the baker. We will use the same costs as from Example 1. However, we will no longer consider special ingredients which add value in an independent (i.e. additive)

fashion. Instead, we will consider if they add value in a complementary way. Consider the graph $G = (V = [6], E)$ in Figure 1.1. Some ingredients improve the final pastry more when paired with other specific ingredients. As such, the value added by a subset $S \subseteq [6]$ is given by $|E(S)|$. For each $i \in [6]$ we can observe the degree of vertex i in G is the same as v_i from Example 1. However, there is a much greater degree of complementarity in this reward function. Indeed, any individual ingredient alone now gives no value to the baker. It can be seen to incentivize exactly a set S of special ingredients, the baker would now need to transfer a $c_i/\deg_S(i)$ fraction of the added value $|E(S)|$ to each supplier $i \in S$. Note $\deg_S(i)$ may be strictly smaller than $v_i = \deg_V(i)$ for subsets $S \subsetneq V$. Just as we saw in Example 1, the baker is unable to incentivize $S = [6]$. However, the suppliers can be even harder to incentivize in smaller subsets! If the baker attempts to incentivize $S = \{5, 6\}$ as she did in Example 1 then she would have utility

$$\left(1 - \sum_{i \in S} \frac{c_i}{\deg_S(i)}\right) \cdot |E(S)| = \left(1 - \frac{13}{10}\right) \cdot 1 < 0\,.$$

The combined facts that incentivizing many suppliers is too expensive, and that incentivizing few suppliers costs more on average makes this problem challenging to work with. In fact, when the reward function behaves in a complementary fashion, like in this example, then the problem becomes more formally hard. Indeed, just as Example 1 was closely related to the PARTITION problem, it can be shown that with appropriately defined costs, this example is closely related to the k-CLIQUE problem [22]. Given a graph, the k-CLIQUE problems asks if the graph contains a clique of size k. This connection rules out not just exact algorithms for the contract problem, but also many approximations. In particular, this rules out any multiplicative approximation to this problem [22].

1.2 The Model and Our Goal

We now formally introduce the model, before ultimately obtaining the function we will attempt to approximately maximize.

We have a single principal who interacts with a set of n agents $[n] = \{1, \ldots, n\}$. We focus on the setting with binary actions where each agent either exerts an effort or not. An agent i exerting an effort incurs a cost $c_i \in \mathbb{R}_{\geq 0}$ for the agent, and otherwise they incur no cost.

Let us define the function $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ representing the expected reward of the principal when a group of agents decides to exert an effort. We assume the function

$f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ is monotone, i.e. $f(S) \leq f(T)$ holds whenever $S \subseteq T$. We assume that f is normalized so that $f(\varnothing) = 0$. In Section 3 we additionally assume $f([n]) \leq 1$. We define the *marginal value* of agent i with respect to the set S as $f(i \mid S) := f(S \cup \{i\}) - f(S)$.

A key challenge in contract theory is that the principal obtains an outcome and the corresponding reward, but cannot observe the actions of the agents. Thus, the principal has to provide incentives for agents to take costly actions, and these incentives have to be based solely on the outcome. To achieve this, the principal has to come up with a linear contract. A linear contract is defined by a single vector $t = (t_1, \ldots, t_n)^T$ and the agent i, $i \in [n]$ always receives a positive transfer $r \cdot t_i$ whenever the principal obtains the reward r.

Let us assume that the agents in S exert an effort while all agents not in S do not. Then, for a linear contract $t = (t_1, \ldots, t_n)^T$ the *expected utility* of an agent i, $i \in [n]$ equals $f(S) \cdot t_i - c_i$ if i is in S; and equals $f(S) \cdot t_i$ if i is not in S. Moreover, the principal's expected utility is

$$f(S) \cdot \left(1 - \sum_{i \in [n]} t_i\right) .$$

Observe that the actions of agents can affect the expected utility of other agents.

We assume that each agent attempts to maximize their expected utility. Following existing contract theory literature [7, 26, 22], we see that the set S, $S \subseteq [n]$ forms a pure Nash equilibrium if and only if

$$\begin{aligned} f(S) \cdot t_i - c_i &\geq f(S \setminus i) \cdot t_i && \text{for all } i \in S \\ f(S) \cdot t_i &\geq f(S \cup i) \cdot t_i - c_i && \text{for all } i \notin S. \end{aligned}$$

Then a linear contract $t^* = (t_1^*, \ldots, t_n^*)^T$ that incentivizes exactly the agents in S to exert an effort while maximizing the principal's expected utility satisfies

$$\begin{aligned} t_i^* &= c_i / f(i \mid S) && \text{for all } i \in S \\ t_i^* &= 0 && \text{for all } i \notin S\,, \end{aligned}$$

where $c_i / f(i \mid S)$ is 0 if $c_i = 0$, and otherwise is infinity if $f(i \mid S) = 0$. Using the contract t^* the principal's expected utility is given by the function

$$g(S) := \left(1 - \sum_{i \in S} \frac{c_i}{f(i \mid S)}\right) f(S)\,. \tag{1.1}$$

Hence, whenever the principal decides to incentivize the set of agents S, the maximum principal's expected utility equals $g(S)$. Now, our goal is to determine S, $S \subseteq [n]$ that maximizes $g(S)$.

1.3 Our Results

In recent years, contract theory has garnered an increasing interest in the theoretical computer science and optimization communities. One factor for this growing interest is the clear connection between contracts and fundamental optimization problems. The arising contract design problems turn out to be hard even in relatively structured settings, e.g. it is $\mathcal{NP}$-hard to find the optimal contract for additive reward functions in a multi-agent setting [26]. Finding an optimal contract tends to be hard for many types of reward functions, so it is natural to ask whether it is possible to efficiently find a contract that achieves *approximately* optimal expected utility. By today, there are various results about approximations in the contract design theory [54, 38, 39]. Also, several works show that there are natural limitations for such approximations and these limitations are related to both the computational hardness [28, 29, 15, 16] and to the communication and query complexity [32, 30].

Recently, [22] studied multi-agent contract design when the reward function is supermodular and is based on graphs with all agents having the same cost. They showed that it is impossible to provide a finite multiplicative approximation or any additive FPTAS, unless $\mathcal{P} = \mathcal{NP}$. Indeed, given a graph one can construct a contract instance where the optimal contract has positive utility if and only if the graph contains a clique of a certain size. On a positive note, they develop an additive PTAS. Their PTAS built on the techniques developed in [6], establishing tight connections to the densest subgraph problems. In [22], it was left as an open question whether it is possible to a provide PTAS not only in the case when all agents have the same cost, but also for the general case. In this thesis, we provide an additive PTAS for the graph based supermodular functions considered by [22] for the case with general agent costs.

[26] provided multiplicative approximation for optimal contracts in the case of XOS functions. To obtain these approximately optimal contracts, [26] gave a set of structural lemmas which established connections between the value of the optimal contract, the marginals, and the costs of participating agents. [26] also established a novel property of XOS functions that permits "scaling" the value of the function while preserving sufficiently high marginal values. We adapt their techniques to provide multiplicative approximation for optimal contracts for functions encoding packing constraints.

Thus, we have two positive results for different kinds of functions based on graphs. In the case of supermodular functions based on graphs, adding an agent changes the function value by the total weight of all edges between the added agent and all other agents in the set. In the case of hyperedge packing functions, adding an agent changes the function value

by at most the weight of one hyperedge between the added agent and other agents in the set.

Our results on the supermodular functions also have close ties to approximations for dense problems. [6] gave an additive PTAS for many instances of dense problems, which uses exhaustive sampling to approximately solve a polynomial program. Similar sampling techniques were used in [19, 11] to give an PTAS for finding small probability mixed Nash Equilibrium. Our results on hyperedge packing functions follow the techniques outlined in [26].

1.4 Previous Work

In the following sections we offer a brief survey of the existing combinatorial contract theory literature. This chapter is intended for the curious reader, and as reference material. The relevant material for other chapters will be summarized at the beginning of each corresponding chapter. In these sections we will always make the following assumptions.

- The principal can observe the outcome but not the actions taken by the agent(s).
- A contract must have limited liability, i.e. non-negative transfers.
- In each setting the agent(s) will take some action(s) which maximize their personal utility, and will tie break in favor of the principal. This tie-breaking can be enforced with contracts by promising an ε greater portion of the reward.
- Each agent has some null action with 0 cost for the agent and 0 expected reward for the principal. If the agent has actions $[n] = \{1, \ldots, n\}$ then the null action will be labelled 1. This represents the agent "doing nothing".

We begin by considering settings closely related to the ones considered in later chapters. Afterwards, we consider other settings which are either less combinatorial in nature or are concerned with different constraints.

We remark that it is common in the literature to restrict focus to a setting where there are binary outcomes $\{0, 1\}$. This ensures linear contracts may be considered without loss of generality. Without binary outcomes, it can be beneficial to consider general contracts which promise arbitrary amounts depending on different outcomes. Such contracts can have very complicated structure, and appear less in practical instances even if they may give improved utility to the principal. Some works, e.g. [28, 13], consider this issue.

1.4.1 Single Agent, Multi-Action

The Setting

We now consider the setting from [25], where we have a single agent with n actions available to them. Each action $i \in [n]$ has a cost c_i. The agent may take any subset of the actions $S \subseteq [n]$ and pay cost $\sum_{i \in S} c_i$. There are binary outcomes $\{0, 1\}$ representing failure and success. The principal derives some non-negative reward r from success and 0 reward from failure. Each strategy $S \subseteq [n]$ of the agent has an associated success probability $f(S) \in [0, 1]$ which is assumed to have $f(\varnothing) = 0$ and be monotonically non-decreasing (i.e. if $S \subseteq S'$ then $f(S) \leq f(S')$). A contract consists of a single value $\alpha \in [0, 1]$, where the principal promises $\alpha \cdot r$ if the outcome is a success, and 0 otherwise.

Known Results

Many results in the contract theory literature depend on *critical values* (sometimes called *break points*). Intuitively, critical values are values for the contract (given by $\alpha \in [0, 1]$ in this setting) which could be the value for an optimal contract. Slightly more formally, they represent points where the agent would start to take an action set which yields different success probability. The optimal contract will be one of these critical values, since the optimal contract will be the minimum cost way for the principal to ensure a desired success probability. Thus, if the principal can enumerate over these critical values and calculate their utility at each critical value then they can calculate an optimal contract.

[25] showed for gross substitutes success probability functions (which includes additive, unit demand, and matroid rank functions) one can compute an optimal contract by enumerating over the critical values. To do this, they show for gross substitutes f there are polynomially many critical values, and one can efficiently iterate them. However, the authors also showed there are coverage functions which have exponentially many critical values. Indeed, finding the optimal contract with a submodular (specifically budget additive) success probability function is $\mathcal{NP}$-Hard. [25] showed this via a reduction from SUBSET-SUM. At the same time, they showed there is a (weakly) FPTAS for arbitrary success probability functions and a weakly polynomial time algorithm for any instance with a polynomially sized set of critical values. [22] gave an improved strongly polynomial for any instance with a polynomially sized set of critical values by using a simple divide-and-conquer algorithm. They also showed for a supermodular success function there are polynomially many critical values. [27] gave an algorithm for any success probability function and cost which enumerates all critical values with polynomially many demand queries.

This suggests if one can compute demand queries efficiently and there are only polynomially many critical values then one can compute the optimal contract efficiently. Indeed, the algorithm [27] gives provides a way to compute the optimal contract for supermodular success probabilities and submodular costs (rather than additive). They show a natural class of matching based instances with XOS rewards which has an exponential number of critical values, but also yields an efficient demand oracle. Thus one cannot simply enumerate over all critical values in this case (although some special cases have (pseudo-)polynomially many critical values). [23] gave a FPTAS for the problem for general f, with value and demand oracle access, improving upon the weakly FPTAS from [25].

Hardness results for this setting were improved in [32]. The authors showed that for submodular f it is $\mathcal{NP}$-Hard to approximate the optimal contract to within any constant factor. This relies on the hardness of a promise problem of coverage functions. For XOS f it is is $\mathcal{NP}$-Hard to obtain a $n^{-1/2+\varepsilon}$ approximation ratio for any $\varepsilon > 0$. This relies on the hardness of approximating the largest clique in a graph. Furthermore, [30] showed that when f is submodular any algorithm that computes the optimal contract requires exponentially many demand queries to f. This dissuades concerns that the hardness was simply from the hardness of computing a demand query for f. They showed this by constructing instances where value and demand oracles had effectively the same power. However, this is not always the case. [30] showed that there are instances where the optimal contract can be found with $O(1)$ demand queries, but also requires exponentially many value queries.

We summarize the results in Table 1.2 and Table 1.3.

Success probability function f	Approximation
gross substitutes	exact [25]
supermodular	exact [22, 27]
polynomially many critical values	exact with demand oracle [27]
submodular	$\approx 1/1120$ with demand oracle [23]
general	FPTAS with demand oracle [23]

Table 1.2: Positive results for the single-agent multi-action setting.

Success probability function f	$\mathcal{NP}$-Hard within
submodular (coverage)	any constant with value oracle [32]
XOS	$n^{-1/2+\varepsilon}$ with value oracle [32]

Table 1.3: Hardness results for the single-agent multi-action setting.

1.4.2 Multi-Agent, Single Actions

The Setting

We now consider the setting from [7]. In this setting we consider when a set of n agents each has a binary action available to them. The agent $i \in [n]$ taking the action incurs a cost c_i, and otherwise they incur no cost. Any subset $S \subseteq [n]$ of the agents may take actions. There are binary outcomes $\{0, 1\}$ representing failure and success. The principal derives some non-negative reward r from success and 0 reward from failure. Each set $S \subseteq [n]$ of agents exerting effort has an associated success probability $f(S) \in [0, 1]$ which is assumed to have $f(\varnothing) = 0$ and be monotonically non-decreasing (i.e. if $S \subseteq S'$ then $f(S) \leq f(S')$). A contract consists of vector $t \in [0, 1]^n$, where the principal promises $t_i \cdot r$ to agent $i \in [n]$ if the outcome is a success, and 0 otherwise.

See also [17] for a different multi-agent binary action setting.

Known Results

In 2006 (remarkably early!) [7] considered Boolean functions represented by read-once networks. For AND networks the authors provide a polynomial time algorithm, and they show the problem becomes $\#\mathcal{P}$-Hard for general networks via a reduction from network reliability problems. [31] show that for OR networks (which is a special case of submodular functions) the problem is $\mathcal{NP}$-Hard via a reduction from EXACT 3SAT. They also provide a FPTAS for OR networks. [26] considers submodular and XOS success probability functions. Using value and demand oracle queries they are able to obtain constant approximations to the optimal contract. To accomplish this they use a novel scaling property of XOS functions and their marginals. This allows them to make a demand query at appropriately chosen prices, then scale back to a set with approximately optimal value for the contract problem. For submodular functions f the authors show that a value oracle is sufficient, since one can simulate an "approximate demand oracle" using algorithms for constrained submodular maximization [52]. They complemented these results by showing with high probability any algorithm that performs polynomially many value and demand queries has approximation ratio ≥ 1.136 for XOS functions, and $\Omega(\sqrt{n})$ for subadditive functions. Both of these hardness results are from communication complexity. They show an optimal contract should incentivize some special set $S, |S| \approx n/2$ and also that finding S with only polynomially many value and demand queries is unlikely. [26] also shows it is $\mathcal{NP}$-Hard to calculate the optimal contract for additive f via a reduction from PARTITION. However, the problem admits a FPTAS in this case. Hardness results were improved

by [32]. The authors showed it is $\mathcal{NP}$-Hard to approximate the optimal contract within a factor of 0.7 for submodular f. They showed this via a reduction from a promise coverage function. In fact, this hardness result holds even for normalized unweighted coverage functions f. Note that this rules out a PTAS for general submodular functions, even though one exists for non-trivial special cases such as OR networks [31]. [23] further shows no PTAS exists with submodular f and value and demand oracles. This is accomplished through a more subtle communication complexity argument. [32] also showed for XOS functions any algorithm that makes only polynomially many value oracle queries (and no demand oracle queries) with high probability cannot approximate the optimal contract within a factor greater than $4n^{-1/6}$. This hardness follows from a communication complexity argument with a special set of size $n^{1/3}$. [22] departed from functions which were complement-free be considering supermodular functions. In particular, given a graph $G = (V, E)$ they consider a supermodular success probability function $f(S) = |E(S)|/\binom{n}{2}$. They show if the agent costs are $(k-2)/(k-1) \cdot (1/\binom{n}{2})$ then the optimal contract has positive value if and only if the graph G contains a k-clique. Thus, it is $\mathcal{NP}$-Hard to obtain any finite multiplicative approximation or FPTAS in this setting. However, they show that an additive PTAS is possible when all agents have the same costs. Their techniques take inspiration from additive PTASs for dense problems, such as [6, 19, 11]. Indeed, the authors show that an approximately optimal contract exists incentivizes a set of with $\Theta(n)$ agents who all have $\Theta(n)$ degree.

The work contained in this thesis extends some of these results. In the complement free hierarchy [45] we consider MPH-k functions. General MPH-2 functions can already capture the supermodular problem considered in [22], who showed the problem is hard to approximate in general. However, the techniques from approximations for XOS in [26] can be extended to MPH-k functions which satisfy some specific packing conditions. We also show it is possible to attain an additive PTAS for the supermodular problem considered in [22] when the agents have arbitrary cost. In this case in an approximately optimal contract some "inexpensive" agents may have degree $O(1)$, and even "expensive" agents may only have degree $o(n)$. Thus, a more technical argument is required to refine the requirements of an approximately optimal contract and find the set S to be incentivized.

We summarize the results in Table 1.4 and Table 1.5.

Success probability function f	Approximation
AND networks	exact [7]
OR networks (submodular)	FPTAS [31]
submodular	$\approx 1/256$ with value oracle [26]
XOS	$\approx 1/256$ with demand oracle [26]
MPMH-k, MPMUH-k	$\approx 1/256k^4$ with demand oracle [this work]
$\lvert E(S)\rvert/\binom{n}{2}$	additive PTAS [22] and [this work]

Table 1.4: Positive results for the multi-agent single-action setting.

Success probability function f	$\mathcal{NP}$-Hard within
submodular (coverage)	0.7 with value oracle only [32]
submodular	0.98 with demand oracle [23]
supermodular (e.g. $\lvert E(S)\rvert/\binom{n}{2}$)	any multiplicative factor, or additive FPTAS [22]
XOS	$4n^{-1/6}$ with value oracle [32]
XOS	1/1.136 with demand oracle [26]
MPH-2	any multiplicative factor, or additive FPTAS [this work]

Table 1.5: Hardness results for the multi-agent single-action setting.

1.4.3 Multi-Agent, Multi-Action

Very recently, [23] considered a setting with multiple agents each with the ability to take a subset of actions. This setting extends both of the previous settings.

See also [13] for a different multi-agent non binary-action setting.

