Revenue Maximization with Nonexcludable Goods

MohammadHossein Bateni[1], Nima Haghpanah[2*], Balasubramanian Sivan[3], and Morteza Zadimoghaddam[4]

[1] Google Research, New York bateni@google.com
[2] Northwestern University nima.haghpanah@gmail.com
[3] Microsoft Research bsivan@microsoft.com
[4] Massachusetts Institute of Technology morteza@mit.edu

Abstract. We study the design of revenue maximizing mechanisms for selling nonexcludable public goods. In particular, we study revenue maximizing mechanisms in Bayesian settings for facility location problems on graphs where no agent can be excluded from using a facility that has been constructed. We show that the optimization problem involved in implementing the revenue optimal mechanism is hard to approximate within a factor of $\Omega(n^{2-\epsilon})$ (assuming $P \neq NP$) even in star graphs, and that even in expectation over the valuation profiles, the problem is APX-hard. However, in a relevant special case we construct polynomial time truthful mechanisms that approximate the optimal expected revenue within a constant factor. We also study the effect of partially mitigating nonexcludability by collecting tolls for using the facilities. We show that such "posted-price" mechanisms obtain significantly higher revenue, and often approach the optimal revenue obtainable with full excludability.

Keywords: Revenue maximization, nonexcludable goods, hardness of approximation, incentive compatibility

1 Introduction

How should a seller maximize his revenue while selling nonexcludable services? As a representative example, consider the following facility location problem: firms in an industrial town want to connect their respective warehouses to their outlets with good road links. But the seller (in this case the construction company that lays the roads) cannot enforce exclusivity on the roads it lays: once a firm buys a connection between its warehouse and outlet, the seller cannot exclude others from using those road links. We study the following two questions in this work. Which potential sites should the seller choose for laying roads to maximize his own revenue in such nonexcludable settings? If the seller could enforce partial excludability by collecting tolls for roads, where only those firms that pay the toll for a road can use it, how much more revenue can he earn?[1]

[*] Part of the work was done while the second and fourth coauthors were visiting Google Research.

[1] Note that collecting tolls only enforces partial excludability because *every* firm that pays the toll is allowed to use the road, and cannot be excluded.

The Mechanism Design Problem. We model the problem with a graph where every unordered pair of vertices denotes an agent and every edge denotes the *possibility* of a road to be laid between two vertices. We assume that each agent/firm has a nonnegative private value for getting her pair of vertices connected, drawn independently from a known distribution. We aim for the design of the optimal mechanism which, given the value reports of all the agents, selects a set of edges to build roads on and computes payments to be made by agents so that the expected revenue is maximized (expectation over the distribution of private values) in equilibrium. We invoke Myerson's [14] characterization which states that the expected revenue of any mechanism in a Bayesian Nash Equilibrium (BNE) is equal to the expected *virtual surplus* of the agents *served*, i.e., the sum of virtual values[2] of those agents whose warehouse and outlet are connected in the solution[3]. This reduces the problem of computing the expected revenue maximizing mechanism to a *pointwise* optimization problem: given a profile of values of all agents, compute the set of edges to build roads on so that the sum of the virtual values of the agents served is maximized.

The pointwise optimization problem to maximize virtual values is an interesting graph theoretic problem, which to our knowledge has not been studied before. Given a graph with a weight on every pair of vertices (the weights represent virtual values, and thus, could be negative), design an algorithm that selects a subset of edges that maximizes the sum of the weights of every vertex pair whose endpoints are connected by the edges selected. If the weights were all positive, clearly the optimal choice for the seller would be to pick all the edges in the graph. However, weights could be negative. In particular, while aiming to connect some subset of agents (i.e., vertex pairs), another subset of agents with negative weights automatically get connected, and these negative weighted pairs cannot be excluded due to nonexcludability. Deciding on the the set of edges to select for maximizing the sum of weights of the vertex pairs whose endpoints are connected is the nontrivial underlying graph theoretic problem.

The Rooted and Unrestricted Versions. We study two versions of the graph theoretic problem described above. In the unrestricted version, every (unordered) pair of vertices in the graph is an agent. In the rooted version (which is a special case of the unrestricted version), a single vertex is designated as root, and only those unordered pairs having root as one of their vertices are agents. This corresponds to the root being a central location where all warehouses are located, and other nodes being outlet locations.

Results for Unrestricted Version (Section 3). The natural place to start is to solve the pointwise optimization problem in graphs presented above. Clearly, any α-approximate monotone[4] solution to the pointwise optimization problem implies an α-approximate

[2] The virtual value of an agent is a function of her value and the distribution from which her value is drawn; it can be negative. When the virtual value function is not monotone, Myerson [14] applies a fix by considering the ironed virtual value function (see section 2 for more details), and all our results go through with virtual values replaced by ironed virtual values.

[3] Note that Myerson's mechanism remains revenue optimal in all single-parameter settings regardless of whether or not the good is excludable.

