View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Dagstuhl Research Online Publication Server

Combinatorial Problems with Discounted Price Functions in Multi-agent Systems

Gagan Goel[1], Pushkar Tripathi[1], and Lei Wang[1]

1 College of Computing
Georgia Institute of Technology
Atlanta, USA
gagan,pushkar.tripathi,lwang@cc.gatech.edu

Abstract

Motivated by economic thought, a recent research agenda has suggested the algorithmic study of combinatorial optimization problems under functions which satisfy the property of decreasing marginal cost. A natural first step to model such functions is to consider submodular functions. However, many fundamental problems have turned out to be extremely hard to approximate under general submodular functions, thus indicating the need for a systematic study of subclasses of submodular functions that are practically motivated and yield good approximation ratios. In this paper, we introduce and study an important subclass of submodular functions, which we call *discounted price functions*. These functions are succinctly representable and generalize linear(additive) price functions. We study the following fundamental combinatorial optimization problems: edge cover, spanning tree, perfect matching and $s-t$ path. We give both upper and lower bound for the approximability of these problems.

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2010.436

1 Introduction

In the algorithmic theory of combinatorial optimization, much of the attention has been focused on problems where one wishes to optimize an *additive* function under some combinatorial constraints. In these problems we are given a ground set E of elements and a collection Ω of subsets of the ground set ($\Omega \subseteq 2^E$) that is usually defined implicitly by some combinatorial property (such as the set of all spanning trees in a graph). We are also given a cost for each element in E and the price of a subset $S \in \Omega$ is defined to be the sum of costs of the elements in S. The objective is to find a subset in Ω with a minimum price.

However, additive price functions do not always model the complex dependencies of the prices in a real-world setting. It is widely believed in economics that price structure satisfies the decreasing marginal cost property. Intuitevly, it says that the price of adding an element to a larger set is less than adding it to a smaller set. Largely motivated by this, in recent years, a research agenda that has emerged out [9, 10, 11, 12, 20] is to study combinatorial optimization problems under *submodular* functions. Submodular functions form a very broad class of functions and mathematically capture the property of decreasing marginal cost. A function $f : 2^E \to R^+$ is said to be submodular if and only if for any two subsets S and $T \subseteq E$ such that $S \subseteq T$, the following holds: $f(T \cup i) - f(T) \leq f(S \cup i) - f(S)$ for all elements $i \in E - T$.

Unfortunately, many of the fundamental optimization problems have turned out to be extremely hard under submodular functions [9, 10, 11, 12, 20]. Thus, from a practical standpoint, the applicability of these results is not very well-founded as the class of submodular functions might be much more general than real-world functions. Moreover, the class of submodular functions is defined over an exponentially large domain and thus requires

© Gagan Goel, Pushkar Tripathi and Lei Wang;
licensed under Creative Commons License NC-ND
IARCS Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010).
Editors: Kamal Lodaya, Meena Mahajan; pp. 436–446
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

exponential time to write down the function explicitly. This may not be the case in real-world applications.

In this paper, we wish to explore functions that lie between the additive functions and the general submodular functions, and that are also succinctly representable. In particular, we study *discounted price functions* in which we are given an additive function c and a discount function $d : R^+ \to R^+$ that is a concave curve. The price of any subset S is defined to be $d(c(S))$. It is not difficult to see that dicounted price functions form a subclass of submodular functions. Discounted price functions have strong theoretical motivation as well. A common technique (due to [10]) for designing optimal algorithms under general submodular functions is to first approximate submodular functions by *ellipsoid* functions. These ellipsoid functions form a special class of discounted price functions.

We study discounted price functions in a multi-agent setting. This is motivated by the observation that often in a real-world scenario there are multiple agents, with different price functions, each of whom can build different parts of the required combinatorial structure. For instance, in the case of spanning tree, it might be more cost effective to buy only a subset of the edges of the final tree from a particular agent. For additive 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 price functions.

Discounted Price Model

We define a function $d : \mathrm{R}^+ \to \mathrm{R}^+$ to be a discounted price function if it satisfies the following properties: (1) $d(0) = 0$; (2) d is increasing; (3) $d(x) \leq x$ for all x; (4) d is concave.

We study combinatorial problems in the following general setting. We are given a set of elements E, and a collection Ω of its subsets. We are also given a set $\mathcal{A}$ of k agents where each agent $a \in \mathcal{A}$ specifies a cost function $c_a : E \to \mathrm{R}^+$ where $c_a(e)$ indicates her cost for the element e. Each agent also declares a discounted price function d_a. If an agent a is assigned a set T of elements, then her total price is specified by $d_a(\sum_{e \in T} c_a(e))$. This is called her *discounted price*. For the ease of notation we will use $d_a(T)$ to denote $d_a(\sum_{e \in T} c_a(e))$. The objective is to select a subset S from Ω and a partition $S_1, S_2, ..., S_k$ of S, such that $\sum_{a \in \mathcal{A}} d_a(S_a)$ is minimized.