The Setting

There are n agents. Each agent $i \in [n]$ has an action set A_i, and may choose to take any subset of actions $S_i \subseteq A_i$. Let $A = \bigcup_{i\in[n]} A_i$. Each action $j \in A$ has a cost c_j, and the cost of any agent taking a subset of actions $S \subseteq A$ is $\sum_{j\in S} c_j$. There are binary outcomes $\{0, 1\}$ representing failure and success. The principal derives some non-negative reward r from success and 0 reward from failure. Each set $S \subseteq [n]$ of actions taken has an associated success probability $f(S) \in [0, 1]$ which is assumed to have $f(\varnothing) = 0$ and be monotonically non-decreasing (i.e. if $S \subseteq S'$ then $f(S) \leq f(S')$). A contract consists of vector $t \in [0, 1]^n$, where the principal promises $t_i \cdot r$ to agent $i \in [n]$ if the outcome is a success, and 0 otherwise.

Known Results

This setting is much more general than the previous two, and in fact captures both of them. A single agent can be obtained by setting $n = 1$, and binary actions can be obtained by setting $A_i = \{0, i\}$ for $i \in [n]$ (where the action 0 represents doing nothing). Unfortunately, we pay for this generality with complexity. This setting loses many of the features that made previous settings more pleasant to work with. Indeed, even characterizing a Nash-equilibrium is less trivial. Nonetheless, for submodular f [23] give a constant approximation using value and demand oracles. In fact, their result is stronger: they compute a contract so that any equilibrium of the this contract gives an approximation to the optimal principal's utility. Their basic idea is similar to the multi-agent case considered in [26] (where either no agent is "large" or it suffices to incentivize a single agent) but it is far from trivial to see that this approach would generalize beyond binary actions. [23] further shows no PTAS exists with submodular f and value and demand oracles, even for binary actions. This is accomplished through a subtle communication complexity argument.

1.4.4 Single Agent, Single-Action

This setting is less related to the problems we consider, since there is not much room to consider combinatorial functions in this setting. Nonetheless, it is good to be aware of it.

The Setting

In this setting we consider when a single agent has n actions available to them. Each action $i \in [n]$ has a cost c_i. The agent may choose to take any single action. We consider when there is a general (not necessarily binary) outcome set $\Omega, \Omega = [m]$. Each outcome $j \in [m]$ is associated with a reward r_j for the principal. Each action $i \in [n]$ is associated with a distribution F_i over the outcomes $[m]$. We denote the probability of action $i, i \in [n]$ resulting in outcome $j, j \in [m]$ as $F_{i,j}$. A contract consists of a vector $t = (t_1, \ldots, t_m)$, where the principal promises t_j to the agent if it observes outcome $j, j \in [m]$. A *linear contract* consists of a single value $\alpha, \alpha \geq 0$, where the principal promises $\alpha \cdot r_j$ if it observes outcome $j, j \in [m]$.

Known Results

Consider a contract t, and suppose it was known the agent will take some action $i, i \in [n]$. Then in expectation the principal transfers the agent

$$\sum_{j \in [m]} F_{i,j} t_j \,,$$

and the agent's expected utility is given by

$$\sum_{j \in [m]} F_{i,j} t_j - c_i \,.$$

We now observe that the optimal contract problem can be solved in the following standard way, as in [28]. Consider if we wished to incentivize some action $i, i \in [n]$. We would like to do this at minimum cost, while ensuring no other action has better utility for the agent. This can be done with the following LP over variables $t_j, j \in [m]$.

$$\begin{aligned} \min \quad & \sum_{j \in [m]} F_{i,j} t_j \\ \text{subject to} \quad & \sum_{j \in [m]} F_{i,j} t_j - c_i \geq \sum_{j \in [m]} F_{i',j} t_j - c_{i'} \qquad \text{for all } i' \neq i, i' \in [n] \\ & t \geq \mathbb{0} \end{aligned} \tag{1.2}$$

By enumerating over all fixed actions $i, i \in [n]$ the principal can find which action is best to incentivize. However, one may notice the contracts output by this process may have hard to understand structure. "Simple" contracts, such as linear contracts, are desirable for many reasons. Indeed, linear contracts appear much more frequently in practice as opposed to complicated general contracts. However, linear contracts may give worse utility for the principal compared to general contracts. Thus, more focus is given to how well these "simple" contracts can approximate the utility to the principal of general contracts. [28] shows if there are N linearly implementable actions then the multiplicative loss in the principal's expected utility from using a linear contract rather than an arbitrary one is at most N. There are in fact also instances where for $\epsilon > 0$ the multiplicative loss is at least $n - \epsilon$, even when assuming desirable properties of the distributions F. The authors also show that if all of the costs $c_i, i \in [n]$ are in the range $[1, C)$ then the multiplicative loss is at most $O(\log C)$. It is also possible to consider "simple" contracts as low degree polynomials, rather than linear functions [28]. [29] considers a succinct representation of the setting, where there are n actions and $\{0,1\}^M$ outcomes, where M is given as input.

Thus, the outcomes are subsets of $[M]$, and outcomes are realized according to a product distribution. This makes the previously considered LP intractable. Indeed, [29] shows it is $\mathcal{NP}$-Hard to compute any constant approximation to the optimal contract. However, if one relaxes the incentive compatibility constraints to δ incentive compatible constraints (i.e. allowing the right hand side of constraint 1.2 to be violated additively by at most δ) then one can compute the optimal contract in time polynomial in $(m, 1/\delta)$. [29] also shows it is still $\mathcal{NP}$-Hard to find a constant approximation even when the setting is relaxed to considering these δ incentive compatible contracts.

1.4.5 Alternative Settings

There is an alternative multi-agent contract model, studied in [17]. In this model each agent determines their own individual outcome, which removes externalities. With some mild regularity conditions, they show the optimal contract in their setting for supermodular functions can be computed in polynomial time, and for submodular functions can be approximated within $(1 - 1/e)$. In [13] the authors consider a multi-agent non-binary-action setting which trivializes the computation of optimal contracts (since it is possible to solve LP profiles similar to those in Section 1.4.4). Thus, the authors focus on considering different mechanisms than standard deterministic contracts, the guarantees of simple contracts, and how externalities affect the principal's utility.

One could also consider a setting where the agent is able to take actions sequentially, after observing the outcomes of previous actions [33, 41]. In this setting there is a natural connection to Pandora's box problems [53]. Finally, one can also consider settings with multiple principals [4].

An expansive number of different settings for contract theory have been explored in recent years, as the setting has recently caught the eye of many researchers in the theoretical computer science community. As a small list, other recent models have included contracts with inspections [34], ambiguous contracts [24], agents choosing from menus of contracts [39, 12, 16], contracts with a learning agent [37], contracts for delegating machine learning tasks [50, 5] or selling data to a machine learner [18], hidden but signaled outcomes [8], and Bayesian settings [3, 2, 54, 38, 39, 16, 14].

1.4.6 Max Constraint Satisfaction Problems

Our work in Chapter 3 is related to approximations for max dense constraint satisfaction problems (Max-CSPs), in particular [6]. Our additive PTAS in Chapter 3 runs in time

$O(n^{\text{poly}(1/\varepsilon)})$ due to similar sampling requirements to the PTAS of [6] for densest k subgraph which also runs in time $O(n^{\text{poly}(1/\varepsilon)})$. [6] started a line of work achieving PTASs for dense instances of Max-CSPs. These are accomplished through various means, including sampling/subsampling [6, 1, 20, 9, 55, 46], regularity lemmas [35], and convex hierarchies [21, 10, 40, 56]. Some of these approaches accomplish their approximations more quickly. The fastest known time for a PTAS for dense Max-CSPs is $O(n/\varepsilon^2) + 2^{O(1/\varepsilon^2)}$ in [55]. However, note that these problems are only concerned with maximizing the number of satisfied constraints in a CSP. In our problem we are also concerned with ensuring the number of satisfied constraints containing any non-zero variable is high. Thus we have some global constraints we must deal with also, and not all of the approaches for Max-CSPs directly translate to our problem. [49] considers approximating Max-CSPs with a global cardinality constraint, and is thus relevant. Furthermore, we use the samples for degrees of "high degree" agents to linearize the terms $1/\deg(v)$ in $L(\cdot)$ (see Section 3.3). Therefore the sampling approaches for Max-CSPs are most relevant to us.

1.5 Organization

In Chapter 2 we briefly go over some preliminary definitions and statements which are assumed to be known in the later chapters. This includes classes of functions we'll be working with, and well known techniques and inequalities such as randomized rounding and Chernoff Bounds.

In Chapter 3 we go over our additive approximation for graph based supermodular functions. Using properties of supermodularity we find the existence of an appropriately structured approximately optimal set. We will not be able to actually find this set. However, we are able to find useful information about this set using sampling or exhaustive enumeration. This allows us to linearize the contract problem and consider a relaxed LP. We then show how to greedily alter a solution to this LP to get an approximately feasible solution to the LP which will satisfy concentration requirements. This allows us to randomly round the approximately feasible solution to the linear program and obtain an approximately optimal solution to the contract problem.

In Chapter 4 we go over our multiplicative approximation for hypergraph packing functions. We show we can obtain approximations for MPH-k functions with certain packing properties. This generalizing known approximation results for XOS = MPH-1 functions. The techniques used here follow the same outline as those for XOS functions. We provide structural lemmas for MPH-k functions similar to those that exist for XOS functions, with a predictable dependence on k. Then the result of a demand queried can be "scaled down"

to give an approximately optimal set. The packing properties of the functions we consider are critical to be able to "scale down" a set in this way. Indeed, we observe that if we place no such properties on our MPH-k functions then there can be no finite multiplicative approximation. This hardness result holds even for MPH-2 functions.

In Chapter 5 we make some remarks about future work.

Chapter 2

Preliminaries

2.1 Approximation Algorithms

We first consider what it means to have an efficient approximation algorithm. In this work we concern ourselves with $\mathcal{NP}$-Hard problems. Under the standard assumption $\mathcal{P} \neq \mathcal{NP}$, it is impossible to compute an optimal solution to such a problem in polynomial time. Thus, we instead look to have algorithms that compute *approximately* optimal solutions, rather than optimal solutions. Say ALG is the value of the output of the algorithm, and OPT is the value of the optimal solution. An α-approximation algorithm guarantees

$$\begin{aligned} \alpha \cdot \mathrm{OPT} \leq \mathrm{ALG} \leq \mathrm{OPT} \qquad & \text{if } \alpha \leq 1\,, \\ \mathrm{OPT} \leq \mathrm{ALG} \leq \alpha \cdot \mathrm{OPT} \qquad & \text{if } \alpha > 1\,. \end{aligned}$$

In some situations we instead guarantee that the output is only off by an additive term of α. In this case we refer to our approximation algorithm as an additive α-approximation algorithm.

Sometimes we may also parameterize the run-time of our algorithm by our error term. For a maximization problem, a polynomial time approximation scheme (PTAS) is a $(1-\varepsilon)$-approximation algorithm which runs in time polynomial in the size of the input for any fixed $\varepsilon > 0$. Similarly, a fully polynomial time approximation scheme (FPTAS) is a $(1-\varepsilon)$-approximation algorithm which runs in time polynomial in the size of the input *and* in $(1/\varepsilon)$ for any fixed $\varepsilon > 0$. Observe that a PTAS runs efficiently in terms of the input, but may be very inefficient in terms of the error term ε. We analogously define an additive PTAS and additive FPTAS by allowing ε additive error, rather than $(1-\varepsilon)$ multiplicative error.

2.2 Probability

We use the following standard inequalities.

Markov's Inequality

Let X be a nonnegative random variable, and $a > 0$. Then

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}.$$

Chernoff Bound

Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0, 1\}$, $\mu = \mathbb{E}\left[\sum_{i \in [n]} X_i\right]$, and $0 < \delta < 1$. Then

$$\Pr\left[\sum_{i \in [n]} X_i \leq (1 - \delta)\mu\right] \leq \exp\left(\frac{-\delta^2 \mu}{2}\right).$$

Hoeffding Bound

Let $X_1, \ldots, X_n$ be independent random variables taking values in $[a, b]$, $\mu = \mathbb{E}\left[\sum_{i \in [n]} X_i\right]$, and $t > 0$. Then

$$\Pr\left[\sum_{i \in [n]} X_i \leq \mu - t\right] \leq \exp\left(\frac{2t^2}{n(b-a)^2}\right).$$

2.3 Linear Relaxations and Rounding

In Chapter 3 we would ultimately like to solve an integer program. Our problem is essentially of the following form

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in \{0, 1\}^n \end{aligned}$$

where A, b, c are appropriately sized. Unfortunately, this problem is in general intractable. Thankfully, it is known how to solve linear programs efficiently e.g. with the ellipsoid method [44] or interior point methods [43]. This means we may solve the following linear program efficiently.

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \in [0,1]^n \end{aligned}$$

However, our original problem remains. Given an optimal $x^* \in [0,1]^n$ for this linear relaxation how can we obtain an (approximately) optimal $\bar{x} \in \{0,1\}^n$? This is a tricky problem in general. We will make use of a common technique known as *randomized rounding*. For $i, i \in [n]$ consider setting $\bar{x}_i$ to 1 with probability x_i^* and to 0 with probability $1 - x_i^*$. If our linear programming relaxation involves sufficiently "large" bounds, then we may hope that this rounding is an effective strategy. Indeed, in this case one can make use of the inequalities in Section 2.2 to show that this "brute force" rounding is sufficiently effective. We will observe in Chapter 3 that our problem is too delicate to naively apply this approach. Nonetheless, the heart of our rounding algorithm is still randomized rounding.

2.4 Some Relevant Classes of Functions

Classes of reward function f

Throughout this thesis we consider the following standard classes of functions

- An *additive function* f satisfies $f(S) = \sum_{i \in S} v_i$ for every $S \subseteq [n]$, where v_i for $i \in [n]$ are some fixed values.
- An *rank k Positive-Hypergraph (*PH-k*) function* f satisfies $f(S) = \sum_{e \in E, e \subseteq S} w_e$ for every $S \subseteq [n]$, where $w_e \geq$ for $e \in E$ are weights on edges of a rank k hypergraph $G = (V, E)$.
- A *submodular function* f satisfies $f(i \mid S) \geq f(i \mid S')$ for every $S, S' \subseteq [n], S \subseteq S', i \in [n]$. A *supermodular function* f instead satisfies $f(i \mid S) \leq f(i \mid S')$.
- A XOS *function* f satisfies $f(S) = \max_{i \in [k]} a_i(S)$ for every $S \subseteq [n]$ where each a_i is an additive function, and $[k]$ is some index set.

- A *rank k Maximum over Positive Hypergraphs (*MPH-k*) function* f satisfies $f(S) = \max_{i\in[k]} w_i(S)$ for every $S \subseteq [n]$ where each w_i is a PH-k function, and $[k]$ is some index set.
- A *subadditive function* f satisfies $f(i \mid S) \leq f(i \mid S')$ for every $S, S' \subseteq [n], S \subseteq S', i \in [n]$.
- A *monotone function* f satisfies $f(S) \leq f(S')$ for every $S, S' \subseteq [n], S \subseteq S'$.

It is known that submodular $\subsetneq$ XOS $=$ MPH-1 $\subsetneq$ subadditive and that MPH-1 $\subsetneq$ MPH-2 $\subsetneq \cdots \subsetneq$ MPH-n $=$ monotone.

In Chapter 4 we introduce a few new classes of functions which generalize XOS functions in a more constrained way than MPH-k functions. They maintain the useful property that every vertex gets value from a single edge in a XOS = MPH-1 function.

- A *rank k Positive Matching Hypergraph (*PMH-k*) function* f is a PH-k function where the underlying hypergaph is a matching. That is, every vertex is incident to at most one hyperedge.
- A *rank k Maximum over Positive Matching Hypergraph (*MPMH-k*) function* satisfies $f(S) = \max_{i\in[k]} w_i(S)$ for every $S \subseteq [n]$ where each w_i is a PMH-k function on the same underlying hypergraph, and $[k]$ is some index set.
- A *rank k Positive Matching Hypergraph (*PMUH-k*) function* f is a PH-k function where the underlying hypergaph is uniform and a matching. That is, every edge has rank k and every vertex is incident to at most one hyperedge.
- A *rank k Maximum over Positive Matching Uniform Hypergraph (*MPMUH-k*) function* satisfies $f(S) = \max_{i\in[k]} w_i(S)$ for every $S \subseteq [n]$ where each w_i is a PMUH-k function, and $[k]$ is some index set.

Oracle Access to a function f

In Chapter 4 we consider set functions for the reward function f. We use the following standard oracles.

- A *value oracle* for f returns $f(S)$ when queried with S.

- A *demand oracle* for f returns $S \in \arg\max_{S \subseteq [n]} f(S) - \sum_{i \in S} p_i$ when queried with a vector of prices $p = (p_1, \ldots, p_n)^T \in \mathbb{R}^n_{\geq 0}$.

We also use the following oracle for MPMH-k and MPMUH-k functions, defined in an analogous way to the more standard XOS oracle.

- A XOS *oracle* for f returns a supporting function a such that $a(S) = f(S)$ when queried with S.
- A MPMH-k *oracle* for f returns a supporting function w such that $w(S) = f(S)$ when queried with S.
- A MPMUH-k *oracle* for f returns a supporting function w such that $w(S) = f(S)$ when queried with S.

Chapter 3

Supermodular Graph Functions

3.1 Definitions

Given a graph $G = (V, E)$ with $|V| = n$ we let $f(S) = |E(S)|/\binom{n}{2}$. Our goal is to find a set $S \subseteq V$ that maximizes the following function, which corresponds to the function in (1.1)

$$\left(1 - \sum_{i \in S} \frac{c_i \cdot \binom{n}{2}}{\deg_S(i)}\right) \frac{|E(S)|}{\binom{n}{2}},$$

where $E(S) := \{uv \in E : u \in S, v \in S\}$ and $\deg_S(i) := |N(i) \cap S|$. For the sake of exposition, we modify the costs c_i, $i \in V$ as follows $c_i \leftarrow c_i \cdot \binom{n}{2}$. Thus, we are maximizing the function

$$g(S) := \left(1 - \sum_{i \in S} \frac{c_i}{\deg_S(i)}\right) \frac{|E(S)|}{\binom{n}{2}}.$$

Let us denote by OPT the maximum value of $g(S)$ over S, $S \subseteq V$, and let S^* be the corresponding optimal set of agents. Before we proceed, let us define C as the set of "cheap" vertices, or more formally $C := \{i \in V : c_i \leq \varepsilon/(2n)\}$. We assume that OPT is at least ε, since otherwise the task of additive approximation (up to ε) is trivial.

We further define functions for the left and right factors of $g(S)$, i.e.

$$L(S) := 1 - \sum_{i \in S} \frac{c_i}{\deg_S(i)} \quad \text{and} \quad R(S) := \frac{|E(S)|}{\binom{n}{2}}.$$

Now we define parameters κ, σ, δ, δ_ε, γ, γ' that do not depend on n. For getting a first impression about our approach, it suffices to ignore their formal definitions and return to them only for verifying technical details and proofs. We define a value that is needed for concentration inequalities κ and a value used for sampling bounds σ. The δ parameters are used for degree estimates and the γ parameters are used for working with "pseudo-cores". We define these parameters as

$$\kappa := \frac{2}{\varepsilon^2}, \qquad \sigma := \varepsilon^6, \qquad \delta := \frac{36e}{(1-\varepsilon)^2 \cdot \varepsilon^5 \cdot \ln(2)}, \qquad \delta_\varepsilon := \frac{\delta}{\varepsilon}, \qquad \gamma := \frac{\varepsilon^4}{36}, \qquad \gamma' := \frac{\varepsilon}{6}.$$

3.2 Techniques and Comparison to Previous Work

The problem of optimal linear contracts for supermodular functions based on graphs, see Section 3.1, was introduced and studied in [22]. It is impossible to get any multiplicative approximation or additive FPTAS unless $\mathcal{P} = \mathcal{NP}$, since one can construct an instance where the optimal contract has positive value if and only if the graph has a clique of certain size [22]. One of the main results in [22] is an additive PTAS when all the agents have the same cost. In the same cost case, the PTAS in [22] follows the plan below.

