Chapter 5

Set Cover

The Set Cover problem this chapter deals with is again a very simple to state – yet quite general – NP-hard combinatorial problem. It is widely applicable in sometimes unexpected ways. The problem is the following: We are given a set U (called *universe*) of n elements, a collection of sets $\mathcal{S} = \{S_1, \dots, S_k\}$ where $S_i \subseteq U$, and a cost function $c : \mathcal{S} \to \mathbb{R}^+$. The task is to find a minimum cost subcollection $\mathcal{S}' \subseteq \mathcal{S}$ that *covers* U, i.e., such that $\cup_{S \in \mathcal{S}'} S = U$.

Example 5.1. Consider this instance: $U = \{1, 2, 3\}$, $\mathcal{S} = \{S_1, S_2, S_3\}$ with $S_1 = \{1, 2\}$, $S_2 = \{2, 3\}$, $S_3 = \{1, 2, 3\}$ and cost $c(S_1) = 10$, $c(S_2) = 50$, and $c(S_3) = 100$. These collections cover U: $\{S_1, S_2\}$, $\{S_3\}$, $\{S_1, S_3\}$, $\{S_2, S_3\}$, $\{S_1, S_2, S_3\}$. The cheapest one is $\{S_1, S_2\}$ with cost equal to 60.

For each set S, we associate a variable $x_S \in \{0, 1\}$ that indicates of we want to choose S or not. We may thus write solutions for Set Cover as a vector $x \in \{0, 1\}^k$. With this, we write Set Cover as a mathematical program.

Problem 5.1 Set Cover

Instance. Universe U with n elements, collection $\mathcal{S} = \{S_1, \dots, S_k\}$, $S_i \subseteq U$, a cost function $c : \mathcal{S} \to \mathbb{R}$.

Task. Solve the problem

$$\begin{aligned} \text{minimize} \quad & \operatorname{val}(x) = \sum_{S \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1 \quad e \in U, \\ & x_S \in \{0, 1\} \quad S \in \mathcal{S}. \end{aligned}$$

Define the *frequency* of an element to be the number of sets it is contained in. Let f denote the frequency of the most frequent element. In this chapter we present several algorithms that either achieve approximation ratio $O(\log n)$ or f. Why are we interested in a variety algorithms? Is one algorithm not sufficient? Yes, but here the focus is on the *techniques* that yield these algorithms.

5.1 Greedy Algorithm

The GREEDY algorithm follows the natural approach of iteratively choosing the most cost-effective set and remove all the covered elements until all elements are covered. Let C be the set of elements already covered at the beginning of an iteration. During this iteration define the *cost-effectiveness* of a set S as $c(S)/|S - C|$, i.e., the average cost at which it covers new elements. For later reference, the algorithm sets the *price* at which it covered an element equal to the cost-effectiveness of the covering set. Further recall that $H_n = \sum_{i=1}^{n} 1/i$ is called the *n-th Harmonic number* and that $\log n \leq H_n \leq \log n + 1$.

Algorithm 5.1 GREEDY

```
Input.   Universe $U$ with $n$ elements, collection $\mathcal{S} = \{S_1, \ldots, S_k\}$, $S_i \subseteq U$, a cost function
         $c : \mathcal{S} \to \mathbb{R}$.

Output.  Vector $x \in \{0,1\}^k$

Step 1. $C = \emptyset$, $x = 0$.

Step 2. While $C \neq U$ do the following:

        (a) Find the most cost-effective set in the current iteration, say $S$.
        (b) Set $x_S = 1$ and for each $e \in S - C$ set price$(e) = c(S)/|S - C|$.
        (c) $C = C \cup S$.

Step 3. Return $x$.
```

Theorem 5.2. *The* GREEDY *algorithm is an H_n-approximation algorithm for the* SET COVER *problem.*

It is an exercise to show that this bound is tight.

Direct Analysis

The following lemma is crucial for the proof of the approximation-guarantee. Number the elements of U in the order in which they were covered by the algorithm, say $e_1, \ldots, e_n$. Let x^* be an optimum solution.

Lemma 5.3. *For each $i \in \{1, \ldots, n\}$,* $\text{price}(e_i) \leq \text{val}(x^*)/(n - i + 1)$.

