Robust Combinatorial Optimization with Exponential Scenarios

Uriel Feige* Kamal Jain* Mohammad Mahdian* Vahab Mirrokni*

November 15, 2006

Abstract

Following the well-studied two-stage optimization framework for stochastic optimization [14, 17], we study approximation algorithms for robust two-stage optimization problems with exponential number of scenarios. Prior to this work, Dhamdhere et al. [7] introduced approximation algorithms for two-stage robust optimization problems with polynomial number of scenarios. To model exponential number of scenarios, we assume the set of possible scenarios is given implicitly, for example by an upper bound on the number of active clients. In two-stage robust optimization, we need to pre-purchase some resources in the first stage before the adversary's action. In the second stage, after the adversary chooses the clients that need to be covered, we need to complement our solution by purchasing additional resources at an inflated price. The goal is to minimize the cost in the worst-case scenario. We give a general approach for solving such problems using LP rounding. Our approach uncovers an interesting connection between robust optimization and online competitive algorithms. We use this approach, together with known online algorithms, to develop approximation algorithms for several robust covering problems, such as set cover, vertex cover, and edge cover. We also study a simple *buy-at-once* algorithm that either covers all items in the first round or does nothing in the first round and waits to build the complete solution in the second round. We show that this algorithm gives tight approximation factors for unweighted variants of these covering problems, but performs poorly for general weighted problems.

*Microsoft Research, Redmond, WA, USA. emails: {urifeige,kamalj,mahdian,mirrokni}@microsoft.com

1 Introduction

In many combinatorial optimization problems, the objective is to minimize the cost of building an installation to serve a number of clients. In classical optimization problems, it is often assumed that the parameters of the system are known in advance. However, in reality, it is almost always impossible or costly to obtain accurate data about various parameters of the optimization problem at the time of planning. For example, the cost of acquiring a resource or the set of clients that need to be serviced might be unknown. The goal of the fields of *stochastic optimization* and *robust optimization* is to provide algorithms for minimizing the cost in presence of uncertainty.

In *stochastic optimization* [5, 6], it is assumed that we have information about the probability distribution governing the data. Given this information, the goal is to plan ahead to minimize the expected cost. In particular, in *two-stage stochastic optimization*, a solution is built in two stages: in the first stage, we need to decide which resources to purchase given only distributional information about the instance. In the second stage, the exact information about the data is revealed and we are allowed to complement the solution built in the first stage by purchasing extra resources at an inflated cost.

Robust optimization [3, 4, 16] can be considered the worst-case analogue of the stochastic optimization. In a robust optimization problem, we are given bounds on various parameters of the system, and the goal is to find a solution that minimizes the cost in a worst-case scenario (or be feasible in a worst-case scenario). A *two-stage robust optimization* problem (introduced in [7]) is similar to a two-stage stochastic problem except instead of a distribution, we have a set of possible scenarios (given either explicitly, or implicitly by giving bounds on various parameters), and instead of expectation, we would like to minimize the maximum cost of the solution, where maximum is taken over the set of all possible scenarios.

During the past few years, stochastic optimization (and in particular, two-stage stochastic optimization) has received considerable attention from the perspective of approximation algorithms. Efficient approximation algorithms are given for a wide class of optimization problems, both for cases where the distribution is given explicitly by listing the set of all possible scenarios and the corresponding probabilities [17, 12], and in the more general case where the distribution is given implicitly, as the product of a number of independent trials, or by an oracle [14, 18, 11, 7].

For robust optimization, only recently Dhamdhere, Goyal, Ravi, and Singh [7] initiated the study of approximation algorithms for two-stage robust covering problems when the set of scenarios is given *explicitly*. In this paper, we take on the task of studying approximation algorithms for two-stage robust optimization problems, where the set of possible scenarios is given *implicitly*. In particular, we focus on the case where the set of possible scenarios is given by an upper bound on the number of active clients, and give approximation algorithms for the robust version of several classical covering problems such as set cover, vertex cover, and edge cover.

1.1 Preliminaries

In this section we give a formal definition of the two-stage robust optimization model that will be studied in this paper. This model is a generalization of the model introduced by Dhamdhere et al. [7] (in the case of explicitly listed scenarios), and is motivated by similar models for two-stage stochastic optimization [14, 12].

In a *covering problem*, we have a set $\mathcal{C}$ of potential *clients*, and a set $\mathcal{R}$ of *resources*. Each resource $r \in R$ can be *purchased* at a cost c_r. In order to serve a set of clients, a set of resources must be purchased. The collection of all sets of resources which can serve the set $S \subset \mathcal{C}$ of clients is denoted by $sol(S)$. In covering problems the collection $sol(S)$ is an upper ideal, i.e., if a set of resources can serve S, so can any superset of this set.

Generally, the collection $sol(S)$ is given implicitly by specifying a set of constraints. Three examples that we will focus on in this paper are set cover, vertex cover, and edge cover. In the *set cover* problem, each resource $r \in \mathcal{R}$ corresponds to a set of clients, and the collection $sol(S)$ consists of all sets of resources whose union covers S. In the *vertex cover* problem, the set of clients and the set of resources are the edge set and the vertex set of a given graph, respectively, and a set $S \subset \mathcal{C}$ can be served by any set of vertices that contain at least one of the endpoints of each edge in S. In the *edge cover* problem, each resource is an edge and each client is a vertex of a given graph, and a set S of clients can be covered by any set of edges that has at least one edge adjacent to any vertex in S.

In a *two-stage robust covering problem*, we have a collection $\mathcal{S}$ of scenarios, each given by a set of *active* clients (i.e., clients that need to be covered). The objective is to purchase a set of resources in the first stage to minimize the cost of these resources plus a given inflation factor λ times the maximum over scenarios S in $\mathcal{S}$, of the cost completing the solution for scenario S. In other words, after we purchase a set of resources in the first stage, an adversary decides in which scenario we are. After that, we need to complete the solution by purchasing more resources at costs inflated by a factor λ (or more generally, by an inflation factors λ_r^S which depends on the resource $r \in \mathcal{R}$ and the scenario $S \in \mathcal{S}$).