[4] In single-parameter settings, a mechanism's allocation is monotone if fixing the values of other agents and agent i alone reporting a higher value results in agent i getting served with no

truthful mechanism to the problem of expected revenue maximization. However, we show that except for very special cases, pointwise optimization is not possible unless $P = NP$. In particular, we show that the pointwise problem is NP-hard to approximate to within a factor $\Omega(n^{2-\epsilon})$ even on star graphs. For the special case where the underlying graph is a path, we give a dynamic program to solve the pointwise optimization. Thus, for paths, we get the expected optimal revenue with nonexcludability for any product distribution over agents' values.

Results for Rooted Version (Section 4). Our results for the rooted version are threefold.

1. First, we give a simple polynomial time algorithm for pointwise optimization in trees—this contrasts with our result that for the unrestricted version the pointwise problem is hard to approximate beyond a factor of $\Omega(n^{2-\epsilon})$ even for star graphs.
2. Second, as our main result, we give a polynomial time algorithm for optimization in expectation (over the distribution of values, and hence virtual values) for arbitrary graphs, when the agents' valuations are drawn i.i.d.
3. Third, we establish APX-hardness of optimization in expectation for arbitrary graphs, when the valuations of all agents are independent but not necessarily identical.

Our main result, which is the second point above, is that we design an algorithm that guarantees a constant factor approximation to the optimal virtual surplus in expectation. To this end, we extensively use the key (yet simple) property that even though virtual values can be negative, the expected virtual value of an agent is nonnegative. This suggests the following high-level approach to the problem. Partition agents into a constant number of sets. Pick a target set at random, and run an algorithm that is a constant factor approximation for the agents of that set (ignoring other agents). The nonnegativity of the expected virtual value implies that the contribution from nontargeted sets is nonnegative. Since each set is targeted with constant probability, this implies a constant factor approximation. We use this approach to solve the abstract edge-weighted version of the problem (in which edges are agents who derive value from being connected to the root), and then reducing the original problem to the edge-weighted version. In particular, we present an algorithm that partitions the edges of any graph into two sets that, loosely speaking, correspond to edges in well-connected parts of the graph and edges in sparse cuts of the graph. We show that once we contract the edges in a set (that corresponds to ignoring their value in the above high level approach), the remaining graph has a nice structure that allows for constant approximations.

Partial Excludability via Pricing (Section 5). If the seller were allowed to set prices on edges (i.e., collect tolls), can he get close to the optimal revenue when excludability is allowed? Typically the optimal revenue with full excludability is much higher than the optimal revenue with nonexcludability, and it is worthwhile for a profit maximizing firm to explore this option. In this problem the seller first has to decide which edges to pick to lay roads on, and decide how to price roads to maximize revenue. We construct a pricing

smaller probability. Monotone allocation is necessary and sufficient for truthfulness, i.e., they alone lend themselves to truthful payments.

scheme so that when the value distributions are i.i.d., the seller obtains (for the rooted version) the optimal revenue possible when full excludability is allowed. Further, if the distributions are independent but not identical, and satisfy a technical MHR condition[5], we show how to price edges to obtain a $\frac{1}{\log n}$ fraction of the optimal revenue possible when full excludability is allowed (again for the rooted version). An implication of these results is that mitigating externalities via tolls is much more remunerative than aiming for the optimal solution with nonexcludability.

Related Work. Approximating the expected version of a problem that is hard-to-approximate in the worst case, via exploiting the fact that the expected virtual value for any distribution is nonnegative is a relatively new idea. To our knowledge, it has been used only in Haghpanah et al. [10]. Auctions with externality (a notion related to nonexcludability where an agent's utility doesn't just depend on the services he received but also on the outcomes for the other agents) have been studied in multiple flavors before. Settings with positive externality include increased value for having a telephone or a music player or a new technology if more of your neighbors have them [15, 11]. A prime and well-studied example for a setting with negative externality is the sale of contiguous ad slots to two competing businesses [1, 4, 8, 9, 12, 13]. Equilibria which are surprising at first sight, like no allocation equilibria can result in large revenue for the auctioneer in settings with negative externality [7]. Posted pricings have often been a mechanism of choice for settings with externalities [3, 2, 5, 11]. The main difference between these and the posted prices studied in our work (apart from the presence of nonexcludability in our settings) is that these works allow agent-specific prices, whereas our setting is more constrained: we place prices on edges that are common for all agents.

2 Model and Notation

Unrestricted Version. We consider a universe of n potential sites, located on the vertices of a graph, $G = (V, E)$, with m undirected edges. An undirected edge $(i, j) \in E$ means that a link/road connecting sites i and j is allowed (but need not necessarily be constructed). An agent is an unordered pair of sites (i, j), interested in having some path constructed between i and j. Thus, there could be up to $\binom{n}{2}$ agents in a mechanism. An instance $\mathcal{I} = (G, A, F)$ of the problem consists of an undirected graph G, a set A of agents (represented by a set of pairs of vertices), and a distribution F_i associated with agent i (distributions are explained below). Sometimes we use i to denote a single site and sometimes to denote an agent, who is actually a pair of vertices. The context will make it clear whether i is a single site or a pair of sites.