We study the following four problems over an undirected graph $G = (V, E)$.

- **Discounted Edge Cover**: In this problem, Ω is chosen to be the collection of edge covers.
- **Discounted Spanning Tree**: In this problem, Ω is the collection of spanning trees of the graph.
- **Discounted Perfect Matching**: In this problem, we assume the graph has an even number of vertices. Ω is chosen to be the collection of perfect matchings of the graph.
- **Discounted $s - t$ Path**: In this problem, we are given two fixed vertices s and t. Ω consists of all paths connecting s and t.

Our Results

In section 2.1, we show that the discounted edge cover and spanning tree problems are hard to approximate within a factor of $(1 - o(1)) \log n$ unless P = NP. In section 2.2 and 2.3, we amplify this result to show that $s - t$ path and matching are hard to approximate within any polylog factor.

On the algorithmic front, in section 3.1 and 3.2, we show that our results are tight by giving $\log n$-approximate algorithms for the discounted edge cover and spanning tree

problems. In section 3.3, we describe simple $O(n)$-approximate algorithms for discounted $s-t$ and perfect matching. We leave the design of sublinear approximation algorithms for these two problems as an open question.

Related work

The classical versions of edge cover, spanning tree, perfect matching and $s-t$ path are well studied and polynomial time algorithms are known for all these problems, see [18, 14, 3, 7, 6, 4]. These have served as part of the most fundamental combinatorial optimization models in the development of computer science and operations research.

Recently, Svitkina and Fleischer [20] generalized the linear(additive) cost settings of some combinatorial optimization problems such as sparsest cut, load balancing and knapsack to a submodular cost setting. They gave $\sqrt{n/\log n}$ upper and lower bounds for all these problems. [11] also studied combinatorial optimization problems in the single agent setting.

Multi-agent setting was first introduced in [9]. In their model of *combinatorial problems with multi-agent submodular cost functions* they gave $\Omega(n)$ lower bounds for the problems of submodular spanning tree and perfect matching, and gave a $\Omega(n^{2/3})$ lower bound for submodular $s-t$ path where the submodular functions are given by value oracles. They also gave the matching upper bounds for all these problems. We remark that the lower bounds presented in [20, 9] are information theoretic and not computational.

Similar generalization from additive objective functions to the more general submodular functions was applied to maximization problems in [8, 15, 2]. The multi-agent generalization of maximization problems which corresponds to *combinatorial auction* has been extensively studied both in computer science and economics [5, 17] and tight information theoretic lower bounds are known for these problems, see [16].

2 Hardness of Approximation

In this section we present hardness of approximation results for the four problems defined earlier. Unlike some of the previous work on combinatorial optimization [9, 20] over non-linear cost functions, the bounds presented here are not information theoretic but are contingent on $\mathrm{P} \neq \mathrm{NP}$. In section 2.1, we show that all our problems are hard to approximate within factor $\log n$. In section 2.2 and 2.3, we amplify the hardness of approximation for discounted $s-t$ path and perfect matching to $O(\log^c n)$ for any constant c.

Recall that in each of the problems we are given a graph $G=(V,E)$ over n vertices. We are also given a set $\mathcal{A}$ of k agents each of whom specifies a cost $c_a : E \to \mathrm{R}^+$. Here $c_a(e)$ is the cost for building edge e for agent a. Each agent also specifies a discounted price function given by $d_a : \mathrm{R}^+ \to \mathrm{R}^+$. The objective is to build a specified combinatorial structure using the edges in E, and allocate these edges among the agents such that the sum of discounted prices for the agents is minimized.

2.1 Basic Reduction

To show the logarithmic hardness of approximation for the problems stated earlier we consider the following general problem and use a reduction from set cover to establish its hardness.

Discounted Reverse Auction: We are given a a set E of n items and a set $\mathcal{A}$ of agents each of whom specifies a function $c_a : E \to \mathrm{R}^+$. Here $c_a(e)$ is the cost for procuring item e from agent a. Each agent also specifies a discounted price function given by $d_a : \mathrm{R}^+ \to \mathrm{R}^+$.

The task is to find a partition $\mathcal{P} = \{P_1 \cdots P_k\}$ of E such that $\sum_{a \in \mathcal{A}} d_a(\sum_{e \in P_a} c_a(e))$ is minimized.