The robust optimization problem can be studied in several different models, depending on how the list of scenarios $\mathcal{S}$ is given to the algorithm. The first model, studied by Dhamdhere et al. [7] and Golovin et al [10], is to assume that the list of all possible scenarios is given explicitly. This model is suitable for situations where the number of possible scenarios is not very large. An alternative model, motivated by the independent trials model of stochastic optimization, is to assume that the list of scenarios is given implicitly by an upper bound on the maximum number of active clients. More formally, in this model an integer k is given and $\mathcal{S}$ is defined as $\{S \subseteq \mathcal{C} : |S| \leq k\}$.[1] Finally, motivated by the oracle model in stochastic optimization, we define an oracle model for robust optimization where the list of possible scenarios is given by an oracle which, given the set of resources purchased in the first stage, outputs the worst-case scenario for the second stage.

An important distinction between our oracle model and the oracle model for stochastic

[1] More generally, we can consider a model where the set of clients is partitioned into subsets $\mathcal{C}_1, \ldots, \mathcal{C}_t$, and the set of scenarios is the collection of all sets that have at most k_i clients from the set $\mathcal{C}_i$. Although the results in Sections 2 and 3 work for this more general model, for clarity of exposition we restrict ourselves to the simpler model.

optimization is that in our model, the problem the oracle needs to solve is often computationally intractable. For example, if the set of scenarios in a robust set cover problem is given by an upper bound on the number of active clients, the oracle needs to find a subset of k clients whose minimum cost of covering is maximized. We call this problem the *max-min set cover problem*, and will observe that it is computationally hard. Considering the hardness of the oracle problem, the algorithms designed for the oracle model need to be able to work with an *approximate oracle* as well. Furthermore, in order to solve a robust optimization problem in the model where the scenarios are given by an upper bound on the number of active clients, in addition to designing an algorithm for the oracle model, we need to give an approximation algorithm for the max-min version of the problem.

1.2 Our Contribution

In this paper, we mostly focus on the model where the scenarios are given implicitly by an upper bound on the number of active scenarios. This is motivated by real-world situations where a good estimate of the total number of clients who will show up is available, but we do not exactly know where they appear. We will also give a general LP-based algorithm for the oracle model, assuming that the oracle gives a good approximation of the worst-case scenario with respect to the fractional solution.

A naive idea to solve the robust optimization problems is a *buy-at-once* algorithm: either cover all items in the first round in which case nothing needs to be done in the second round. Or do nothing in the first round and construct a solution in the second round, after the adversary makes its choices. The choice of which of the two options to use is based on a polynomial-time test that is problem specific. We study this algorithm in Section 4 and prove that when the inflation factor is the same for all scenarios the approximation ratio of this algorithm for robust unweighted set cover, vertex cover, and edge cover problems are $\max(\log m, \log n)$, 2, and 2 respectively. However, the following example shows that for the weighted version of robust vertex cover, any buy-at-once algorithm (even with unbounded computing power) performs poorly. The input of the robust vertex cover problem are a node-weighted graph G, a parameter k (for number of edges to be chosen by adversary), and an inflation factor $\lambda > 1$. Consider a clique on n vertices, with $k = 1$ and $\lambda = \sqrt{n}$. All vertices have weight 1, except for two verteices that have weight $w = \sqrt{n}$. The buy-at-once algorithm will either pay at least n in the first round, or at least $\lambda w = n$ in the second round. However, an optimal algorithm can choose only the heavy vertex in the first round, and then pay at most $w + \lambda k = 2\sqrt{n}$. Hence the approximation ratio of the buy-at-once algorithm for weighted robust vertex cover is no better than $\Omega(\sqrt{n})$. This example indicates the need for more sophisticated approximation algorithms for robust two-stage optimization problems.

In Section 2, we give a general LP-based framework for solving robust covering problems given access to an oracle that solves the *max-min problem* (or the adversary's problem) and another oracle that rounds the LP solution for the classical (i.e., non-robust) optimization problem. More precisely, the separation oracle for this exponential LP's, we need to solve a *max-min* variant of the fractional covering problem. For example, in the max-min fractional set cover problem, given a collection of subsets $S_1, S_2, \ldots, S_m$ of a universe F each with a cost $c(S_i)$ and a parameter k, we need to find a subset $T \subseteq F$ of size at most k for which the cost of

fractional set cover is maximized. In Section 3, we show how an online algorithm can be used to solve the max-min problem when the set of feasible scenarios are given by an upper bound on the number of active clients. We use this to give $O(\log m)$-approximation algorithms for max-min fractional set cover. We also show that the max-min fractional set cover problem is not approximable within a factor better than $\Omega(\frac{\log m}{\log \log m})$ under reasonable complexity assumptions. As a result of this framework, we get an $O(\log n \log m)$-approximation for the robust set cover problem. Following similar ideas, we design constant-factor approximation algorithms for robust vertex cover and edge cover problems. This framework can be extended easily to more general settings in which the scenarios are given implicitly in more general ways. The main step for these extensions are to design good approximation algorithms for the max-min problems.

As mentioned earlier, in Section 4, we give a simple *buy-at-once* algorithm for robust unweighted set cover, vertex cover and edge cover with approximation ratio are $\max(\log m, \log n)$, 2, and 2 respectively. We show that no buy-at-once algorithm (even without computational constraint) can perform better than $\Omega(\log n + \log m)$ for unweighted set cover. For vertex cover and edge cover problems, we provide evidence (under "reasonable" complexity assumptions) that no polynomial-time algorithm can produce an approximation ratio better than two.

2 An LP-rounding approach for robust set cover

In this section, we give an LP-based approach for robust set cover. Our techniques work for a more general covering problem where each resource $r \in \mathcal{R}$ can be picked an integer number of times x_r, and a client is covered if a corresponding inequality of the form $\sum_r a_{ir} x_r \geq 1$ (where a_{ir} are given non-negative coefficients) is satisfied. The details of this generalization are omitted here.

We start by giving an LP formulation of the two-stage robust set cover problem.[2]

$$\text{minimize} \qquad T + \sum_{r \in \mathcal{R}} c_r y_r^0 \tag{1}$$

$$\text{subject to} \qquad \forall S \in \mathcal{S}, \forall i \in S : \sum_{r:\ i \in r} (y_r^0 + y_r^S) \geq 1 \tag{2}$$

$$\forall S \in \mathcal{S} : \sum_{r \in \mathcal{R}} \lambda c_r y_r^S \leq T. \tag{3}$$

The variable y_r^0 in the above LP indicates whether the resource r is purchased in the first stage. Similarly, the variable y_r^S indicates whether this resource is purchased in the second stage, if the adversary selects the set S as the set of active clients. The variable T indicates the maximum cost of the second stage, where the maximum is taken over all possible scenarios. Clearly, if the variables y_r^0 and y_r^S are restricted to be integers, the above integer program captures the robust set cover problem precisely. Therefore, relaxing the integrality condition

[2] Although we present this LP in the case that the inflation factor λ does not depend on the resource r or the scenario S, it is easy to see that all proofs in this section apply to the more general case.

gives us a linear program whose solution is a lower bound on the cost of the optimal solution to the robust set cover problem.

