

Approximability of Combinatorial Problems with Multi-agent Submodular Cost Functions

Gagan Goel
College of Computing
Georgia Tech
Atlanta, USA
gagan@gatech.edu

Chinmay Karande
College of Computing
Georgia Tech
Atlanta, USA
ckarande@cc.gatech.edu

Pushkar Tripathi
College of Computing
Georgia Tech
Atlanta, USA
tpushkar@cc.gatech.edu

Lei Wang
College of Computing
Georgia Tech
Atlanta, USA
lwang@cc.gatech.edu

Abstract— Applications in complex systems such as the Internet have spawned recent interest in studying situations involving multiple agents with their individual cost or utility functions. In this paper, we introduce an algorithmic framework for studying combinatorial problems in the presence of multiple agents with submodular cost functions. We study several fundamental covering problems (Vertex Cover, Shortest Path, Perfect Matching, and Spanning Tree) in this setting and establish tight upper and lower bounds for the approximability of these problems.

1. INTRODUCTION

A multitude of fundamental computational problems with real-world applications can be cast in the following framework: We are given a set X of elements, a collection C of subsets of X (i.e. $C \subseteq 2^X$) and a cost function f over the subsets of X . The collection C is typically specified via a combinatorial structure like a matroid or a graph property (for instance, the set of all spanning trees in a graph). The objective is to select a set $S \in C$ that minimizes $f(S)$.

A major focus in theoretical computer science has been on linear cost functions. The study of combinatorial problems with linear cost functions has led to great developments in the theory of exact and approximation algorithms. However, linear cost functions do not always model the complex dependencies of the costs in a real-world setting. Often, they only serve as an approximation to the original functions. As a result, even though we might have a good algorithm for solving some linear optimization problem, the output solution can still be suboptimal.

Another feature that arises in a real-world setting is the presence of multiple agents, where each agent has her own cost function. Thus, in the optimal solution, each agent might build only a part of the required combinatorial structure. For example, the Internet is a complex multi-agent system where each service provider owns only a part of the network. For linear cost functions, it is easy to see that having multiple agents doesn't change the complexity of the original problem. However, this is not the case for more general cost functions.

Motivated by these considerations, we define the following class of *combinatorial problems with multi-agent submodular cost functions* (MSCP) - We are given a set of elements X and a collection $C \subseteq 2^X$. We are also given k agents, where each agent i specifies a normalized monotone submodular cost function $f_i : 2^X \rightarrow R^+$. The goal is to find a set $S \in C$ and a partition S_1, \dots, S_k of S such that $\sum_i f_i(S_i)$ is minimized.

Submodular functions form a rich class and capture the natural properties of economies of scale or the law of diminishing returns. A function $f : 2^X \rightarrow R^+$ is said to be submodular iff for any two sets S and $T \subseteq X$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. Function f is said to be monotone if $f(S) \leq f(T)$ for any $S \subseteq T$, and normalized if $f(\emptyset) = 0$. Note that linear functions, i.e. functions of the form $f(S) = \sum_{i \in S} a_i$, are a special and classically studied subcase of submodular functions. Since a submodular function is defined over an exponentially large domain, we will work with the *value oracle* model in which an oracle will return the value of $f(S)$, when queried with the set $S \subseteq X$.

In the past, there has been some work along these lines (See [3], [5], [23], [25], [7]), but to the best of our knowledge, none of them has studied multi-agent submodular functions over a truly combinatorial structure. For instance, the work of [3] studies submodular function maximization over matroid constraints only in the single agent setting, the work of [23], [25] considers the Multi-Agent submodular functions setting but the combinatorial structure (or the collection C) used in their problem is either the set of all subsets or the whole set itself.

Notice that by fixing the collection C to any particular combinatorial structure, one can define a subclass of the problems of interest. In this contribution, we study the following fundamental subclass of problems in MSCP :

- **Submodular Vertex Cover (MS-VC):** We are given an undirected graph $G(V, E)$. Element set X is the same as the set of vertices V and the collection C consists of all the vertex covers of the graph. Recall that a set $S \subseteq V$ is a vertex cover if every $e \in E$ is incident on a vertex in S .

* Research supported by NSF Grant CCF-0728640.

Table I
RESULTS

	Single-agent		Multi-agent	
	Lower bound	Upper bound	Lower bound	Upper bound
Vertex Cover	$2 - \epsilon$	2	$\Omega(\log n)$	$2 \log n$
Shortest Path	$\Omega(n^{2/3})$	$O(n^{2/3})$	$\Omega(n^{2/3})$	$O(n^{2/3})$
Perfect Matching	$\Omega(n)$	n	$\Omega(n)$	n
Spanning Tree	$\Omega(n)$	n	$\Omega(n)$	n

- **Submodular Shortest Path (MS-SP):** We are given a connected undirected graph $G(V, E)$, and a pair of vertices $s, t \in V$. Element set X is the same as the set of edges E and the collection C consists of all the paths from s to t .
- **Submodular Minimum Perfect Matchings (MS-MPM):** In this setting, we have a undirected graph $G = (V, E)$ with cost functions over E . G contains at least one perfect matching. Element set X is the set of all edges, and the collection C is defined as the set of all perfect matchings of G . Recall that a set $M \subseteq E$ is a perfect matching of G if exactly one edge in M is incident on every vertex.
- **Submodular Minimum Spanning Tree (MS-MST):** We are given a connected undirected graph $G = (V, E)$ with cost functions over E . Element set X is the set of all edges, and the collection C is the set of spanning trees of G . Recall that a spanning tree is a minimal connected subgraph of G .

For each of the above problems, we study both the single agent and the multi-agent setting.

1.1. Motivation and Applications

We propose to extend the algorithmic study of covering problems to the more general setting of submodularity and multiple agents. From a practical viewpoint, each of the problem we study is meaningful in its own right. Shortest path and spanning trees are used in network design problems, and it is natural to assume that different agents could have different submodular cost functions depending on the set of edges they can construct cheaply. Similarly one can motivate the other problems as they are used in many settings, especially those related to algorithmic game theory.

From a theoretical perspective, one would like to extend the tools and techniques developed for combinatorial problems with linear cost functions to as general a setting as possible. Submodular functions are a natural generalization where one would expect to be able to extend the techniques. Despite the significant recent progress on some of the fundamental problems in this area [3], [5], [23], [25], [7], the algorithmic theory for combinatorial problems with submodular cost functions is not substantially developed yet. Many more basic questions remain to be identified and

solved, which could form the basis of tools and techniques for solving more complex problems.