Rooted version. The rooted version is a special case of the setting described above. A special vertex r is designated as the root. Only vertex pairs of the form (i, r) are agents, i.e., the root r is one of the end points of the path desired for every agent.

[5] Roughly speaking, this means that the tail of the distribution is no heavier than the exponential distribution. Many natural classes of distributions like the uniform, exponential, Gaussian (Normal) distributions satisfy the MHR condition. See Section 2 for a formal definition.

An outcome $o \in \Omega = \{0,1\}^m$ is the set of edges constructed. Agent i has a valuation function $v_i : \Omega \to \mathbf{R}^+ \cup \{0\}$, which maps outcomes to nonnegative real numbers. We study mechanisms in the single-parameter setting where the function $v_i(\cdot)$ takes only two values: v_i and zero. An agent i has a nonzero value v_i if and only if the set of edges selected contains a path between the two sites she represents, and in this case we say that agent i was served. Let $\mathcal{S}$ denote the set of all feasible sets of agents, i.e., the set of all sets of agents that can be simultaneously served. Note that this set system $\mathcal{S}$, is not downward closed: $S \in \mathcal{S}$ does not necessarily mean that $S' \in \mathcal{S}$ for all $S' \subset S$. Clearly, the non-downward closedness stems from the inability to exclude agents from using the roads.

We study mechanisms for this problem in a Bayesian setting, i.e., for every i, the single parameter v_i is assumed to be drawn independently from a publicly known distribution function F_i. Thus $F = F_1 \times F_2 \times \ldots F_n$ denotes the product distribution from which the vector of types $\mathbf{v}$ is drawn. The mechanisms in this paper assume the availability of any expectation defined with respect to F.

A direct revelation mechanism or an auction solicits sealed bids $(b_1, b_2, \ldots, b_n)$ from all the agents, and determines the outcome $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and payments $\mathbf{p} = (p_1, p_2, \ldots, p_n)$. Each agent i is risk-neutral and has a linear utility $u_i(\mathbf{b}) = v_i \cdot x_i(\mathbf{b}) - p_i(\mathbf{b})$.

Let $(\mathbf{x}(\cdot), \mathbf{p}(\cdot))$ denote a mechanism. When agent i is bidding in the auction, she knows only her own value v_i. A mechanism $(\mathbf{x}(\cdot), \mathbf{p}(\cdot))$ is incentive compatible (IC) if $v_i x_i(v_i, v_{-i}) - p_i(v_i, v_{-i}) \geq v_i x_i(z, v_{-i}) - p_i(z, v_{-i})$ for all i, z. A necessary and sufficient condition on $\mathbf{x}(\cdot)$ for IC payments to exist is monotonicity: for all $i, v_i, {v_i}' \geq v_i, v_{-i}$, we have $x_i(v_i, v_{-i}) \leq x_i({v_i}', v_{-i})$. Since we focus on the class of IC mechanisms, we have $\mathbf{b} = \mathbf{v}$.

Optimal auctions. To solve for optimal auctions, Myerson [14] defined *virtual valuations* for agents as $\phi_i(v_i) = v_i - \frac{1-F_i(v_i)}{f_i(v_i)}$ and proved that the expected payment of an agent, $\mathbf{E}_{v_i}[p_i(v_i)]$, in any truthful mechanism, is equal to her expected virtual value $\mathbf{E}_{v_i}[\phi_i(v_i)x_i(v_i)]$ [6]. The distribution F_i is said to be *regular* if the virtual valuation function is monotone. For regular distributions, maximizing virtual values pointwise results in an incentive-compatible allocation rule and therefore the corresponding revenue-optimal auction serves that feasible set of agents who maximize virtual value. For irregular distributions, where maximizing virtual values may result in non-IC allocation rules, Myerson applies a fix by describing a general *ironing* technique. The ironing procedure converts any virtual valuation function $\phi_i(\cdot)$ to an *ironed virtual value function* $\bar{\phi}_i(\cdot)$ such that maximizing $\bar{\phi}_i(\cdot)$ pointwise results in an IC allocation rule.

Theorem 1. *[14] The revenue optimal auction in single-parameter Bayesian settings serves the set S of agents where $S = \mathrm{argmax}_{S \in \mathcal{S}} \sum_{i \in S} \bar{\phi}_i(v_i)$. Further, for all values v_i for which $\bar{\phi}_i(v_i)$ remains the same, agent i's allocation remains the same.*

As a corollary, when $\mathcal{S}$ contains all possible 2^n sets, the revenue optimal auction puts a price of $\bar{\phi}_i^{-1}(0)$ for agent i and makes a take-it-or-leave-it offer to each of them.