The main difficulty with this LP formulation is that it contains an exponential number of constraints *and* an exponential number of variables, and therefore cannot be solved directly using the ellipsoid method. We can deal with this problem using a technique developed by Shmoys and Swamy [18] for stochastic optimization: we consider the projection of the above LP onto the space corresponding to the variables y_r^0's and T, and then give a separation oracle for the reduced LP. The projection of the above LP corresponds to the following program.

$$\begin{aligned} &\text{minimize} && T + \sum_{r \in \mathcal{R}} c_r y_r^0 && (P) \\ &\text{subject to} && \forall S \in \mathcal{S} :\ T \geq \text{cost}_2(S, y_r^0) \end{aligned}$$

Here $\text{cost}_2(S, y_r^0)$ denotes the cost of the optimal fractional solution for the second stage when the set of active clients is S, given that resource r is already purchased to the extent of y_r^0 in the first stage.

The separation oracle for this LP corresponds to an algorithm that computes the optimal strategy for the adversary of the robust fractional set cover problem. We call this *the max-min fractional set cover problem*. More precisely, the max-min fractional set cover problem is the following: given a fractional first-stage solution (i.e., y_r^0's), select a scenario (in the example we will focus on in this paper, a set of at most k clients) so that the cost of a fractional solution for the second stage is maximized. The following lemma, proved using a simple application of the ellipsoid method, shows that given an approximation algorithm for the max-min fractional problem, we can compute an approximate solution of the above LP in polynomial time.

Lemma 1 *Assume we have a polynomial time γ-approximation algorithm for the max-min fractional problem. Then, we can compute a γ-approximation to the solution of the linear program (P) in polynomial time.*

Proof Sketch: The proof is based on an idea introduced by Shmoys and Swamy [18] in the context of stochastic optimization. The idea is to use binary search to find the smallest value of R such that the ellipsoid algorithm (using a separation oracle described below) decides that the following set of inequalities has a solution.

$$\forall S \in \mathcal{S} :\ \text{cost}_2(S, y_r^0) + \sum_{r \in \mathcal{R}} c_r y_r^0 \leq R \qquad (P')$$

For the separation oracle, we run the approximation algorithm for the max-min fractional problem with the input (y_r^0). This algorithm finds a scenario S^* such that $\text{cost}_2(S^*, y_r^0) \geq \frac{1}{\gamma}\,\text{cost}_2(S, y_r^0)$ for every scenario S. (Observe that given a scenario S^*, the value of $\text{cost}_2(S^*, y_r^0)$ can be computed in polynomial time, because computing this cost involves solving a fractional set cover problem. Hence the approximation algorithm for the max-min fractional problem returns a scenario together with its exact cost, rather than its approximate cost.) If $\text{cost}_2(S^*, y_r^0) + \sum_{r \in \mathcal{R}} c_r y_r^0 \leq R$, the separation oracle accepts (y_r^0) as a feasible solution.

Otherwise, it rejects and outputs a separating hyperplane of the polynomial-sized program $\text{cost}_2(S^*, y_r^0) + \sum_{r \in \mathcal{R}} c_r y_r^0 \leq R$. Consider the smallest value R^* (found by binary search) for which the ellipsoid algorithm decides that (P') has a solution. Notice that during the search for R^*, every time the ellipsoid algorithm is run for an $R < R^*$, all calls to the separation oracle resulted in a reject and a hyperplane in (P') that separates the given point. Therefore, for any such R, if we run the ellipsoid algorithm with an exact separation oracle on (P'), we should get the same answer. This means that the program (P') is infeasible for every $R < R^*$, and hence, the optimal solution of the program (P) is at least R^*.

For $R = R^*$, our ellipsoid algorithm terminates with a point (y_r^0) that is accepted by the separation oracle. The approximation guarantee of the separation oracle implies that (y_r^0) is a feasible solution for the program (P') with $R = \gamma R^*$. This means that (y_r^0) gives a solution of the program (P) with the objective function value at most $\gamma.R^*$, which is at most γ times the optimal solution of this program. ■

The above lemma requires us to be able to solve the max-min fractional set cover problem given a *fractional* first-stage solution. In other words, for each client i we are given a fractional value θ_i, so that if the adversary chooses i in the set of active clients, we will have to cover i to the extent of θ_i. In the following lemma, we show that it is enough to be able to solve the max-min problem given that θ_i's are zero or one. In other words, given a subset C' of the clients (corresponding to those with $\theta_i = 1$), we need to be able to find a set of at most k clients in C' whose minimum fractional cost of covering is maximized. We call this problem the max-min fractional set cover problem with integer requirements.

Lemma 2 *Assume we have a polynomial time γ-approximation algorithm for the max-min set cover problem with integer requirements. Then, we can compute a 2γ-approximation to the solution of the linear program (P) in polynomial time.*

Proof Sketch: We use the ellipsoid algorithm as in the proof of Lemma 1, except instead of running the algorithm for the max-min fractional problem with the input (y_r^0), we do the following: for each client i, if this client is covered by the fractional first-stage solution (y_r^0) to the extent of at least $1/2$, we do not require the client to be covered in the second stage; otherwise, we require the client to be fully covered, and then run the algorithm for the max-min problem with integer requirement on the resulting instance, and divide the value of the output of this algorithm by two. The result of the algorithm is as described in the proof of Lemma 1. It is easy to see that if this separation oracle rejects an input, an exact separation oracle would also reject the input. However, if the separation oracle accepts (y_r^0), then doubling the fractional solution would be a feasible fractional solution for the program (P') with $R = \gamma R^*$. Hence there is a feasible solution to the program (P) with cost at most 2γ times the cost of (y_r^0). ■

Finally, we notice that the solution obtained by solving the linear program (P) can be rounded into an integral solution using an LP-based algorithm that solves the (non-robust) optimization problem. Combining this with the previous lemmas, we obtain the following.