1.2. Our Results

We give an approximation algorithm and a matching information theoretic lower bound for each of the subclass of problems that we mentioned earlier. We present these results in the table above.

Note that the minimum perfect matching and minimum spanning tree problems, which are polynomial time solvable with linear cost functions, have a large hardness factor with submodular cost functions. We would like to draw attention to our lower bound result for the Vertex Cover problem in the single agent case. In the classical vertex cover problem, the best known approximation factor is 2, and the best known hardness of approximation is 1.3606 (assuming $P \neq NP$) [4]. Khot et al. [16] showed that achieving a factor of $2 - \epsilon$ ‘might be’ hard by showing hardness result based on UGC conjecture [14]. Our results for Vertex Cover in the single agent case implies that, if the cost function over the set of vertices is submodular, then the optimal approximation factor is indeed 2.

Our hardness results use information theoretic arguments and follow the framework explained in Section 2, with some modifications specific to each problem. Our algorithms are based on LP rounding or greedy methods.

We would like to point out that our results for perfect matchings and spanning trees extend to the class of subadditive cost functions, and to related combinatorial structures such as steiner trees.

Remark: Independent of our work, recently, Iwata and Nagano [12] also gave factor 2 approximation algorithm for the single agent submodular vertex cover. They also study submodular cost set cover and submodular edge cover problem. This work [12] also appears in this conference.

1.3. Related Work

Submodular functions have been of great interest in Optimization in the past. The most fundamental of them is, perhaps, the non-monotone submodular function minimization problem. A sequence of works in this direction [20], [11], [9], [19], [10], [13] has resulted in fast strongly polynomial time combinatorial algorithms. Another related work is that of non-monotone submodular function maximization [5]. Both these algorithms are sometimes used as a subroutine in

solving the configuration LPs corresponding to some other submodular combinatorial optimization problem.

Another body of work in optimization over submodular functions deals with welfare maximization [3], [25], [6], [15]. Calinescu et al. [3] studied submodular function maximization subject to matroid constraints. They showed that their problem contains as a subcase many other allocation problems, thus giving a unified framework for studying such problems. Matching information theoretic lower bounds were established in [17].

Very recently, Svitkina and Fleischer [23] studied submodular objective function for problems like Sparsest cut, load balancing, and knapsack. They gave $O(\sqrt{\frac{n}{\log n}})$ upper and lower bounds for all these problems, showing that all these problems become much harder in the submodular setting. For the submodular procurement auctions, where a set of n goods has to be allocated to k agents (i.e. collection set $C = \{X\}$) with submodular cost functions to minimize the overall cost, a simple greedy algorithm is known to have a factor $\log(n)$ [8]. Goemans et al. [7] gave an algorithm for constructing explicit approximate submodular functions by querying polynomial number of times to the original submodular function. Some other related work in optimization that uses submodular functions include [21], [8], [24], [22], [26].

Recall that the problems we study in this contribution are very well studied in the linear cost setting. Shortest path, perfect matching and spanning tree can be solved exactly in polynomial time. Algorithm for vertex cover with factor 2 for weighted graphs was first given by [1]. The best known hardness of approximation for Vertex Cover is 1.3606 (assuming $P \neq NP$) [4]. Using UGC conjecture [14], Khot and Regev [16] showed that achieving a factor of $2 - \epsilon$ is hard.

2. PRELIMINARIES: INFORMATION THEORETIC LOWER BOUNDS

In this contribution we establish the complexity of some multi-agent submodular combinatorial optimization problems. A problem is said to have information theoretic lower bound of α if any randomized algorithm that approximates the optimum to a factor α with high probability requires superpolynomial number of queries to the value oracle.

By Yao's principle, it suffices to establish the lower bounds for deterministic algorithms acting on an input which is picked randomly from some fixed distribution. To show these approximation gaps, we follow the general framework which was also used in [23], [7], [5]. We will outline this framework in the single agent setting.

The idea is first to choose a problem instance which has a suitably large collection set $C \subseteq 2^X$ of interest. For example, for the spanning tree problem, we choose a graph that has exponentially many spanning trees. Then we design two submodular cost functions f and g . Typically,

g is deterministically picked, whereas f is chosen from a distribution. The choice of f and g relies on the following two properties: a) f and g must be 'hard to distinguish' in the sense that they return the same value on almost all queries and b) The optimum values of f and g over C must differ by a large factor. Intuitively this amounts to 'hiding' a particular set $Q \in C$ in f by setting $f(Q)$ to a low value. The set Q is chosen from a distribution over C . Since C is designed to be extremely large, this leaves an exponentially small probability of an arbitrary query S made by an algorithm differentiating f from g . By the union bound and a computation path argument [23], [5], an algorithm making polynomially many number of queries cannot distinguish between f and g . Combining this with the gap in the optima of f and g , one proves the lower bound.

We note an important observation from the above discussion, which we will use in our proofs of lower bounds:

Observation 2.1. *To prove an information theoretic lower bound using two submodular functions f and g with a gap in their optimum values, it suffices to prove that $\Pr[f(Q) \neq g(Q)]$ is exponentially small over the random choice of $Q \subseteq X$.*

3. VERTEX COVER

In this section, we consider the submodular vertex cover problem. We first prove an information theoretic lower bound of $2 - \epsilon$ (for any fixed ϵ) for the single agent case and provide an algorithm with approximation ratio of 2. We then present a $2 \log n$ approximation algorithm for the multi-agent case and an information theoretic lower bound of $\Omega(\log n)$.

3.1. Single Agent Case

We are given an undirected graph $G(V, E)$ and a normalized monotone submodular function $f : 2^V \rightarrow \mathbb{R}$. We wish to find a vertex cover $U \subseteq V$ of graph G such that $f(U)$ is minimized.

Theorem 3.1. *For every fixed $\epsilon > 0$, any randomized algorithm for the submodular vertex cover problem with an approximation ratio of $2 - \epsilon$ needs exponentially many queries to the value oracle.*

Proof:

Consider a bipartite graph $G(A \cup B, E)$ such that $|A| = |B| = n$. The edge set consists of n edges which forms a matching between A and B . Let R be a random minimum cardinality vertex cover of this graph, which can be picked by choosing one endpoint of every edge uniformly at random.

Define the following two submodular cost functions.

$$\begin{aligned} f_R(S) &= \min\{|S \cap \bar{R}| + \\ &\quad \min\{|S \cap R|, \frac{(1 + \delta)n}{2}\}, n\} \\ g(S) &= \min\{|S|, n\} \end{aligned}$$

Here δ is chosen such that $2/(1+\delta) = 2 - \epsilon$. Notice that the optimum value of the vertex cover for the function f_R is $\frac{(1+\delta)n}{2}$, and for g it is n . Thus if we can show that any randomized algorithm, cannot distinguish between f_R and g with high probability, then it will imply an inapproximability ratio of $2/(1+\delta)$ or $2 - \epsilon$ for the submodular vertex cover problem.

