Approximation algorithms for partial covering problems (Submission to ICALP 2001, Track A)

Rajiv Gandhi * Samir Khuller † Aravind Srinivasan ‡

Abstract

We study the generalization of covering problems to *partial covering*. Here we wish to cover only a desired number of elements, rather than covering all elements as in standard covering problems. For example, in k-set cover, we wish to choose a minimum number of sets to cover at least k elements. For k-set cover, if each element occurs in at most f sets, then we derive a primal-dual f-approximation algorithm (thus implying a 2-approximation for k-vertex cover) in polynomial time. In addition to its simplicity, this algorithm has the advantage of being parallelizable. For instances where each set has cardinality at most three, we obtain an approximation of 4/3. We also present better-than-2-approximation algorithms for k-vertex cover on bounded degree graphs, and for vertex cover on expanders of bounded *average* degree. We obtain a polynomial-time approximation scheme for k-vertex cover on planar graphs, and for covering points in R^d by disks.

Key Words and Phrases: Approximation algorithms, partial covering, set cover, vertex cover, primal-dual methods, randomized rounding.

*Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965. Fax: +1-301-405-6707. E-mail: gandhi@cs.umd.edu.

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF Award CCR-9820965 and an NSF CAREER Award CCR-9501355. Fax: +1-301-405-6707. E-mail: samir@cs.umd.edu.

‡Bell Labs, Lucent Technologies, 600-700 Mountain Avenue, Murray Hill, NJ 07974. Fax: +1-908-582-3340. E-mail: srin@research.bell-labs.com.

1. Introduction

Covering problems are widely studied in discrete optimization: basically, these problems involve picking a least-cost collection of sets to cover elements. Classical problems in this framework include the general set cover problem, of which a widely studied special case is the vertex cover problem. (The vertex cover problem is a special case of set cover in which the edges correspond to elements and vertices correspond to sets; in this set cover instance, each element is in exactly two sets.) Both these problems are NP-hard and polynomial-time approximation algorithms for both are well studied. For set cover see [12, 26, 29]. For vertex cover see [6, 7, 13, 21, 22, 30].

In this paper we study the generalization of "covering" to "partial covering" [27, 31]. Specifically, in k-set cover, we wish to find a minimum number (or, in the weighted version, a minimum weight collection) of sets that cover at least k elements. When k is the total number of elements, we obtain the regular set cover problem; similarly for k-vertex cover. (We sometimes refer to k-set cover as "partial set cover", and k-vertex cover as "partial vertex cover"; the case where k equals the total number of elements is referred to as "full coverage".) This generalization is motivated by the fact that real data (in clustering for example) often has errors (also called outliers). Thus, discarding the (small) number of constraints posed by such errors/outliers is permissible.

Suppose we need to build facilities to provide service within a fixed radius to a certain fraction of the population. We can model this as a partial set cover problem. The main issue in partial covering is: which k elements should we choose to cover? If such a choice can be made judiciously, we can then invoke a set cover algorithm. Other facility location problems have recently been studied in this context

[11].

We begin our discussion by focusing on vertex cover and k-vertex cover. A very simple approximation algorithm for unweighted vertex cover (full coverage) is attributed to Gavril and Yannakakis, and can be found, e.g., in [14]: take a maximal matching and pick all the matched vertices as part of the cover. The size of the matching (number of edges) is a lower bound on the optimal vertex cover, and this yields a 2-approximation. This simple algorithm fails for the partial covering problem, since the lower bound relies on the fact that all the edges have to be covered. The first approximation algorithm for k-vertex cover was given in [9]. Their 2-approximation algorithm is based on a linear programming (LP) formulation: suitably modifying and rounding the LP's optimal solution. A faster approximation algorithm achieving the same factor of 2 was given by Hochbaum [24] in which the key idea is to relax the constraint limiting the number of uncovered elements and searching for the dual penalty value. More recently, Bar-Yehuda [8] studied the same problem and gave a 2-approximation for k-vertex cover based on the elegant "local ratio" method.

1.1. Problem Definitions and Previous Work

- **k-Set Cover:** Given a set $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, a collection $\mathcal{S}$ of subsets of $\mathcal{T}$, $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, a cost function $c : \mathcal{S} \rightarrow \mathcal{Q}^+$, and an integer k, find a minimum cost sub-collection of $\mathcal{S}$ that covers at least k elements of $\mathcal{T}$.

  Previous Results: For the full coverage version, a $\ln n + 1$ approximation was proposed by Johnson [26] and Lovász [29]. This analysis of the greedy algorithm can be improved to $H(\Delta)$ (see the proof in [14]) where Δ is the size of the largest set[1]. Chvátal [12] generalized this to the case when sets have costs. Slavík [32] shows the same bound for the partial cover problem. When $\Delta = 3$, Duh and Fürer [15] gave a 4/3-approximation for the full coverage version. They extended this result to get a bound of $H(\Delta) - \frac{1}{2}$ for full coverage. When an element belongs to at most f sets Hochbaum [21] gives a f-approximation.