Theorem 1 *Assume, we have an α-approximation algorithm A_1 for the max-min set cover problem with integer requirements, and an algorithm A_2 that given a subset S of clients, finds*

an integral solution that covers the clients in S and whose cost is at most β times the minimum cost of fractionally covering S. Then there is a $2\alpha\beta$-approximation algorithm for the robust optimization problem.

Proof Sketch: As in the proof of the previous lemmas, we use the ellipsoid algorithm (with a separation oracle as described in the proof of Lemma 2) and binary search to find the smallest $R = R^*$ for which the ellipsoid algorithm decides that the program (P') has a feasible solution. Using the same argument, one can see that the value R^* is a lower bound on the solution of the linear program (P). On the other hand, for $R = R^*$, the algorithm finds a solution (y_r^0) which is accepted by the oracle. Let T be the set of clients covered to the extent of $1/2$ by (y_r^0), and use the algorithm A_2 to construct a solution of value at most 2β times $\sum_r c_r y_r^0$ to cover the elements in T. Finally, for any second stage scenario, the fractional cost of covering the uncovered elements of this scenario in the second stage is at most 2α by the property of the separation oracle, and hence these elements can be covered with cost at most $2\alpha\beta$ integrally. Hence, the total cost of the solution is at most $2\alpha\beta$ times the solution of (P). Hence, this algorithm is a $2\alpha\beta$ approximation algorithm for the robust set cover problem. ■

By the above theorem, the main ingredient in solving a robust optimization problem with implicitly given scenarios is the algorithm for the max-min problem. In the next section, we show how this problem can be approximated in several special cases.

3 The max-min problems

The results of the previous section show that in order to solve the LP relaxation of the robust set cover problem, we need to consider the max-min problem. In Appendix A we give a general reformulation of the max-min problem using LP duality, and then use this formulation to give a simple $\sqrt{n}$-approximation algorithm for this problem. In this section, we design $O(\log m)$-approximation for max-min fractional set cover problem. Here, we present a general framework to design approximation algorithms for the max-min problems using online competitive algorithms. Note that the max-min problems that we need to solve for approximating the robust covering problems are the fractional variant of covering problems.

Given a universe F of clients and a subset $T \subseteq F$, let $\text{opt}(T)$ be the cost of an optimal (fractional) solution to cover all clients in T. Let $\mathcal{A}$ be an α-competitive online algorithm for a covering problem. Namely, upon the arrival of any client a_k to an existing set of clients $a_1, a_2, \ldots, a_{k-1}$, $\mathcal{A}$ augments the current solution to a feasible solution for $a_1, \ldots, a_{k-1}, a_k$. The algorithm is α-competitive if for every sequence of clients $a_1, \ldots, a_k$ the cost of the online solution produced by $\mathcal{A}$ is at most α times the cost of the optimal (offline) solution for $a_1, a_2, \ldots, a_k$. Let $\mathcal{A}(b|a_1, a_2, \ldots, a_k)$ denote the *marginal increase* in the cost of the solution constructed by algorithm $\mathcal{A}$ when we add a new element b to an existing sequence of clients $(a_1, \ldots, a_k)$.

Consider two solutions w and w' for a fractional covering problem. Solution w' dominates solution w if for each set S the fractional value given to its respective variable in w' is at least as large as that given in w. We say that the covering problem satisfies the *monotonicity* property, if any given two solutions w and w' such that w' dominates w and any element a, the optimal marginal increase in expanding w' to cover a is not more than the optimal marginal increase

in expanding w to cover a. It is not hard to prove that the set cover problem and its special cases satisfy this property.

The following theorem presents a relation between competitive online algorithms and approximation algorithms for the max-min problem.

Theorem 2 *Let $\mathcal{A}$ be an α-competitive online algorithm for a covering problem. If the covering problem satisfies the monotonicity property then the corresponding max-min problem admits a $(\frac{e}{e-1})\alpha$-approximation algorithm.*

Proof: Given the online algorithm $\mathcal{A}$ for the covering problem, we prove that the following algorithm $\mathcal{B}$ is a $(1-\frac{1}{e})\alpha$-approximation algorithm for the max-min problem:

1. $T = \emptyset$.
2. for $i = 1, \ldots, k$ do
   (a) Find a client a_i that maximizes $\mathcal{A}(a_i|a_1, a_2, \ldots, a_{i-1})$ and add it to T.

Let the optimal solution to the max-min problem be the set $\{b_1, \ldots, b_k\}$ of clients. Let OPT^* be the optimal cost of covering $\{b_1, \ldots, b_k\}$. Let W_i be the cost of the solution of the online algorithm after i elements have arrived and $L_i = \max[0, \mathrm{OPT}^* - W_i]$. We prove that $L_i \leq (1-\frac{1}{k})L_{i-1}$. Consider expanding the solution of the online algorithm for $\{a_1, \ldots, a_{i-1}\}$ in the optimal way so that it covers $\{b_1, b_2, \ldots, b_k\}$. The cost of this new solution is at least OPT^*. Hence there is some item b_j (with $1 \leq j \leq k$) such that there is difference of $\frac{\mathrm{OPT}^* - W_{i-1}}{k}$ in cost between the case in which the clients $b_1, \ldots, b_{j-1}$ are added (or no clients at all, if $j = 1$) and the case in which the clients $b_1, \ldots, b_j$ are added. Since the covering problem satisfies the monotonicity property, adding b_j alone to $\{a_1, \ldots, a_{i-1}\}$ requires an increase in cost of at least $\frac{\mathrm{OPT}^* - W_{i-1}}{k}$ compared to the cost of $\mathcal{A}$ covering $\{a_1, \ldots, a_{i-1}\}$. Hence $W_{i-1} + (\frac{\mathrm{OPT}^* - W_{i-1}}{k})$ is a lower bound on the cost of $\mathcal{A}$ for covering $\{a_1, \ldots, a_{i-1}, b_j\}$. Since algorithm $\mathcal{B}$ chooses in the ith step the a_i that maximizes the marginal increase in the cost of $\mathcal{A}$, we will indeed have that $W_i \geq W_{i-1} + \frac{\mathrm{OPT}^* - W_{i-1}}{k}$, and thus, $L_i \leq L_{i-1}(1-\frac{1}{k})$. Thus $L_i \leq (1-\frac{1}{k})^i L_0$. Therefore, $L_k \leq (1-\frac{1}{k})^k \mathrm{OPT}^* \leq \frac{1}{e}\mathrm{OPT}^*$. This shows that $W_k \geq (1-\frac{1}{e})\mathrm{OPT}^*$. Since $\mathcal{A}$ is an α-competitive algorithm, the true cost of covering $\{a_1, \ldots, a_k\}$ is at least W_k/α, and algorithm $\mathcal{B}$ is a $(\frac{e}{e-1})\alpha$-approximation algorithm for the max-min problem.