Step 1. Show that there exists a near optimal set S' such that $\deg_{S'}(i)$ is $\Omega(n)$ for all $i \in S'$. **Approach.** This step is accomplished by proving that a *core* of an optimal solution can be selected as S'. A core of a graph is an inclusion-wise maximal subgraph such that all vertices in the subgraph have degree larger than some specified threshold, i.e. $\deg_{S'}(i) \geq \sigma n$ for all $i \in S'$.

Step 2. Using the existing techniques for densest subgraphs, as in [6, 19], obtain sufficiently accurate estimates for $\deg_{S'}(i)$, $i \in V$. **Approach.** These estimates are obtained by sampling a set of size $\Theta(\ln n)$ uniformly from V, then finding the number of sampled neighbours for each $i \in V$. One can then scale the number of sampled neighbours for i, $i \in V$, to obtain an estimate for $\deg_{S'}(i)$. If one repeats this process $n^{\text{poly}(1/\varepsilon)}$ times then with high probability one of the sampled sets will provide accurate estimates for $\deg_{S'}(i)$, for $i \in V$ with $\deg_{S'}(i) \geq \sigma n = \Omega(n)$. Furthermore, every $i \in V$ with $\deg_{S'}(i) < \sigma n$ will have low estimated degree.

Step 3. Using the estimates for $\deg_{S'}(i)$, $i \in V$ set up an LP with a "small" number of constraints. Then solve the LP and randomly round the solution. **Approach.** The estimates allow us to linearize the terms in the function $L(\cdot)$. Then solve the constructed LP and round each entry in the found optimal solution independently. Because $\deg_{S'}(i) = \Omega(n)$, for $i \in S'$, the LP involves relatively "large bounds". Hence, one can show good

concentration for each constraint in the LP after randomized rounding. Since the LP has a relatively "small" number of constraints, one can thus demonstrate that the rounded solution is likely to be feasible for the LP.

All of the known techniques start to fall apart in the case of different costs! Here is a list of the main challenges for the general costs case, and the techniques that we develop.

Challenge 1. It is unclear whether there is a near optimal set S' such that $\deg_{S'}(i)$ is $\Omega(n)$ for all $i \in S'$. Generally speaking, one can ignore every "cheap" agent $i \in S'$ because their contribution to $L(\cdot)$ is negligible regardless of $\deg_{S'}(i)$. What about "expensive" agents? Some straightforward modifications of techniques in [22] show that each "expensive" agent $i \in S'$ has to satisfy $\deg_{S'}(i) = \Omega(\sqrt{n})$, but that bound is too weak for later steps to work. **Our approach.** We introduce a concept of *pseudo-core*, see Section 3.4.2. We then develop a procedure, called *iterative pseudo-coring*, that iteratively increases costs for "expensive" agents and removes agents that are not a part of the desired pseudo-core, see Algorithm 3. The iterative nature of the procedure allows us to control changes in functions that we use to approximate $L(\cdot)$. The total cost increase for "expensive" agents guarantees that remaining "expensive" agents have sufficiently high degree $\deg_{S'}(\cdot)$. In particular, we show that each "expensive" agent $i \in S'$ has to satisfy $\deg_{S'}(i) = \Omega(n/\ln\ln n)$.

Challenge 2. Sufficiently accurate estimates for $\deg_{S'}(i)$, $i \in V$ can be efficiently obtained only for i with $\deg_{S'}(i) = \Omega(n)$. **Our approach.** We still use the accurate estimate for each agent i with $\deg_{S'}(i) = \Omega(n)$, that can be obtained through sampling. For all "expensive" agents in S' without accurate estimates for $\deg_{S'}(i)$, our iterative pseudo-coring guarantees a sufficiently high lower bound on $\deg_{S'}(\cdot)$ and we use this lower bound instead.

Challenge 3. The absence of accurate estimates for some "expensive" agents in S' prevents us from linearizing the terms for those agents in the function $L(\cdot)$. Moreover, since there are no accurate estimates for $\deg_{S'}(i)$ for many agents $i \in V$, we need to design a new LP. This new LP makes any naive rounding approach infeasible due to concentration requirements. **Our approach.** With concerns about linearizing $L(\cdot)$ in mind, we demonstrate that we can ignore the terms in $L(\cdot)$ that correspond to agents without accurate estimates of their degrees. To circumvent the concentration issues, we design a preprocessing procedure for an optimal LP solution. Our preprocessing procedure guarantees that "expensive" agents have their variable set to zero or have their variable set to a sufficiently large value. The LP solution achieved by the preprocessing allows us to obtain necessary concentration results.

We believe that our approaches for tackling the first and last challenges can be of

independent interest in contract theory and in a broader range of areas.

3.3 Proof Overview and Intuition

We would like to emphasize that the exposition in this section is done with a lot of omissions and without taking care of some technical details. The sole purpose of this section is to provide intuition for the objects and procedures introduced in later sections. No part of this section should be understood as a formal definition of any object.

Let S^* be an optimal set, i.e. the set S^*, $S^* \subseteq V$ that achieves $\max_{S \subseteq V} g(S)$. Roughly speaking, our approach attempts to obtain enough of information about S^* so that we can set up an appropriate linear program and afterwards round its optimal solution.

As a first step we generalize the approach from [22] built on cores. We change it to *pseudo-core* and show that there exists a set S', $S' \subseteq S^*$ such that:

(i) the value $g(S')$ of S' is not far from the optimal value $g(S^*) = OPT$, in particular $g(S') \geq g(S^*) - \varepsilon$;

(ii) the degree $\deg_{S'}(v)$ of each vertex v in S' is sufficiently high in comparison to its cost c_v, in particular for each vertex $v \in S'$ we have

$$\frac{c_v}{\deg_{S'}(v)(\deg_{S'}(v)+1)} \leq \frac{1}{\gamma\,(n^2 + (6/\varepsilon)n)}\,. \tag{3.1}$$

Then following the idea from [22], we assume that we **know** the vertices in V with sufficiently high number of neighbours in S', in particular we **know** the set

$$H := \{v \in V \,:\, \deg_{S'}(v) \geq \sigma n\}$$

and for each $v \in H$ we **know** $\deg_{S'}(v)$. We can assume that we know the set H and the degrees $\deg_{S'}(v)$, $v \in H$, because through sampling [22, 19] we can obtain accurate estimates for $\deg_{S'}(v)$, $v \in V$ whenever $\deg_{S'}(v)$ is sufficiently high. Analogously to [22] we make an observation that the following inequality holds

$$R(S^*) \geq R(S' \cap H) \geq R(S^*) - 2\varepsilon/3\,.$$

From now on, due to a possibility of an exhaustive inspection we assume that we **know** also $R(S' \cap H)$.

First Linear Program. At this point, one can try to set up a linear program similar to the one in [22]. Here is a rough idea of how such a linear program could look.

$$\min_{\{x_v\}_{v\in V}} \quad \sum_{v\in H} \frac{c_v}{\deg_{S'}(v)} \cdot x_v \tag{3.2}$$

$$\text{subject to} \quad \sum_{v\in H} \deg_{S'}(v) \cdot x_v \geq 2 \cdot R(S' \cap H) \tag{3.3}$$

$$\sum_{u\in N(v)} x_u \geq \deg_{S'}(v) \qquad \text{for all } v \in H \tag{3.4}$$

$$0 \leq x_v \leq 1 \qquad \text{for all } v \in V\,. \tag{3.5}$$

The objective function (3.2) attempts to minimize the negative effect of agents in H on the function $L(\cdot)$, while the constraint (3.3) attempts to guarantee a sufficiently high value of $R(\cdot)$. The constraint in (3.4) guarantees that the degrees are behaving properly for agents in H.

In the case when all costs are equal, one can also add the constraints $x_v = 0$ for all $v \in V \setminus H$, see [22]. In the general case, we need to be more careful with agents in $V \setminus H$, because it is not clear how to handle the agents in $V \setminus H$. Indeed, the constraint (3.4) potentially relies on the neighbours in $V \setminus H$, while the objective function (3.2) does not account for the agents in $V \setminus H$. This becomes an issue, since in general it is not feasible to efficiently obtain accurate estimates for $\deg_{S'}(v)$, $v \in V \setminus H$. So, we cannot simply add constraints analogous to the constraint (3.4) for $v \in V \setminus H$.

Final Linear Program. Now, we need to take a closer look at the agents in $V \setminus H$ and H. We split all agents into four different groups A, B, C and D depending on their cost and on their presence in H. Intuitively, we have the following situation:

- for agents in A and in B, we are not sure whether we should include them or not;
- for agents in C, we are sure that we include them whenever possible, because they have relatively low costs;
- for agents in D, we are sure that we never include them; because they have relatively high costs.

An agent $v \in H$ lies in A if the agent satisfies the inequality (3.1). Otherwise an agent $v \in H$ lies in D. Recall, that we assume that for each agent $v \in H$ we know $\deg_{S'}(v)$.

An agent $v \in V \setminus H$ lies in B, C, D, respectively, if $c_v \in \left(\varepsilon/(2n), \frac{\sigma n(\sigma n+1)}{\gamma(n^2+(6/\varepsilon)n)}\right]$, $c_v \in \left[0, \frac{\varepsilon}{2n}\right]$, $c_v \in \left(\frac{\sigma n(\sigma n+1)}{\gamma(n^2+(6/\varepsilon)n)}, +\infty\right)$, respectively. Note, that the costs of agents in $D \setminus H$ are so high, that they cannot satisfy the inequality (3.1) due to the definition of the set H.

Note that by definition, we **know** the sets A, B, C and D. Now, we can obtain a lower bound d_v on $\deg_{S'}(v)$ for all $v \in B \cap S'$ using the inequality (3.1). So, we can set a final linear program.

$$\min_{\{x_v\}_{v\in V}} \quad \sum_{v\in A} \frac{c_v}{\deg_{S'}(v)} \cdot x_v \tag{3.6}$$

$$\text{subject to} \quad \sum_{v\in H} \deg_{S'}(v) \cdot x_v \geq 2 \cdot R(S' \cap H) \tag{3.7}$$

$$\sum_{u\in N(v)} x_u \geq \deg_{S'}(v) \qquad \text{for all } v \in A \tag{3.8}$$

$$\sum_{u\in N(v)} x_u \geq d_v \cdot x_v \qquad \text{for all } v \in B \tag{3.9}$$

$$x_v = 1 \qquad \text{for all } v \in C \tag{3.10}$$

$$x_v = 0 \qquad \text{for all } v \in D \tag{3.11}$$

$$0 \leq x_v \leq 1 \qquad \text{for all } v \in V\,. \tag{3.12}$$

Randomized Rounding. We can now solve the final linear program and obtain an optimal solution. Then we can try to apply randomized rounding to the found optimal solution. This strategy does not immediately work though. To make it work, we need to strengthen the constraint (3.9) in the final linear program, and before randomized rounding we need to preprocess the optimal solution for the final linear program.

Further note that the objective function (3.6) of the final linear program still does not account for all agents, but only for the agents in A. However, we can show that if the randomized rounding has sufficient concentration with respect to the constraint (3.9) then the costs of all agents outside of A can be ignored.

It is challenging to achieve the sufficient concentration with respect to constraint (3.9). For this, we need to strengthen the previously obtained bounds d_v, $v \in B$. We prove that d_v, $v \in B$ can be set to the maximum between $\Omega(n/\ln\ln(n))$ and the previous bound obtained only from (3.1).

After that, to achieve the concentration, we show how to obtain a near optimal (and also "near feasible") solution x^* to the final linear program, where each non-zero x^*_v, $v \in A \cup B$ is sufficiently large, i.e. non-zero variables x^*_v, $v \in A \cup B$ are not too close to 0. All these properties allow us to prove the desired concentration results and so successfully round the solution x^*.

3.4 Formal Statements

Our algorithm heavily relies on the existence of a well-structured near optimal solution S'. We neither know the set S' nor intend to find it. However, its existence allows us to set up an LP that is *up to some extent* a relaxation of our problem. Indeed, the LP ignores the cost of some agents, and forces some agents to have their variable set to 0 or 1 when not all near optimal solutions may require doing so. Nonetheless, the LP will capture well-structured near optimal solutions, such as S'.

The next lemma captures the desired properties of a near optimal solution S'. In all remaining, parts of the paper S' refers to a set satisfying the properties in Lemma 3.4.1.

Lemma 3.4.1. *There is a set $S' \subseteq V$ which has the following properties*

(i) $g(S') \geq OPT - \varepsilon$,

(ii) $\left|\{v \in S' \mid \deg_{S'}(v) \geq \frac{\varepsilon}{6}n\}\right| \geq \frac{\varepsilon}{6}n = \gamma' n$,

(iii) for all $v \in S'$ either

- *(a)* $\deg_{S'}(v) \geq \frac{(1-\varepsilon)^2 \cdot \varepsilon^5 \ln(2)}{36e} \cdot \frac{n}{\ln\ln(n)} = \delta^{-1} \cdot \frac{n}{\ln\ln(n)}$, *and* $\deg_{S'}(v)(\deg_{S'}(v)+1)/c_v \geq \gamma\,(n^2 + (6/\varepsilon)n)$, *or*
- *(b)* $c_v \leq \varepsilon/(2n)$.

At first glance, the next theorems might appear mysterious. Indeed, we start with a near optimal S' and set up an LP based on S'. Then we solve the LP and round its optimal solution to another near optimal set. Actually, as we will see later, the information used in the LP about the near optimal set S' can be efficiently obtained without knowing S'. This can be done through sampling or exhaustive inspection due to the properties guaranteed by Lemma 3.4.1.

Note that in the next definition the agents are partitioned into four parts A, B, C and D. As defined before, C are the "cheap" agents, i.e. the agents satisfying (iiib) in Lemma 3.4.1.

The agents in A are the agents that could potentially lie in the set mentioned in (ii) and at the same time could satisfy (iiia) in Lemma 3.4.1. The agents in B are the agents that cannot lie in the set mentioned in (ii) but could satisfy (iiia) in Lemma 3.4.1. The agents in D can satisfy neither (iiia) nor (iiib) in Lemma 3.4.1. We note that the second bound of (iiia) in Lemma 3.4.1 is equivalent to $\deg_{S'}(v) \geq \sqrt{c_i \cdot \gamma\,(n^2 + (6/\varepsilon)n) + (1/4)} - (1/2)$. Here, the parameters $\hat{d}_i$, $i \in H$ represent estimates of degrees $\deg_{S'}(i)$.

Definition 1. Let $S' \subseteq V$ be a set as in Lemma 3.4.1. Recall $C = \{i \in V : c_i \leq \varepsilon/(2n)\}$. Let us consider H such that $\{i \in S' \setminus C : \deg_{S'}(i) \geq \sigma \cdot n\} \subseteq H \subseteq \{i \in V \setminus C : \deg_{S'}(i) \geq (1-\varepsilon)\sigma \cdot n/(1+\varepsilon)\}$. Let $\hat{d}_i$, $i \in H$ be such that we have $(1-\varepsilon)\deg_{S'}(i) \leq \hat{d}_i \leq (1+\varepsilon)\deg_{S'}(i)$. Let d_i, $i \in V \setminus C$ be defined as

$$d_i := \max\left(\frac{(1-\varepsilon)^2 \cdot \varepsilon^5 \ln(2)}{36e} \cdot \frac{n}{\ln\ln(n)}, \qquad \sqrt{c_i \cdot \gamma\left(n^2 + \frac{6}{\varepsilon}n\right) + \frac{1}{4}} - \frac{1}{2}\right).$$

Now define $A := \{i \in H : \hat{d}_i/(1-\epsilon) \geq d_i\}$, $D := (H \setminus A) \cup \{i \in V \setminus (H \cup C) : d_i \geq \sigma n\}$, and $B := V \setminus (A \cup C \cup D)$.

Now we establish that the objective function in our LP reflects the value of $L(\cdot)$. Note that in the objective function we consider only agents in A, and we ignore those in B and C. We will later show the agents in B and C do not contribute much to $L(\cdot)$.

Lemma 3.4.2. *Consider the following LP and let OPT_{LP} be its optimal value.*

$$\min_{\{x_v\}_{v\in V}} \quad \sum_{v\in A} \frac{c_v}{\hat{d}_v} \cdot x_v \tag{3.13}$$

$$\textit{subject to} \quad \sum_{v\in A} \hat{d}_v x_v \geq 2(1-\varepsilon) \cdot |E(S')| \tag{3.14}$$

$$\sum_{u\in N(v)} x_u \geq \left(\frac{1}{1+\varepsilon}\right) \cdot \hat{d}_v \qquad \textit{for all } v \in A \tag{3.15}$$

$$\sum_{u\in N(v)} x_u \geq \left(\frac{1}{1+\varepsilon}\right) \cdot d_v \cdot x_v \qquad \textit{for all } v \in B \tag{3.16}$$

$$x_v = 1 \qquad \textit{for all } v \in C \tag{3.17}$$

$$x_v = 0 \qquad \textit{for all } v \in D \tag{3.18}$$

$$0 \leq x_v \leq 1 \qquad \textit{for all } v \in V\,. \tag{3.19}$$

Then OPT_{LP} is less than or equal to $\frac{1}{1-\varepsilon}(1 - L(S'))$, i.e. less than or equal to

$$\frac{1}{1-\varepsilon}\left(\sum_{v\in S'}\frac{c_v}{\deg_{S'}(v)}\right).$$

Next we establish that there is a desirable near-feasible solution x^* which we can compute efficiently. This x^* satisfies necessary properties for concentration in randomized rounding, is optimal, and is near feasible.

Theorem 3.4.3. *Let OPT_{LP} be the optimal value of the LP (3.13)-(3.19). Then there exists x^* satisfying*

$$x^*_v > 0 \implies x^*_v \geq 1/n^{1/2} \qquad \text{for all } v \in A \tag{3.20}$$

$$x^*_v > 0 \implies \sum_{u\in N(v)} x^*_u \geq \kappa \cdot n^{1/2} \cdot \ln\left(2n^2\right) \qquad \text{for all } v \in B\,, \tag{3.21}$$

and also

$$\sum_{v\in A}\frac{c_v}{\hat{d}_v}\cdot x^*_v \leq OPT_{LP} \tag{3.22}$$

$$\sum_{v\in A}\hat{d}_v x^*_v \geq 2(1-\varepsilon)^2\cdot|E(S')| \tag{3.23}$$

$$\sum_{u\in N(v)} x^*_u \geq \left(\frac{(1-\varepsilon)^2}{1+\varepsilon}\right)\cdot\hat{d}_v \qquad \text{for all } v \in A \tag{3.24}$$

$$\sum_{u\in N(v)} x^*_u \geq \left(\frac{1-2\varepsilon(1+\varepsilon)}{1+\varepsilon}\right)\cdot d_v\cdot x^*_v \qquad \text{for all } v \in B \tag{3.25}$$

$$x^*_v = 1 \qquad \text{for all } v \in C \tag{3.26}$$

$$x^*_v = 0 \qquad \text{for all } v \in D \tag{3.27}$$

$$0 \leq x^*_v \leq 1 \qquad \text{for all } v \in V\,. \tag{3.28}$$

Moreover, such an x^* can be efficiently computed.