- **k-Vertex Cover:** Given a graph $G = (V, E)$, a cost function $c : V \rightarrow \mathcal{Q}^+$, and an integer k, find a minimum cost subset of V that covers at least k edges of G.

  Previous Results: For the partial coverage version several 2-approximation algorithms are known (see [9, 24, 8]).

- **Geometric Covering Problem:** Given n points in a plane, find a minimally sized set of disks of diameter D that would cover at least k points.

  Previous Results: The full coverage version is well-studied. This problem is motivated by the location of emergency facilities as well as from image processing (see [23] for additional references). For the special case of geometric covering problems, Hochbaum and Maass [25] have developed a polynomial approximation scheme.

1.2. Methods and Results

- **k-Set Cover:** For the special case when each element is in at most f sets, we combine a primal-dual algorithm [13, 18] with a thresholding method to obtain an f-approximation.

  One advantage of our method, in addition to its simplicity, is that it can be easily parallelized by changing the algorithm slightly. The resulting approximation factor is $f(1 + \epsilon)$, where $\epsilon > 0$ is any desired constant. The number of parallel rounds is $O(\log n)$ once we fix $\epsilon > 0$. The number of processors required in linear in the problem size. This is the first parallel approximation algorithm for any partial covering problem.

  Our general method is as follows: we first "guess" the cost of the maximum cost set in the optimal solution. We then modify the original cost function by raising the costs of some of the sets to infinity, so that these sets are never chosen in our solution. This leads to dual feasible solutions for the instance with modified costs (which we use as a lower bound) that may be *infeasible* for the original problem. However, if

[1] $H(k) \doteq \sum_{i=1}^{k} 1/i = \ln k + \Theta(1)$.

we only raise the costs of sets that are guaranteed to not be in the optimal solution, we do not change the optimal IP solution. Hence the dual feasible solution for this modified instance is still a lower bound for the optimal IP.

To parallelize the above algorithm, at each "round" when we update the dual variables we include all sets whose constraints are "almost" tight. This is similar to the method described in [28], but does not work directly. The main difficulty is that in each round many sets are chosen. As long as we have covered fewer than the target number of elements there is no problem in accounting for the cost of the chosen sets. However, in the last round (when we cross the required threshold), we have to carefully pick an appropriate subset of the chosen sets.

For set cover where the sets have cardinality at most Δ there are results (starting from [16, 19]) by Duh and Fürer [15] for set cover (full coverage) that improve the $H(\Delta)$ bound to $H(\Delta)-\frac{1}{2}$. For example, for $\Delta = 3$ they present a $\frac{4}{3}$ ($= H(3)-\frac{1}{2}$) approximation using "semi-local" optimization rather than a $\frac{11}{6}$-approximation obtained by the simple greedy algorithm.

For the case $\Delta = 3$, we can obtain a $\frac{4}{3}$ bound for the partial coverage case. This does suggest that perhaps the $H(\Delta) - \frac{1}{2}$ bound can be obtained as well. This would improve Slavík's result [32].

- **k-Vertex Cover:** By switching to a probabilistic approach to rounding the LP relaxation of the problem, we obtain improved results for k-vertex cover, where we wish to choose a minimum number of vertices to cover at least k edges. An outstanding open question for vertex cover (full coverage) is whether the approximation ratio of 2 is best-possible; see, e.g., [17]. Thus, it has been an issue of much interest to identify families of graphs for which *constant-factor approximations better than* 2 (which we denote by Property (P)) are possible. In the full coverage case, Property (P) is true for graphs of bounded *maximum* degree; see, e.g., [20]. How can we extend such a result? Could Property (P) hold for graphs of constant *average* degree? This is probably not the case, since this would imply that Property (P) holds for all graphs. (Given a graph G with n vertices, suppose we add a star with $\Theta(n^2)$ vertices to G by connecting the center of the star by an edge to some vertex of G. The new graph has bounded average degree, and its vertex-cover number is one more than that of G.) However, we show that for *expander* graphs of bounded average degree, Property (P) is indeed true. We also show Property (P) for k-vertex cover in the case of bounded maximum degree and arbitrary k; this is the first Property (P) result for k-vertex cover, to our knowledge. Our result on expanders uses an expectation analysis and the expansion property. Expectation analysis is insufficient for our result here on k-vertex cover, and we show that a random process behaves close to its mean on bounded-degree graphs: the degree-boundedness helps us show that many sub-events related to the process are (pairwise) independent. We also present certain new results for multi-criteria versions of k-vertex cover.