Proof. In any iteration, the leftover sets of the optimal solution x^* can cover the remaining elements at a cost of at most $\text{val}(x^*)$. Therefore, among these, there must be one set having cost-effectiveness of at most $\text{val}(x^*)/|U - C|$. In the iteration in which element e_i was covered, $U - C$ contained at least $n - i + 1$ elements. Since e_i was covered by the most cost-effective set in this iteration, we have that

$$\text{price}(e_i) \leq \frac{\text{val}(x^*)}{|U - C|} \leq \frac{\text{val}(x^*)}{n - i + 1}$$

which was claimed. $\square$

Proof of Theorem 5.2. Since the cost of each set is distributed evenly among the new elements covered, the total cost of the set cover picked is

$$\mathrm{val}(x) = \sum_{i=1}^{n} \mathrm{price}(e_i) \leq \mathrm{val}(x^*) H_n,$$

where we have used Lemma 5.3. □

Dual-Fitting Analysis

Here we will give an alternative analysis of the Greedy algorithm for Set Cover. We will use the *dual fitting* method, which is quite general and helps to analyze a broad variety of combinatorial algorithms.

For sake of exposition we consider a minimization problem, but the technique works similarly for maximization. Consider an algorithm Alg which does the following:

(1) Let (P) be an integer programming formulation of the problem of interest. We are interested in its optimal solution x^*, respectively its objective value $\mathrm{val}(x^*)$. Let (D) be the dual of a linear programming relaxation of (P).

(2) The algorithm Alg computes a feasible solution x for (P) and a "solution" y for (D), where we allow that y is *infeasible* for (D). But the algorithm has to ensure that

$$\mathrm{val}(x) \leq \overline{\mathrm{val}}(y),$$

where val is the objective function of (P) and $\overline{\mathrm{val}}$ is the objective function of (D).

(3) Now divide the entries of y by a certain quantity α until $y' = y/\alpha$ *is* feasible for (D). (The method of dual fitting is applicable only if this property can be ensured.) Then $\overline{\mathrm{val}}(y')$ is a lower bound for $\mathrm{val}(x^*)$ by weak duality, i.e.,

$$\overline{\mathrm{val}}(y') \leq \mathrm{val}(x^*)$$

by Lemma 3.8.

(4) Putting these things together, we obtain the approximation guarantee of α by

$$\mathrm{val}(x) \leq \overline{\mathrm{val}}(y) = \overline{\mathrm{val}}(\alpha y') = \alpha \overline{\mathrm{val}}(y') \leq \alpha \mathrm{val}(x^*).$$

Now we apply this recipe to Set Cover and consider the Greedy algorithm. For property (1) we use our usual formulation

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1 \quad e \in U, \\ & x_S \in \{0,1\} \quad S \in \mathcal{S}. \end{aligned} \tag{P}$$

When we relax the constraints $x_S \in \{0,1\}$ to $0 \leq x_S \leq 1$ and dualize the corresponding linear program we find

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in U} y_e, \\ \text{subject to} \quad & \sum_{e \in S} y_e \leq c(S) \quad S \in \mathcal{S}, \\ & y_e \geq 0. \end{aligned} \tag{D}$$

This dual can be derived purely mechanically (by applying the primal-dual-definition and rewriting constraints if needed), but this program also has an intuitive interpretation. The constraints of (D) state that we want to "pack stuff" into each set S such that the cost $c(S)$ of each set is not exceeded, i.e., the sets are not overpacked. We seek to maximize the total amount packed.

How about property (2)? The algorithm Greedy computes a certain feasible solution x for (P), i.e., a solution $x_S = 1$ if the algorithm picks set S and $x_S = 0$ otherwise. What about the vector y? Define the following vector: For each $e \in U$ set $y_e = \text{price}(e)$, where $\text{price}(e)$ is the value computed during the execution of the algorithm.

By construction of the algorithm we have

$$\text{val}(x) = \sum_{S \in \mathcal{S}} c(S) x_S = \sum_{e \in U} \text{price}(e) = \sum_{e \in U} y_e = \overline{\text{val}}(y),$$

i.e., Greedy satisfies property (2) of the dual fitting method (even with equality).