Finally, we establish that we can indeed round the solution x^* to a near optimal solution.

Theorem 3.4.4. *Let us be given a vector x^* as in Theorem 3.4.3, i.e. a vector x^* satisfying (3.20)-(3.28). Let us construct a set S'' by independently including each $v \in A \cup B$ into the set S'' with probability x^*_v; and afterwards including each $v \in C$ into the set S'' if $N(v)\cap S'' \neq \varnothing$. Then with probability at least $1-\sqrt{\varepsilon}-1/n$ we obtain $g(S'') \geq g(S') - 4\sqrt{\epsilon} \geq OPT - 5\sqrt{\epsilon}$.*

We observe that Theorem 3.4.4 allows us to independently repeat the rounding in Theorem 3.4.4 to achieve a good approximation with high probability. In particular, if we independently run our algorithm $\ln(1/n)/\ln(\sqrt{\varepsilon}+1/n)$ times then we obtain a set S'' with $g(S'') \geq OPT - 5\sqrt{\epsilon}$ with probability at least $1 - 1/n$.

Before we move ahead let us explain the intuition behind the constraints in LP (3.13)-(3.19) from Lemma 3.4.2. Let us assume that we are dealing with an integral solution $\bar{x}$ and the corresponding set $\bar{S}$, $\bar{S} \subseteq V$. The objective function (3.13) is estimating the portion of the principal's reward transferred to the agents in A. Transfers done to all other agents are not taken into account by the LP. Later, we see that with such an objective function OPT_{LP} is sufficiently close to $1 - L(\bar{S})$. The constraint (3.14) guarantees that the value $R(\bar{S} \cap A)$ is at least $(1-\varepsilon) \cdot R(S')$. The constraints (3.15) and (3.16) guarantee that for every $v \in A$ and $v \in \bar{S} \cap B$ the value $\deg_{\bar{S}}(v)$ has a lower bound similar to the one in Lemma 3.4.1. The constraint (3.17) guarantees that all agents in V with sufficiently small costs are included in $\bar{S}$. The constraint (3.18) guarantees that we do not include agents that do not have a lower bound on $\deg_{\bar{S}}(v)$ similar to the one in Lemma 3.4.1.

Let us explain the way we set up the LP (3.13)-(3.19). One of the parameters present in the LP is $|E(S')|$. Note, that $|E(S')|$ can take only integral values between 0 and $n(n-1)/2$ and hence permits us to do exhaustive search. It remains to verify that the set H and the values $\hat{d}_v$ also can be found. For this, we use the oblivious samplers of [22, 19].

Lemma 3.4.5. *[22, 19] Let $K \subseteq V$ be any subset of nodes with $|K| \geq \frac{\varepsilon}{6} \cdot n$. There is an $O\left(n^{\text{poly}\left(\frac{1}{\varepsilon}\right)}\right)$-time algorithm unaware of the nodes K that computes $t = O\left(n^{\text{poly}\left(\frac{1}{\varepsilon}\right)}\right)$ different marginal value estimates $\widetilde{d}_v^{\,1}, \ldots, \widetilde{d}_v^{\,t}$ of $\deg_K(v)$ for each node $v \in V$ such that with probability at least $1 - \frac{1}{\text{poly}(n)}$ there exists an $j \in [t]$ whose estimates satisfy,*

(i) $\Pr\left[\widetilde{d}_v^{\,j} < (1+\varepsilon)\sigma \cdot n\right] \geq 1 - \frac{1}{2n^3}, \quad \forall v \in V$ *with* $\deg_K(v) < \frac{\varepsilon(1+\varepsilon)\sigma}{9} n$

(ii) $\Pr\left[(1-\varepsilon) \cdot \deg_K(v) \leq \widetilde{d}_v^{\,j} \leq (1+\varepsilon) \cdot \deg_K(v)\right] \geq 1 - \frac{1}{2n^3}, \quad \forall v \in V$ *with* $\deg_K(v) \geq \frac{\varepsilon(1+\varepsilon)\sigma}{9} n$

We remark that Lemma 3.4.5 allows us to set the LP (3.13)-(3.19) even without knowing the set S'. Note that by Lemma 3.4.1 (ii) we have $|S'| \geq \varepsilon n/6$, and so we can apply Lemma 3.4.5 for K equal to S'. Then we run the algorithm outlined in Theorem 3.4.3 and Theorem 3.4.4 independently for every $j \in [t]$ by setting $\hat{d}_v := \widetilde{d}_v^{\,j}$ and then selecting

$$H := \{i \in V \setminus C \,:\, \hat{d}_i \geq \sigma n(1-\varepsilon)\}\,.$$

Let us argue that with high probability at least for one $j \in [t]$ the selected above $\hat{d}_i$, $i \in H$ and the set H are correct. Lemma 3.4.5 guarantees that with probability $1 - 1/\mathrm{poly}(n)$ there exists $j \in [t]$ such that both (i) and (ii) are satisfied. Conditioned on this, by (i) and (ii) with probability at least $1 - 1/(2n^2)$ we have

$$\{i \in S' \setminus C \,:\, \deg_{S'}(i) \geq \sigma \cdot n\} \subseteq H \subseteq \{i \in V \setminus C \,:\, \deg_{S'}(i) \geq (1-\varepsilon)\sigma \cdot n/(1+\varepsilon)\}\,,$$

and for all $i \in H$ we have

$$(1-\varepsilon)\deg_{S'}(i) \leq \hat{d}_i \leq (1+\varepsilon)\deg_{S'}(i)\,.$$

3.4.1 Linear Relaxation and Rounding

In this section, we assume that we have the information about S' that is required to set up the LP (3.13)-(3.19). With this assumption, we show how to find a near optimal solution. To do this we prove Lemma 3.4.2, Theorem 3.4.3 and Theorem 3.4.4.

Proof of Lemma 3.4.2

Let S' be a set of agents satisfying Lemma 3.4.1. Clearly, the characteristic vector x' of the set $S' \cup C$ is a feasible solution for the LP (3.13)-(3.19). For comparison's sake, we note that the bound of $\deg_{S'}(v)(\deg_{S'}(v)+1) \geq \gamma\,(n^2 + (6/\varepsilon)n) \cdot c_v$ implies $\deg_{S'}(v)^2/c_v \geq \frac{(1-\varepsilon)\varepsilon^4}{36} \cdot n^2$. Moreover, the value of x' equals

$$\sum_{v \in A} \frac{c_v}{\hat{d}_v} \cdot x'_v \leq \sum_{v \in A} \frac{c_v}{(1-\varepsilon)\deg_{S'}(v)} \leq \sum_{v \in S'} \frac{c_v}{(1-\varepsilon)\deg_{S'}(v)} = \frac{1}{1-\varepsilon}\left(1 - L(S')\right)\,, \tag{3.29}$$

showing the desired statement.

Proof of Theorem 3.4.3

In this section, we design an efficient preprocessing procedure, i.e. Algorithm 1, for feasible solutions of the LP (3.13)-(3.19). We apply this procedure to an optimal solution of the LP (3.13)-(3.19). The following lemma directly implies Theorem 3.4.3.

Given a vector x, Algorithm 1 iteratively sets to zero entries x_v which correspond to the inequalities in the LP (3.13)-(3.19) without sufficient concentration guarantee. Intuitively, in the proof of Lemma 3.4.6 we show that Algorithm 1 does not set too many coordinates

to zero. In particular, Algorithm 1 consists of two loops. The first loops "zeroes out" in x coordinates that sum up to at most $n^{1/2}$. Later, we show that $n^{1/2}\left(\kappa \cdot \ln\left(2n^2\right)+1\right)\ln(n)$ is a rough upper bound on the sum of coordinates "zeroed" in the second loop. These two upper bounds allow us to obtained the desired statement.

Algorithm 1: Fractional Coring

Input: Feasible solution x to LP (3.13)-(3.19)

1 $x^* \leftarrow x, j \leftarrow 1$
2 **for** $v \in A$ **do**
3 | **if** $x_v^* \leq 1/n^{1/2}$ **then**
4 | | $x_v^* \leftarrow 0$
5 **while** *true* **do**
6 | $v \leftarrow \arg\min_{\substack{v\in B \\ x_v^*>0}} \left(\sum_{u\in N(v)} x_u^*\right)$
7 | $w \leftarrow \arg\min_{\substack{w\in B \\ x_w^*>0}} \left(\varepsilon d_w x_w^* - \sum_{i=1}^{j-1} x_{v_i}\right)$
8 | **if** $\left(\sum_{u\in N(v)} x_u^*\right) < \kappa \cdot n^{1/2} \cdot \ln\left(2n^2\right)$ **then**
9 | | $x_v^* \leftarrow 0, v_j \leftarrow v, j \leftarrow j+1$
10 | **else if** $\left(\varepsilon d_w x_w^* - \sum_{i=1}^{j-1} x_{v_i} - n^{1/2}\right) < 0$ **then**
11 | | $x_w^* \leftarrow 0, v_j \leftarrow w, j \leftarrow j+1$
12 | **else**
13 | | Exit loop

Output: x^*

Lemma 3.4.6. *Given an optimal solution x for the LP (3.13)-(3.19), Algorithm 1 outputs a vector x^* such that (3.20)-(3.28) hold.*

Proof. We first consider the changes caused by the loop on line 2. Let X be the set of v such that x_v^* was set to 0 in this loop. Observe that we have $\sum_{v\in X} x_v \leq n^{1/2}$. This implies

$$\sum_{u\in N(v)} x_u \leq \sum_{u\in N(v)} x_u^* + n^{1/2} \qquad \text{for all } v \in A \cup B$$

at the time point when the algorithm moves to line 5. Thus, we obtain

$$\sum_{u\in N(v)} x^*_u \geq \left(\frac{1-\varepsilon}{1+\varepsilon}\right)\cdot \hat{d}_v \qquad \text{for all } v\in A$$

at this point of the algorithm. Similarly, notice that at this point we have

$$\sum_{v\in A} \hat{d}_v x_v - \sum_{v\in A} \hat{d}_v x^*_v \leq n^{3/2}\,,$$

which implies that (3.23) is true at this point. Now we consider how the rest of the algorithm changes x^* from this point onwards, i.e. from the moment when the algorithm moves to line 5 for the first time.

Let J be the value of j immediately before Algorithm 1 terminated. Observe that at the end of the algorithm we have $\sum_{u\in N(v)} x^*_u \geq \kappa\cdot n^{1/2}\cdot \ln(2n^2)$ for all $v\in B$, $x^*_v>0$, implying that (3.21) holds for x^*. Note, the entries of x^* indexed by $v\in A\cup C$ did not change after the first loop of the algorithm, implying that (3.20), (3.22), (3.23) and (3.26) hold for x^*, and we have $x^*_v\geq 1/n^{1/2}$ or $x^*_v=0$ for all $v\in A$. By line 10 in Algorithm 1, for all $v\in B$ with $x^*_v>0$ we have

$$\varepsilon\cdot d_v\cdot x^*_v > \sum_{i=1}^{J} x_{v_i} + n^{1/2}\,.$$

So x^* satisfies also (3.25). Clearly, the output x^* satisfies (3.27) and (3.28). In the remaining part of the proof we show that x^* satisfies also (3.24).

Consider the jth iteration, i.e. the iteration right before $x^*_{v_j}$ gets set to 0. For compactness in this proof, we introduce the following notation

$$\bar{\kappa} := \kappa\cdot n^{1/2}\cdot \ln\left(2n^2\right), \quad \bar{\delta} := \delta\cdot\frac{\ln\ln(n)}{n}, \quad \bar{\delta}_\varepsilon := \frac{\bar{\delta}}{\varepsilon}\,.$$

We have the following facts:

(i) $\displaystyle\sum_{u\in N(v_j)} x^*_u < \bar{\kappa}, \quad \text{or} \quad x^*_{v_j}/\bar{\delta}_\varepsilon \leq \varepsilon d_{v_j} x^*_{v_j} \leq \sum_{i=1}^{j-1} x_{v_i} + n^{1/2},$

(ii) $\displaystyle\sum_{u\in N(v_j)} x_u \leq \sum_{u\in N(v_j)} x^*_u + \sum_{i=1}^{j-1} x_{v_i} + n^{1/2}$, and

(iii) $x_{v_j}/\bar{\delta} \leq \sum_{u \in N(v_j)} x_u$.

Here, (i) follows from the fact that $x^*_{v_j}$ is set to 0 in the current iteration. Due to the definition of $v_1, \ldots, v_{j-1}$ we have that $\sum_{u \in V} x_u - \sum_{u \in V} x^*_u$ is at most $\sum_{i=1}^{j-1} x_{v_i} + n^{1/2}$, resulting in (ii). The inequality (iii) holds since x^* after the loop on line 2 is feasible for the LP (3.22)-(3.28).

We combine (i), (ii) and (iii) to obtain that one of the following inequalities holds

$$x^*_{v_j} \leq \bar{\delta} \cdot \left(\bar{\kappa} + n^{1/2} + \sum_{i=1}^{j-1} x_{v_i} \right) \qquad \text{or} \qquad x^*_{v_j} \leq \bar{\delta}_\varepsilon \cdot \left(\sum_{i=1}^{j-1} x_{v_i} + n^{1/2} \right) ,$$

depending on the outcome in (i).

Let us prove that

$$x_{v_j} \leq \left(\bar{\kappa} + n^{1/2} \right) \bar{\delta}_\varepsilon \left(\bar{\delta}_\varepsilon + 1 \right)^{j-1} \tag{3.30}$$

holds for every $j = 1, \ldots, J-1$. We prove the inequality (3.30) by induction. It is straightforward to notice that $x_{v_1} \leq \bar{\delta}_\varepsilon \cdot (\bar{\kappa} + n^{1/2})$. Let us assume that for all $1 \leq i \leq j$ the inequality (3.30) holds.

Case $\sum_{u \in N(v_{j+1})} x^*_u < \bar{\kappa}$. In this case, we have

$$\begin{aligned}
x^*_{v_{j+1}}/\bar{\delta} &\leq \left(\bar{\kappa} + n^{1/2} \right) + \sum_{i=1}^{j} x_{v_i} \\
&\leq \left(\bar{\kappa} + n^{1/2} \right) + \sum_{i=1}^{j} \left(\bar{\kappa} + n^{1/2} \right) \bar{\delta}_\varepsilon \left(\bar{\delta}_\varepsilon + 1 \right)^{i-1} \\
&= \left(\bar{\kappa} + n^{1/2} \right) + \left(\bar{\kappa} + n^{1/2} \right) \cdot \bar{\delta}_\varepsilon \cdot \sum_{i=1}^{j} \left(\bar{\delta}_\varepsilon + 1 \right)^{i-1} \\
&= \left(\bar{\kappa} + n^{1/2} \right) \left(1 + \bar{\delta}_\varepsilon \cdot \frac{(\bar{\delta}_\varepsilon + 1)^j - 1}{\bar{\delta}_\varepsilon} \right) \\
&= \left(\bar{\kappa} + n^{1/2} \right) \cdot \left(\bar{\delta}_\varepsilon + 1 \right)^j ,
\end{aligned}$$

leading us to the inequality (3.30) by rearranging for $x^*_{v_{j+1}}$.

Case $x^*_{v_{j+1}}/\bar{\delta}_\varepsilon \le \sum_{i=1}^{j} x_{v_i} + n^{1/2}$. In this case, we have

$$\begin{aligned} x^*_{v_{j+1}} &\le \bar{\delta}_\varepsilon \cdot \left(\sum_{i=1}^{j} x_{v_i} + n^{1/2} \right) \\ &\le \bar{\delta}_\varepsilon \cdot \left(\sum_{i=1}^{j} \left(\bar{\kappa} + n^{1/2}\right) \bar{\delta}_\varepsilon \left(\bar{\delta}_\varepsilon + 1\right)^{i-1} + n^{1/2} \right) \\ &= \bar{\delta}_\varepsilon \cdot \left(\left(\bar{\kappa} + n^{1/2}\right) \left(\left(\bar{\delta}_\varepsilon + 1\right)^{j} - 1 \right) + n^{1/2} \right) \\ &\le \left(\bar{\kappa} + n^{1/2}\right) \bar{\delta}_\varepsilon \left(\bar{\delta}_\varepsilon + 1\right)^{j} , \end{aligned}$$

leading us again to the inequality (3.30). Thus, the inequality (3.30) always holds.

By (3.30), we have

$$\begin{aligned} \sum_{j=1}^{J} x_{v_j} &\le \sum_{j=1}^{J} \left(\bar{\kappa} + n^{1/2}\right) \bar{\delta}_\varepsilon \left(\bar{\delta}_\varepsilon + 1\right)^{j-1} \\ &= \left(\bar{\kappa} + n^{1/2}\right) \left(\left(\bar{\delta}_\varepsilon + 1\right)^{J} - 1 \right) \le \left(\bar{\kappa} + n^{1/2}\right) \left(\bar{\delta}_\varepsilon + 1\right)^{J} \\ &= \left(\bar{\kappa} + n^{1/2}\right) \left(\frac{36e}{(1-\varepsilon)^2 \cdot \varepsilon^6 \ln(2)} \cdot \frac{\ln\ln(n)}{n} + 1 \right)^{J} \\ &\le \left(\bar{\kappa} + n^{1/2}\right) \cdot \exp\left(\frac{36e}{(1-\varepsilon)^2 \cdot \varepsilon^6 \ln(2)} \cdot \ln\ln(n) \right) \\ &= \left(\bar{\kappa} + n^{1/2}\right) \cdot \ln(n)^{\alpha} , \end{aligned}$$

where $\alpha := (36e) / ((1-\varepsilon)^2 \cdot \varepsilon^6 \ln(2))$ and so α is an expression depending only on ε. Here, the first inequality is due to (3.30). The third inequality is due to the definition of e. Thus, we have that for the output x^* the value $\sum_{u \in V} x_u - \sum_{u \in V} x^*_u$ is at most $\left(\bar{\kappa} + n^{1/2}\right) \cdot \ln(n)^{\alpha}$. Note that by definition for all $i \in A$ we have $\hat{d}_i \ge (1-\varepsilon)\sigma \cdot n$. Hence, $\sum_{u \in V} x_u - \sum_{u \in V} x^*_u \le \left(\bar{\kappa} + n^{1/2}\right) \cdot \ln(n)^{\alpha}$ imply that x^* satisfies also (3.24). □

Proof of Theorem 3.4.4

Let us assume that vector x^* satisfies (3.20)-(3.28). Let us construct a set S'' by independently including each $v \in A \cup B$ into the set S'' with probability x^*_v. Afterwards, we include each $v \in C$ into the set S'' if $N(v) \cap S'' \neq \varnothing$. This is precisely the randomized construction of S'' described in Theorem 3.4.4.