From Observation 2.1 it suffices to show that for a deterministic query Q , $Pr[f_R(Q) \neq g(Q)]$ is exponentially small, where the probability space is defined over the random choice of set R . Since $f_R(S) \leq g(S)$ for all $S \subseteq V$, $f_R(Q) \neq g(Q)$ implies $f_R(Q) < g(Q)$.

Let Q^* be the optimal query for which $Pr[f_R(Q) < g(Q)]$ is maximized. We will show that $|Q^*| = n$. First, suppose that $|Q| \geq n$, then

$$\begin{aligned} Pr[f_R(Q) < g(Q)] &= Pr[f_R(Q) < n] \\ &= Pr[|Q \cap \bar{R}| + \min \{ |Q \cap R|, (1+\delta)n/2 \} < n] \end{aligned}$$

which increases as the size of Q is reduced. Thus the size of the optimal query in this case is n .

Now suppose $|Q| \leq n$. In this case,

$$\begin{aligned} Pr[f_R(Q) < g(Q)] &= Pr[f_R(Q) < |Q|] \\ &= Pr[|Q \cap \bar{R}| + \min \{ |Q \cap R|, (1+\delta)n/2 \} < |Q|] \\ &= Pr[\min \{ |Q \cap R|, (1+\delta)n/2 \} < |Q \cap R|] \\ &= Pr[|Q \cap R| > (1+\delta)n/2] \end{aligned}$$

which increases as $|Q|$ is raised. Therefore, the optimal query size in this case is also n .

Hence $|Q^*| = n$. Let k be the number of edges for which both the end points are contained in Q^* . Call the corresponding set of vertices Q_1 ($|Q_1| = 2k$). Let $Q_2 = Q^* - Q_1$. Now $Pr[f_R(Q^*) < g(Q^*)] = Pr[|Q^* \cap R| > (1+\delta)n/2] = Pr[|Q_2 \cap R| > (1+\delta)n/2 - k] \leq e^{-n\delta^2/3}$. Last inequality follows from Chernoff bounds. Hence, the probability that an arbitrary query Q can distinguish between f and g is exponentially small. \blacksquare

Theorem 3.2. *There exists an algorithm which finds a 2-approximate solution to the single agent vertex cover problem with submodular costs.*

Proof: We use the LP rounding method to give our algorithm. Let variable x_S be an indicator variable for the set S of vertices being the vertex cover. Then the following LP is a lower bound on the value of the optimal integral solution.

$$\begin{aligned} \min \sum_{S \subseteq V} x_S f(S) & \quad (LP1) \\ \sum_{S:u \in S} x_S + \sum_{S:v \in S} x_S & \geq 1 \quad \forall (u,v) \in E \\ x_S & \geq 0 \quad \forall S \subseteq V \\ \max \sum_{e \in E} y_e & \quad (Dual) \\ \sum_{v \in S} \sum_{e \in \delta(v)} y_e & \leq f(S) \quad \forall S \subseteq V \\ y_e & \geq 0 \quad \forall e \in E \end{aligned}$$

It is not difficult to see that the function $\sum_{v \in S} \sum_{e \in \delta(v)} y_e$ is a modular function. Thus $f(S) - \sum_{v \in S} \sum_{e \in \delta(v)} y_e$ is a submodular function, and we can use the submodular minimization algorithm as a subroutine to construct a separation oracle for the dual. This allows us to find an optimal fractional solution to the LP1 with value OPT. Let x^* be this solution. Output $Q = \{ u \in V : \sum_{S:u \in S} x_S^* \geq 1/2 \}$ as the vertex cover. Clearly, for any $(u,v) \in E$, either $\sum_{S:u \in S} x_S^* \geq 1/2$ or $\sum_{S:v \in S} x_S^* \geq 1/2$ must hold, thus Q is a valid vertex cover of G . Since $2x^*$ is a fractional cover of Q , submodularity implies that $f(Q) \leq 2 \sum_{S \subseteq V} x_S^* f(S) = 2 \cdot \text{OPT}$. \blacksquare

3.2. Multi-agent Case

We are given an undirected graph $G(V,E)$ and a normalized monotone submodular function $f_i : 2^V \rightarrow \mathbb{R}$ for each agent i . We wish to find a vertex cover $U \subseteq V$, and a partition U_1, U_2, \dots, U_k of U such that $\sum_i f(U_i)$ is minimized.

We will use the following theorem (proof in appendix) to prove our lower bound for the multi-agent case.

Theorem 3.3. *Given n items and k agents, each with her own normalized monotone submodular cost function over the set of items, the problem of allocating these items to the agents so as to minimize the total cost has an information-theoretic lower bound of $\Omega(\log n)$.*

Now we will sketch the lower bound proof for the multi-agent case. Consider a suitable instance graph such as the one used in the proof of Theorem 3.1, and fix a vertex cover Q . Define the cost functions of various agents over the set Q to be same as the one defined in the proof of theorem 3.3. For every other set S (i.e. set which contains elements from $V - Q$) assign a very high value to $f_i(S)$ for every agent i . Essentially, the problem of finding minimum cost vertex cover reduces to that of assigning vertices in Q to the various agents so as to minimize the total cost - which is constrained by theorem 3.3 to an information theoretic lower bound of $\Omega(\log n)$. Thus we get the following theorem.

Theorem 3.4. *Any randomized algorithm for the multi-agent submodular vertex cover problem with an approximation ratio $c \log n$ for some constant $c < 1$ needs exponentially many queries to the value oracle.*

2 log n-approximate algorithm: We begin by finding an optimal fraction solution \bar{x} using the LP relaxation LP2, which gives a lower bound on the optimal integral solution. The given LP can be solved by constructing a separation oracle of the dual program as shown earlier in the single agent case. Consider the set $Q = \left\{ u \in V : \sum_{S:u \in S} \sum_i \bar{x}_{i,S} \geq 1/2 \right\}$, which forms a valid vertex cover. We will now round $2\bar{x}$ to find an allocation of vertices in Q to the various agents. Let W denote the total cost of the solution $2\bar{x}$.

$$\begin{aligned} \min \sum_{S \subseteq V} \sum_i x_{i,S} f_i(S) \quad (LP2) \\ \sum_{S:u \in S} \sum_i x_{i,S} + \sum_{S:v \in S} \sum_i x_{i,S} \geq 1 \quad \forall (u,v) \in E \\ x_{i,S} \geq 0 \quad \forall S \subseteq V, \forall i \end{aligned}$$

At any step of the algorithm let Z contain the uncovered elements in Q . For any fractional cover x of Z , define $\alpha_x(u) = \sum_{S:u \in S} \sum_i \frac{x_{i,S} f_i(S)}{|S \cap Z|}$. Note that $\sum_u \alpha_x(u) = \sum_{i,S} x_{i,S} f_i(S)$. Pick $u \in Z$ that minimizes $\alpha_{2\bar{x}}(u)$. Among the sets containing u , choose a set (i, S) randomly with probability proportional to $x_{i,S}$. Remove all the newly covered elements from Z and iterate until all the elements in Q are covered. Let y denote this integral cover.