■

Using Theorem 2 and the known competitive online algorithms, we design approximation algorithms for the max-min problems. For the sake of completeness, we describe an $O(\log m)$-competitive algorithm for the online fractional set cover problem by Alon et. al [1]. In the online algorithm, every set gets a weight, where the weight of set S at each time will be denoted by $w(S)$. Every client will have a covering factor, where the covering factor of client i is $f(i) = \sum_{S \in \mathcal{S} | i \in S} w(S)$. Similar to [1], we assume that the cost of each set is between 1 and $2m^2$, i.e, for each $S_i \in \mathcal{S}$, $1 \leq c(S_i) \leq 2m^2$. We initialize the weight of every set S as $w(S) = \frac{1}{2m^3}$. After the arrival of a new client a, the online algorithm performs the following:

1. if $f(a) \geq 1$, then do nothing (the new client a is already fractionally covered), else
   (a) Let $\epsilon = \frac{1-f(a)}{\sum_{S \in \mathcal{S}: a \in S} \frac{w(S)}{c(S)}}$.
   (b) for each $S \in \mathcal{S}$ that contains a, $w(S) = w(S)(1 + \frac{\epsilon}{c(S)})$.
   (c) for each b, $f(b) = \sum_{S \in \mathcal{S}: b \in S} w(S)$.

Alon et. al [1] proved that the above algorithm is $O(\log m)$-competitive for the online fractional set cover problem. This algorithm along with Theorem 2 gives an $O(\log m)$-approximation algorithm for the max-min fractional set cover problem. In Appendix B, we show that this result is nearly best possible (assuming certain complexity theoretic assumptions).

The above result also implies an $O(\log m \log k)$-approximation algorithm for the max-min (integral) set cover problem. Moreover, using the randomized rounding method for the set cover problem and the above results, we have the following:

Theorem 3 *There exists an $O(\log m \log n)$-approximation algorithm for the robust two-stage set cover problem.*

Using the ideas of the 2-approximation algorithm for vertex cover by Bar-Yehuda and Even [2], we can design a 2-competitive online algorithm for vertex cover problem as follows. In the online algorithm, we keep track of a value $r(u)$ for each vertex u of the graph. We initialize these values to $r(u) = w(u)$. At any moment, the fractional vertex cover solution is to pick $1 - \frac{r(u)}{w(u)}$ fraction of each vertex u. Upon the arrival of a new edge $e = uv$, the online algorithm sets $r_u = r_u - \min(r_u, r_v)$ and $r_v = r_v - \min(r_u, r_v)$. Observe that after this update either $r(u) = 0$ or $r(v) = 0$. Note that the fractional solution is to pick $1 - \frac{r(u)}{w(u)}$ of each vertex u. This means that we fully pick u or v for edge $e = uv$ and this solution is a feasible fractional vertex cover. Similar to the proof of Bar-Yehuda and Even [2], we can prove that this algorithm is a 2-competitive online algorithm. Using Theorem 2, this 2-competitive online algorithm implies a $(\frac{2e}{e-1})$-approximation algorithm for the max-min fractional vertex cover problem. Applying the above results and the 2-approximate rounding procedure for the vertex cover problem, we get a $(\frac{4e}{e-1})$-approximation algorithm for the robust (weighted) vertex cover problem. The details are omitted. Also, for the edge cover problem, it is straightforward to design a constant-factor approximation algorithm for the max-min problem and hence a constant-factor approximation for the robust edge cover problem. We do not optimize the constants of the approximation ratio for the weighted problems. However, in Section 4 we show a tight buy-at-once 2-approximation for unweighted vertex cover and edge cover.

4 Improved Algorithms for Unweighted Problems

In this section, we give buy-at-once approximation algorithms for unweighted variants of robust set cover, vertex cover, and edge cover.

4.1 Robust Unweighted Set Cover

In the robust unweighted set cover problem, all sets have unit cost. The input of the problem include n items, a collection of m sets, a parameter k (for number of items to be chosen by adversary), and an inflation factor $\lambda > 1$. To simplify notation, we assume here that parameters such as k, m and n are sufficiently large, and hence we shall ignore effects such as rounding $\ln m$ to the nearest integer. They affect the approximation ratio only by low order terms. The buy-at-once approximation algorithm for robust set cover is as follows:

1. Compute a minimum fractional set cover and let t be its size.
2. If $t < \frac{\lambda k}{\ln n}$, use the greedy algorithm to find a set cover. It will be of size at most $t \ln n$. Nothing needs to be done in the second round.
3. If $t \geq \frac{\lambda k}{\ln n}$, do nothing in first round. In the second round, use a greedy algorithm to cover the items chosen by the adversary.

Lemma 3 *The above buy-at-once algorithm achieves an approximation ratio no worse than* $\max[\ln n, \ln m]$ *(up to low order terms) for unweighted robust set cover.*

Proof: Observe that by duality, t is the size of the maximum fractional packing. Let αt be the number of sets chosen by opt in the first round. Removing all items covered by these sets, the remaining set cover instance still has a fractional packing of value at least $(1-\alpha)t$. (We may assume that $\alpha \leq 1$, as otherwise the analysis becomes even simpler.) Pick a set T of items, where each item is selected into T independently, with probability equal to its fractional value in the maximum fractional packing. The expected size of T is exactly $(1-\alpha)t$. In fact, known bounds imply that $|T| \geq \lfloor (1-\alpha)t \rfloor$ with probability at least $1/2$. Moreover, every set is expected to contain at most one item from T, and known bounds imply that with probability at least $1/2$, no set will contain more than $\ln 2m$ items. For simplicity of notation, we shall assume that T contains exactly $(1-\alpha)t$ items, and no set contains more than $\ln m$ items from T. (This assumption affects only low order terms in the approximation ratio.) Hence in the second round opt will pay at least $\min[(1-\alpha)t, k]\frac{\lambda}{\ln m}$, and in the two rounds combined opt pays at least $\alpha t + \min[(1-\alpha)t, k]\frac{\lambda}{\ln m}$. This is a piecewise linear function in α, and its minimum is achieved when α is either 0 or 1, or when $(1-\alpha)t = k$. It follows that opt pays at least $\min[t, \frac{k\lambda}{\ln m}]$.