[6] In fact the equality holds even for a specific $\mathbf{v}_{-i}$, i.e., $\mathbf{E}_{v_i}[p_i(\mathbf{v})] = \mathbf{E}_{v_i}[\phi_i(v_i)x_i(\mathbf{v})]$.

Monotone Hazard Rate. A class of distributions that always satisfy the regularity condition described above are the ones which satisfy the monotone hazard rate condition. A distribution with cdf F satisfies the MHR condition if $\frac{f(x)}{1-F(x)}$ is nondecreasing in x. The MHR condition holds for many natural classes of distributions like the uniform, exponential and normal distributions.

Nonnegative Virtual Valuations. An important property of virtual valuations (and ironed too) is that when the support of the distribution is nonnegative (which is true in our case since values are nonnegative), the virtual valuation in expectation over an agent's distribution is always nonnegative. This property crucially helps us in providing approximations in expectation, for problems which are very hard to approximate for every single realization of values.

3 The Unrestricted Version

In this section, we obtain the following results for the unrestricted version of our problem.

1. A simple polynomial time dynamic program that solves the pointwise problem optimally in paths. (see Appendix A.)
2. A hardness result showing that the pointwise problem is hard to approximate within a factor of $\Omega(n^{2-\epsilon})$ for any $\epsilon > 0$ (assuming $P \neq NP$) even in stars, i.e., we cannot generalize the result on paths even to stars (proof in full version).
3. Given that pointwise approximation is ruled out for arbitrary graphs, the only possible approximation we can hope for is in expectation over values. We show the APX-hardness of this problem in arbitrary graphs. The reduction resembles the one given by Haghpanah et al. [10]. In fact, our hardness result holds even for the rooted version, and hence the proof is presented in Section 4.3 along with other results for the rooted version.

4 The Rooted Version for i.i.d. Agents

Given the hardness results in Section 3 even for undirected graphs, we consider an important special case of our problem in this section: there is a designated root node in an undirected graph, and each nonroot vertex is an agent. Agents values are i.i.d., and an agent derives value only if there is a path constructed from his node to the root node. As before, construction of an edge e can happen only if $e \in G$.

4.1 Pointwise Optimization in Trees

In contrast to the unrestricted version where we showed that the pointwise problem in hard to approximate within a factor of $\Omega(n^{2-\epsilon})$ (assuming $P \neq NP$) even in stars, we show that for the rooted version, we can solve the pointwise problem in polynomial time in trees. The proof is presented in the full version.

4.2 Main Result: Optimization in Expectation in Arbitrary Graphs

Next we present a constant-factor approximation algorithm for optimizing the expected revenue for the rooted, node-weighted version of the problem in arbitrary graphs (the virtual value of an agent is the weight of the nonroot node of that agent). In Section 4.2 we explain how the problem can be solved on edge-weighted graphs, i.e., agents are edges, and they get a value when they are connected to the root. This is later used in Section 4.2 as a subroutine to tackle the vertex-weighted problem.

Edge-Connectivity for the Rooted Version. Given a connected undirected graph, we show how to partition its edges into two parts such that contracting the former edge set yields a 3-edge connected subgraph whereas contracting the latter edge set results in a *roulette* subgraph—a special series-parallel graph to be defined below. We then demonstrate that it is possible to solve the problem (approximately) on each of the two subgraphs, and finally argue that this suffices to obtain a constant-factor approximation for the general rooted edge-weighted case.

Let us define some notations first. For a set S of edges in a graph G, subgraph G/S is obtained from G after contracting all edges S one at a time, where contracting an edge simply refers to removing the edge and identifying its endpoint vertices. We recursively define the class of roulette graphs as follows. A simple cycle is a roulette, and so is a cycle each of whose vertices is replaced by a roulette (with one or two vertices of the inner roulette taking the place of an original vertex of the cycle). Finally, any graph whose 2-edge connected components are roulettes is itself a roulette.

Now we can present the main structural lemma that reduces our general problem into two tractable subproblems.

Lemma 1. *There exists a polynomial-time algorithm that, given a graph $G(V, E)$, partitions the edge set E into two sets S_1 and S_2 such that graphs $G_1 = G/S_1$ and $G_2 = G/S_2$ are respectively 3-edge connected and roulette.*

Proof. Let S_1 be the set of all edges in G that belong to a cut of size at most 2. We note that all cuts of size at most 2 can be found in polynomial time, e.g., using a naïve brute-force search. Since G_1 is obtained by a series of edge contractions, every cut in G_1 represents a cut in G as well. Therefore, since all edges of S_1 are contracted in G_1, no cut of size at most 2 is present in G_1, hence G_1 is 3-edge connected.