Analysis: Let u_1, u_2, \dots be the order in which the vertices in Q get covered. We claim that $E[\alpha_y(u_j)] \leq W/(|Q| - j + 1)$. Suppose u_j was picked during the algorithm. Then, $E[\alpha_y(u_j)] \leq \alpha_{2\bar{x}}(u_j)$. Since $2\bar{x}$ covers the remaining $|Q| - j + 1$ elements in Q , $\alpha_{2\bar{x}}(u_j) \leq W/(|Q| - j + 1)$. On the other hand, if u_j was not picked during the algorithm, then $E[\alpha_y(u_j)] = \alpha_{2\bar{x}}(u'_j) \leq W/(|Q| - j' + 1) \leq W/(|Q| - j + 1)$ for some $j' < j$. Summing over j , we have

$$\begin{aligned} \sum_{i,S} y_{i,S} f_i(S) &= \sum_{u \in Q} \alpha_y(u) \leq \sum_{u \in Q} \alpha_{2\bar{x}}(u) \\ &\leq W \log n \leq 2\text{OPT} \cdot \log n \end{aligned}$$

This algorithm can be derandomized using standard techniques.

4. SHORTEST PATH

In this problem we are given an undirected graph $G = (V, E)$ and a monotone submodular cost function $f_i : 2^E \rightarrow \mathbb{R}$ for each agent i . The goal is to find a path P between two given vertices, and partition of P into

P_1, P_2, \dots, P_k such that $\sum_i f_i(P_i)$ is minimized. We first consider the single agent case and provide an information-theoretic lower bound of $n^{2/3-\epsilon}$ for all fixed $\epsilon > 0$. We also present an $O(n^{2/3})$ approximation algorithm for this case. Lastly, we give a sketch of an $O(n^{2/3})$ approximation algorithm for the multi-agent case.

4.1. Single agent case

As in previous sections, we proceed by designing two submodular functions that are hard to distinguish in polynomially many queries but have different optimal values. In the general framework outlined in section 2, this is accomplished by ‘hiding’ a random element of lower cost from the target collection C in one of the functions. In this case, C is the set of all $s-t$ paths. However an identical analysis does not work in this case. This is because for a pair of adjacent edges, the events that these edges belong to the random shortest $s-t$ path are not independent precluding the use of Chernoff bounds which makes the analysis a lot more involved. In this section we use a simple pigeon hole principle argument to solve this problem.

Theorem 4.1. *For every fixed $\epsilon > 0$, any randomized approximation algorithm for the submodular shortest path problem with factor $n^{2/3-\epsilon}$ needs exponentially many queries.*

Proof: Consider the graph G which is a level graph having $n^{2/3} + 2$ levels of vertices. First level contains only vertex s and the last level contains only t . Each other level has $n^{1/3}$ vertices and there exists a complete bipartite graph between successive levels. Let R be a randomly chosen $s-t$ path of length $n^{2/3} + 1$.

Define the following two submodular cost functions $f, g : 2^E \rightarrow \mathbb{R}^+$:

$$\begin{aligned} f(Q) &= \min\{ |Q \cap \bar{R}| + \\ &\quad \min\{ |Q \cap R|, (1+n^\epsilon) \}, n^{2/3} + 1 \} \\ g(Q) &= \min\{ |Q|, n^{2/3} + 1 \} \end{aligned}$$

Clearly the ratio of optima in g and f is at least $\frac{n^{2-\epsilon}+1}{(1+n^\epsilon)}$.

Now we look at the probability that the algorithm can not distinguish f and g . By observation 2.1 it suffices to prove that $Pr[f(Q) \neq g(Q)]$ is exponentially small for an arbitrary query Q . This happens if and only if $f(Q) < g(Q)$. Making arguments analogous to the proof of theorem 3.1, $Pr[f(Q) < g(Q)]$ is maximized when $|Q| = 1 + n^{2/3}$. Therefore,

$$Pr[f(Q) < g(Q)] = Pr[|Q \cap R| > 1 + n^\epsilon]$$

Let E_{even} and E_{odd} be the set of edges which are at distance even and odd respectively from the vertex s . Define $Q_{\text{even}} = Q \cap E_{\text{even}}$ and $Q_{\text{odd}} = Q \cap E_{\text{odd}}$. Similarly define R_{even}

and R_{odd} . Without loss of generality, let $|Q_{odd}| \geq |Q_{even}|$. Thus,

$$\begin{aligned} Pr[|Q \cap R| > 1 + n^\epsilon] &= Pr[|Q_{even} \cap R_{even}| + |Q_{odd} \cap R_{odd}| > 1 + n^\epsilon] \\ &\leq 2 \cdot Pr[|Q_{odd} \cap R_{odd}| > \frac{1 + n^\epsilon}{2}] \end{aligned}$$

Note that the edges in R_{odd} were chosen independently at random since R was chosen uniformly at random. Also $E[|Q_{odd} \cap R_{odd}|] = O(1)$. Thus by chernoff bounds we conclude that is $Pr[|Q \cap R| > 1 + n^\epsilon] \leq O(e^{-n^{2\epsilon}})$. This proves the theorem. ■

Theorem 4.2. *There exists an algorithm which finds an $O(n^{2/3})$ approximate solution to the single agent shortest path problem with submodular costs.*

Proof: We analyse two simple approaches to get an $O(n)$ approximate algorithm and then using ideas from both the approaches, we get an $O(n^{2/3})$ approximate algorithm for the problem.