▶ **Lemma 1.** *It is hard to approximate the discounted reverse auction problem within factor* $(1 - o(1)) \log n$ *unless* $\mathrm{P} = \mathrm{NP}$.

Proof. We reduce set cover to the discounted reverse auction problem to prove this result. Consider an instance $\mathcal{I} = (U, C, w)$ of set cover where we wish to cover all elements in the universe U using sets from C and minimize the sum of weights under the weight function $w : C \to \mathrm{R}^+$. We define an instance, $\mathcal{I}'$ of our discounted reverse auction problem corresponding to $\mathcal{I}$ in the following way. Let U be the set of items. For every set $S \in C$ define an agent a_S, whose cost function c_a assigns the value $w(S)$ for every element $s \in S$ and sets the cost of all other elements in U to be infinity. The discounted price function for the agent is shown in figure 1. Here the slope of the second segment is small enough.

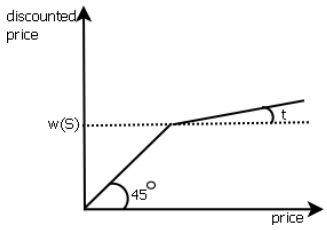

Figure 1 Discount function for agent corresponding to set S

Consider a solution for $\mathcal{I}'$ where we procure at least one item from agent a_S; then we can buy all elements in S from a_S without a significant increase in our payment. So the cost of the optimal solution to $\mathcal{I}$ can be as close to the price of the optimal solution for $\mathcal{I}'$ as we want. By [1, 19], set cover is hard to approximate beyond a factor of $\log n$ unless $\mathrm{P} = \mathrm{NP}$. Therefore the discounted reverse auction problem can not be approximated within factor $(1 - o(1)) \log n$ unless $\mathrm{P} = \mathrm{NP}$. ◀

This reduction can be extended to other combinatorial problems in this setting to give logarithmic hardness of approximation for many combinatorial problems. This can be achieved by considering an instance of the problem where we have just one combinatorial object and our task is to allocate it optimally among the agents. For example, for the discounted spanning tree problem we consider the instance when the input graph is itself a tree and we have to optimally allocate its edges among the agents to minimize the total price. Thus, we have the following:

▶ **Theorem 2.** *It is hard to approximate discounted edge cover, spanning tree, perfect matching and* $s - t$ *path within factor of* $(1 - o(1)) \log n$ *on a graph with* n *vertices unless* $\mathrm{P} = \mathrm{NP}$.

2.2 Amplification: Hardness for Discounted $s - t$ Path

In this section we consider the discounted $s - t$ path problem between two given vertices s and t. We show that unless $\mathrm{P} = \mathrm{NP}$, this problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant c. The proof is based on amplification of the result of theorem 2. This is done by repeatedly applying a transformation σ on the given family of problem instances, which amplifies the approximation factor on every application. Each

application of σ also increases the size of the graph but only by a polynomial(in n) factor. We now describe the transformation formally.

Consider the following instance $(G, \mathcal{A}, \mathcal{U})$: We are given a graph $G = (V, E)$, vertices s,t and a set $\mathcal{A}$ of agents. We are also given a collection $\mathcal{U} = \{U_a\}_{a \in \mathcal{A}}$. Here $U_a \subseteq E$ specifies the set of edges that can be assigned to agent a, i.e., $c_a(e) = 1$ for all $e \in U_a$ and $c_a(e) = +\infty$ otherwise. The discounted price function d_a is such that $d_a(x) = x$ for all $x \leq 1$ and 1 for all $1 < x < +\infty$. Observe that under this assumption, for any set S of edges, $d_a(S)$ has value 1 if $S \subseteq U_a$ and ∞ otherwise. We may assume that the sets U_a for $a \in \mathcal{A}$ are pairwise disjoint, by replacing a single edge that can be assigned to multiple agents by parallel edges and assigning them to each of the agents. In future discussion, we will use $\mathcal{F}$ to denote the family of instances $\{(G, \mathcal{A}, \mathcal{U})\}$.