For property (3) the following result is useful.

Lemma 5.4. *For every $S \in \mathcal{S}$ we have that*

$$\sum_{e \in S} y_e \leq H_n c(S).$$

Proof. Let $S \in \mathcal{S}$ with, say, m elements. Consider these in the ordering the algorithm covered them, say, $e_1, \ldots, e_m$. At the iteration when e_i gets covered S contains $m - i + 1$ uncovered elements. Since Greedy chooses the most cost-effective set we have that

$$\text{price}(e_i) \leq \frac{c(S)}{m - i + 1},$$

i.e., the cost-effectiveness of the set the algorithm chooses can only be smaller than the cost-effectiveness of S. (Be aware that "smaller" is "better" here.)

Summing over all elements gives

$$\sum_{e \in S} y_e = \sum_{i=1}^{m} \text{price}(e_i) \leq c(S) \sum_{i=1}^{m} \frac{1}{m - i + 1} = c(S) H_m \leq c(S) H_n$$

as claimed. □

Now we are in position to finalize the dual-fitting analysis using property (4).

Proof of Theorem 5.2. Define the vector $y' = y / H_n$, where y is defined above. Observe that for each set $S \in \mathcal{S}$ we have

$$\sum_{e \in S} y'_e = \sum_{e \in S} \frac{y_e}{H_n} = \frac{1}{H_n} \sum_{e \in S} y_e \leq c(S)$$

using Lemma 5.4. That means y' is feasible for (D). Using the property (4) of the dual fitting method proves the approximation guarantee of at most H_n. □

5.2 Primal-Dual Algorithm

The primal-dual schema introduced here is the method of choice for designing approximation algorithms because it often gives algorithms with good approximation guarantees and good running times. After introducing the ideas behind the method, we will use it to design a simple factor f algorithm, where f is the frequency of the most frequent element.

The general idea is to work with an LP-relaxation of an NP-hard problem and its dual. Then the algorithm iteratively changes a primal and a dual solution until the relaxed primal-dual complementary slackness conditions are satisfied.

Primal-Dual Schema

Consider the following primal program:

$$\begin{aligned} \text{minimize} \quad & \operatorname{val}(x) = \sum_{j=1}^{n} c_j x_j, \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m, \\ & x_j \geq 0 \quad j = 1, \ldots, n. \end{aligned}$$

The dual program is:

$$\begin{aligned} \text{maximize} \quad & \overline{\operatorname{val}}(y) = \sum_{i=1}^{m} b_i y_i, \\ \text{subject to} \quad & \sum_{i=1}^{m} a_{ij} y_i \leq c_j \quad j = 1, \ldots, n, \\ & y_i \geq 0 \quad i = 1, \ldots, m. \end{aligned}$$

Most known approximation algorithms using the primal-dual schema run by ensuring one set of conditions and suitably relaxing the other. We will capture both situations by relaxing both conditions. If primal conditions are to be ensured, we set $\alpha = 1$ below, and if dual conditions are to be ensured, we set $\beta = 1$.

Primal Complementary Slackness Conditions. Let $\alpha \geq 1$. For each $1 \leq j \leq n$:

$$\text{either} \quad x_j = 0 \quad \text{or} \quad c_j/\alpha \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j.$$

Dual Complementary Slackness Conditions. Let $\beta \geq 1$. For each $1 \leq i \leq m$:

$$\text{either} \quad y_i = 0 \quad \text{or} \quad b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i.$$