[7] addresses the problem of finding approximate explicit representations for submodular functions. The representation returned by [7] assigns a weight to every element of the base set. The cost of a set is simply the square root of the sum of the weights of its elements. They develop a \sqrt{n} approximate explicit representation for a submodular function over a ground set of size n . A possible approach would be to find the weights for all the of the graph and then find the shortest path using these weights. This approach yields a $O(\sqrt{E})$ approximation algorithm. For dense graphs this can be as bad as $O(n)$. This approach can be useful if the given graph has few edges.

Another simple algorithm to get an $O(n)$ approximation for this problem is to do the following. Define *cost* of an edge e as $f(\{e\})$. The algorithm runs in multiple phases. In each phase choose a new edge and drop all edges that weigh more than the given edge and return any $s-t$ path(if it exists) in the pruned graph. We finally select the smallest $s-t$ paths among those returned during the phases. It is easy to see that this is an $O(l)$ approximate algorithm where l is the number of edges in the optimal path. Once again this can be as bad as $O(n)$ for some graphs. This approach can be useful if the given graph is *dense*, since sufficiently dense graphs are known to have small diameter.

The basic idea of our algorithm is to decompose the graph in to sparse and dense clusters. Then we use the first approach to account for sparse regions of the graph and deal with the dense regions using ideas from the second algorithm.

The algorithm runs in multiple phases, where after each phase we output a path. Final solution is the minimum cost path among these paths. Each phase is identified by a unique edge in the edge set, thus there are $|E|$ phases. Following are the steps, in order, which constitutes a single phase.

Pruning Step: Let e be the edge corresponding to the current phase. Delete all edges that weigh more than e . Let G_e denote the pruned graph. The phase terminates prematurely if the s and the t are disconnected in G_e .

Separation Step: In this step we partition the edge set into those that are in dense regions of the graph and those which belong to sparse regions. Succisively remove vertices from G_e whose degree in the remaining graph is at most $n^{1/3}$. Also remove the edges incident on these vertices and add them to the set S_e . Continue removing vertices until all the remaining vertices have degree more than $n^{1/3}$. Let R_e be the remaining edges in G_e . Edges in S_e belong to the sparse part of the graph while those in R_e constitute the dense part of the graph.

Search Step: Using [7] we find an explicit representation for the function f restricted to the S_e . Redefine the costs for edges in S_e to be the weights returned by the subroutine and set the cost of each edge in R_e to be a zero. Treating these edge weights as additive quantities, find the shortest $s-t$ path passing through e . Let P_e be this path.

Compression Step: The path returned by the search step might contain too many edges, which could be bad for the algorithm. In this step, we compress the path P_e by replacing some of its subpaths by smaller paths(in terms of number of edges). For this we analyze the intersection of P_e with every connected component of $G[V, R_e]$. Let H be an arbitrary connected component of $G[V, R_e]$ and let a be the first vertex where P_e enters H and b be the final vertex that it passes through before leaving H for the last time. Replace the subpath of P_e between a and b with the shortest path in H (in terms of the number of edges) connecting them(refer to the figure below). Do this for every connected component of $G[V, R_e]$. Report this modified path as the solution for this phase.

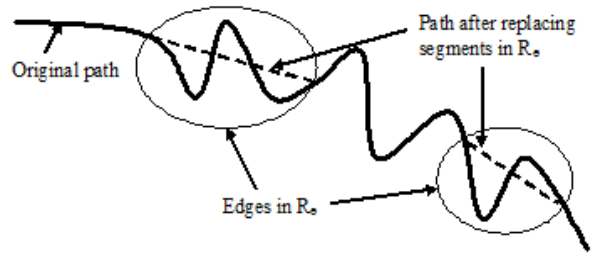


Figure 1. Taking short cuts in dense regions

Analysis: To prove that the above algorithm achieves an approximation ratio of $O(n^{2/3})$ we will use the following observation.

Observation: Since all the vertices remaining after the first step have degree greater than $n^{1/3}$, any connected component C of $G[V, R_e]$ has diameter at most $|V(C)|/n^{1/3}$.

Let P_{OPT} be an optimal solution for the problem. Let

α be the heaviest edge in this path. Consider the phase corresponding to α . During the separation step of every phase we remove at most $n^{4/3}$ edges since each of the chosen vertices have a degree less than $n^{1/3}$. Thus the subroutine gives an explicit cost function that is a $O(n^{2/3})$ approximation for all subsets of S_α . Let $\hat{f}(A)$ denote the cost of any $A \subseteq S_\alpha$ returned by the subroutine. Thus we have $f(A) \leq \hat{f}(A) \leq n^{2/3}f(A)^1 \forall A \subseteq S_\alpha$.

Let P_α be the solution returned by this phase. Define $X_\alpha = P_{OPT} \cap S_\alpha$, $Y_\alpha = P_\alpha \cap S_\alpha$ and $Z_\alpha = P_\alpha \cap R_\alpha$. It follows that,

$$2n^{2/3}f(P_{OPT}) \geq n^{2/3}f(P_{OPT}) + n^{2/3}f(\{\alpha\}) \quad (1)$$

$$\geq n^{2/3}f(X_\alpha) + n^{2/3}f(\{\alpha\}) \quad (2)$$

$$\geq \hat{f}(X_\alpha) + n^{2/3}f(\{\alpha\}) \quad (3)$$

$$\geq \hat{f}(Y_\alpha) + n^{2/3}f(\{\alpha\}) \quad (4)$$

$$\geq \hat{f}(Y_\alpha) + f(Z_\alpha) \quad (5)$$

$$\geq f(Y_\alpha) + f(Z_\alpha) \quad (6)$$

$$\geq f(Y_\alpha \cup Z_\alpha) \quad (7)$$

$$= f(P_\alpha) \quad (8)$$

Equations (1) and (2) follow from monotonicity. Equation(4) follows from equation(3) since P_α is the shortest path under the function \hat{f} . Also, using the observation above and summing over all components of $G[V, R_\alpha]$ we conclude that $|Z_\alpha|$ can not be more than $n^{2/3}$. Thus equation(5) follows from equation(4) since each edge in $|Z_\alpha|$ costs at most $f(\{\alpha\})$ and f is submodular. We arrive at equations(6), (7), (8) by the definition of \hat{f} and using submodularity of function f . ■

4.2. Multi-agent case

Now we give an outline of the algorithm for the multiagent case, which also has a factor $O(n^{2/3})$. Just like the single agent case we proceed in a number of phases each having similar steps as before. At the end of each phase we output a path and the final solution is the best solution across all phases as the answer. Each phase is identified by a unique agent-edge pair. Thus there are $k|E|$ phases.