Lemma 3.4.7. *Let S'' be constructed as in Theorem 3.4.4. Then we have*

$$\Pr\left[\sum_{v\in A\cap S''}\hat{d}_v \geq (1-\varepsilon)\sum_{v\in A}\hat{d}_v x^*_v\right] \geq 1-\frac{1}{n^2}\,, \tag{3.31}$$

$$\Pr\left[\sum_{v\in S''\cap A}\frac{c_v}{\hat{d}_v} \leq \sum_{v\in A}\frac{c_v}{\hat{d}_v}x^*_v+\varepsilon\right] \geq 1-\frac{1}{n^2}\,, \tag{3.32}$$

and for all $v\in A\cup B$ we have

$$\Pr\left[\deg_{S''}(v) \geq (1-\varepsilon)\sum_{u\in N(v)} x^*_u \;\mid\; v\in S''\right] \geq 1-\frac{1}{n^2}\,. \tag{3.33}$$

Proof. Let us use Chernoff's inequality to bound the probability that the event in (3.31) does not occur, and obtain

$$\begin{aligned}
\Pr\left[\sum_{v\in A\cap S''}\hat{d}_v > (1-\varepsilon)\sum_{v\in A}\hat{d}_v x^*_v\right] &= \Pr\left[\sum_{v\in A\cap S''}\hat{d}_v \leq (1-\varepsilon)\mathbb{E}\left[\sum_{v\in A\cap S''}\hat{d}_v\right]\right]\\
&\leq \exp\left(-\frac{2\varepsilon^2\mathbb{E}\left[\sum_{v\in A\cap S''}\hat{d}_v\right]^2}{n\left(\max_{v\in A}\hat{d}_v\right)^2}\right)\\
&\leq_\star \exp\left(-\frac{2\varepsilon^2(1-\varepsilon)^4\left(\varepsilon/6\cdot n\right)^4}{n^3}\right)\\
&\leq \frac{1}{n^2}\,.
\end{aligned}$$

Here the inequality $\star$ follows from $\hat{d}_v\leq n$ for all $v\in A$; and from the definition of $\hat{d}_v$ in Definition 1 and the fact that the set S' satisfies (ii) in Lemma 3.4.1.

Now we use Hoeffding's inequality to bound the probability the event in (3.32) does not occur. To use this inequality we first obtain some preliminary bounds. Recall by definition for all $v\in A$ we have $(\hat{d}_v/(1-\varepsilon))\geq d_v$. Then if $c_v\geq n^{1/4}$, we can expand and rearrange the definition of d_v to obtain

$$\frac{(\hat{d}_v)^2}{c_v} \geq (1-\varepsilon)^2\cdot\left(\gamma\left(n^2+\frac{6}{\varepsilon}n\right)-\frac{\hat{d}_v}{(c_v(1-\varepsilon))}\right) \geq (1-\varepsilon)^3\gamma\left(n^2+\frac{6}{\varepsilon}n\right)\,,$$

and otherwise $\hat{d}_v/c_v \geq (1-\varepsilon)\sigma n^{3/4}$. Finally we recall $x^*_v \geq 1/n^{1/2}$ for $v \in A, x^*_v > 0$, and observe

$$\sum_{v\in A} \frac{c_v}{n^{3/2}} \leq \sum_{v\in A} \frac{c_v}{\hat{d}_v} \cdot x^*_v \leq_\star \sum_{v\in V} \frac{c_v}{\hat{d}_v} x' < 1\,,$$

where x' corresponds to the set of agents satisfying Lemma 3.4.1, and inequality $\star$ follows from the feasibility of the desired set, equation (3.29). With this, we obtain

$$\begin{aligned}
\Pr\left[\sum_{v\in S''\cap A} \frac{c_v}{\hat{d}_v} > \sum_{v\in A} \frac{c_v}{\hat{d}_v} x^*_v + \varepsilon\right] &= \Pr\left[\sum_{v\in S''\cap A} \frac{c_v}{\hat{d}_v} > \mathbb{E}\left[\sum_{v\in S''\cap A} \frac{c_v}{\hat{d}_v}\right] + \varepsilon\right] \\
&\leq \exp\left(-\frac{2\varepsilon^2}{\sum_{v\in S''\cap A}\left(\frac{c_v}{\hat{d}_v}\right)^2}\right) \\
&\leq_\star \exp\left(-\frac{2\varepsilon^2}{\sum_{\substack{v\in S''\cap A\\ c_v\leq n^{1/4}}}\left(\frac{1}{(1-\varepsilon)\sigma n^{3/4}}\right)^2 + \sum_{\substack{v\in S''\cap A\\ c_v> n^{1/4}}}\left(\frac{c_v}{(1-\varepsilon)^3\gamma(n^2+(6/\varepsilon)n)}\right)}\right) \\
&\leq_{\star\star} \exp\left(-\frac{2\varepsilon^2}{\frac{1}{(1-\varepsilon)^2\sigma^2 n^{1/2}} + \frac{n^{3/2}}{(1-\varepsilon)^3\gamma(n^2+(6/\varepsilon)n)}}\right) \\
&\leq \frac{1}{n^2}\,.
\end{aligned}$$

Here the inequality $\star$ follows from the above discussion bounding costs, and the inequality $\star\star$ follows from noting $|A| \leq n$ and the fact that $\sum_{v\in S''\cap A} c_v < n^{3/2}$. Also note $\gamma\left(n^2 + (6/\varepsilon)n\right)$ is at least $\varepsilon^7 n^2$.

Now we will again use Chernoff bounds for our final concentration result. This time we bound the probability that event (3.33) does not occur, and obtain

$$\begin{aligned}
\Pr\left[\deg_{S''}(v) < (1-\varepsilon)\sum_{u\in N(v)} x^*_u \mid v \in S''\right] &\leq \exp\left(-\frac{\varepsilon^2 \mathbb{E}\left[\deg_{S''}(v) \mid v \in S''\right]}{2}\right) \\
&\leq_\star \exp\left(-\frac{\varepsilon^2 \kappa \cdot n^{1/2} \cdot \ln\left(2n^2\right)}{2}\right) \\
&\leq \frac{1}{n^2}\,,
\end{aligned}$$

where inequality $\star$ is from our guaranteed bound $\mathbb{E}\left[\deg_{S''}(v) \mid v \in S''\right] \geq \kappa \cdot n^{1/2} \cdot \ln\left(2n^2\right)$ for any v with $x_v > 0$. □

Bound on $R(S'')$.

Let us use Lemma 3.4.7 to prove Theorem 3.4.4. First let us estimate $R(S'')$. Let us consider the case when all events described in (3.31) and (3.33) happen. Thus, with probability at least $1-2/n$, we have

$$\begin{aligned} R(S'') \cdot \binom{n}{2} = |E(S'')| &\geq \frac{1}{2} \sum_{v \in S'' \cap A} \deg_{S''}(v) \\ &\geq_{\star} \frac{1}{2} \cdot \frac{(1-\varepsilon)^2}{1+\varepsilon} \sum_{v \in S'' \cap A} \hat{d}_v \\ &\geq_{\star\star} \frac{1}{2} \cdot \frac{(1-\varepsilon)^3}{1+\varepsilon} \sum_{v \in A} \hat{d}_v x^*_v \\ &\geq \frac{(1-\varepsilon)^5}{(1+\varepsilon)} |E(S')| \\ &= \frac{(1-\varepsilon)^5}{(1+\varepsilon)} \cdot R(S') \cdot \binom{n}{2}, \end{aligned}$$

where the inequality $\star$ is due to the events in (3.33) and since x^* satisfies (3.24), the inequality $\star\star$ holds due to the event in (3.31). The last inequality holds since x^* satisfies (3.23).

Bound on $L(S'')$.

Let us now estimate $L(S'')$. Recall that we have

$$L(S'') = 1 - \sum_{v \in S'' \cap A} \frac{c_v}{\deg_{S''}(v)} - \sum_{v \in S'' \cap B} \frac{c_v}{\deg_{S''}(v)} - \sum_{v \in S'' \cap C} \frac{c_v}{\deg_{S''}(v)}\,. \tag{3.34}$$

By the definition of C and by the construction of S'', we have that $\deg_{S''}(v) \geq 1$ and $c_v \leq \varepsilon/(2n)$ for every $v \in S'' \cap C$. Thus, the last term in (3.34) is always at most $\varepsilon/2$, i.e.

$$\sum_{v \in S'' \cap C} \frac{c_v}{\deg_{S''}(v)} \leq \varepsilon/2\,.$$

Let us consider the case when all events described in (3.31), (3.32) and (3.33) happen. With probability at least $1-2/n$ we have that the second term in (3.34) is at most $(OPT_{LP}+\varepsilon)(1+\varepsilon)/(1-\varepsilon)^3$, i.e.

$$\sum_{v \in S'' \cap A} \frac{c_v}{\deg_{S''}(v)} \leq \frac{(OPT_{LP}+\varepsilon)(1+\varepsilon)}{(1-\varepsilon)^3}\,.$$

Indeed,

$$\begin{aligned}\sum_{v\in S''\cap A}\frac{c_v}{\deg_{S''}(v)} &\leq_{\star} \sum_{v\in S''\cap A}\frac{c_v}{(1-\varepsilon)\sum_{u\in N(v)}x^*_u}\\ &\leq_{\star\star}(1+\varepsilon)\sum_{v\in S''\cap A}\frac{c_v}{(1-\varepsilon)^3\hat{d}_v}\\ &\leq_{\star\star\star}\frac{1+\varepsilon}{(1-\varepsilon)^3}\left(\sum_{v\in A}\frac{c_v}{\hat{d}_v}x^*_v+\varepsilon\right)\\ &\leq \frac{1+\varepsilon}{(1-\varepsilon)^3}\left(OPT_{LP}+\varepsilon\right),\end{aligned}$$

where the inequality $\star$ is due to the events in (3.33), the inequality $\star\star$ is due to (3.24) for x^*, the inequality $\star\star\star$ is due to the event in (3.32).

Now it remains to estimate the third term in (3.34), i.e. the term corresponding to the set B. As in the previous case, we assume that all events in (3.31), (3.32) and (3.33) happen. Let us start with an estimate for the expected value of the term corresponding to the set B.

$$\begin{aligned}\mathbb{E}\left[\sum_{v\in S''\cap B}\frac{c_v}{\deg_{S''}(v)}\right] &= \sum_{v\in B}\mathbb{E}\left[\frac{c_v}{\deg_{S''}(v)}\;\middle|\; v\in S''\right]\cdot\Pr[v\in S'']\\ &\leq_{\star}\sum_{v\in B}\left(\frac{(1+\varepsilon)}{(1-2\varepsilon(1+\varepsilon))}\frac{c_v}{d_v\cdot x^*_v}\right)\cdot x^*_v\\ &= \frac{(1+\varepsilon)}{(1-2\varepsilon(1+\varepsilon))}\sum_{v\in B}\frac{c_v}{d_v}\\ &\leq \frac{(1+\varepsilon)}{(1-2\varepsilon(1+\varepsilon))}\sum_{v\in B}\frac{d_v+1}{\gamma\left(n^2+(6/\varepsilon)n\right)}\\ &\leq \frac{(1+\varepsilon)^2}{(1-2\varepsilon(1+\varepsilon))}\sum_{v\in B}\frac{\sigma n}{\gamma\left(n^2+(6/\varepsilon)n\right)}\\ &\leq \varepsilon,\end{aligned}$$

where the inequality $\star$ is due to (3.25) for x^* and due to the event in (3.33). The last two inequalities are due to $c_v \leq d_v(d_v+1)/(\gamma\left(n^2+(6/\varepsilon)n\right))$ for all $v\in B$ and due to the definition of σ,γ.

Using Markov's inequality, we get

$$\Pr\left[\sum_{v\in S''\cap B}\frac{c_v}{\deg_{S''}(v)}\geq\sqrt{\varepsilon}\right]\leq\sqrt{\varepsilon}\,.$$

Hence, with probability at least $1-\sqrt{\varepsilon}-1/n$ we have for sufficiently small ε

$$\begin{aligned}L(S'')&\geq 1-\frac{1+\varepsilon}{(1-\varepsilon)^3}\left(OPT_{LP}+\varepsilon\right)-\sqrt{\varepsilon}-\varepsilon/2\\&=1-\frac{1+\varepsilon}{(1-\varepsilon)^3}\left(\frac{1}{1-\varepsilon}\left(1-L(S')\right)+\varepsilon\right)-2\sqrt{\varepsilon}\\&\geq L(S')-3\sqrt{\varepsilon}\,.\end{aligned}$$

Bound on $g(S'')$.

Thus, with probability at least $1-\sqrt{\varepsilon}-1/n$ we have

$$\begin{aligned}g(S'')&=L(S'')\cdot R(S'')\\&\geq\left(L(S')-3\sqrt{\varepsilon}\right)\cdot\frac{(1-\varepsilon)^5}{1+\varepsilon}R(S')\\&\geq L(S')R(S')-4\sqrt{\varepsilon}\\&=g(S')-4\sqrt{\varepsilon},\end{aligned}$$

using that $L(S')$ and $R(S')$ are at most 1. Thus we obtain the desired statement of Theorem 3.4.4.

3.4.2 Structured Near Optimal Solutions

In this section we prove Lemma 3.4.1. The desired set S' is the output of Algorithm 3 with input given by the optimal set S^* and the original costs c_i, $i\in V$.

Before we proceed, let us make some observations about the structure of an optimal set S^*. The next lemma can be shown by starting with the set S^* and iteratively removing all agents with at most $\varepsilon n/3$ adjacent agents in the set.

Lemma 3.4.8. *[22] If a set S satisfies $R(S)\geq\varepsilon$, then there exists $\widetilde{S}$, $\widetilde{S}\subseteq S$ such that $R(\widetilde{S})\geq R(S^*)-2\varepsilon/3$, $|\widetilde{S}|\geq\frac{\varepsilon}{3}n$ and $\deg_{\widetilde{S}}(i)\geq\frac{\varepsilon}{3}n$ for all $i\in\widetilde{S}$.*

At several places, we need to bound the total cost of agents in the optimal set S^* or its subsets. For this, we use the following observation.

Observation 3.4.9. If a set S^* satisfies $g(S^*) \geq \varepsilon$, or $L(S^*) > 0$, then for every $S \subseteq S^*$ we have $\sum_{i \in S} c_i \leq n$.

Proof. By $g(S^*) \geq \varepsilon$ and $R(S^*) > 0$, we have $L(S^*) > 0$. In its turn rearranging $L(S^*) > 0$ gives us

$$1 > \sum_{i \in S^*} \frac{c_i}{\deg_{S^*}(i)} \geq \sum_{i \in S} \frac{c_i}{n}\,.$$

□

Let us consider a set $S \subseteq V$ and introduce its pseudo-core. Note, that the definition of the pseudo-core does not impose any conditions on the vertices in $S \cap C$, i.e. it does not impose any conditions on the vertices with sufficiently small costs. Let us define the *β-pseudo-core* of S with respect to the costs c'_i, $i \in S$ as an inclusion-wise maximal subset $\bar{S}$ of S such that

$$\frac{\deg_{\bar{S}}(i) \cdot (\deg_{\bar{S}}(i) + 1)}{c'_i} \geq \beta \qquad \text{for all } i \in \bar{S} \setminus C\,,$$

alternatively one can reformulate the above condition as

$$\deg_{\bar{S}}(i) \geq \sqrt{c'_i \cdot \beta + \frac{1}{4}} - \frac{1}{2} \qquad \text{for all } i \in \bar{S} \setminus C\,.$$

It is straightforward to check the following observation, it is fully analogous to the similar results on core.

Observation 3.4.10. The β-pseudo-core with costs $\{c'_i\}_{i \in S}$ of a set S is in fact unique, and it can be found with the greedy Algorithm 2.

Let us now explain how to construct the desired set S' from an optimal set S^*.

First note in Algorithm 3 we will be taking β-pseudo-cores with $\beta = \gamma\,(n^2 + (6/\varepsilon)n)$. Let us define $M := \log_2 \ln(\gamma(n + 6/\varepsilon))$ and for each k, $1 \leq k \leq M$ let us define

$$\Delta_k := \frac{\varepsilon^{\left(2 - \frac{1}{2^k}\right)} \cdot \left(\gamma\,(n^2 + (6/\varepsilon)n)\right)^{\left(1 - \frac{1}{2^k}\right)}}{M^{\left(2 - \frac{1}{2^{k-1}}\right)} \cdot n^{\left(2 - \frac{1}{2^k}\right)}}\,.$$

Algorithm 2: PseudoCore

Input: Initial set S, costs c'_i for $i \in S$, pseudo-core threshold value β

$\bar{S} \leftarrow S$

while $\bar{S} \setminus C \neq \varnothing$ **do**

  $i \leftarrow \arg\min_{v \in \bar{S} \setminus C} \dfrac{\deg_{\bar{S}}(i) \cdot (\deg_{\bar{S}}(i) + 1)}{c'_i}$

  if $\dfrac{\deg_{S'}(i) \cdot (\deg_{S'}(i) + 1)}{c'_i} < \beta$ **then**

    $\bar{S} \leftarrow \bar{S} \setminus \{v\}$

  else

    Break

Output: $\bar{S}$

Algorithm 3: Iterated PseudoCoring

Input: Initial set S^*, costs c_i for $i \in S^*$

For every $i \in S^*$, $c_{i,1} \leftarrow c_i$

$S_1 \leftarrow \texttt{PseudoCore}(S^*, \{c_{i,1}\}_{i \in S^*}, \gamma\,(n^2 + (6/\varepsilon)n))$

for $k = 2, \ldots M$ **do**

  for $i \in S_{k-1}$ **do**

    if $i \notin C$ **then**

      $c_{i,k} \leftarrow c_{i,k-1} + \Delta_{k-1}$

  $S_k \leftarrow \texttt{PseudoCore}(S_{k-1}, \{c_{i,k}\}_{i \in S_{k-1}}, \gamma\,(n^2 + (6/\varepsilon)n))$

Let S' be S_M after removing all agents i, $i \in S_M \cap C$ with $\deg_{S_M}(i) = 0$

Output: S'

Algorithm 3 iteratively increases costs and applies `PseudoCore`. In this manner, this process iteratively removes agents that have large cost and small degree. The iterative nature of Algorithm 3 allows to do an analysis of the properties of the resulting set S'.

For k, $1 \leq k \leq M$ let us define

$$\mathcal{L}_k(S) := 1 - \sum_{i \in S \setminus C} \frac{c_{i,k}}{\deg_S(i) + 1}\,.$$

The next lemma captures two facts. The first fact is that $\mathcal{L}_1(\cdot)$ is the same as $L(\cdot)$ up to ignoring the cost terms of agents in C. The second fact is that each application (moreover, each iteration) of Algorithm 2 within Algorithm 3 improves the value of $\mathcal{L}_k(\cdot)$.