Lemma 5.5. *If x and y are primal and dual feasible solutions respectively satisfying the complementary slackness conditions stated above, then*

$$\operatorname{val}(x) \leq \alpha\beta\overline{\operatorname{val}}(y).$$

Proof. We calculate directly using the slackness conditions and obtain

$$\begin{aligned}\mathrm{val}(x) = \sum_{j=1}^{n} c_j x_j &\le \alpha \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j \\ &= \alpha \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \le \alpha\beta \sum_{i=1}^{m} b_i y_i = \overline{\mathrm{val}}(y)\end{aligned}$$

which was claimed. □

The algorithm starts with a primal infeasible solution and a dual feasible solution; usually these are $x = 0$ and $y = 0$ initially. It iteratively improves the feasibility of the primal solution and the optimality of the dual solution ensuring that in the end a primal feasible solution is obtained and all conditions stated above, with a suitable choice for α and β, are satisfied. The primal solution is always extended integrally, thus ensuring that the final solution is integral. The improvements to the primal and the dual go hand-in-hand: the current primal solution is used to determine the improvement to the dual, and vice versa. Finally, the cost of the dual solution is used as a lower bound on the optimum value, and by Lemma 5.5, the approximation guarantee of the algorithm is $\alpha\beta$.

Primal-Dual Algorithm

Here we derive a factor f approximation algorithm for Set Cover using the primal-dual schema. For this algorithm we will choose $\alpha = 1$ and $\beta = f$. We will work with the following primal LP for Set Cover

$$\begin{aligned}\text{minimize} \quad & \mathrm{val}(x) = \sum_{S \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \ge 1 \quad e \in U, \\ & x_S \ge 0 \quad S \in \mathcal{S}.\end{aligned}$$

and its dual

$$\begin{aligned}\text{maximize} \quad & \overline{\mathrm{val}}(y) = \sum_{e \in U} y_e, \\ \text{subject to} \quad & \sum_{e \in S} y_e \le c(S) \quad S \in \mathcal{S}, \\ & y_e \ge 0 \quad e \in U.\end{aligned}$$

For these LPs the primal and dual complementary slackness conditions are:

Primal Complementary Slackness Conditions. For each $S \in \mathcal{S}$:

$$\text{either} \quad x_S = 0 \quad \text{or} \quad \sum_{e \in S} y_e = c(S).$$

A set S will be said to be *tight* if $\sum_{e \in S} y_e = c(S)$. So, this condition states that: “Pick only tight sets into the cover.”

Dual Complementary Slackness Conditions. For each $e \in U$:

$$\text{either} \quad y_e = 0 \quad \text{or} \quad \sum_{S: e \in S} x_S \leq f.$$

Since we will find a 0/1 solution for x, these conditions are equivalent to: "Each element having non-zero dual value can be covered at most f times." Since each element is in at most f sets, this condition is trivially satisfied for all elements.

These conditions suggest the following algorithm:

Algorithm 5.2 Primal-Dual Set Cover

Input. Universe U with n elements, collection $\mathcal{S} = \{S_1, \ldots, S_k\}$, $S_i \subseteq U$, a cost function $c : \mathcal{S} \to \mathbb{R}$.

Output. Vector $x \in \{0,1\}^k$

```
Step 1. $x = 0$, $y = 0$. Declare all elements uncovered.
Step 2. Unless all elements are covered, do:
    (a) Pick an uncovered element, say $e$, and raise $y_e$ until some set goes tight.
    (b) Pick all tight sets $S$ in the cover, i.e., set $x_S = 1$.
    (c) Declare all the elements occuring in these sets as covered.
Step 3. Return $x$.
```

Theorem 5.6. *The algorithm* Primal-Dual Set Cover *is a f-approximation algorithm for* Set Cover.

Proof. At the end of the algorithm, there will be no uncovered elements. Further no dual constraint is violated since we pick only tight sets S into the cover and no element $e \in S$ will later on be a candidate for increasing y_e. Thus, the primal and dual solutions will both be feasible. Since they satisfy the primal and dual complementary slackness conditions with $\alpha = 1$ and $\beta = f$, by Lemma 5.5, the approximation guarantee is f. □

Example 5.7. A tight example for this algorithm is provided by the following set system. The universe is $U = \{e_1, \ldots, e_{n+1}\}$ and $\mathcal{S}$ consists of $n-1$ sets $\{e_1, e_n\}, \ldots, \{e_{n-1}, e_n\}$ of cost 1 and one set $\{e_1, \ldots, e_{n+1}\}$ of cost $1+\varepsilon$ for some small $\varepsilon > 0$. Since e_n appears in all n sets, this system has $f = n$.