We define the transformation $\sigma : \mathcal{F} \to \mathcal{F}$ that takes an instance $I = (G, \mathcal{A}, \mathcal{U})$ in $\mathcal{F}$, and generates another instance $I^{\otimes} = (G^{\otimes}, \mathcal{A} \times \mathcal{A}, \mathcal{U}^{\otimes})$ as follows. The graph $G^{\otimes} = (V^{\otimes}, E^{\otimes})$ is constructed from G by replacing each edge $(u, v) \in E$ with a copy of the graph G such that s coincides with u and t coincides with v. Thus any edge $e \in E$ can be identified with a subgraph G^e, of $G^{\otimes}$ that is isomorphic to G. Each $e' \in G^e$ has a cost $c_a(e)$ for each agent a and we define $\rho(e') = e$ and define $\gamma(e')$ to be the edge corresponding to e' in G under this isomorphism. There are $|\mathcal{A}|^2$ agents in the new instance who are indexed by $\mathcal{A} \times \mathcal{A}$. We define the elements of $\mathcal{U}^{\otimes}$ as $U^{\otimes}_{(a_1,a_2)} = \{e' \mid \rho(e') \in U_{a_1} \text{ and } \gamma(e') \in U_{a_2}\}$.

Note that $|E^{\otimes}| = |E|^2$, i.e. the size of instance $I^{\otimes}$ is bounded by a polynomial in the size of I. We define $\sigma(\mathcal{F}) = \{\sigma(I) \mid \forall I \in \mathcal{F}\}$. In lemma 3, we show that we can amplify that hardness result from theorem 2 by applying the transformation σ repeatedly.

▶ **Lemma 3.** *If $\mathcal{H} = \sigma^r(\mathcal{F})$ is a family of instances for the $s - t$ path problem that is hard to approximate to a factor better than α, then $\sigma(\mathcal{H})$ is hard to approximate within a factor $O(\alpha^2)$.*

Proof. Let $I = (G, s, t, \mathcal{A}, \mathcal{U})$ be an instance in $\mathcal{H}$. Let us begin by making some observations about the structure of an optimal solution for $\sigma(I) = (G^{\otimes}, \mathcal{A}, \mathcal{U}^{\otimes})$.

▶ Claim 1. If there is a $s - t$ path of price β in G, then there is a $s - t$ path in $G^{\otimes}$ of price at most β^2.

Proof of claim 1. Let $P = e_1, e_2 \ldots e_t$ be a path of price β in G. We can construct a $s - t$ path in $G^{\otimes}$ by considering the set of graphs $G^{e_1} \ldots G^{e_t}$ and picking the edges corresponding to the edges in P in each of these copies. It can be verified that this gives us a valid path that has price β^2. ◀

Next we note that the converse is also true.

▶ Claim 2. If there is a $s - t$ path of price β^2 in $G^{\otimes}$ then there is a $s - t$ path in G of price at most β.

Proof of claim 2. Let P be a path of price β^2 in $G^{\otimes}$. Let $G^{e_1} \ldots G^{e_t}$ be the copies of G that have non-empty intersection with P. Two cases may arise. Either the set of edges $\{e_1 \ldots e_t\}$ belong to at most β distinct agents in $\mathcal{A}$ or they belong to more than β agents in $\mathcal{A}$. Note that the set of edges $\{e_1 \ldots e_t\}$ form a path in G, and in the first case this path has price at most β. In the second case, the price of edges in $P \cap G^{e_i}$ must be less than β for some copy G^{e_i} of graph G. These edges also form a $s - t$ path in G of price at most β. Thus in both cases we can find a path of price at most β in G. ◀

Using the observations above, if the price of the optimal solution to I is OPT, then the price of the optimal solution to $\sigma(I)$ is OPT^2. Furthermore, if we can approximate the optimal solution to $\sigma(I)$ to within a factor of $o(\alpha^2)$ then we can approximate the optimal solution for I to better than $o(\alpha)$, using the construction in claim 2. This yields the desired contradiction. ◀

By theorem 2, $\mathcal{F}$ is hard to approximate within a factor of $\log n$. Using this as the base case and applying lemma 3 repeatedly we have the following theorem.

▶ **Theorem 4.** *The discounted shortest $s-t$ path problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant $c > 0$.*

2.3 Reduction: Hardness for Discounted Perfect Matching

In this section we consider the discounted perfect matching problem. We show that unless P = NP, this problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant c. The proof is based on a factor preserving reduction from the $s-t$ path problem. We now describe our reduction:

▶ **Lemma 5.** *Let A be a β-approximate algorithm for the perfect matching problem, then we can get a β-approximation for the $s-t$ path problem using A as a subroutine.*

Proof. Suppose we are given a graph $G = (V, E)$. Construct an auxiliary graph G^* in the following way: Replace every vertex $v \in V$ by v' and v'' and add an edge connecting them. The price of this edge is zero for every agent. We replace each edge $uv \in E$ with the gadgets shown in figure 2.