On the other hand, every edge in G_2 belongs to a cut of size at most 2 in G, hence in G_2. The bridges in G_2 do not hurt the roulette structure if the 2-edge connected subgraphs of G_2 are roulettes. Thus we assume that the graph G_2 is 2-edge connected. We claim that if each of $\{e_1, e_2\}$ and $\{e_1, e_3\}$ is a cut in G_2, then so is $\{e_2, e_3\}$. Therefore, the edges of G_2 form an equivalence class. To see this equivalence relation, it suffices to focus on the following alternative definition of edge cuts of size 2. In a 2-edge connected graph, two edges e and e' form a cut of size 2 if and only if the set of cycles in the graph that contain e is the same as the set of cycles in the graph that contain e'. Based on this new definition, the two sets of cycles containing e_2 and e_3 respectively are both equal to the set of cycles containing e_1, and therefore they are equal to each other as well which means that e_2 and e_3 form a cut of size 2.

Therefore, the graph looks like a cycle of this equivalence class (the equivalence class of e_1) where some vertices are replaced by another structure; the same argument applies to each of these smaller structures, giving rise to the inductive definition of roulette graphs. We should note that two edges from two of these smaller structures do not belong to the same equivalence class (they cannot form a cut of size 2), and therefore each of the other equivalence classes belong to one smaller structure and is not split between different structures. Thus we can inductively claim each smaller structure is a roulette graph. □

The following decomposition lemma serves as the starting point for our 3-approximation algorithm of the 3-edge connected graph G_1.

Lemma 2. *There exists a polynomial-time algorithm that finds* 3 *spanning trees* T_1, T_2*, and* T_3 *in a* 3*-edge connected graph* G *such that every edge of* G *is missing in at least one of these spanning trees.*

Proof. We replace each edge of G by two parallel edges to get the 6-edge connected graph G^2 with the same vertex set. Catlin et al. [6] show, among other things, that any $2k$-edge connected graph has k edge-disjoint spanning trees, while Roskind and Tarjan [16] show how to find k edge-disjoint spanning trees in a graph (if they exist) in quadratic time. Therefore, we can find 3 edge-disjoint spanning trees T_1, T_2, and T_3 in G^2. Edge-disjointness guarantees that each edge of G can belong to at most two of these spanning trees. □

On the other hand, the problem can be solved optimally for roulette graphs. The intuition behind the algorithm is that roulettes can be shown to have treewidth of at most two, hence, as are many similar problems on bounded-treewidth graphs, our problem can be solved via the dynamic-programming method.

Lemma 3. *There exists a polynomial-time algorithm that finds the optimal solution for roulette graphs.*

Proof. Let us say we have a *cycle decomposition* of our roulette graph, which describes the recursive structure of its 2-edge connected components. Each cycle representing one equivalence class is called an *essential* cycle of the graph. Instead of solving the rooted problem, we consider a slightly more general problem where up to two vertices s, t on an essential cycle are specified, and these vertices should both appear in the connected subgraph of the output. At the beginning we are going to have only one vertex $s = t$ which is the root vertex.

Let us ignore the bridges at this point and assume the graph is 2-edge connected. Focus on the essential cycle, and imagine that all the recursive structures are contracted for now. In the resulting graph, the optimal solution looks like a path or it is the entire cycle. There are polynomially many cases to consider, and we will output the best solution among them. In each case we have a subproblem for each contracted piece if we decide that the optimal solution passes through or ends at the contracted vertex. Since the base cases of the induction (i.e., vertices or cycles) are easily solvable, we can argue inductively that our sligthly more general problem can be solved using a dynamic program.

For each bridge connected to the essential cycle, we compute the best solution for the rooted problem on the other side of the bridge (whose root is the endpoint of the bridge) and add that to the main solution if its weight plus the weight of the bridge turns out positive. □

We conclude this section by putting together the above ideas to obtain a 4-approximation algorithm for the rooted edge-weighted problem.

Lemma 4. *There exists a polynomial-time monotone algorithm that achieves an approximation factor of* 4 *for any graph* G*.*

Proof. If the graph is not connected, we can focus on the connected component that contains the root and disregard the rest of the connected components. Using Lemma 1, we partition the edges of G into two parts S_1 and S_2. We also use Lemma 2 to find three spanning trees T_1, T_2, and T_3 in graph G_1.

We consider four candidate solutions. One solution is the union of S_1 and the edges in tree T_1. Since T_1 is a spanning tree in $G_1 = G/S_1$, the union of T_1 and S_1 is a connected spanning subgraph of G. Serving this connected spanning subgraph allows us to choose any subset of the remaining edges to serve. Among the remaining edges (edges not in $S_1 \cup T_1$), we serve those with positive realized virtual values. In a similar fashion we construct two other candidate solutions based on T_2 and T_3. The fourth and last candidate solution is to contract set S_2 of edges, and solve the problem optimally in the roulette graph G_2 using Lemma 3. The optimal solution we find in G_2 is a connected subgraph of G_2, however, it might not be a connected subgraph of G that includes the root. Nonetheless, it is always possible to serve a subset of edges $S'_2 \subseteq S_2$ to make the whole solution not only a connected subgraph of G but also one that includes the root vertex as well. Our fourth candidate solution consists of the edges in S'_2 and the optimal soluion for G_2.