Lemma 3.4.11. *We have $L(S^*) \leq \mathcal{L}_1(S^*) \leq \mathcal{L}_1(S_1)$; and for all k, $1 \leq k \leq M-1$ we have $\mathcal{L}_{k+1}(S_k) \leq \mathcal{L}_{k+1}(S_{k+1})$.*

Proof. Indeed, we have

$$\begin{aligned} L(S^*) &= 1 - \sum_{i \in S^*} \frac{c_i}{\deg_S(i)} \\ &= 1 - \sum_{i \in S^* \setminus C} \frac{c_i}{\deg_{S^*}(i)} - \sum_{i \in S \cap C} \frac{c_i}{\deg_{S^*}(i)} \\ &\leq 1 - \sum_{i \in S^* \setminus C} \frac{c_i}{\deg_{S^*}(i)} \\ &= \mathcal{L}_0(S^*)\,, \end{aligned}$$

Consider k, $1 \leq k \leq M$. Consider Algorithm 2, i.e. `PseudoCore` applied to a set S. Let $v \in \arg\min_{i \in S}\{\deg_S(i)(\deg_S(i)+1)/c_{i,k}\}$ be about to be removed from S during the current iteration of Algorithm 2. Let us show that $\mathcal{L}_k(S \setminus \{v\}) \geq \mathcal{L}_k(S)$.

$$\begin{aligned} \mathcal{L}_k(S \setminus \{v\}) - \mathcal{L}_k(S) &= \frac{c_{v,k}}{\deg_S(v)+1} - \sum_{i \in S \cap N(v) \setminus C} \frac{c_{i,k}}{\deg_S(i) \cdot (\deg_S(i)+1)} \\ &\geq \frac{c_{v,k}}{\deg_S(v)+1} - \sum_{i \in S \cap N(v) \setminus C} \frac{c_{v,k}}{\deg_S(v) \cdot (\deg_S(v)+1)} \\ &= \frac{c_{v,k}}{\deg_S(v)+1} \left(1 - \sum_{i \in S \cap N(v) \setminus C} \frac{1}{\deg_S(v)}\right) \\ &\geq 0\,, \end{aligned}$$

where the first inequality is due to the "greedy" choice of v in Algorithm 2, the second inequality is from the definition of $N(v)$ and $\deg_S(v)$. This concludes the proof, since the value $\mathcal{L}_k(\cdot)$ is improved with each iteration when Algorithm 2 is applied within Algorithm 3. □

Lemma 3.4.12. *For all k, $1 \leq k \leq M$ we have $\mathcal{L}_k(S_k) \leq \mathcal{L}_{k+1}(S_k) + 2\varepsilon/M$.*

Proof. We want to show $\mathcal{L}_k(S_k) - \mathcal{L}_{k+1}(S_k) \leq 2\varepsilon/M$. First of all notice, that S_k is the output of Algorithm 2 applied on S_{k-1} (in case of $k = 1$, applied on S^*) and costs $c_{i,k}$,

$i \in S_{k-1}$. Recall we set $c_{i,k+1} := c_{i,k} + \Delta_k$ for all $i \in S_k \setminus C$ and so we have $c_{i,k+1} \geq \Delta_k$. So, by Algorithm 2, we have

$$\deg_{S_{k+1}}(i)^2 + \deg_{S_{k+1}}(i) \geq \gamma \left(n^2 + \frac{6}{\varepsilon} n \right) \cdot c_{i,k+1} \,,$$

and

$$\deg_{S_{k+1}}(i) \geq \left(\gamma \left(n^2 + \frac{6}{\varepsilon} n \right) \cdot c_{i,k+1} \right)^{1/2} - 1 \geq \left(\gamma \left(n^2 + \frac{6}{\varepsilon} n \right) \cdot \Delta_k \right)^{1/2} - 1 \,.$$

Thus, we have

$$\begin{aligned}
\mathcal{L}_{k+1}(S_{k+1}) - \mathcal{L}_{k+2}(S_{k+1}) &= \left(1 - \sum_{i \in S_{k+1} \setminus C} \frac{c_{i,k+1}}{\deg_{S_{k+1}}(i) + 1} \right) \\
&\quad - \left(1 - \sum_{i \in S_{k+1} \setminus C} \frac{c_{i,k+2}}{\deg_{S_{k+1}}(i) + 1} \right) \\
&= \sum_{i \in S_{k+1} \setminus C} \frac{c_{i,k+2} - c_{i,k+1}}{\deg_{S_{k+1}}(i) + 1} \\
&\leq \sum_{i \in S_{k+1} \setminus C} \frac{\Delta_{k+1}}{\deg_{S_{k+1}}(i) + 1} \\
&\leq \sum_{i \in S_{k+1} \setminus C} \frac{\Delta_{k+1}}{\left(\gamma \left(n^2 + (6/\varepsilon) n \right) \cdot \Delta_k \right)^{1/2}} \\
&\leq \frac{n \cdot \Delta_{k+1}}{\left(\gamma \left(n^2 + (6/\varepsilon) n \right) \cdot \Delta_k \right)^{1/2}} \,.
\end{aligned}$$

Notice that by the definition of Δ_k we have

$$\frac{\Delta_{k+1}}{\Delta_k^{1/2}} = \frac{\varepsilon \cdot \gamma \left(n^2 + (6/\varepsilon) n \right)^{1/2}}{M \cdot n} \,,$$

leading us to the desired inequality for $k \geq 2$.

For $k = 1$, the inequality can be checked in the same manner. In the case $k = 1$, we use that for all $i \in S_1 \setminus C$ we have $c_i \geq \varepsilon/(2n)$ and so

$$\deg_{S_1}(i) \geq \left(\gamma \left(n^2 + \frac{6}{\varepsilon} n \right) \cdot \frac{\varepsilon}{2n} \right)^{1/2} - 1 \,.$$

This gives us an estimate as follows

$$\begin{aligned}
\mathcal{L}_1(S_1) - \mathcal{L}_2(S_1) &= \left(1 - \sum_{i \in S_1 \setminus C} \frac{c_{i,1}}{\deg_{S_1}(i)+1}\right) - \left(1 - \sum_{i \in S_1 \setminus C} \frac{c_{i,2}}{\deg_{S_1}(i)+1}\right) \\
&= \sum_{i \in S_1 \setminus C} \frac{c_{i,2} - c_{i,1}}{\deg_{S_1}(i)+1} \\
&\leq \sum_{i \in S_1 \setminus C} \frac{\Delta_1}{\deg_{S_1}(i)+1} \\
&\leq \sum_{i \in S_1 \setminus C} \frac{\Delta_1}{\left(\gamma\left(n^2 + (6/\varepsilon)n\right) \cdot \frac{\varepsilon}{2n}\right)^{1/2}} \\
&\leq 2\frac{n^{3/2}}{\left(\gamma\left(n^2 + (6/\varepsilon)n\right) \cdot \varepsilon\right)^{1/2}} \cdot \Delta_1 \\
&= 2\frac{n^{3/2}}{\left(\gamma\left(n^2 + (6/\varepsilon)n\right) \cdot \varepsilon\right)^{1/2}} \cdot \frac{\varepsilon}{M} \cdot \frac{\left(\gamma\left(n^2 + (6/\varepsilon)n\right) \cdot \varepsilon\right)^{1/2}}{n^{3/2}} \\
&\leq 2\frac{\varepsilon}{M}.
\end{aligned}$$

□

Finally we show that our actual $L(\cdot)$ value is not too far off from our relaxed value at the end of this process.

Lemma 3.4.13. *We have* $\mathcal{L}_M(S_M) \leq L(S^*) + 2\varepsilon$.

Proof. Note that Algorithm 3 does not decrease the costs of agents, so $c_{i,M} \geq c_i$ for each $i \in S_M$. Note also that S' is obtained from S_M by excluding all agents $i \in S_M \cap C$ such

that $\deg_{S_M}(i) = 0$. So we have

$$\begin{aligned}
\mathcal{L}_M(S_M) - L(S') &= \left(1 - \sum_{i \in S_M \setminus C} \frac{c_{i,M}}{\deg_{S_M}(i)+1}\right) \\
&\quad - \left(1 - \sum_{i \in S_M \setminus C} \frac{c_i}{\deg_{S_M}(i)} - \sum_{i \in S' \cap C} \frac{c_i}{\deg_{S_M}(i)}\right) \\
&\leq \sum_{i \in S_M \setminus C} \frac{c_i}{\deg_{S_M}(i)} - \sum_{i \in S_M \setminus C} \frac{c_i}{\deg_{S_M}(i)+1} + |S \cap C| \frac{\varepsilon}{2n} \\
&\leq \sum_{i \in S_M \setminus C} \frac{c_i + \deg_{S_M}(i) \cdot (c_i - c_{i,M})}{\deg_{S_M}(i) \cdot (\deg_{S_M}(i)+1)} + \frac{\varepsilon}{2} \\
&\leq_{\star} \sum_{i \in S_M \setminus C} \frac{c_i}{\deg_{S_M}(i) \cdot (\deg_{S_M}(i)+1)} + \frac{\varepsilon}{2} \\
&\leq_{\star\star} n \cdot \left(\frac{36e}{\varepsilon^5 \cdot (1-\varepsilon)^2} \cdot \frac{\log_2 \ln(n)}{n}\right)^2 + \frac{\varepsilon}{2} \\
&\leq 2\varepsilon\,,
\end{aligned}$$

where the inequality $\star$ follows from $c_i \leq c_{i,M}$ for all $i \in S_M \setminus C$, and the inequality $\star\star$ follows from Lemma 3.4.15 and Observation 3.4.9, and the final inequality is true for large enough n. □

Let us first state the corollary of Lemma 3.4.11, Lemma 3.4.12 and Lemma 3.4.13.

Corollary 3.4.14. *We have* $L(S^*) \leq L(S') + 4\varepsilon$.

Proof. This follows directly from the inequalities in Lemma 3.4.11, Lemma 3.4.12, and Lemma 3.4.13.

$$\begin{aligned}
L(S^*) \leq \mathcal{L}_1(S^*) &\leq \mathcal{L}_1(S_1) \leq \mathcal{L}_2(S_1) + 2\varepsilon/M \\
&\leq \mathcal{L}_2(S_2) \leq \mathcal{L}_3(S_2) + 4\varepsilon/M \\
&\leq \dots \\
&\leq \mathcal{L}_M(S_M) + 2\varepsilon \\
&\leq L(S') + 4\varepsilon\,.
\end{aligned}$$

□

Lemma 3.4.15. *For all $i \in S' \setminus C$, we have*

$$\deg_{S'}(i) \geq \frac{\varepsilon^5 \cdot (1-\varepsilon)^2}{36e} \cdot \frac{n}{\log_2 \ln(n)}\,, \qquad \text{and} \qquad \frac{\deg_{S'}(i)(\deg_{S'}(i)+1)}{c_i} \geq \gamma\left(n^2 + \frac{6}{\varepsilon}n\right).$$

Proof. The second property is immediate from the application of Algorithm 2 in Algorithm 3. Consider the first property. We can apply our degree bound and our bound on Δ_{M-1} to S_M to get

$$\begin{aligned}
\deg_{S_M}(i) &\geq \left(\gamma\left(n^2 + \frac{6}{\varepsilon}n\right) \cdot c_{i,M-1}\right)^{1/2} - 1 \\
&\geq \frac{\varepsilon^{\left(1-\frac{1}{2^M}\right)} \cdot \left(\gamma\left(n^2 + (6/\varepsilon)n\right)\right)^{\left(1-\frac{1}{2^M}\right)}}{M^{\left(1-\frac{1}{2^{M-1}}\right)} \cdot n^{\left(1-\frac{1}{2^M}\right)}} - 1 \\
&\geq \varepsilon \cdot \frac{\gamma\left(n^2 + (6/\varepsilon)n\right)}{n} \cdot \frac{1}{M}\left(\frac{\gamma\left(n^2 + (6/\varepsilon)n\right)}{n}\right)^{-\frac{1}{2^M}} - 1\,.
\end{aligned}$$

So by running Algorithm 3 for $M = \log_2 \ln(\gamma(n + 6/\varepsilon))$ iterations, for all $i \in S_M \setminus C$ we get

$$\deg_{S'}(i) \geq \frac{\varepsilon}{e} \cdot \frac{\gamma(n+6/\varepsilon)}{\log_2 \ln(\gamma(n+6/\varepsilon))} - 1 \geq \frac{\varepsilon^5 \cdot (1-\varepsilon)^2 \ln(2)}{36e} \cdot \frac{n}{\ln\ln(n)}\,,$$

which gives the desired bound by substituting and simplifying. □

Now let us finally observe that our set S_M retains approximately optimal $R(\cdot)$ value. We observe that if the pseudo-coring Algorithm 2 only removes elements $i \in S$ with $\deg_S(i) < \gamma' n$ then in the course of our iterated coring Algorithm 3 we remove at most $\gamma' n \cdot n = \frac{\varepsilon}{6} \cdot n^2$ value from $f(S^*)$, which reduces $R(\cdot)$ by less than ε. However, the pseudo-coring algorithm may remove elements of degree up to n, if the costs of those elements are high enough. We bound the number of such elements the algorithm may remove.

Lemma 3.4.16. *We have $R(S^*) \leq R(S') + \varepsilon$, $R(S') \geq \frac{\varepsilon}{3}$ and*

$$|\{i \in S' \mid \deg_{S'}(i) \geq \frac{\varepsilon}{6} \cdot n\}| \geq \frac{\varepsilon}{6}n\,.$$

Proof. Let us consider $\widetilde{S}$ as in Lemma 3.4.8 and let us show that $|\widetilde{S} \setminus S_M| \leq \varepsilon n/6$, implying $|\widetilde{S} \setminus S'| \leq \varepsilon n/6$. This would finish the proof, since $|\widetilde{S}| \geq \varepsilon n/3$ and for all $i \in \widetilde{S}$ we have $\deg_{\widetilde{S}}(i) \geq \varepsilon n/3$. Thus, we need to show that at most $\varepsilon n/6$ agents from $\widetilde{S}$ are removed during Algorithm 3.

Let us consider an iteration of Algorithm 3 at the moment when the total number of removed agents in $\widetilde{S}$ is at most $\varepsilon n/6$. Consider iteration k, $1 \leq k \leq M$ of Algorithm 3, i.e. the iteration when Algorithm 2 is called with costs $c_{i,k}$. Let us assume that during the call to Algorithm 2 a vertex $v \in \widetilde{S}$ is removed from $\bar{S}$ in Algorithm 2. Thus, we have

$$\frac{\deg_{\bar{S}}(v) \cdot (\deg_{\bar{S}}(v) + 1)}{c_{v,k}} < \gamma \left(n^2 + \frac{6}{\varepsilon} n \right) ,$$

but also $\deg_{\bar{S}}(v) \geq \gamma' n = \varepsilon n/6$, where $\bar{S}$ is the set in our algorithm at time of removal. We have

$$\begin{aligned} c_{v,k} &= c_v + \sum_{j=1}^{k} \Delta_j \\ &\leq c_v + \frac{\varepsilon \cdot \gamma \left(n^2 + (6/\varepsilon) n \right)}{n^2} \cdot \left(\frac{\gamma \left(n^2 + (6/\varepsilon) n \right)}{n} \right)^{-\frac{1}{2^k}} \\ &\leq c_v + \frac{\varepsilon \cdot \gamma \left(n^2 + (6/\varepsilon) n \right)}{e \cdot n^2} , \end{aligned}$$

where the first inequality is due to the definition of Δ_k, and the second is from the fact that $k \leq M = \log_2 \ln(\gamma(n + 6/\varepsilon))$. Now we also have

$$\gamma \left(n^2 + \frac{6}{\varepsilon} n \right) > \frac{\deg_{\bar{S}}(v) \cdot (\deg_{\bar{S}}(v) + 1)}{c_{v,k}} \geq \frac{\gamma' n \cdot (\gamma' n + 1)}{c_{i,k}} ,$$

since v is removed in Algorithm 2 and at the moment of removal $\deg_{\bar{S}}(v) \geq \gamma' n$. Rearranging and putting this together yields

$$\begin{aligned} c_v &\geq \frac{\gamma' n \cdot (\gamma' n + 1)}{\gamma \left(n^2 + (6/\varepsilon) n \right)} - \frac{\varepsilon \cdot \gamma \left(n^2 + (6/\varepsilon) n \right)}{e \cdot n^2} \\ &= \frac{n}{\varepsilon^2 \cdot n} - \frac{\varepsilon^2}{6en} \left(\frac{\varepsilon}{3} n - 1 \right) \\ &\geq \frac{(1 - \varepsilon)}{\varepsilon^2} . \end{aligned}$$

Hence, for every agent $v \in \widetilde{S}$ that was removed in Algorithm 2 before in total more than $\varepsilon n/6$ agents were removed, we have

$$c_v \geq (1 - \varepsilon)/(\varepsilon^2) .$$

By Observation 3.4.9, we have $\sum_{i \in \widetilde{S}} c_i < n$. This shows that Algorithm 2 removes at most $\varepsilon^2 \cdot n/(1-\varepsilon)$ vertices from $\widetilde{S}$. Thus, as desired $|\widetilde{S} \setminus S_M| \leq n/(1-\varepsilon) \leq \varepsilon n/6$.

To finish the proof notice that

$$R(S^*) - \varepsilon/3 \leq R(S') \leq R(S') + \frac{\varepsilon n}{6} \cdot n \cdot \frac{1}{\binom{n}{2}} = R(S') + \varepsilon/3\,.$$

□

Chapter 4

Packing Hypergraph Functions

4.1 Techniques and Comparison to Previous Work

Our results on hyperedge packing functions follow and adapt the techniques from [26] for XOS functions. [26] developed a series of crucial structural results about contracts for XOS functions relating values, marginals, and costs for optimal contracts. One of the crucial results in [26] is the "scaling" for XOS functions that allows to select an agents subset while getting the value of the subset within a fixed range and keeping the marginals for the subset sufficiently high. We show how to adapt the techniques outlined for hyperedge packing functions.

Let us consider a scenario where "active" agents are expected to break into groups. Let us assume that we have a limit on the size of such groups, so no group can have too many agents. If an agents' group is formed then it generates some expected reward for the principal. After that, the reward of the principal equals to the sum of the rewards generated by all groups. For example, the principal is responsible for hiring agents to form teams for some project. Additionally, there might be policies enforcing certain constraints on the teams that can be formed, for example the restrictions can be due to competences, diversity, geographical proximity, etc. To model such setting we introduce the class of functions MPMH-k for natural numbers k. The functions MPMUH-k correspond to the case when all teams need to have exactly k agents, and all "active" agents do not need to work on a specific project but jointly select a most profitable available project.

Let us now consider a generalization of the functions studied in Section 3. Say we are given a rank k hypergraph $G = (V, E)$ with non-negative weights on its edges $w(e)$,

$e \in E$. For $S \subseteq V$ let us define $w(S)$ as $\sum_{e \subseteq S} w(e)$. We call such $w : 2^V \to \mathbb{R}_+$ a *rank k Positive-Hypergraph* (PH-k) function. Note that the functions in Section 3 are PH-2 functions where $w(e) = 1/\binom{n}{2}$ for all $e \in E$. Thus the results from [22] lead us to the following observation.

Observation 4.1.1. There is no finite multiplicative approximation or additive FPTAS for the optimal contract problem on general PH-2 functions unless $\mathcal{P} = \mathcal{NP}$, even with value and demand oracles.

Proof. Consider the graph supermodular function considered in Section 3. Recall that finding any finite multiplicative approximation or additive FPTAS to the optimal contract problem for this function is impossible unless $\mathcal{P} = \mathcal{NP}$. We also observe that value and demand oracle queries can be computed efficiently. It is clear that value oracle queries are easy to compute in polynomial time. Solving demand queries for this function is a special case of submodular minimization, which may be done in polynomial time, [51]. This is because the function $|E(S)|$ is supermodular, and demand prices are modular. □

In the light of the above observation, to provide multiplicative approximation for PH-k functions we need to impose further constraints on PH-k functions. We call a PH-k function *rank k Positive-Matching-Hypergraph* (PMH-k) if the underlying hypergraph G is a hypermatching, i.e. each vertex is contained in at most one hyperedge.