Now we can analyze the approximation ratio of our algorithm. When $t < \frac{\lambda k}{\ln n}$, the algorithm pays at most $t \ln n \leq \lambda k$, which is a factor of $\ln n$ larger than t, and at most a factor of $\ln m$ larger than $\frac{k\lambda}{\ln m}$. Hence the approximation ratio in this case is at most $\max[\ln m, \ln n]$.

When $t \geq \frac{\lambda k}{\ln n}$, the algorithm pays nothing in the first round, and at most $\lambda k \leq t \ln n$ in the second round. Again, the approximation ratio can be seen to be at most $\max[\ln m, \ln n]$. ■

4.2 Robust Unweighted Vertex Cover and Edge Cover

A 2-approximation algorithm for robust unweighted vertex cover is as follows: Compute a maximum matching M in G, and let $|M|$ denote its size. If $|M| < \lambda k$, then we pick a vertex cover of size no larger than $2|M|$ in the first round (for example, by picking both endpoints of every edge in M) and nothing needs to be done in the second round. If $|M| \geq \lambda k$, we do nothing in first round and in the second round, we use a factor 2 approximation algorithm for vertex cover to cover the edges chosen by the adversary. The proof is left to Appendix.

Lemma 4 *The buy-at-once algorithm achieves an approximation ratio of 2 for unweighted robust vertex cover.*

Proof: For case 1, we show that $opt \geq |M|$, giving an approximation ratio no worse than 2. If $k \geq |M|$, then any solution would need to cover at least $|M|$ disjoint edges (in first and second round combined), and hence must cost at least $|M|$. If $k < |M|$, let M_1 denote the number of vertices picked by the optimal solution in the first round. If $|M| - M_1 \geq k$, opt pays in the second round $\lambda k > |M|$. If $|M| - M_1 < k$ then $opt \geq M_1 + \lambda(|M| - M_1) \geq |M|$, where the last inequality used the fact that $\lambda > 1$.

For case 2, our algorithm is in fact optimal. The inequalities $|M| \geq \lambda k$ and $\lambda \geq 1$ imply that the graph has a matching of size at least k. The algorithm will then pay λk in the second round. Everything that opt pays in the first round is sunk cost unless less than k matching edges remain for the second round, in which case we can use the inequality $M_1 + \lambda(|M| - M_1) \geq |M| \geq \lambda k$. It follows that also opt does nothing in the first round and pays λk in the second round. ■

Any algorithm that approximated robust vertex cover within a ratio better than 2 will need to approximate minimum vertex cover within a ratio better than 2 (e.g., by setting $\lambda = \infty$), and achieving this would resolve a long standing open problem.

The input of the unweighted edge cover problem is a graph with n vertices, m edges, a parameter k (for number of vertices to be chosen by adversary), and an inflation factor $\lambda > 1$. All edges have unit cost. Observe that the number of edges needed to cover ℓ vertices is always between $\ell/2$ and ℓ. This fact serves as a basis for a tight 2-approximation for the robust unweighted edge cover problem. The algorithm and the proof is left to the appendix. Moreover, we prove that if P≠NP, then the max-min variant and the robust two-stage variant of the edge cover problem cannot be approximated better than a factor 2. More interestingly, even the max-min fractional edge cover problem cannot be approximated better than a factor 2.

References

[1] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. In *SODA*, pages 577–586, 2004.

[2] R. Bar-Yehuda and S. Even. A linear time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2:198–203, 1981.

[3] D. Bertsimas and M. Sim. The price of robustness. *Operation Research*, 52:35–53.

[4] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming Series B*, 98:49–71.

[5] J. Birge and F. Louveaux. Introduction to stochastic programming. *Springer*, Berlin, 1997.

[6] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1:197–206, 1955.

[7] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. *FOCS*, 2005.

[8] I. Dinur, V. Guruswami, S. Khot, and O. Regev. New multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal of Computing*, 34(5):1129–1146, 2005.

[9] U. Feige. A threshold of $\ln n$ for approximating set cover. *JACM*, 45(4):634–652, 1998.

[10] D. Golovin, V. Goyal, and R. Ravi. Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems. *STACS*, 2006.

[11] A. Gupta, M. Pal, R. Ravi, and A. Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In *STOC*, pages 170–178, 2004.

[12] A. Gupta, R. Ravi, and A. Sinha. An edge in time saves nine: Lp rounding approximation algorithms for stochastic network design. *FOCS*, 45, 2004.

[13] J. Hastad. Clique is hard to approximate. In *FOCS*, pages 627–636, 1996.

[14] N. Immorlica, D. Karger, M. Minkoff, and V. S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In *SODA*, 2004.

[15] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *JACM*, 41(5):960–981, 1994.

[16] Y. Nikulin. Robustness in combinatorial optimization and scheduling theory: An annotated bibliography. *Technical Report SOR-91-13, Statistics and Operation Research*, `http://www.optimization-online.org/DB_FILE/2004/11/995.pdf`, 2004.

[17] R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *IPCO*, pages 101–115, 2004.

[18] D. Shmoys and S. Swamy. Stochastic optimization is (almost) as easy as deterministic optimization. In *FOCS*, 2004.