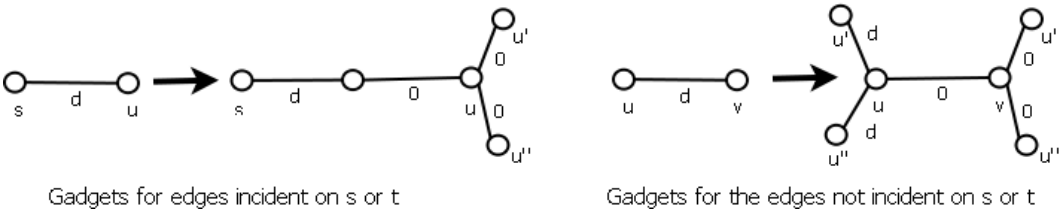

Figure 2 Gadgets

On this graph G^*, use the algorithm A to get the minimum weight matching. Let M be the matching returned. We can interpret M as a $s-t$ path in G in the following way. Let $g(uv)$ be the edges in G^* corresponding to the edge uv for the gadget shown in figure 2. Observe that either one or two edges of every such gadget must belong to M. Let S be the set of edges in G such that two edges in their corresponding gadget belong to M. One can check that every vertex in V is incident with zero or two edges from S, whereas s and t are each incident with exactly one edge in S. Therefore S consists of an $s-t$ path P_S and some other circuits. Now the circuits in S must have cost zero. This is because if a circuit has positive cost then the cost of the matching can be reduced further by pairing up the vertices in the circuit as shown in figure 3.

◀

Note that the reduction defined in lemma 5 defines a cost preserving bijection between $s-t$ paths in G to perfect matchings in G^*. Thus, using theorem 4 we have :

▶ **Theorem 6.** *The discounted perfect matching problem is hard to approximate within a factor of $O(\log^c n)$ for any fixed constant $c > 0$.*

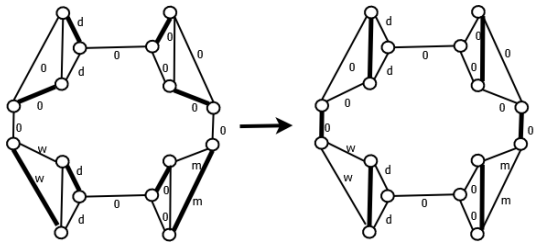

Figure 3 Circuits not involving edges in S should have zero costs

3 Algorithms for Discounted Combinatorial Optimization

In this section, we present approximation algorithms for the four problems defined earlier.

3.1 Discounted Edge Cover

We will establish a factor $O(\log n)$ algorithm for the discounted edge cover problem.

Given a discounted edge cover instance, we construct a set cover instance such that: 1) An optimal edge cover corresponds to a set cover with the same cost and 2) A set cover corresponds to an edge cover with a smaller price. For the set cover instance, we apply the greedy algorithm from [13] to get a set cover whose cost is within $\log n$ of the optimal cost. The corresponding edge cover gives a $\log n$ approximation of the optimum edge cover. We remark that we will have exponentially many sets in the set cover instance that we construct for our problem. To apply the greedy algorithm, we need to show that in each step, the set with the lowest *average cost* can be found in polynomial time.

Now we state our algorithm formally. Consider a set cover instance where we have to cover the set of vertices, V, with $k2^n$ subsets which are indexed by $(a, S) \in \mathcal{A} \times 2^V$. The cost of the set (a, S), denoted by $cost(a, S)$, is defined as the minimum discounted price of an edge cover for the vertices in S that can be built by agent a. For the instance of set cover described above, we apply the greedy algorithm [13] to get a set cover $\mathcal{S}$. Let U_a be the set of vertices covered by sets of the form $(a, S) \in \mathcal{S}$. Let C_a be agent a's optimal discounted edge cover of the vertices in U_a. We output $\{C_a : a \in \mathcal{A}\}$ as our solution.

The correctness of the algorithm follows from the observation that each C_a is a cover of U_a thus their union must form an edge cover for V.

Now we show that the running time of the algorithm is polynomial in k and n. Given U_a, C_a can be found in polynomial time. Thus our algorithm can be implemented in polynomial time, if we can implement the greedy algorithm on our set cover instance efficiently.

Recall that the greedy algorithm from [13] covers the ground set iteratively. Let Q be the set of covered elements at the beginning of a phase. The *average cost* of a set (a, S) is defined as $\alpha_a(S) = cost(a, S)/|S - Q|$. In every iteration the algorithm picks the set with the smallest average cost until all the vertices are covered. To show that this algorithm can be implemented efficiently, we only need to show the following lemma.