Following the known generalization of PH-k functions to so called MPH-k functions, we define *rank k Maximum-over-Positive-Matching-Hypergraphs* MPMH-k functions equal to maximum of all PMH-k functions corresponding to some hypergraph with weights for its hyperedges. Given a hypergraph $G = (V, E)$ with non-negative weights $w(e)$, $e \in E$ the corresponding MPMH-k function $f : 2^V \to \mathbb{R}_+$ is defined as

$$f(S) := \max\{w(M) \,:\, M \text{ hypermatching in } G[S]\}\,,$$

where $G[S]$ is a sub-hypergraph of G induced by vertices S. It is straightforward to notice that such a function f is not XOS. Consider the graph $G = (\{u, v\}, \{uv\})$ and the edge weight $w(uv) = 1$. Then $f(\{u\}) + f(\{v\}) = 0$ while $f(\{u, v\}) = 1$, violating the inequality $f(\{u\}) + f(\{v\}) \geq f(\{u, v\})$ that needs to hold for XOS functions.

Working with an MPMH-k function $f : 2^V \to \mathbb{R}_+$ we require access to a so called MPMH-k *oracle*: given a set T the oracle returns the value $f(T)$ and a hypermatching in the hypergraph $G = (V, E)$ such that $w(M) = f(T)$.

Theorem 4.1.2. *Let us be given a fixed number k and a fixed parameter $\xi > 1$. For each* MPMH-k *function $f : 2^V \to \mathbb{R}_+$ we can compute a $1/(256k^4\xi + 2)$ approximation to the*

optimal contract with respect to f in polynomial time and using a polynomial number of calls for MPMH-k *oracle, value oracle and demand oracle.*

Analogous results can be obtained when the underlying hypergraphs are uniform. We call a PMH-k function *rank k Positive-Matching-Uniform-Hypergraph* (PMUH-k) if the underlying hypergraph G is a k-uniform hypermatching. A *rank k Maximum-over-Positive-Matching-Uniform-Hypergraphs* MPMUH-k function equals to maximum of several PMUH-k functions, i.e. maximum of *supporting* PMUH-k functions. Note, that the PMUH-k functions used in the above definition do not need to correspond to the same hypergraph. The proof of the below theorem is identical to the proof of Theorem 4.1.2, and we omit it for the sake of space.

Theorem 4.1.3. *Let us be given a fixed number k and a fixed parameter $\xi > 1$. For each* MPMUH-k *function $f : 2^V \to \mathbb{R}_+$ we can compute a $1/(256k^4\xi + 2)$ approximation to the optimal contract with respect to f in polynomial time and using a polynomial number of calls for* MPMUH-k *oracle, value oracle and demand oracle.*

Here, MPMUH-k oracle called for a MPMUH-k function f and a set S returns a *supporting* PMUH-k function ω such that $\omega(S) = f(S)$.

4.2 Scaling Properties of MPMH-k Functions

Lemma 4.2.1. *Let $f : 2^{[n]} \to \mathbb{R}_{\geq 0}$ be an* MPMH-k *function and let sets T and S satisfy $S \subseteq T \subseteq [n]$. Let M be the hypermatching in G such that $w(M) = f(T)$ and M matches only vertices in T. Let us assume that for every hyperedge $e \in M$ we have $e \subseteq S$ or $e \cap S = \varnothing$. Then we have*

$$\sum_{i \in S} f(i \mid T \setminus \{i\}) \leq k \cdot f(S).$$

Proof. By the definition of the functions f and w, for every $i \in S$ we have

$$f(i \mid T \setminus \{i\}) \leq \sum_{e \in M \colon e \subseteq T,\, i \in e} w(e) = \sum_{e \in M \colon e \subseteq S,\, i \in e} w(e)\,,$$

where the equality follows from the fact that for every hyperedge $e \in M$ we have $e \subseteq S$ or $e \cap S = \varnothing$. Then we have

$$\sum_{i \in S} f(i \mid T \setminus \{i\}) \leq \sum_{i \in S} \sum_{e \in M \colon e \subseteq S,\, i \in e} w(e) \leq \sum_{e \in M \colon e \subseteq S} |e| \cdot w(e) \leq k \cdot w(S) \leq k \cdot f(S)\,.$$

□

Let us now show how we can "scale" MPMH-k functions while preserving sufficiently high marginals. Given a set $T \subseteq [n]$ and an MPMH-k function $f : 2^{[n]} \to \mathbb{R}$, let us show how to obtain another set U, $U \subseteq T$ such that $f(U)$ is within a certain range while the marginals of f with respect to U are comparable with the marginals with respect to T. For this we start with a certain hypergraph $G = (V, E)$ and iteratively remove subsets of agents from T, where in each step the removed subset of agents forms a hyperedge in G.

Algorithm 4: Scaling for MPMH-k Functions

Input: An MPMH-k function $f : 2^{[n]} \to \mathbb{R}_{\geq} 0$, a set T, parameters $0 \leq \Psi < f(T)$ and $\delta \in (0, 1]$

$t \leftarrow 1$
while $T_t \neq \varnothing$ **do**
  $j \leftarrow \arg\min_{i \in T_{t-1}} \frac{f(i|T_{t-1} \setminus \{i\})}{f(i|T_0 \setminus \{i\})}$, $s \leftarrow t$
  Let $e \in M$ be the unique hyperedge containing the agent j.
  forall $i \in e$, starting with $i = j$ **do**
    $i_t \leftarrow i$, $s_t \leftarrow s$, $T_t \leftarrow T_{t-1} \setminus \{i_t\}$, $\delta_t \leftarrow \frac{f(i_t|T_t)}{f(i_t|T_0 \setminus \{i_t\})}$, $t \leftarrow t + 1$
$j^* \leftarrow \min\{j \mid f(T_j) \leq \Psi\}$
$s' \leftarrow s_{j^*} - 1$
$k^* \leftarrow \max\{j^*, \min\{k \mid f(T_k) \leq (1 - \delta) f(T_{s'})\}\}$
$t^* \leftarrow \arg\max_{t \in \{j^*, \ldots, k^*\}} \delta_t$

Output: $U = T_{t^*-1}$

Lemma 4.2.2. *Let f be a* MPMH-k *function. Let us be given a set $T \subseteq [n]$ and parameters $\delta \in (0, 1]$ and $0 \leq \Psi < f(T)$. Let T_0 be an inclusion-wise minimal subset of T with $f(T_0) = f(T)$. Let M be the hypermatching in G such that $f(T_0) = w(M)$ and M matches only vertices in T_0.*

Then Algorithm 4 finds a set $U \subseteq T$ such that

$$(1 - \delta)\Psi \leq f(U) \leq \Psi + \max_{e \in M} w(e), \qquad \text{and} \qquad f(i \mid U \setminus \{i\}) \geq \frac{\delta}{k} \cdot f(i \mid T \setminus \{i\}) \quad \forall i \in U\,.$$

Moreover, Algorithm 4 runs in polynomial time and does one call to an MPMH-k *oracle.*

Proof. Since T_0 is an inclusion-wise minimal set satisfying $f(T_0) = f(T)$, we have $f(i \mid T_0 \setminus \{i\}) > 0$ for all $i \in T_0$. Also every agent i in T_0 is contained in a unique hyperedge $e \in E$. Due to the iterative removal of agents, we have $T_0 \supseteq T_1 \supseteq \cdots \supseteq T_{|T_0|} = \varnothing$, and so by monotonicity we have $f(T_0) \geq f(T_1) \geq \cdots \geq f(T_{|T_0|}) = 0$. Thus j^* is well defined.

We also have $s_{j^*} \geq 1$ due to $\Psi < f(T) = f(T_0)$. Thus k^* is also well defined. We observe that $j^* \leq k^*$, which ensures that t^* is well defined. Note also that we have $t^* \geq j^* \geq 1$; so $U := T_{t^*-1}$ is well defined.

We now consider why the returned set U satisfies the properties stated in the lemma. Consider the first inequality in the lemma. By the monotonicity of f and by the definition of t^*, k^*, j^* we have

$$f(T_{t^*-1}) \geq f(T_{k^*-1}) \geq (1-\delta) f(T'_s) \geq (1-\delta)\Psi\,,$$

which leads to the first inequality in the lemma. By the definition of j^* we have $f(T_{j^*}) \leq \Psi$. Then by definition of s' we have $f(T_{s'}) \leq \Psi + w(e^*)$, where e^* is the unique hyperedge in M such that j^* is in e^*. The second inequality in the lemma follows from monotonicity.

Consider an agent $i \in U$. By definition $f(T_{s'}) = f(T_{k^*}) + \sum_{t \in \{s'+1,\ldots,k^*\}} f(i_t \mid T_t)$. Now by using the definition of δ_t, $t = 1, \ldots, |T_0|$ we obtain

$$\begin{aligned}
f(T_{s'}) - f(T_{k^*}) &= \sum_{t \in \{s'+1,\ldots,k^*\}} \delta_t \cdot f(i_t \mid T_0 \setminus \{i_t\}) \\
&\leq_{\star} \sum_{t \in \{s'+1,\ldots,k^*\}} \delta_{t^*} \cdot f(i_t \mid T_0 \setminus \{i_t\}) \\
&\leq_{\star\star} \sum_{t \in \{s'+1,\ldots,|T_0|\}} \delta_{t^*} \cdot f(i_t \mid T_0 \setminus \{i_t\}) \\
&\leq_{\star\star\star} \delta_{t^*} \cdot k \cdot f(T_{s'})\,.
\end{aligned}$$

The inequality $\star$ follows from the definition of δ^*, and the inequality $\star\star$ from the fact f is monotonically increasing. Now consider the inequality $\star\star\star$. Note for every hyperedge $e \in M$, we have $e \subseteq T_{s'}$ or $e \cap T_{s'} = \varnothing$. This is because of the definition of s', and the loop on line 5 of Algorithm 4. Thus the inequality $\star\star\star$ follows from Lemma 4.2.1.

With this inequality in hand, we observe for all $i \in U = T_{t^*-1}$

$$\begin{aligned}\frac{f(i \mid T_{t^*-1} \setminus \{i\})}{f(i \mid T_0 \setminus \{i\})} &\geq \delta_{t^*} \\ &\geq \frac{1}{k} \cdot \frac{f(T_{s'}) - f(T_{k^*})}{f(T_{s'})} \\ &= \frac{1}{k}\left(1 - \frac{f(T_{k^*})}{f(T_{s'})}\right) \\ &\geq \frac{1}{k}\left(1 - \frac{(1-\delta)f(T_{s'})}{f(T_{s'})}\right) \\ &= \frac{\delta}{k}\,.\end{aligned}$$

Finally observe that $f(i \mid T_0 \setminus \{i\}) \geq f(i \mid T \setminus \{i\})$ follows from $f(T_0) = f(T)$ and the fact that f is monotonically increasing. Combining these inequalities yields the last inequality in the statement of the lemma.

It is straightforward to see Algorithm 4 runs in polynomial time with MPMH-k oracle access to the function f. Also note that T_0 can be obtained by iteratively removing agents from T whose removal does not decrease the function value. □

4.3 Structural Lemmas

Let us define sets of agents that could form hyperedges in these hypergraphs, while the formed hyperedges have acceptable marginal values in terms of the function g (or more specifically in terms of the function L, see Section 3).

$$A := \left\{ e \in E \,:\, \sum_{i \in e} \left(\frac{c_i}{f(i \mid e \setminus \{i\})} \right) \leq \frac{1}{2} \right\}.$$

Next, the following structural lemmas help us to design and analyze the function g, or more specifically the functions L and R, see Section 3. We defer their proofs to the later parts of the section.

Lemma 4.3.1. *Let f be an* MPMH-k *function and let $g(S^*)$ be positive. Let M be the hypermatching such that $w(M) = f(S^*)$ and M matches only vertices in S^*. We have $|M \setminus A| \leq 1$; and if there exists a hyperedge e' in $M \setminus A$ then we also have*

$$g(S^*) \leq f(S^* \setminus e') + \max\{0, g(e')\}\,.$$

Lemma 4.3.2. *Let f be an* MPMH-k *function, and S be a set such that $S \subseteq S^*$. Let M be a hypermatching such that $w(M) = f(S^*)$ and M matches only vertices in S^*. Let us assume that for every hyperedge $e \in M$ we have $e \subseteq S$ or $e \cap S = \varnothing$. Then we have*

$$\sum_{i \in S} \sqrt{c_i} \leq \sqrt{k \cdot f(S)}\,.$$

Lemma 4.3.3. *Let f be an* MPMH-k *function, and α be a fixed positive value. Let S be a set such that $f(S) > 0$ and $f(i \mid S \setminus \{i\}) \geq \sqrt{\alpha \cdot c_i \cdot f(S)}$ for every $i \in S$. Then we have*

$$g(S) \geq \left(1 - \frac{k}{\alpha}\right) f(S)\,.$$

4.4 Obtaining a Good Set

With the structural lemmas in hand, we consider how to obtain an approximately optimal contract. Note that through exhaustive search we can assume that e' from Lemma 4.3.1 is known to us. We fix such a hyperedge e' throughout the rest of our proofs.

The next lemma assumes that apart from e' we are also given an estimate for the value $f(S^* \setminus e')$. Using the estimate for the value $f(S^* \setminus e')$ we show how to obtain an approximately optimal contract by using an MPMH-k oracle, demand oracle, value oracle and Algorithm 4. Afterwards, we will be left only with describing how to obtain an estimate for the value $f(S^* \setminus e')$.

Algorithm 5: Approximately optimal contract given an estimate of $f(S^* \setminus e')$

Input: A MPMH-k function $f : 2^{[n]} \to \mathbb{R}_{\geq} 0$, an estimate $\tilde{y} \geq f(S^* \setminus e')$, costs c_i for $i \in [n]$

$U \leftarrow \varnothing$

Let $\Psi = \frac{\tilde{y}}{32k^4} - \max_{e \in A \setminus \{e'\}} w(e)$.

For every i in $V \setminus e'$, $\quad p_i \leftarrow \frac{1}{2\sqrt{k}} \cdot \sqrt{c_i \cdot \tilde{y}}$.

Let T be a demand set with prices p_i over the set $V \setminus e'$.

if $0 < \Psi < f(T)$ **then**

  $U' \leftarrow$ the output of Algorithm 4 on input $(f, T, \Psi, \delta = \frac{1}{2})$

  $U \leftarrow \arg\max_{U'' \in \{U, U'\}} g(U'')$

Output: U

Lemma 4.4.1. *Algorithm 5 runs in polynomial time with access to an* MPMH-k *oracle, demand and value oracle. If $\tilde{y} \leq f(S^* \setminus e')$ then the returned set U satisfies*

$$g(U) \geq \max\left\{\frac{1}{4} \cdot \left(\frac{1}{32k^4}\tilde{y} - \max_{e \in A \setminus \{e'\}} w(e)\right), 0\right\}.$$

Proof. It is clear the algorithm runs in polynomial time with access to an MPMH-k oracle, demand and value oracle, since k is constant and Algorithm 4 runs in polynomial time with access to an MPMH-k oracle, demand and value oracle.

Let M be the hypermatching such that $w(M) = f(S^*)$ and M matches only vertices in S^*.

Recall we assume that $\tilde{y} \leq f(S^* \setminus e')$. If $\Psi \leq 0$ then the algorithm returns the empty set, and the statement of the lemma is satisfied. Otherwise $\Psi > 0$ and we have

$$\begin{aligned}
f(T) &\geq_{\star} f(T) - \sum_{i \in T} p_i \\
&\geq_{\star\star} f(S^* \setminus e') - \sum_{i \in (S^* \setminus e')} p_i \\
&= f(S^* \setminus e') - \frac{\sqrt{\tilde{y}}}{2\sqrt{k}} \sum_{i \in S^* \setminus e'} \sqrt{c_i} \\
&\geq_{\star\star\star} f(S^* \setminus e') - \frac{1}{2\sqrt{k}} \sqrt{k \cdot f(S^* \setminus e') \cdot \tilde{y}} \\
&\geq_{\star\star\star\star} \frac{1}{2} f(S^* \setminus e') \\
&> \Psi\,.
\end{aligned}$$

The inequality $\star$ holds since the prices are non-negative, and the inequality $\star\star$ uses that T is a demand set over $V \setminus e'$. The inequality $\star\star\star$ is due to Lemma 4.3.2. The inequality $\star\star\star\star$ holds because of the assumption $\tilde{y} \leq f(S^* \setminus e')$. Finally, the last inequality holds by definition of Ψ.

Furthermore, Lemma 4.2.2 ensures that the output U' of Algorithm 4 in this iteration, satisfies

$$f(U') \geq (1-\delta)\Psi = \frac{1}{2} \cdot \left(\frac{1}{32k^4}\tilde{y} - \max_{e \in A \setminus \{e'\}} w(e)\right), \tag{4.1}$$

$$f(U') \leq \Psi + \max_{e \in A \setminus \{e'\}} w(e) = \frac{\tilde{y}}{32k^4}, \tag{4.2}$$

and also

$$f(i \mid U' \setminus \{i\}) \geq \frac{\delta}{k} \cdot f(i \mid T \setminus \{i\}) \geq \frac{\delta}{k} \cdot p_i = \frac{\delta}{2k^{3/2}} \cdot \sqrt{c_i \cdot \tilde{y}} \geq \sqrt{2k \cdot c_i \cdot f(U')}\,.$$

The second inequality is from $f(i \mid T \setminus \{i\}) \geq p_i$ for all $i \in T$, since T is a demand set. The final inequality follows from (4.1). Then by Lemma 4.3.3 we have $g(U) \geq (1 - 1/2)\, f(U')$, which gives us

$$g(U) \geq \left(1 - \frac{1}{2}\right) f(U) \geq \frac{1}{4} \cdot \left(\frac{1}{32k^4}\tilde{y} - \max_{e \in A \setminus \{e'\}} w(e)\right)\,,$$

where the last inequality is due to (4.2). □

With Algorithm 5 and Lemma 4.4.1 in hand, all that remains is to show that we can obtain such an estimate $\tilde{y}$. We do this now, and complete the proof of Theorem 4.1.2.

Algorithm 6: Approximately optimal contract

Input: A MPMH-k function $f : 2^{[n]} \to \mathbb{R}_{\geq} 0$, costs c_i for $i \in [n]$, parameter $\xi > 1$

Let $C = A$
if $A \neq \varnothing$ and $\max_{e \in A} w(e) > 0$ **then**
  Let $x = \max_{e \in A} f(e)/2$
  for $j = 0, \ldots, \lceil \log_\xi 2n \rceil$ **do**
    Let $x_j = x \cdot \xi^j$
    $U_j \leftarrow$ the output of Algorithm 5 on $(f, \{c_i\}_{i \in A}, \tilde{y} = x_j)$
    $C \leftarrow C \cup \{U_j\}$

Output: $\arg\max_{S \in C} g(S)$

Proof of Theorem 4.1.2. We first note that Algorithm 6 runs in polynomial time for any $\xi > 1$ and constant k with access to a MPMH-k and demand oracle, since Algorithm 5 does. Now we show that Algorithm 6 gives the desired approximation. First suppose $f(S^* \setminus e') \leq 128k^4\xi \cdot \max_{e \in A} g(e)$. Then we see that by choosing one of our initial sets of agents in C we obtain the desired approximation ratio. Indeed, by Lemma 4.3.1 we have

$$g(S^*) \leq f(S^* \setminus e') + \max\{0, g(e')\} \leq (128k^4\xi + 1) \cdot \max_{e \in A} g(e)\,.$$

Now suppose otherwise, so $f(S^* \setminus e') > 128k^4\xi \cdot \max_{e \in A} g(e)$. We have $\max_{e \in A} g(e) \geq x$ since $g(e) \geq (1/2) f(e)$ for $e \in A$. This also implies that we have $f(S^* \setminus e') \geq x$ in this case.