We now show that for every instance (i.e., a graph with a designated root and a distribution) one of these candidate solutions that guarantees a 4-approximate solution. Thus picking one at random will guarantee a 4-approximation. For $1 \leq i \leq 3$, if we stick to solution i all the time, our gain from edges in S_1 and T_i is nonnegative (i.e., the sum of their expected virtual values), and in addition we get all edges with positive virtual value outside $S_1 \cup T_i$. Since each edge of S_2 is missing in at least one $S_1 \cup T_i$, the total value we get from the first three solutions is at least the projection of optimal solution on set S_2 of edges. On the other hand in the fourth solution, our expected revenue from edges added from S_2 is nonnegative (once again since the expectation of the virtual values are positive), and our expected revenue from S_1 is at least the amount that the optimal solution gains from them. Therefore, these four solutions together achieve no less than the optimal revenue. Thus, one candidate solution has expected revenue at least a quarter of the optimum.

The algorithm is monotone because firstly it decides which of the four candidate solutions to use independent of the realized values. Further each candidate solution is monotone. The first three solutions are monotone because the edge selection criteria (for the edges they consider) is just non-negative virtual value. The fourth solution is monotone because it is an optimal algorithm for the edges it focuses on. □

Vertex-Connectivity for the Rooted Version. In this section we provide a constant-factor monotone approximation algorithm for the vertex-connectivity problem using the algorithm for the edge-connectivity version of the problem described above. Let V_2 be the set of degree-2 vertices in graph G. Similar to the edge-connectivity approach, we describe two algorithms (one using a reduction to the edge-connectivity problem) that achieve constant-factor approximations to the problems where the values of $V \setminus V_2$ and V_2 are replaced by zero, respectively. Again, since the expected value of each vertex is nonnegative, this implies a constant approximation for the vertex-connectivity problem.

First consider the instance in which the values of vertices in $V \setminus V_2$ are replaced by zero. Notice that this results in an instance in which any vertex with nonzero value has degree 2. Construct another instance in which each vertex of degree 2 is replaced by an edge with the same value. We can solve this instance using our edge-connectivity algorithm.

Next consider the instance in which the values of vertices in V_2 are replaced by zero. We convert the instance to one without any degree-2 vertices via replacing all paths consisting only of degree-2 vertices by an edge. The following lemma shows that this graph has a spanning tree where at least $\frac{1}{7}$ of its vertices are leaves. The algorithm uses the *internal* nodes (i.e., nonleaves) of this tree to connect the leaves that are positive, to the root. This gives a 7-approximation to the problem because the vertex values are drawn i.i.d.

Putting these two algorithms together, we obtain a constant-factor approximation algorithm for the vertex-connectivity rooted revenue maximization in expectation. In particular, a balancing argument puts a bound of 11 on its approximation ratio.

Monotonicity of the resulting algorithm follows from the monotonicity of the algorithms used for the two subcases. When we use the algorithm for the edge-connected version, the monotonicity of the edge-connected version implies the same here. For the other case, note that a leaf is picked whenever its virtual value is positive thus resulting in a monotone allocation.

Lemma 5. *Given a graph with no degree-*2 *vertices, a spanning tree can be constructed in polynomial-time where at least* $\frac{1}{7}$ *of the vertices are leaves.*

Proof. Start from an arbitrary spanning tree T of G. Let T_2 be the set of vertices of degree 2 in T, and let $\hat{T}_2 \subseteq T_2$ be those vertices in T_2 both whose neighbors, too, are in T_2. Modify T as long as any of the following two rules apply.

1. If there exists a vertex $v \in \hat{T}_2$ that has an edge in G to an internal vertex u of T, update T by adding the edge (v, u) to T, and removing the edge incident to v in the unique cycle formed after adding (v, u). Since both neighbors of v in T had degree 2, this process generates a new leaf without removing any of the old leaves.
2. If two vertices $v, u \in \hat{T}_2$ have edges in G to the same leaf l of T, add edges from v and u to that leaf, and remove two edges from T as follows. The addition of edge (u, l) to T produces a cycle that passes through exactly one of the two neighbors of u; call it u'. Note that u' has degree 2 in T by definition of $\hat{T}_2$; let u, u'' be its neighbors. Remove the edge (u', u'') from T, removing the cycle u and maintaining the connectivity of T. We carry out a similar operation, mutatis mutandis, for v. The

result will be a tree T on the same set of vertices with one more leaf (increasing the degree of l but turning two other internal vertices into leaves).