Suppose the algorithm raises y_{e_n} in the first iteration. When y_{e_n} is raised to 1, all sets $\{e_i, e_n\}$, $i = 1, \ldots, n-1$ go tight. They are all picked in the cover, thus covering the elements $e_1, \ldots, e_n$. In the second iteration $y_{e_{n+1}}$ is raised to ε and the set $\{e_1, \ldots, e_{n+1}\}$ goes tight. The resulting set cover has cost $n + \varepsilon$, whereas the optimum cover has cost $1 + \varepsilon$.

5.3 LP-Rounding Algorithms

The central idea behind algorithms that make use of the *LP-rounding* technique is as follows: Suppose you have an LP-relaxation of a certain NP-hard problem. Then you can solve this optimally and try to "round" the optimal fractional solution to an integral one.

Here we derive a factor f approximation algorithm for Set Cover but this time by rounding the fractional solution of an LP to an integral solution (instead of the primal-dual schema). We consider our usual LP relaxation for Set Cover

$$\begin{aligned} \text{minimize} \quad & \mathrm{val}(x) = \sum_{s \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S: e \in S} x_S \geq 1 \quad e \in U, \\ & x_S \geq 0 \quad S \in \mathcal{S}. \end{aligned}$$

Simple Rounding Algorithm

The idea of the algorithm below is to include those sets S into the cover for which the corresponding value z_S in the optimal solution z of the LP is "large enough".

Algorithm 5.3 Simple Rounding Set Cover

Input. Universe U with n elements, collection $\mathcal{S} = \{S_1, \dots, S_k\}$, $S_i \subseteq U$, a cost function $c : \mathcal{S} \to \mathbb{R}$.

Output. Vector $x \in \{0, 1\}^k$

Step 1. Set $x = 0$, solve the LP relaxation below, and call the optimal solution z.

$$\begin{aligned} \text{minimize} \quad & \mathrm{val}(x) = \sum_{S \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S: e \in S} x_S \geq 1 \quad e \in U, \\ & x_S \geq 0 \quad S \in \mathcal{S}. \end{aligned}$$

Step 2. For each set S set $x_S = 1$ if $z_S \geq 1/f$.

Step 3. Return x.

Theorem 5.8. *The algorithm* Simple Rounding Set Cover *is an f-approximation algorithm for* Set Cover.

Proof. Let x be the solution returned by the algorithm and z be the optimal solution of the LP. Consider an arbitrary element $e \in U$. Since e is in at most f sets, one of these sets must be picked to the extent of at least $1/f$ in the fractional solution z. If this were not the case then $\sum_{S:e \in S} z_S < \sum_{S:e \in S} 1/f \leq f \cdot 1/f = 1$ yields a contradiction to the feasibility of z. Thus e is covered due to the definition of the algorithm and x is hence a feasible cover. We further have $x_S \leq f z_S$ and thus

$$\mathrm{val}(x) \leq f \mathrm{val}(z) \leq f \mathrm{val}(x^*)$$

where x^* is an optimal solution for the Set Cover problem. □

Randomized Rounding

Another natural idea for rounding fractional solutions is to use randomization: For example, for the above relaxation, observe that the values z_S are between zero and one. We may thus interpret these values as probabilities for choosing a certain set S.

Here is the idea of the following algorithm: Solve the LP-relaxation optimally and call the solution z. With probability z_S include the set S into the cover.

This basic procedure yields a vector x with expected value equal to the optimal fractional solution value but might not cover all the elements. We thus repeat the procedure "sufficiently many" times and include a set into our cover if it was included in any of the iterations. We will show that $O(\log n)$ many iterations suffice yielding an $O(\log n)$-approximation algorithm.

Algorithm 5.4 Randomized Rounding Set Cover

Input. Universe U with n elements, collection $\mathcal{S} = \{S_1, \dots, S_k\}$, $S_i \subseteq U$, a cost function $c : \mathcal{S} \to \mathbb{R}$.

Output. Vector $x \in \{0,1\}^k$

Step 1. Set $x = 0$, solve the LP relaxation below, and call the optimal solution z.

$$\begin{aligned} \text{minimize} \quad & \text{val}(x) = \sum_{S \in \mathcal{S}} c(S) x_S, \\ \text{subject to} \quad & \sum_{S : e \in S} x_S \geq 1 \quad e \in U, \\ & x_S \geq 0 \quad S \in \mathcal{S}. \end{aligned}$$

Step 2. Repeat $\lceil 3 \log n \rceil$ times: For each set S set $x_S = 1$ with probability z_S.