▶ **Lemma 7.** *For any $Q \subset V$, we can find $min\{cost(a, S)/|S - Q| : (a, S) \in \mathcal{A} \times 2^V\}$ in polynomial time.*

Proof. We can iterate over all choices of agent $a \in \mathcal{A}$, thus the problem boils down to finding $\min\{cost(a, S)/|S - Q| : S \subseteq V\}$ for each $a \in \mathcal{A}$.

For each integer d, if we can find $\min\{cost(a, S) : |S - Q| = d\}$ in polynomial time, then we are done since then we can just search over all the possible sizes of $S - Q$. Unfortunately,

it is NP-hard to compute $\min\{cost(a, S) : |S - Q| = d\}$ for all integer d. We will use claim 3 to circumvent this problem.

▶ Claim 3. For any graph $G = (V, E)$ and $Q \subseteq V$ and for any positive integer d, we can find the set (a, S) minimizing $cost(a, S)$ such that $|S - Q|$ is at least d, in polynomial time.

Proof. To find the desired set we construct a graph $G' = (V', E')$ as follows: Add a set $X \cup Y$ to the set of vertices in G, where $|X| = |Q|$ and $|Y| = |V| - |Q| - d$. Match every vertex in X to a vertex in Q with an edge of cost 0. Connect each vertex in Y to each vertex in V by an edge of very large cost. Set the cost of each edge $e \in E$ as $c_a(e)$. Find the minimum cost edge cover in G'. Let S^* be the set of vertices not adjacent to $X \cup Y$ in such a cover. It is easy to verify that S^* is the desired set. ◀

By claim 3 above we can generate a collection of subsets $\{S_i \subseteq V : 1 \leq i \leq n\}$, such that (a, S_i) has the lowest value of $cost(a, S)$ among all sets S which satisfy $|S - Q| \geq i$.

▶ Claim 4. $\min\{cost(a, S)/|S - Q| : S \subseteq V\} = min\{cost(a, S_i)/|S_i - Q| : 1 \leq i \leq n\}$.

Proof. Let S^* be the set that has the minimum average cost with respect to agent a. Suppose $|S^* - Q| = d$. By our choice of S_d, we have $|S_d - Q| \geq d = |S^* - Q|$ and $cost(a, S_d) \leq cost(a, S^*)$. Therefore we have $cost(a, S_d)/|S_d - Q| \leq cost(a, S^*)/|S^* - Q|$, hence they must be equal. ◀

By iterating over all $a \in \mathcal{A}$, we can find $\min\{cost(a, S)/|S - Q| : (a, S) \in \mathcal{A} \times 2^V\}$ in polynomial time. ◀

Next we show that the approximation factor of our algorithm is $\log n$. Let OPT_{EC} and OPT_S denote the costs of the optimal solutions for the discounted edge cover instance and the corresponding set cover instance respectively. Let $\{C_a : a \in \mathcal{A}\}$ be the edge cover reported by our algorithm. For all $a \in \mathcal{A}$ let O_a be the set of vertices covered by agent a in the optimal edge cover. The sets $\{(a, O^a) : a \in \mathcal{A}\}$ form a solution for the set cover instance. Therefore $OPT_S \leq OPT_{EC}$.

Since we use the greedy set cover algorithm to approximate OPT_S, we have

$$\sum_{a \in \mathcal{A}} d_a(C_a) \leq (\log n) OPT_S \leq (\log n) OPT_{EC}$$

Thus we have the following theorem.

▶ **Theorem 8.** *There is a polynomial time algorithm which finds a $O(\log n)$-approximate solution to the discounted edge cover problem for any graph over n vertices.*

3.2 Discounted Spanning Tree

In this section, we study the discounted spanning tree problem and establish an $O(\log n)$ approximation algorithm for this problem.

Let us first consider a simple $O(\log^2 n)$-approximation algorithm. Observe that a spanning tree is an edge cover with the connectivity requirement. If we apply the greedy edge cover algorithm in section 3.1, there is no guarantee that we will end up with a connected edge cover. We may get a collection of connected components. We can subsequently contract these components and run the greedy edge cover algorithm again on the contracted graph. We repeat this until there is only one connected component. By this method, we will get a connected edge cover containing a spanning tree.