Consider the phase corresponding to agent A and edge e . Let f_A be the valuation function for A . We define the *minimized cost* of an edge as the minimum valuation for that edge(as a singleton set) across all agents. In the pruning step we drop all edges whose *minimized cost* is more than $f_A(\{e\})$. The separation step is identical to that for the single agent case where we separate the edge set into edges belonging to the dense part of the graph and those lying in the sparse regions. For the search step, we find the explicit representation for edges in S_e for every agent. For every edge in S_e we define its *modified weight* as the minimum weight returned by the subroutines(using the algorithm in

[7]) for that edge, across all agents. All edges in R_e carry zero weight. We find the shortest path passing through e under this modified weight function treating all weights to be additive. The compression step is once again identical to that for the single agent case, where we reduce the number of edges in the path found in the previous step by replacing sub paths with smaller paths.

One last step remains, that is to allocate edges to the agents. We assign all edges in R_e to the corresponding agent for that phase. We also assign the edge e to this agent. For every edge in S_e , we allocate it to the agent who had the lowest weight for that edge under the explicit representations found during the search step.

The analysis for the algorithm is similar to that for the single agent case.

5. PERFECT MATCHING

In this section, we consider the multi-agent submodular minimum perfect matching problem. In this problem we are given a bipartite graph $G(V, E)$ containing at least one perfect matching and a normalized monotone submodular function $f_i : 2^E \rightarrow R^+$ for each agent i . We wish to find a perfect matching M , and a partition of M into M_1, M_2, \dots, M_k such that $\sum_i f_i(M_i)$ is minimized. We first prove an information theoretic lower bound of $\Omega(n)$ on the approximability of the single agent case, which also implies the same bound for the multi-agent case. Then, we give an n approximate algorithm for the multi-agent case.

As in previous sections, we proceed by designing two submodular functions that are hard to distinguish in polynomially many queries but have different optimal values. In the general framework outlined in section 2, this is accomplished by ‘hiding’ a random element of lower cost from the target collection C in one of the functions. In this case, C is the set of all perfect matchings. Once again choosing a random matching from C however does not serve our purpose because for a fixed pair of edges, the events that these edges belong to the random matching are not independent precluding the use of Chernoff bounds. This time we circumvent this problem by using the following result from the theory of random graphs [2]:

Lemma 5.1. *Let $G(n, n, p)$ be a random bipartite graph on $2n$ vertices such that each edge is present independently with probability p . Then*

$$Pr[G(n, n, p) \text{ contains no perfect matching}] = O(ne^{-np})$$

Now instead of hiding a randomly chosen perfect matching, we hide a collection of randomly and independently chosen edges that contains a perfect matching with high probability. We prove the following theorem.

Theorem 5.1. *For every fixed $\epsilon, \delta > 0$, any randomized approximation algorithm for the submodular minimum cost perfect matching problem with factor $n^{1-3\epsilon}/(1+\delta)$ needs*

¹This is a slight deviation from the result in [7] but it can be derived by scaling their result by \sqrt{n}

exponentially many queries. Also there exists an algorithm that approximately finds an n -approximate minimum cost matching in polynomial time.

Proof: Consider the graph G which is a union of n^ϵ different copies of the complete bipartite graph $K_{n^{1-\epsilon}, n^{1-\epsilon}}$. Let G_i be the i^{th} copy. In each copy G_i , we pick a random subset R_i of edges by picking each edge independently with probability $p = 1/n^{1-2\epsilon}$. Let $R = \bigcup R_i$. By applying lemma 5.1 followed by the union bound, R contains a perfect matching with probability $1 - O(\text{poly}(n)e^{-n^\epsilon})$.

Define the following two submodular cost functions $f_R, g : 2^E \rightarrow \mathbb{R}^+$:

$$\begin{aligned} f_R(Q) &= \min \{ |Q \cap \bar{R}| + \\ &\quad \min\{ |Q \cap R|, (1 + \delta)n^{3\epsilon} \}, n^{1+\epsilon} \} \\ g(Q) &= \min \{ |Q|, n^{1+\epsilon} \} \end{aligned}$$

With probability $1 - O(\text{poly}(n)e^{-n^\epsilon})$, R contains a perfect matching and hence the minimum cost of a perfect matching in f is at most $(1 + \delta)n^{3\epsilon}$. Therefore the ratio of optima in g and f is at least $\frac{n^{1-3\epsilon}}{(1+\delta)}$ with high probability.

Now we look at the probability that the algorithm can not distinguish f and g . By observation 2.1 it suffices to prove that $\Pr[f_R(Q) \neq g(Q)]$ is exponentially small for an arbitrary query Q . It's easy to see that $f_R(S) \leq g(S)$, thus these two functions differ on Q if and only if $f_R(Q) < g(Q)$.

Making arguments analogous to the proof of theorem 3.1, $\Pr[f_R(Q) < g(Q)]$ is maximized when $|Q| = n^{1+\epsilon}$. Therefore,

$$\Pr[f_R(Q) < g(Q)] = \Pr[|Q \cap R| > (1 + \delta)n^{3\epsilon}]$$

Since $E[|Q \cap R|] = n^{3\epsilon}$ and edges were picked uniformly at random, we can apply Chernoff bounds to conclude that this probability is $O(e^{-n^{2\epsilon}\delta^2})$. This proves the theorem. ■

Factor n approximation algorithm for MS-MPM: We are given a graph $G(V, E)$ and submodular cost functions f_i for each agent. Define a new cost function w over E as $w_e = \min_i f_i(\{e\})$ and define $w(Z) = \sum_{e \in Z} w_e$ for all $Z \subseteq E$. Since w is an additive valuation function we can find a minimum value perfect matching in polynomial time. Let M be such a matching. Assign each edge $e \in M$ to the agent having the minimum valuation for that edge. Let the cost of this solution under the original valuation functions be W .

Analysis: We now prove that this is an n -approximate algorithm. By submodularity we have $W \leq w(M)$. Let M_0 be an optimal solution of MS-MPM having value OPT . Since M is a minimum weight matching under w , $w(M) \leq w(M_0)$.

Let $w_{\max} = \max_{e \in M_0} \{f_i(e) \mid e \text{ assigned to agent } i \text{ in } M_0\}$. By submodularity of the cost functions, $w(M_0) \leq n \cdot w_{\max}$. By monotonicity we have $w_{\max} \leq \text{OPT}$. Therefore,

$$W \leq w(M) \leq w(M_0) \leq n \cdot w_{\max} \leq n \cdot \text{OPT}$$

This completes the analysis.