The process terminates in a linear number of iterations since the number of leaves increases in each step. We end up with a tree T for which neither of the rules applies. Let T_1 and $T_{\geq 3}$, respectively, denote subsets of vertices of degrees one and at least three in T. We argue below that $|T_1|$ is at least a constant fraction of $|T_2|+|T_{\geq 3}|$. As no vertex of G has degree two, any vertex in $\hat{T}_2$ is bound to have degree at least three in G, hence an edge not in T. This edge cannot be to an internal vertex of T because Rule (1) no longer applies to T. Rule (2), on the other hand, implies that these leaves are distinct for different vertices of $\hat{T}_2$. Therefore, we have

$$|T_1| \geq |\hat{T}_2|. \tag{1}$$

As trees have average degree less than two, we know

$$|T_1| > |T_{\geq 3}|. \tag{2}$$

To bound $|T_2| - |\hat{T}_2|$, if this quantity is not zero, orient T from an arbitrary vertex in $T_2 \setminus \hat{T}_2$ towards the leaves. Assign each vertex $v \in T_2 \setminus \hat{T}_2$ to its closest descendant in the oriented tree that is in $T_1 \cup T_{\geq 3}$. Such an assignment is always possible since no vertex in the former group is a leaf of the (oriented) tree. Each vertex in the latter group is assigned to at most twice, otherwise there should be a path of vertices of degree two with more than two vertices in $T_2 \setminus \hat{T}_2$—a contradiction. As a result, we get

$$|T_2| - |\hat{T}_2| \leq 2(|T_1| + |T_{\geq 3}|) \leq 4|T_1|, \tag{3}$$

where the last inequality is due to (2).

Summing up (1), (2) and (3) with $|T_1| \geq |T_1|$, we obtain $7|T_1| \geq |T_1|+|T_2|+|T_{\geq 3}|$ as desired. □

4.3 APX-Hardness of Optimization in Expectation in Arbitrary Graphs

In this section we prove the APX-hardness of the rooted version in arbitrary graphs when the valuations of agents are independent but not necessarily identical.

Definition 1. *The prize-collecting set cover problem (PCSCP) consists of a collection of sets $S_1, S_2, \ldots, S_n$ over a universe U. For a collection C of sets, let $Q_C = \cup_{i \in C} S_i$. The goal is to find a collection C^* that maximizes $\alpha|Q_{C^*}| + n - |C^*|$ for some given $\alpha > 0$.*

We first show that there is an approximation preserving reduction from PCSCP to our problem, and then invoke the result from [10] that establishes the APX-hardness of PCSCP, and that its approximation ratio is at least $\frac{530}{529} = 1.002$.

Lemma 6. *There is an approximation preserving reduction from PCSCP to our problem*

See Appendix B for a proof.

5 Posted-Pricing Results

As mentioned in the introduction, one goal of this work to compare two kinds of mechanisms: a) natural mechanisms for nonexcludable public goods, such as a direct revelation mechanism, that have to deal with nonexcludabilities and b) posted price mechanisms which mitigate nonexcludability to a certain extent by ensuring that only those who pay for an edge can use that edge. We show that for the rooted version with i.i.d. agents, a posted price mechanism obtains the optimal revenue possible when full excludability is allowed — i.e., even partially mitigating nonexcludability via tolls gets us the benefit of full excludability. When the agents values are independent but not necessarily identical, we design a pricing scheme that obtains a $O(\frac{1}{\log n})$ fraction of the optimal revenue possible when full excludability is allowed. The details of the prices set, and the proof, are presented in the full version of the paper.

References

[1] AGGARWAL, G., FELDMAN, J., MUTHUKRISHNAN, S., AND PÁL, M. 2008. Sponsored search auctions with markovian users. *Internet and Network Economics*, 621–628.

[2] AKHLAGHPOUR, H., GHODSI, M., HAGHPANAH, N., MIRROKNI, V., MAHINI, H., AND NIKZAD, A. 2010. Optimal iterative pricing over social networks. *Internet and Network Economics*, 415–423.

[3] ANARI, N., EHSANI, S., GHODSI, M., HAGHPANAH, N., IMMORLICA, N., MAHINI, H., AND MIRROKNI, V. 2010. Equilibrium pricing with positive externalities. *Internet and Network Economics*, 424–431.

[4] ATHEY, S. AND ELLISON, G. 2011. Position auctions with consumer search. *The Quarterly Journal of Economics 126,* 3, 1213–1270.

[5] CANDOGAN, O., BIMPIKIS, K., AND OZDAGLAR, A. 2010. Optimal pricing in the presence of local network effects. *Internet and Network Economics*, 118–132.

[6] CATLIN, P. A., LAI, H.-J., AND SHAO, Y. 2009. Edge-connectivity and edge-disjoint spanning trees. *Discrete Mathematics 309,* 5, 1033–1040.

[7] DENG, C. AND PEKEC, S. 2011. Money for nothing: exploiting negative externalities. In *Proceedings of the ACM Conference on Electronic commerce (EC)*. 361–370.