Step 3. Return x.

Theorem 5.9. *With probability at least* $1 - 1/n^2$ *the algorithm* Randomized Rounding Set Cover *returns a feasible solution, which is expected* $\lceil 3 \log n \rceil$*-approximate for* Set Cover.

Proof. Let z be an optimal solution for the LP. We estimate the probability that an element $e \in U$ is covered in one iteration in Step 2. Let e be contained in m sets and let $z_1, \dots, z_m$ be the probabilities given in the solution z. Since e is fractionally covered we have $z_1 + \cdots + z_m \geq 1$. With easy but tedious calculus we see that – under this condition – the probability for e being covered is minimized when the z_i are all equal, i.e., $z_1 = \cdots = z_m = 1/m$:

$$\Pr[x_S = 1] = 1 - (1 - z_1) \cdot \dots \cdot (1 - z_m) \geq 1 - \left(1 - \frac{1}{m}\right)^m \geq 1 - \frac{1}{e}.$$

Each element is covered with probability at least $1 - 1/e$. But maybe we have not covered all elements after $\lceil 3 \log n \rceil$ iterations. The probability that the element e is not covered at the end of the algorithm, i.e., after $\lceil 3 \log n \rceil$ iterations is

$$\Pr[e \text{ is not covered}] \leq \left(\frac{1}{e}\right)^{\lceil 3 \log n \rceil} \leq \frac{1}{n^3}.$$

Thus the probability that there is an uncovered element is at most

$$\sum_{e\in U} \Pr\left[e \text{ is not covered}\right] \leq n \cdot \frac{1}{n^3} \leq \frac{1}{n^2}.$$

Hence the retured solution x is feasible with probability at least $1 - 1/n^2$.

Consider a single iteration in Step 2 and let $y \in \{0,1\}^k$ be the vector that indicates which sets are included in this particular iteration. For each set S let $y_S = 1$ with probability z_S. Then we have

$$\mathbb{E}\left[\text{val}(y)\right] = \sum_{S\in\mathcal{S}} \mathbb{E}\left[c(S)y_S\right] = \sum_{S\in\mathcal{S}} c(S)\Pr\left[y_S = 1\right] = \sum_{S\in\mathcal{S}} c(S)z_S = \text{val}(z).$$

Now we consider all iterations in Step 2 and clearly have

$$\mathbb{E}\left[\text{val}(x)\right] \leq \lceil 3\log n\rceil \cdot \mathbb{E}\left[\text{val}(y)\right] \leq \lceil 3\log n\rceil \cdot \text{val}(z) \leq \lceil 3\log n\rceil \cdot \text{val}(x^*),$$

where x^* is an optimal solution for Set Cover. So, the algorithm returns a feasible solution, with probability at least $1-1/n^2$, whose expected value is $\lceil 3\log n\rceil$-approximate. □

The proof above shows that the algorithm is a $\lceil 3\log n\rceil$-approximation in expectation. But we can actually state that the approximation ratio is $4 \cdot \lceil 3\log n\rceil$ with probability around 3/4. Use Markov's inequality $\Pr\left[X > t\right] \leq \mathbb{E}\left[X\right]/t$ to show

$$\Pr\left[\text{val}(x) > 4 \cdot \lceil 3\log n\rceil \cdot \text{val}(z)\right] \leq \frac{\mathbb{E}\left[\text{val}(x)\right]}{4 \cdot \lceil 3\log n\rceil \cdot \text{val}(z)} \leq \frac{1}{4}$$

The probability that either not all elements are covered or the obtained solution has value larger than $4 \cdot \lceil 3\log n\rceil$ times the optimal value is at most $1/n^2 + 1/4 \leq 1/2$ for all $n \geq 2$. Thus we have to run the whole algorithm at most two times in expectation to actually get a $4 \cdot \lceil 3\log n\rceil$-approximate solution.