6. SPANNING TREE

In this section, we consider the multi-agent submodular minimum spanning tree problem. We are given a connected graph $G(V, E)$ and a normalized monotone submodular function $f_i : 2^E \rightarrow \mathbb{R}^+$ for each agent i . We want to find a spanning tree T of G , and a partition of T into T_1, T_2, \dots, T_k such that $\sum_i f_i(T_i)$ is minimized. We first prove an information theoretic lower bound of $\Omega(n)$ on the approximability of the single agent case, which also implies the same bound for the multi-agent case. Then, we give an n approximate algorithm for the multi-agent case.

To prove the lower bound we will provide two submodular functions that can not be distinguished in polynomially many queries and have widely differing optimal values. As in Section 5, we will use the following lemma [2] in the proof.

Lemma 6.1. *Let $G(n, p)$ be a random graph on n vertices such that each edge is present independently with probability p . Then*

$$\Pr[G(n, p) \text{ is disconnected}] \leq n(1 - p)^n.$$

Theorem 6.1. *For every fixed $\epsilon, \delta > 0$, any randomized approximation algorithm for the submodular minimum spanning problem on a with factor $n^{1-3\epsilon}/(1 + \delta)$ needs exponentially many queries. Also there exists an algorithm that approximately finds an n -approximate spanning tree in polynomial time.*

Proof: Consider n^ϵ cliques G_1 to G_{n^ϵ} , each with $n^{1-\epsilon}$ vertices. Let G be the graph formed by joining these cliques at one of the their vertices, thus making the graph G connected. We choose a random subset of edges R_i , in each copy G_i , by picking each edge independently with probability $p = 1/n^{(1-2\epsilon)}$. Let $R = \bigcup R_i$. By applying Lemma 6.1 followed by the union bound, R is connected with probability at least $1 - ne^{-n^{2\epsilon}}$.

Define the following two submodular cost functions $f_R, g : 2^E \rightarrow \mathbb{R}^+$:

$$\begin{aligned} f_R(Q) &= \min \{ |Q \cap \bar{R}| + \\ &\quad \min\{ |Q \cap R|, (1 + \delta)n^{3\epsilon} \}, n^{1+\epsilon} \} \\ g(Q) &= \min \{ |Q|, n^{1+\epsilon} \} \end{aligned}$$

With probability $1 - O(\text{poly}(n)e^{-n^\epsilon})$, R is connected and hence, the cost of the optimal spanning tree in f is at most $(1 + \delta)n^{3\epsilon}$. Therefore, the ratio of optimal solution values in g and f is at least $\frac{n^{1-3\epsilon}}{(1+\delta)}$ with high probability.

Making arguments similar to the proof of Theorem 5.1, we conclude that $\Pr[f_R(Q) < g(Q)] = O(\text{poly}(n)e^{-n^\epsilon})$ for any query Q . By observation 2.1, this suffices to prove the theorem.

Factor n approximation algorithm for MS-MST: We are given a graph $G(V, E)$ and submodular cost functions f_i for each agent. Define a new cost function w over E as $w_e = \min_i f_i(\{e\})$. Run Kruskal's algorithm on G treating w_e as the cost of the edge e to get a minimum spanning tree T . Assign each edge $e \in T$ to the agent i minimizing $f_i(\{e\})$. The proof that this constitutes an n -approximate solution follows similar arguments as the analysis of the n -approximate algorithm for perfect matching. ■

7. DISCUSSION

The setting that we have considered in this work is quite general, and is a very exciting avenue of research. There are many other interesting problems in this class such as minimum graph cut, edge cover which could be studied in the future work. We have considered the covering problems in this work, one can ask the same questions for packing problems like maximum matching. One could also consider even more general functions like Subadditive functions. Extension to multi-agent makes a natural connection to Game Theory. Mechanism design of these combinatorial problem also has interesting applications. Another area of interest is the characterization of functions for which the computational complexity doesn't change when considered in this general setting compared to the linear one.

Acknowledgement: We would like to thank Vijay Vazirani for valuable inputs and discussions.

REFERENCES

- [1] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.
- [2] B. Bollobas. *Random Graphs*. Cambridge University Press, 2001.
- [3] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *IPCO*, pages 182–196, 2007.
- [4] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:2005, 2005.
- [5] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *FOCS*, pages 461–471, 2007.
- [6] Uriel Feige and Jan Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1 - 1/e$. In *FOCS*, pages 667–676, 2006.
- [7] Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544, 2009.
- [8] Ara Hayrapetyan, Chaitanya Swamy, and Éva Tardos. Network design for information networks. In *SODA*, pages 933–942, 2005.
- [9] Satoru Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Comput.*, 32(4):833–840, 2003.

- [10] Satoru Iwata. Submodular function minimization. *Math. Program.*, 112(1):45–64, 2008.
- [11] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001.
- [12] Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. *Foundations of Computer Science, Annual IEEE Symposium on*, 2009.
- [13] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 1230–1237, 2009.
- [14] Subhash Khot. On the power of unique 2-prover 1-round games. In *STOC*, pages 767–775, 2002.
- [15] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. *Algorithmica*, 52(1):3–18, 2008.
- [16] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.
- [17] Vahab S. Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *ACM Conference on Electronic Commerce*, pages 70–77, 2008.
- [18] Noam Nisan. The communication complexity of approximate set packing and covering. In *ICALP*, pages 868–875, 2002.
- [19] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. In *IPCO*, pages 240–251, 2007.
- [20] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000.
- [21] Yogeshwer Sharma, Chaitanya Swamy, and David P. Williamson. Approximation algorithms for prize collecting forest problems with submodular penalty functions. In *SODA*, pages 1275–1284, 2007.
- [22] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.
- [23] Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS*, pages 697–706, 2008.
- [24] Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. In *SODA*, pages 153–161, 2006.
- [25] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.
- [26] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. In *Combinatorica*, pages 385–393, 1982.

APPENDIX

Proof of theorem 3.3 : The problem referred to by the statement of the theorem is called *combinatorial reverse auction* (CRA) and it belongs to the class MSCP as follows: The ground set X consists of the items, and the collection C consists of exactly one set - the set X itself. For the lower bound, again the idea is to construct a deterministic instance

and a random instance of the CRA so that the optimal solution of these two instances differ by a factor of $\Omega(\log n)$, and then show that with high probability, a deterministic algorithm which uses only polynomially many value queries can not distinguish between these two instances.