[8] GIOTIS, I. AND KARLIN, A. 2008. On the equilibria and efficiency of the gsp mechanism in keyword auctions with externalities. *Internet and Network Economics*, 629–638.

[9] GOMES, R., IMMORLICA, N., AND MARKAKIS, E. 2009. Externalities in keyword auctions: An empirical and theoretical assessment. *Internet and Network Economics*, 172–183.

[10] HAGHPANAH, N., IMMORLICA, N., MIRROKNI, V., AND MUNAGALA, K. 2011. Optimal auctions with positive network externalities. In *Proceedings of the 12th ACM conference on Electronic commerce*. EC '11. 11–20.

[11] HARTLINE, J., MIRROKNI, V., AND SUNDARARAJAN, M. 2008. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web*. ACM, 189–198.

[12] JEZIORSKI, P., SEGAL, I., ET AL. 2009. What makes them click: Empirical analysis of consumer demand for search advertising. *SSRN eLibrary 33*.

[13] KEMPE, D. AND MAHDIAN, M. 2008. A cascade model for externalities in sponsored search. *Internet and Network Economics*, 585–596.

[14] MYERSON, R. B. 1981. Optimal Auction Design. *Mathematics of Operations Research 6,* 1, 58–73.

[15] ROHLFS, J. 1974. A theory of interdependent demand for a communications service. *The Bell Journal of Economics and Management Science*, 16–37.

[16] ROSKIND, J. AND TARJAN, R. E. 1985. A note on finding minimum-cost edge-disjoint spanning trees. *Mathematics of Operations Research 10,* 4, 701–708.

A The Unrestricted Version

A.1 Pointwise Optimization in Paths

In this section, we show that in paths, a polynomial-time dynamic program is all that is necessary to implement the pointwise optimization involved in implementing Myerson's mechanism.

Consider a path with vertices 1 to n. For $1 \leq i \leq j \leq n$, let $S(i, j)$ be the sum of virtual values of the set of all agents (i.e., vertex pairs) whose both endpoints are in the interval $[i, j]$. We set $S(i, i) = 0$ for all i—assuming that there is no trivial agent whose vertex pairs are the same. Define $OPT(i)$ to be optimal solution when we are restricted to choose only among edges connecting vertices 1 through i. The unrestricted optimal solution OPT is therefore $OPT(n)$. We set $OPT(0) = 0$. The following recursive formula can be used in the dynamic program to solve for OPT.

$$\forall i \leq n, OPT(i) = \max_{0 \leq k < i} OPT(k) + S(k + 1, i).$$

In the above recursion, $k+1$ is the leftmost vertex that is connected to i. All vertices $k + 1, \ldots, i$ are connected, and $S(k + 1, i)$ is by definition their contribution to the objective. Since the edge connecting k to $k + 1$ is not included, the set of edges chosen among the first k vertices must be equal to $OPT(k)$.

B The Rooted Version

B.1 Proof of Approximation Preserving Reduction from PCSCP

Proof of Lemma 6. Given an instance of the PCSCP, where the sets are denoted $S_1, S_2, \ldots S_n$ and the elements are denoted $e_1, e_2, \ldots e_m$, we construct an instance of our problem as follows.

Vertices. We start with a root vertex r. For every element e_i, we construct one vertex with the same name e_i. For every set S_i, we construct one vertex denoted by S_i as well.

Edges. For each i, set vertex S_i is connected by an edge to root r and all $e_j \in S_i$.

Agents. The agents (r, S_i) are called set-agents, and (r, e_i) are called element-agents.

Distribution The value distribution of element agents is deterministic α. For set agents, the value is drawn from the distribution Bernoulli$(L-1, 1/L)$—i.e., the value is equal to $L-1$ with probability $1/L$, and zero otherwise—where $L \gg mn\alpha$. Thus, the virtual value for these agents is -1 w.p. $1-1/L$ and $L-1$ w.p. $1/L$.

The optimal revenue in our problem, as $L \to \infty$, can be analyzed in two cases.

1. If at least one set-agent has positive virtual valuation (which happens w.p. approximately $n/L \to 0$), the solution chooses all the edges incident on those set agents (with positive virtual valuation) to obtain expected revenue n. The expected revenue from the remaining agents (set-agents with negative virtual valuation, and element-agents) is at most $\alpha mn/L \ll 1$. Therefore, the optimal solution has contribution n from this event as $L \to \infty$, and this solution is trivial to compute.
2. If no set has positive virtual valuation (which happens w.p. $1 - n/L \to 1$), the value of the solution is precisely $\alpha|Q_{C^*}| - |C^*|$. This is because once a set-agent is chosen, clearly all the edge-agents that is contained by this set must be chosen since they all have deterministic positive virtual value α. We should also note that you can obtain the α virtual value of an element agents only if you connect it to root using one of the set agents it belongs to, and for each set you pick you have -1 virtual value in this case.

Therefore the value of the optimal solution is $\alpha|Q_{C^*}| + n - C^*$. □