On the other hand, we have $f(S^* \setminus e') \leq 2n \cdot x$ by the definition of x. This is because we can have at most n edges each with at most value $2x$ in any hypermatching M such that $w(M) = f(S^* \setminus e')$. This implies that there is a unique $j^* \in \{0, 1, \ldots, \lceil \log_\xi 2n \rceil\}$ for which $x_j \leq f(S^* \setminus e') < \xi x_j$ holds. Note this also means $(1/\xi) f(S^* \setminus e') \leq x_{j^*} \leq f(S^* \setminus e')$. Now we have

$$\begin{aligned} g(U_{j^*}) &\geq \frac{1}{4} \cdot \left(\frac{\tilde{y}}{32k^4} - \max_{e \in A} w(e) \right) \\ &\geq \left(\frac{f(S^* \setminus e')}{128k^4\xi} - \frac{\max_{e \in A} g(e)}{2} \right) \\ &\geq f(S^* \setminus e') \cdot \left(\frac{1}{256k^4\xi} \right) \\ &\geq g(S^*) \cdot \left(\frac{128k^4}{128k^4 + 1} \right) \cdot \left(\frac{1}{256k^4\xi} \right) \\ &= \frac{g(S^*)}{256k^4\xi + 2} \,. \end{aligned}$$

The first inequality is due to Lemma 4.4.1, the second inequality uses the definition of A, and the third and fourth inequalities both are due to $f(S^* \setminus e') > 128k^4\xi \cdot \max_{e \in A} g(e)$. The fourth inequality also uses Lemma 4.3.1. □

4.5 Proof of Structural Lemmas

Proof of Lemma 4.3.1. Let us first show $|M \setminus A| \leq 1$.

$$\begin{aligned} 0 < g(S^*) \\ &= f(S^*) \left(1 - \sum_{i \in S^*} \frac{c_i}{f(i \mid S^* \setminus \{i\})} \right) \\ &= f(S^*) \left(1 - \sum_{e \in M} \sum_{i \in e} \frac{c_i}{f(i \mid S^* \setminus \{i\})} \right) \\ &\leq f(S^*) \left(1 - \sum_{e \in M \setminus A} \sum_{i \in e} \frac{c_i}{f(i \mid S^* \setminus \{i\})} \right) \\ &\leq_\star f(S^*) \left(1 - \frac{|M \setminus A|}{2} \right) . \end{aligned}$$

The inequality $\star$ is due to the definition of A and due to $f(i \mid S^* \setminus \{i\}) \leq f(i \mid e \setminus \{i\})$ for every $e \in M$ and $i \in e$. If $|S^* \setminus A| = 0$ then we have the statement of the lemma. So suppose otherwise, and let e' be the unique edge in $M \setminus A$. Now we obtain

$$g(S^*) \leq_\star f(S^* \setminus e') + f(e') \left(1 - \sum_{i \in e'} \frac{c_i}{f(i \mid S^* \setminus \{i\})}\right) \leq_{\star\star} f(S^* \setminus e') + g(e') \,.$$

The inequality $\star$ is due to the definition of the the function g and since $f(S^*) = f(S^* \setminus e') + f(e')$ by the choice of the hypermatching M. The inequality $\star\star$ is due to $f(i \mid S^* \setminus \{i\}) \leq f(i \mid e' \setminus \{i\})$. □

Proof of Lemma 4.3.2. First note $f(i \mid S^* \setminus \{i\}) > 0$ for all $i \in S*$ with $c_i > 0$, since otherwise $g(S^*) = -\infty$. Since $g(\varnothing) = 0$, the above is a contradiction on the optimality of the set S^*. Also note that if $f(S^*) = 0$ then the claim is trivially satisfied. So suppose $f(S^*) > 0$. Then we observe that

$$g(S^*) = f(S^*) \left(1 - \sum_{i \in S^*} \frac{c_i}{f(i \mid S^* \setminus i)}\right) \geq 0 \qquad \Rightarrow \qquad \sum_{i \in S^*} \frac{c_i}{f(i \mid S^* \setminus i)} \leq 1 \,.$$

Combined with Cauchy-Schwarz and Lemma 4.2.1 this gives us

$$\left(\sum_{i \in S} c_i^{1/2}\right)^2 \leq \left(\sum_{i \in S} \frac{c_i}{f(i \mid S^* \setminus i)}\right) \left(\sum_{i \in S} f(i \mid S^* \setminus i)\right) \leq k \cdot f(S) \,.$$

Taking square roots completes the proof. □

Proof of Lemma 4.3.3. Consider an agent $i \in S$. If $c_i > 0$ then we have $f(i \mid S \setminus \{i\}) > 0$; and so the condition $f(i \mid S \setminus \{i\}) \geq \sqrt{\alpha \cdot c_i \cdot f(S)}$ is equivalent to

$$\frac{c_i}{f(i \mid S \setminus \{i\})} \leq \frac{1}{\alpha} \frac{f(i \mid S \setminus \{i\})}{f(S)} \,.$$

For every $i \in S$ with $c_i = 0$, we have $\frac{c_i}{f(i|S\setminus\{i\})} = 0$. Thus if $c_i = 0$ then the inequality above holds in that case too. Summing over $i \in S$ gives us

$$\sum_{i \in S} \frac{c_i}{f(i \mid S \setminus \{i\})} \leq \frac{1}{\alpha} \frac{\sum_{i \in S} f(i \mid S \setminus \{i\})}{f(S)} \leq \frac{k}{\alpha} \,,$$

where the last inequality is due to Lemma 4.2.1. This gives us the desired inequality

$$g(S) = \left(1 - \sum_{i \in S} \frac{c_i}{f(i \mid S \setminus \{i\})}\right) f(S) \geq \left(1 - \frac{k}{\alpha}\right) f(S)\,.$$

□

Chapter 5

Conclusion

We have seen in Chapter 3 and in Chapter 4 that the types of approximation one is able to attain for optimal linear contracts can be quite different depending on the properties of the reward function. Indeed, in Chapter 4 we attain constant factor multiplicative approximations. However, in Chapter 3 no multiplicative approximation is possible and so we focus on achieving additive approximations.

5.1 Questions Regarding Chapter 3

In Chapter 3 our work was related to a classic PTAS for the densest k-subgraph problem, as found in [6]. [6] considers a wide variety of dense instances of smooth polynomial programming problems, and gives PTASs for them.

It is worth considering if we can achieve our additive approximations in less time. Our additive PTAS in Chapter 3 runs in time $O(n^{\text{poly}(1/\varepsilon)})$. Recall this is similar to the PTAS of [6] for densest k subgraph which runs in time $O(n^{\text{poly}(1/\varepsilon)})$. Some of the other PTASs for dense instances of Max-CSPs could thus be helpful for improving the runtime of our algorithm. Recall these are accomplished through various means, including sampling/subsampling [6, 1, 20, 9, 55, 46], regularity lemmas [35], and convex hierarchies [21, 10, 40, 56]. The fastest known time for a PTAS is $O(n/\varepsilon^2) + 2^{O(1/\varepsilon^2)}$ in [55]. Also recall we use the sampled degrees to linearize our objective function, and thus we need some amount of sampling for our approach. Nonetheless, some other approaches to approximating Max-CSPs could yield other apporaches for approximating the contract problem in Chapter 3. [22] wondered if one can use regularity to obtain an additive PTAS for the case

of general costs. It would be interesting to see if this is possible, and if this approach would give a simplified additive PTAS. Additionally, determining exactly how quickly a PTAS can run while achieving a given approximation could be an interesting question. [47] gives such a characteration for the run-time of approximations for Max-CSPs.

It is natural to ask whether our techniques in Chapter 3 extend to other functions considered by [6]. Indeed, it seems likely that our approach could be directly applied to supermodular smooth polynomial programming problems of degree 2. However, we run into a hurdle when considering problems only modelled by polynomials of degree ≥ 3.

For example, consider finding an approximately optimal linear contract for the function $f(S) = |E(S)|/\binom{n}{3}$ in a rank 3 hypergraph (V, E) (as opposed to finding an approximately optimal linear contract for the function $f(S) = |E(S)|/\binom{n}{2}$ in a graph (V, E) as in Chapter 3). We can again apply our pseudo-coring process (see Section 3.4.2). This yields a near optimal solution where every agent has degree $\Omega(n^2/\ln\ln n)$. Then "high degree" agents (similar to those in A, see Section 3.3) will have degree $\Omega(n^2)$, and "low degree" agents (similar to those in B, see Section 3.3) will have degree $\Omega(n^2/\ln\ln n)$. We can again sample for the degrees of "high degree" agents, and obtain accurate estimates of their degrees. Next we naively need constraints of the form

$$\sum_{uvw \in E} x_u x_v \geq \Omega(n^2) \qquad \text{for all } w \in A \tag{5.1}$$

$$\sum_{uvw \in E} x_u x_v \geq x_w \cdot \Omega(n^2/\ln\ln n) \qquad \text{for all } w \in B \tag{5.2}$$

to mimic the constraints (3.8) and (3.9) in Section 3.3. However, we note that these are no longer linear constrains. The natural linearization, as in [6], introduces additive error εn^2 to these constraints. This does not introduce problems for constraints (5.1), but renders constraints (5.2) useless.

Thus if we want our approach to tackle more general functions, such as $f(S) = |E(S)|/\binom{n}{3}$ in a rank 3 hypergraph then we must give an alternative linearization of these constraints which adds less error to these constraints. However, this may not be necessary to solve the problem if we use a modified approach. We may consider modifying our approach to solve a NLP using only the "high degree" constraints (e.g. (5.1)) and then preprocess and round the resulting solution more carefully. Indeed, if it can be shown that one can ignore the cost of agents with "low degree" then the actual degrees of the "low degree" agents are irrelevant for an additive approximation. Of course, showing that such agents have sufficient degree is critical to showing their costs can be ignored, and so more careful rounding is necessary if one plans to not place sufficient degree constraints on

these agents in the NLP itself. It might be possible to get around this issue by using one of the other approaches for obtaining a PTAS for dense Max-CSPs. In particular, using regularity lemmas could be useful for avoiding these issues.

5.2 Questions Regarding Chapter 4

Our results in Chapter 4 naturally extend previous results on XOS functions to more general classes of functions. These classes are subsets of MPH-k. We also observed general MPH-2 functions are much harder to approximate, as they yield no multiplicative approximation. Our approximations come from the observation that we can still "scale" the functions we consider, in a similar way to the "scaling" for XOS functions in [26]. It would be interesting to see if these "scaling" properties are in some way necessary to achieve good approximations. It is also worth considering if one can achieve better approximations, both for the functions considered in Chapter 4 and even for XOS functions. The current gap between approximations and hardness of approximation could be improved.

References

[1] Noga Alon, W Fernandez De La Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csps. *Journal of computer and system sciences*, 67(2):212–243, 2003.

[2] Tal Alon, Paul Duetting, Yingkai Li, and Inbal Talgam-Cohen. Bayesian analysis of linear contracts. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 66–66, 2023.

[3] Tal Alon, Paul Dütting, and Inbal Talgam-Cohen. Contracts with private cost per unit-of-effort. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 52–69, 2021.

[4] Tal Alon, Inbal Talgam-Cohen, Ron Lavi, and Elisheva Shamash. Incomplete information vcg contracts for common agency. *Operations Research*, 72(1):288–299, 2024.

[5] Nivasini Ananthakrishnan, Stephen Bates, Michael Jordan, and Nika Haghtalab. Delegating data collection in decentralized machine learning. In *International Conference on Artificial Intelligence and Statistics*, pages 478–486. PMLR, 2024.

[6] Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 284–293, 1995.

[7] Moshe Babaioff, Michal Feldman, and Noam Nisan. Combinatorial agency. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 18–28, 2006.

[8] Yakov Babichenko, Inbal Talgam-Cohen, Haifeng Xu, and Konstantin Zabarnyi. Information design in the principal-agent problem. *arXiv preprint arXiv:2209.13688*, 2022.

[9] Boaz Barak, Moritz Hardt, Thomas Holenstein, and David Steurer. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 512–531. SIAM, 2011.

[10] Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *2011 ieee 52nd annual symposium on foundations of computer science*, pages 472–481. IEEE, 2011.

[11] Siddharth Barman. Approximating nash equilibria and dense bipartite subgraphs via an approximate version of caratheodory's theorem. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 361–369, 2015.

[12] Martino Bernasconi, Matteo Castiglioni, and Andrea Celli. Agent-designed contracts: How to sell hidden actions. *arXiv preprint arXiv:2402.16547*, 2024.

[13] Federico Cacciamani, Martino Bernasconi, Matteo Castiglioni, and Nicola Gatti. Multi-agent contract design beyond binary actions, 2024.

[14] Matteo Castiglioni, Junjie Chen, Minming Li, Haifeng Xu, and Song Zuo. A reduction from multi-parameter to single-parameter bayesian contract design, 2024.

[15] Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Bayesian agency: Linear versus tractable contracts. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 285–286, 2021.

[16] Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Designing menus of contracts efficiently: the power of randomization. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 705–735, 2022.

[17] Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Multi-agent contract design: How to commission multiple agents with individual outcomes. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 412–448, 2023.

[18] Junjie Chen, Minming Li, and Haifeng Xu. Selling data to a machine learner: Pricing via costly signaling. In *International Conference on Machine Learning*, pages 3336–3359. PMLR, 2022.

[19] Constantinos Daskalakis and Christos H Papadimitriou. On oblivious ptas's for nash equilibrium. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 75–84, 2009.

[20] W Fernandez de la Vega, Marek Karpinski, Ravi Kannan, and Santosh Vempala. Tensor decomposition and approximation schemes for constraint satisfaction problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 747–754, 2005.

[21] Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of maxcut. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 53–61. Citeseer, 2007.

[22] Ramiro Deo-Campo Vuong, Shaddin Dughmi, Neel Patel, and Aditya Prasad. On supermodular contracts and dense subgraphs. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 109–132. SIAM, 2024.

[23] Paul Duetting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Multi-agent combinatorial contracts. *arXiv preprint arXiv:2405.08260*, 2024.

[24] Paul Duetting, Michal Feldman, and Daniel Peretz. Ambiguous contracts. In Kevin Leyton-Brown, Jason D. Hartline, and Larry Samuelson, editors, *Proceedings of the 24th ACM Conference on Economics and Computation, EC 2023, London, United Kingdom, July 9-12, 2023*, page 539. ACM, 2023.

[25] Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Combinatorial contracts. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 815–826. IEEE, 2022.

[26] Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Multi-agent contracts. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1311–1324, 2023.

[27] Paul Dutting, Michal Feldman, and Yoav Gal Tzur. Combinatorial contracts beyond gross substitutes. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 92–108. SIAM, 2024.

[28] Paul Dütting, Tim Roughgarden, and Inbal Talgam-Cohen. Simple versus optimal contracts. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 369–387, 2019.

[29] Paul Dutting, Tim Roughgarden, and Inbal Talgam-Cohen. The complexity of contracts. *SIAM Journal on Computing*, 50(1):211–254, 2021.

[30] Paul Dütting, Michal Feldman, Yoav Gal-Tzur, and Aviad Rubinstein. The query complexity of contracts, 2024.

[31] Yuval Emek and Michal Feldman. Computing optimal contracts in combinatorial agencies. *Theoretical Computer Science*, 452:56–74, 2012.

[32] Tomer Ezra, Michal Feldman, and Maya Schlesinger. On the (in) approximability of combinatorial contracts. In *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.

[33] Tomer Ezra, Michal Feldman, and Maya Schlesinger. Sequential contracts, 2024.

[34] Tomer Ezra, Stefano Leonardi, and Matteo Russo. Contracts with inspections. *arXiv preprint arXiv:2402.16553*, 2024.

[35] Alan Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *Proceedings of 37th conference on foundations of computer science*, pages 12–20. IEEE, 1996.

[36] Sanford J. Grossman and Oliver D. Hart. *An Analysis of the Principal-Agent Problem*, pages 302–340. Springer Netherlands, Dordrecht, 1992.

[37] Guru Guruganesh, Yoav Kolumbus, Jon Schneider, Inbal Talgam-Cohen, Emmanouil-Vasileios Vlatakis-Gkaragkounis, Joshua R Wang, and S Matthew Weinberg. Contracting with a learning agent. *arXiv preprint arXiv:2401.16198*, 2024.

[38] Guru Guruganesh, Jon Schneider, and Joshua R Wang. Contracts under moral hazard and adverse selection. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 563–582, 2021.

[39] Guru Guruganesh, Jon Schneider, Joshua R Wang, and Junyao Zhao. The power of menus in contract design. In *Proceedings of the 24nd ACM Conference on Economics and Computation*, 2023.

[40] Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with psd objectives. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 482–491. IEEE, 2011.

[41] Martin Hoefer, Conrad Schecker, and Kevin Schewior. Contract design for pandora's box, 2024.

[42] Bengt Holmström. Moral hazard and observability. *The Bell Journal of Economics*, 10(1):74–91, 1979.

[43] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[44] Leonid Genrikhovich Khachiyan. A polynomial algorithm in linear programming. In *Doklady Akademii Nauk*, volume 244, pages 1093–1096. Russian Academy of Sciences, 1979.

[45] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18–28, 2001.

[46] Pasin Manurangsi and Dana Moshkovitz. Approximating dense max 2-csps. *arXiv preprint arXiv:1507.08348*, 2015.

[47] Pasin Manurangsi and Prasad Raghavendra. A birthday repetition theorem and complexity of approximating dense csps. *arXiv preprint arXiv:1607.02986*, 2016.

[48] Royal Swedish Academy of Sciences. Scientific background on the 2016 nobel prize in economic sciences. 2016.

[49] Prasad Raghavendra and Ning Tan. Approximating csps with global cardinality constraints using sdp hierarchies. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 373–387. SIAM, 2012.

[50] Eden Saig, Inbal Talgam-Cohen, and Nir Rosenfeld. Delegated classification. *Advances in Neural Information Processing Systems*, 36, 2024.

[51] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[52] Maxim Sviridenko, Jan Vondrák, and Justin Ward. Optimal approximation for submodular and supermodular optimization with bounded curvature. *Mathematics of Operations Research*, 42(4):1197–1218, 2017.

[53] Martin Weitzman. *Optimal search for the best alternative*, volume 78. Department of Energy, 1978.

[54] Shenke Xiao, Zihe Wang, Mengjing Chen, Pingzhong Tang, and Xiwang Yang. Optimal common contract with heterogeneous agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7309–7316, 2020.

[55] Grigory Yaroslavtsev. Going for speed: Sublinear algorithms for dense r-csps. *arXiv preprint arXiv:1407.7887*, 2014.

[56] Yuichi Yoshida and Yuan Zhou. Approximation schemes via sherali-adams hierarchy for dense constraint satisfaction problems and assignment problems. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 423–438, 2014.