Consider the following deterministic instance of CRA: There are m agents and a set J of $n = m(m+1)^2/4$ elements. The elements are equally partitioned into m blocks J_1, J_2, \dots, J_m . We will choose m such that $m = 2^d - 1$, for some d . Now each number i between 1 and m can be represented as a vector \mathbf{a}_i in $GF[2]^d$. Let $G_i = \bigcup_{1 \leq k \leq m, \mathbf{a}_i \cdot \mathbf{a}_k = 1} J_k$. For each i , $1 \leq i \leq m$, agent i is only interested in elements in G_i . It can be shown that $|\{\mathbf{a}_k : \mathbf{a}_i \cdot \mathbf{a}_k = 1\}| = (m+1)/2$. Thus each agent is interested in only $(m+1)/2$ blocks of elements and for each block there are $(m+1)/2$ agents who are interested in it. Now, we define the cost function $f_i : 2^J \rightarrow \mathbb{R}^+$ as follows:

$$f_i(S) = \begin{cases} \min\{|S|, (m+1)^2/4\}, & \text{if } S \subseteq G_i \\ \infty, & \text{otherwise} \end{cases}$$

Let us analyze the optimal cost of this instance. We say that an agent is *marked* if the total size of elements assigned to him is at least $(m+1)^2/4$. Among all the optimal solutions, let OPT be the one that maximizes the number of marked agents. We claim that at least d agents are marked in OPT . Suppose not, then without loss of generality, we may assume $M = \{1, 2, \dots, t\}$ to be the set of marked agents and $t < d$. The system of linear equations $\mathbf{a}_i \cdot \mathbf{x} = 0, \forall 1 \leq i \leq t$ has at least one solution $\mathbf{x}^* \in GF[2]^d$, since number of equations is less than the number of variables. Let k be the number between 1 and m corresponding to the vector \mathbf{x}^* . This implies that no agent in M is interested in block J_k . Let $A_k = \{i_1, i_2, \dots, i_w\}$ be the set of agents who are assigned some elements in J_k . Then $A_k \cap M = \emptyset$. Therefore, we can mark one more agent by transferring the elements in J_k from agents i_2, i_2, \dots, i_w to agent i_1 without changing the cost of the new solution. This is a contradiction because of the choice of OPT . Hence, the optimal cost of this instance is at least $(m+1)^2 d/4$.

For the random instance, we have the same sets of agents and elements. Also, each agent is interested in the same set of elements. However, the cost function for each agent is defined by a probability distribution on the set assigned to her. Next we describe our construction of the random cost functions explicitly.

For each element, assign it uniformly at random to one of the agents who is interested in it. Let S_i be the set of elements which agent i gets. Clearly (S_1, S_2, \dots, S_m) forms a partition of the element set J . We define the cost function $g_i : 2^J \rightarrow \mathbb{R}^+$, for agent i as follows:

$$g_i(S) = \begin{cases} \min\{|S \cap \overline{S}_i| + \min\{|S \cap S_i|, (1+\delta)(m+1)/2\}, (m+1)^2/4\}, & \text{if } S \subseteq G_i \\ \infty, & \text{otherwise} \end{cases}$$

where $\delta > 0$ is a fixed constant.

Now we show that with high probability, a deterministic algorithm using only polynomially many value queries can not distinguish between $\mathbf{f} = (f_1, f_2, \dots, f_m)$ and $\mathbf{g} = (g_1, g_2, \dots, g_m)$. We prove the following lemma.

Lemma A.1. *For any subset S of elements and any i , $1 \leq i \leq m$, $Pr[f_i(S) \neq g_i(S)] = e^{-\Omega(m)}$.*

Proof: Suppose S is a subset of elements and $1 \leq i \leq m$. By our construction, $g_i(S) \leq f_i(S)$. Therefore $Pr[f_i(S) \neq g_i(S)] = Pr[g_i(S) < f_i(S)]$.

First of all, we claim that the above probability is maximized when $S \subseteq G_i$ and $|S| = (m+1)^2/4$. For this, if $S \not\subseteq G_i$, then $f_i(S) = g_i(S) = \infty$ hence $Pr[f_i(S) < g_i(S)] = 0$. Now suppose $S \subseteq G_i$ and $|S| \geq (m+1)^2/4$. Then $f_i(S) = (m+1)^2/4$. Therefore $Pr[g_i(S) < f_i(S)] =$

$$Pr\{|S \cap \overline{S}_i| + \min\{|S \cap S_i|, (1+\delta)(m+1)/2\} < (m+1)^2/4\}$$

This probability can only increase when we remove elements from S . For the case when $|S| \leq (m+1)^2/4$, we get

$$\begin{aligned} & Pr[g_i(S) < f_i(S)] \\ &= Pr\{|S \cap \overline{S}_i| + \min\{|S \cap S_i|, (1+\delta)(m+1)/2\} < |S|\} \\ &= Pr\{\min\{|S \cap S_i|, (1+\delta)(m+1)/2\} < |S \cap S_i|\} \\ &= Pr\{|S \cap S_i| > (1+\delta)(m+1)/2\} \end{aligned}$$

Thus, this probability can only increase when more elements are added to S . Hence under the condition $S \subseteq G_i, |S| \leq (m+1)^2/4$, the probability is also maximized when $|S| = (m+1)^2/4$.

Now we assume $S \subseteq G_i$ and $|S| = (m+1)^2/4$. In this case, $Pr[g_i(S) < f_i(S)] = Pr\{|S \cap S_i| > (1+\delta)(m+1)/2\}$, which by a standard Chernoff bound arguments, can be shown to be bounded by $e^{-\Omega(m)}$. ■

If we define $\mathbf{f}(S) = (f_1(S), \dots, f_m(S))$ and $\mathbf{g}(S) = (g_1(S), \dots, g_m(S))$, then by a simple union bound, as a corollary of the lemma, we have $Pr[\mathbf{f}(S) \neq \mathbf{g}(S)] = poly(m)e^{-\Omega(m)}$. Now suppose \mathcal{A} is a deterministic algorithm which makes polynomially many queries to the value oracle. Then by the union bound, with probability at most $poly(m) \cdot e^{-\Omega(m)}$, \mathcal{A} can distinguish between \mathbf{f} and \mathbf{g} . Notice that for the cost function $\mathbf{g} = (g_1, \dots, g_m)$, the optimal solution is at most $(1+\delta)m(m+1)/2$ achieved by assigning S_i to agent i . However, as we showed, the optimal solution for the cost function $\mathbf{f} = (f_1, f_2, \dots, f_m)$ has cost at least $d(m+1)^2/4$, thus with high probability, \mathcal{A} can not approximate a CRA instance within factor $\frac{(m+1)^2 d/4}{(1+\delta)m(m+1)/2} \simeq d = c \log n$ for some $c < 1$.

At last, by Yao's principle, we get the theorem. □