A Reformulating the problem using LP duality

As described in the previous section, the max-min problem can be formalized as follows: Given a first-stage solution $\vec{y^0}$ (which, with the help of Lemma 2, can be assumed to be integral), we need to find a subset F of size at most k for which the cost of the fractional covering problem

is maximized. This can be written as the following maximization problem. In this program, U denotes the set of clients *not* covered by the first-stage integral solution.

$$\begin{aligned} \underset{F:\ |F|=k}{\text{maximize}} \quad \min_{y} \quad & \sum_{r\in\mathcal{R}} c_r y_r \\ \text{subject to} \quad & \forall i \in F \cap U : \sum_{r:\ i\in r} y_r \geq 1 \\ & \forall r \in \mathcal{R} :\ y_r \geq 0 \end{aligned} \tag{4}$$

Using LP duality, we replace the minimization part of the above max-min mathematical program by its dual as follows:

$$\begin{aligned} \underset{F:\ |F|=k}{\text{maximize}} \quad \max_{\alpha} \quad & \sum_{i\in F\cap U} \alpha_i \\ \text{subject to} \quad & \forall r : \sum_{i\in r\cap U} \alpha_i \leq c_r \\ & \forall i \in F \cap U :\ \alpha_i \geq 0 \end{aligned} \tag{5}$$

This is a single maximization program, maximizing over the choice of F and α. We can write the general form of the above mathematical program for a general polytope $\mathcal{P}$ as follows: Given a polytope $\mathcal{P}$ with variables (x, y) where x is a vector of variables x_i for all $i \in \mathcal{F}$ and y is the vector of additional variables, we want to find a point in $\mathcal{P}$ so that the value of the sum of the k largest variables in x is maximized. Formally,

$$\begin{aligned} \max_{F\subseteq\mathcal{F}:|F|=k,x,y} \quad & \textstyle\sum_{i\in F} x_i \\ \text{s.t.} \quad & (x, y) \in \mathcal{P} \end{aligned} \tag{6}$$

Let n be the number of x variables in above program. Here, we give a general $\sqrt{n}$-approximation algorithm for the general problem defined above for any polytope $\mathcal{P}$. In particular, this gives a $\sqrt{n}$-approximation algorithm for the max-min problem for all our covering problems. The approximation algorithm is as follows:

- If $k \geq \sqrt{n}$, then output the solution of the following linear program:

$$\begin{aligned} \max_{x,y} \quad & \textstyle\sum_i x_i \\ \text{s.t.} \quad & (x, y) \in \mathcal{P} \end{aligned} \tag{7}$$

- If $k < \sqrt{n}$, then for each i, solve the following linear program,

$$\begin{aligned} \max_{x,y} \quad & x_i \\ \text{s.t.} \quad & (x, y) \in \mathcal{P} \end{aligned} \tag{8}$$

and output the maximum solution.

Theorem 4 *The above algorithm is a $\sqrt{n}$-approximation algorithm for the mathematical program (6).*

Proof: Let (x^*, y^*) and (x^0, y^0) be the optimal solution of program (6) and (7). If $k \geq \sqrt{n}$, then $\max_F \sum_{i \in F} x_i^0 \geq \frac{1}{\sqrt{n}} \sum_{i \in \mathcal{F}} x_i^0 \geq \frac{1}{\sqrt{n}} \sum_{i \in \mathcal{F}} x_i^* \geq \frac{1}{\sqrt{n}} \max_F \sum_{i \in F} x_i^*$. Therefore, the algorithm is a $\sqrt{n}$-approximation algorithm in this case. Now, let (x^i, y^i) for $i \in \mathcal{F}$ be the optimal solution of program (8) when the objective function is x_i. If $k < \sqrt{n}$, $\max_{i \in \mathcal{F}} x_i^i \geq \max_{i \in \mathcal{F}} x_i^* \geq \frac{1}{\sqrt{n}} \max_{F \subset \mathcal{F}} \sum_{i \in F} x_i^*$. Therefore, the output of the algorithm is a $\sqrt{n}$-approximate solution. ■

B Hardness of Max-min Fractional Set Cover

In this section, we give a strong inapproximability result for the max-min (fractional) set cover problem. The proof of Proposition 6 can be adopted easily for max-min fractional edge cover problem. This implies that for any $\epsilon > 0$, it is NP-hard to approximate the max-min fractional set cover problem (or even max-min fractional edge cover problem) within a factor better than $2-\epsilon$. This hardness ratio can be strengthened to nearly logarithmic factors for the fractional set cover problem, but proving this using current techniques seems to require assumptions stronger than $P \neq NP$. Picking $p(n) = \sqrt{n}$ in Theorem 5 shows that the max-min (fractional) set cover problem cannot be approximated within a ratio better than $\Omega(\frac{\log N}{\log \log N})$ (on instances of size N) unless 3SAT can be solved in time $2^{O(\sqrt{n})}$ (on instances of size n).

Theorem 5 *For every $0 < \delta < 1$ and $p(n) = n^\delta$, the max-min fractional set cover problem cannot be approximated within a ratio better than $\Omega(\frac{p(n)}{\log p(n)})$ on instances of size $N = 2^{O(p(n))}$ (in time polynomial in N), unless NP problems (say 3SAT) can be solved in time $2^{O(p(n))}$.*

Proof: The proof is presented for the integral set cover problem, but the approximation hardness applies also to the max-min fractional set cover problem, because in the *yes* instance the cover is disjoint.

The proof is based on the structure of instances of set cover that are generated by the reduction described in [9], and specifically, on the parameters given in Section 6 in [9]. Here we only sketch the proof.

Recall that in [9], the hardness of approximation result is based on a certain multiple-prover proof system. We shall need the number of provers (denoted in [9] by k) to be $p(n)$. (Hence one cannot use here the earlier [15] instead of [9].) In [9] it suffices that the number of parallel repetitions ℓ is logarithmic in the number of provers, hence we can have $\ell = O(\log(p(n)))$. (Remark: later work [8] used a version of a multilayered PCP which is somewhat simpler than the multiple prover system of [9]. This simpler version requires ℓ to grow at a faster rate than $p(n)$, and would result in weaker conclusions if used in the proof of Theorem 5.) This results in a set cover instance with $2^{O(p(n))}$ clients and sets.

Each subset in [9] would be an item in the max-min set cover problem. Each item in [9] would be a set in the max-min set cover problem. Note that in [9] all sets are of the same size, and there is a disjoint set cover for *yes* instances, say, by t sets. We shall set k for the max-min set cover problem to be equal to this t. Hence *yes* instances of [9] correspond to *yes* instances of set cover adversary's for which k clients can be selected that require k sets in order to be covered.

The property of *no* instances of [9] that we shall use is the following: for every $q < p(n)$, for every collection of $tq/p(n)$ sets, there is some item that belongs to $O(p(n)/q)$ of the sets. Extensions of the analysis in [9] can be used in order to prove this property, but this is omitted from the current paper.