Now we will analyze the above algorithm. Let OPT_{ST} be the price of the minimum discounted spanning tree. After each execution of the greedy edge cover algorithm, there is no isolated vertex, hence the contraction decreases the number of vertices by at least a a factor of half, therefore we will have to run the greedy edge cover algorithm at most $O(\log n)$ times. Let OPT^r_{EC} be the price of the minimum edge cover for the graph obtained after the rth contraction and let C_r be the edge cover that we produce for this iteration. Using theorem 8, the price of C_r is at most $(\log n)OPT^r_{EC}$. It is easy to see that OPT^r_{EC} is at most OPT_{ST} for every r. Hence the price of C_r is bounded by $\log n \cdot OPT_{ST}$. Since there are at most $O(\log n)$ iterations, the price of the spanning tree produced by the above algorithm is bounded by $O(\log^2 n)OPT_{ST}$.

We observe that two main steps in the above algorithm are greedy edge cover and contraction. Intuitively, they are used to satisfy the covering and connectivity requirements respectively. The algorithm proceeds by alternately invoking these subroutines. Based on this observation, our idea to get an $O(\log n)$ approximation algorithm is to apply the following greedy algorithm: rather than apply contraction after each complete execution of the greedy edge cover, we interleave contraction with the iterations of the greedy edge cover algorithm. After each iteration, we modify the graph to coerce our algorithm to get a connected edge cover at the end.

Now we describe our greedy algorithm. For every agent a and subset of vertices S we define $cost(a, S)$ as the cost of the optimal edge cover for S. We define the *average cost* of a set (a, S) as $\alpha^S_a = cost(a, S)/|S|$. The algorithm proceeds in phases and each phase has two steps, **search** and **contraction**. In the rth phase, during the search step we find the set (a_r, S_r) with the lowest average cost and set the *potential* of each vertex $v \in S_r$ as $p(v) = \alpha^{S_r}_{a_r}$. The search step is followed by a contraction step, where we modify the graph by contracting every connected component in the induced subgraph of agent a_r's optimal edge cover for the set S_r. After this we begin the next phase. The algorithm terminates when we have contracted the original graph to a single vertex. For every agent, we find the set of all edges assigned to her across all the search steps declare this as her bundle of assigned edges. Finally remove unnecessary edges from the set of assigned edges to get a spanning tree.

It is easy to see that we get a connected edge cover at the end of the algorithm, which proves the correctness of the algorithm. To analyze the running time, we observe that there can be at most n phases and by Lemma 7, each phase can be implemented in polynomial time. Hence the algorithm runs in polynomial time.

Next, we prove that the approximation factor of the algorithm is $O(\log n)$. Let OPT_{ST} be the price of the optimal solution of the discounted spanning tree instance and let $\{T_a : a \in \mathcal{A}\}$ be our solution. Let V' be the set of contracted vertices we produced during the algorithm. Number the elements of V and V' in the order in which they were covered by the algorithm, resolving ties arbitrarily. Suppose $V = \{v_1, ..., v_n\}$ and $V' = \{z_1, ..., z_{n'}\}$. Obviously, $n' \leq n$.

It is easy to verify that $\sum_a d_a(T_a) \leq \sum_i p(v_i) + \sum_j p(z_j)$. Therefore we only need to bound the potentials of the vertices in $V \cup V'$.

▶ Claim 5. $p(v_i) \leq \frac{OPT_{ST}}{n-i+1}$ for any $i \in \{1 \cdots n\}$ and $p(z_j) \leq \frac{OPT_{ST}}{n'-j}$ for any $j \in \{1 \cdots n'\}$.

Proof. For $i \in \{1 \cdots n\}$, suppose v_i is covered in phase r. Let G^r be the underlying graph at the beginning of phase r. Since $v_i, v_{i+1}, ..., v_n$ are not covered before phase r, G^r contains at least $n - i + 1$ vertices. Since the optimal spanning tree can cover the vertices in G^r by a price of OPT_{ST}, by our greedy choice, $p(v_i) \leq OPT_{ST}/(n - i + 1)$.

Similarly, let $1 \leq j \leq n'$ and assume z_j is covered in phase r. Since we should be able to produce $z_{j+1}, z_{j+2}, ..., z_{n'}$ from contraction on vertices of G^r, there are at least $n' - r$ vertices in G^r. Therefore we have $p(z_j) \leq OPT_{ST}/(n' - j)$. ◀

From the above claim, we have

$$\sum_a d_a(T_a) \leq \sum_{1\leq i\leq n} \frac{OPT_{ST}}{n-i+1} + \sum_{1\leq j\leq n'} \frac{OPT_{ST}}{n'-j} \leq (\log n + \log n')OPT_{ST} \leq O(\log n)OPT_{ST}$$

Therefore we have the following theorem:

▶ **Theorem 9.** *There is a polynomial time algorithm which finds an $O(\log n)$-approximate solution to the discounted spanning tree problem for any graph with n vertices.*

3.3 Discounted $s-t$ Path and Perfect Matching

In sections 2.2 and 2.3 we showed that unlike edge cover and spanning tree, no polylog-approximate algorithm is likely to exist for discounted $s-t$ path and perfect matching.

Now we describe a simple n-approximate algorithm for discounted $s-t$ path problem. For each edge e, define $w_e = \min_{a\in\mathcal{A}} d_a(c_a(e))$ and for each $s-t$ path P, define its weight $w(P) = \sum_{e\in P} w_e$. Use Dijkstra's algorithm to find a path P_0 with the minimum weight and output it as the solution. Allocate the edges in P_0 as follows: for each edge $e \in P$, we allocate e to the agent a such that $d_a(c_a(e)) = w_e$, with ties broken arbitrarily.

For the analysis, let us define S_a to be the set of edges allocated to agent a in our solution. Since d_a is concave, we have $d_a(S_a) \leq \sum_{e\in S_a} d_a(c_a(e))$. Therefore, the total price of our solution is bounded by $\sum_{a\in\mathcal{A}}\sum_{e\in S_a} d_a(c_a(e))$ which is exactly $w(P_0)$. Let OPT be the path chosen in the optimal solution (as an abuse of notation, we also use OPT to denote the optimal value) and OPT_a be the set of edges on the path allocated to agent a. By our choice of P_0 and weight w, we have $w(P_0) \leq w(OPT) = \sum_{a\in\mathcal{A}}\sum_{e\in OPT_a} w_e \leq \sum_{a\in\mathcal{A}}\sum_{e\in OPT_a} d_a(c_a(e))$. Since d_a is increasing, we have

$$\sum_{e\in OPT_a} d_a(c_a(e)) \leq |OPT_a| \cdot d_a(OPT_a) \leq n \cdot d_a(OPT_a).$$

Therefore $w(OPT) \leq n\sum_{a\in\mathcal{A}} d_a(OPT_a) = n \cdot OPT$. This implies that our algorithm is an n-approximate algorithm.

We apply the same idea for discounted perfect matching problem. Define the weight of a perfect matching M as $w(M) = \sum_{e\in M} w_e$. Use Edmond's algorithm to find a minimum weight perfect matching M_0 for this weight function. For every $e \in M_0$, allocate it to the agent a such that $c_a(e) = w_e$. By a similar argument as above, we can show that this is a n approximate algorithm.

Acknowledgements

We would like to thank Vijay Vazirani for is continued guidance and support throughout the development of this paper. Also, we would like to thank the reviewers for many useful suggestions.

References

1 Noga Alon, Dana Moshkovitz, and Shmuel Safra. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms*, 2(2):153–177, 2006.

2 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *IPCO '07: Proceedings of the 12th international conference on Integer Programming and Combinatorial Optimization*, pages 182–196, Berlin, Heidelberg, 2007. Springer-Verlag.

3 William Cook and Andre Rohe. Computing minimum-weight perfect matchings. *INFORMS J. on Computing*, 11(2):138–148, 1999.

4 Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2001.

5 Sven de Vries and Rakesh V. Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, (3):284–309, 2003.

6 Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

7 Jack Edmonds. Paths,trees and flowers. *Canad J.Math*, 17, 1965.

8 Uriel Feige and Vahab S. Mirrokni. Maximizing non-monotone submodular functions. In *In Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS*, page 2007, 2007.

9 Gagan Goel, Chinmay Karande, Pushkar Tripathi, and Lei Wang. Approximability of combinatorial problems with multi-agent submodular cost functions. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

10 M.X. Goemans, N.J.A. Harvey, S. Iwata, and V. Mirrokni. Approximating submodular functions everywhere. In *SODA '09: Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.

11 Satoru Iwata and Kiyohito Nagano. Submodular function minimization under covering constraints. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009.

12 Stefanie Jegelka and Jeff Bilmes. Cooperative cuts: Graph cuts with submodular edge weights. *Technical Report*, 2010.

13 David S. Johnson. Approximation algorithms for combinatorial problems. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49, New York, NY, USA, 1973. ACM.

14 Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, February 1956.

15 Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 323–332, New York, NY, USA, 2009. ACM.

16 Vahab Mirrokni, Michael Schapira, and Jan Vondrak. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *EC '08: Proceedings of the 9th ACM conference on Electronic commerce*, pages 70–77, New York, NY, USA, 2008. ACM.

17 Noam Nisam, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, September 2007.

18 Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, 2002.

19 Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484, New York, NY, USA, 1997. ACM.

20 Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 697–706, Washington, DC, USA, 2008. IEEE Computer Society.