The property above implies that for *no* instances in [9], for every collection of t sets there are $O(t\frac{\log(p(n))}{p(n)})$ clients that hit all the sets. This implies that in *no* instances of the max-min set cover problem, the optimum solution has value $O(t\frac{\log(p(n))}{p(n)})$. ■

C Tight Example for Lemma 3

In this section, we prove that the analysis of Lemma 3 is tight. In fact, we show that no buy-at-once algorithm can perform better than $\Omega(\ln m)$ in the case that m is much larger than n. Consider an instance of the two-stage robust set cover where the ground set consists of $n+n^{1/4}$ elements and $k = n^{1/4}$ and $\lambda = n^{1/4}$. The family of subsets in the set cover instance is the family of all subsets of size $n^{1/4}$ of set $\{1, 2, \ldots, n\}$ and all singleton sets $\{n+1\}, \{n+2\}, ..., \{n+n^{1/4}\}$. The optimal solution is to buy all singleton sets in advance and wait for the scenario. Since the adversary should choose a set of size $n^{1/4}$ of $\{1..n\}$ and this set is in the family of sets in the set cover instance, we can cover any scenario by buying one set at cost $\lambda = n^{1/4}$ later. Thus, the cost of the optimal solution is $2n^{1/4}$. If we do not buy any set in advance, the adversary selects $\{n+1, n+2, ..., n+n^{1/4}\}$ and we should pay $\lambda * k = n^{1/2}$ later. On the other hand, in order to cover all the elements in the first round, we need to buy at least $n^{3/4}$ sets to cover all elements of $1..n$, so the cost of covering all the elements in advance is at least $n^{3/4} + n^{1/4}$. Both of these cases are more than a factor of $n^{1/4} = \Omega(\log m)$ larger than the optimal solution (which is $n^{1/4}$).

In addition, observe that the term $\ln n$ in the approximation ratio in Lemma 3 cannot be improved by any polynomial-time algorithm (e.g., when $\lambda = \infty$), due to the hardness of approximating minimum set cover [9]. It is not clear whether the term $\ln m$ can be improved.

D Tight algorithm for robust unweighted edge cover problem

The factor 2 approximation for unweighted fractional edge cover problem is obtained by the following algorithm.

1. Compute a minimum edge cover (by computing a maximum matching) and let t be its size.

2. If $t < \lambda k$, pick this edge cover in the first round. Nothing needs to be done in the second round.

3. If $t \geq \lambda k$, do nothing in first round. In the second round, find a minimum edge cover that covers those vertices chosen by the adversary.

Lemma 5 *The buy-at-once algorithm achieves an approximation ratio of 2 for unweighted robust edge cover.*

Proof: Observe that the algorithm pays at most $\min[t, \lambda k]$. As for the optimal solution, assume that opt picks $t' < t$ edges in the first round.

- If k vertices remain uncovered, then in the second round opt pays at least $k\lambda/2$.
- If less than k vertices remain uncovered after the first round, the adversary can pick all these vertices in the second round. Their number is at least $t - t'$, and opt then pays at least $(t - t')\lambda/2$ in the second round. Altogether, opt pays

$$t' + \frac{\lambda}{2}(t - t') = \frac{1}{2}t + t(\frac{\lambda}{2} - \frac{1}{2}) + t'(1 - \frac{\lambda}{2}) \geq \frac{t}{2}$$

  where the last inequality follows because the sum is increasing in λ, and $\lambda \geq 1$.

Hence opt pays at least $\min[t/2, \lambda k/2]$, showing that the approximation ratio is no worse than 2. ■

The approximation ratio of the buy-at-once algorithm is no better than 2, as can be seen by a graph that has k isolated edges and a clique on the remaining $n - 2k$ vertices. Taking an edge cover in the first round costs $n/2$. Doing nothing in the first round costs λk in the second round. A hybrid solution can take the isolated edges in the first round and $k/2$ edges in the second round, paying $k + \lambda k/2$. The approximation ratio thus approaches 2 when $1 \ll \lambda \ll n/k$.

In fact, it is hard to get an approximation ratio better than 2 for the robust edge cover problem. This is based on the hardness of the max-min problem, namely, the problem of picking k vertices so as to maximize the minimum number of edges required in order to cover them. The parameters in the proof of the following proposition are chosen in such a way that the proof will later extend to showing hardness of approximation for robust edge cover.

Proposition 6 *For every $\epsilon > 0$, it is NP-hard to approximate the unweighted edge cover max-min problem within a ratio better than $2 - \epsilon$.*

Proof: The proof is by reduction from the maximum independent set problem. As shown in [13], for every sufficiently small $\epsilon > 0$, it is NP-hard to distinguish between the following classes of graphs:

Yes instances. Graphs on n vertices that contain an independent set of size ϵn.

No instances. Graphs on n vertices that contain no independent set of size $\epsilon^5 n$.

A graph G serves as an instance of the edge cover adversary's problem, with $k = \epsilon n$.

On yes instances, one can select k vertices that form an independent set in G, and then k edges are needed in order to cover them.

On no instances, whenever there are more than $\epsilon^5 k$ vertices, two of them share an edge in G. It follows that any selection of k vertices can be covered by $\epsilon^5 k + (1-\epsilon^5)k/2 < k/(2-\epsilon)$ edges. ■

To show that the robust edge cover is hard, take ℓ graphs, one of which is a *yes* instance of Proposition 6 and the rest are *no* instances. We make the following *indistinguishability assumption*: picking $\ell/10$ of the graphs in such a way that includes the yes instance is hard. The indistinguishability assumption can be shown to hold under various complexity assumptions, such as the computational indistinguishability of encryptions of 0 and 1 in computationally secure encryption schemes (details omitted). Set the parameters such that ϵ (from the proof of Proposition 6) is very small, $\lambda = 1/\epsilon^2$, $\ell = 1/\epsilon^3$, and $k = \epsilon n$. Then a good solution covers the *yes* graph in the first round, and then in the second round the graph induced on the k vertices selected by the adversary cannot contain an independent set larger than $\ell\epsilon^5 n = \epsilon^2 n \ll k$. Hence opt pays a total of at most roughly $n + \lambda k/2 \simeq n/2\epsilon$. But under the indistinguishability assumption, any polynomial time algorithm will need to either spend $\epsilon n \frac{\ell}{10} = n/10\epsilon^2 \gg n/2\epsilon$ in the first round, or spend roughly λk in the second round. In any case, the approximation ratio is not better than 2 (up to low order terms that depend on ϵ).