- **Geometric Covering:** There is a polynomial approximation scheme based on dynamic programming for the full coverage version [25]. For the partial coverage version since we do not know which k points to cover, we have to define a new dynamic program. This makes the implementation of the approximation scheme due to Hochbaum and Maass [25] more complex, although it is still a polynomial-time algorithm.

- **k-Vertex Cover for Planar Graphs:** We are able to use the dynamic programming ideas developed for the geometric covering problem to design a polynomial-time approximation scheme (PTAS) for k-vertex cover for planar graphs. This is based on Baker's method for the full covering case [3].

2. k-Set Cover

The k-Set Cover problem can be formulated as an integer program as follows. We assign a binary variable x_j for each $S_j \in \mathcal{S}$ i.e $x_j \in \{0,1\}$. In this formulation, $x_j = 1$ iff set S_j belongs to the cover. A binary variable y_i is assigned to each element $t_i \in \mathcal{T}$.

$y_i = 1$ iff t_i is not covered. Clearly, there could be at most $n-k$ such uncovered elements.

The corresponding LP relaxation can be obtained by letting the domain of x_j and y_i be $0 \leq x_j, y_i \leq 1$. Notice that the upper bound on x_j and y_i is unnecessary and is thus dropped.

$$\min \quad \sum_{j=1}^{m} c(S_j) \cdot x_j$$

subject to

$$y_i + \sum_{j:t_i \in S_j} x_j \geq 1, i = 1, 2, \ldots, n$$

$$\sum_{i=1}^{n} y_i \leq n-k$$

$$x_j \geq 0, j = 1, 2, \ldots, m$$

$$y_i \geq 0, i = 1, 2, \ldots, n$$

The dual LP contains a variable u_i (for each element $t_i \in \mathcal{T}$) corresponding to each of the first n constraints in the above LP. The dual variable z corresponds to the $(n+1)^{th}$ constraint in the above LP formulation. The dual LP is to maximize $\sum_{i=1}^{n} u_i - (n-k) \cdot z$ subject to: (i) $\sum_{i:t_i \in S_j} u_i \leq c(S_j)$ for $j = 1, 2, \ldots, m$, (ii) $0 \leq u_i \leq z$ for $i = 1, 2, \ldots, n$, and (iii) $z \geq 0$.

The algorithm SetCover (pseudo-code can be found in Figure 1 in Appendix A) does the following. The algorithm "guesses" the set with the highest cost in the optimal solution by considering each set in turn to be the highest cost set. For each set that is chosen, to be the highest cost set, say S_j, S_j along with all the elements it contains is removed from the instance and is included as part of the cover for this guess of the highest cost set. The cost of all sets having a higher cost than $c(S_j)$ is raised to ∞. $I_j = (\mathcal{T}^j, \mathcal{S}^j, c', k_j)$ is the modified instance. SetCover then calls Primal-Dual on I_j which uses a primal dual approach [18] to return a set cover for I_j. In Primal-Dual, the dual variables u_i are increased for all $t_i \in \mathcal{T}^j$ until there exists a set S_i such that $\sum_{i:t_i \in S_i} u_i = c'(S_i)$. Sets are chosen this way until the cover is feasible. The algorithm then chooses the minimum cost solution among the m solutions found. For the purpose of clarity of exposition in the following pseudo-code (refer to Figure 1 in Appendix A) we assume that costs of all sets in $\mathcal{S}$ are distinct.

Theorem 2.1: SetCover$(\mathcal{T}, \mathcal{S}, c, k)$ *returns a f-approximate solution, where f is the highest frequency of any element i.e. an element appears in at most f sets.*

Proof: Let OPT refer to an optimal solution. We will use OPT to mean either an optimal solution or the cost of an optimal solution. The meaning will be clear from the context in which it is used. Let I be the given instance of the problem. Let I_j refer to the modified instance of the problem i.e. $I_j = (\mathcal{T}^j, \mathcal{S}^j, c', k_j)$. Let S_h be the set with the highest cost in OPT. Let $OPT(I_h)$ be the optimal integer solution for the instance I_h. $OPT = OPT(I_h) + c(S_h)$. Let $DFS(I_h)$ refer to the dual feasible solution for the instance I_h. Again, $DFS(\cdot)$ will be used to mean the dual feasible solution or the cost of the dual feasible solution. $DFS(I_h)$ may not be a feasible solution[2] to the instance $(\mathcal{T}^j, \mathcal{S}^j, c, k_j)$ (note the original cost function). However, since $DFS(I_h) \leq OPT(I_h)$ and $OPT = OPT(I_h) + c(S_h)$, we have $DFS(I_h) + c(S_h) \leq OPT$. SC_h is the set cover chosen by our algorithm during the iteration $j = h$. Let S_l be the last set chosen. Let $ASC = SC_h \setminus \{S_l\}$ (ASC stands for Almost Set Cover). Note that since S_h is the costliest set, $c(S_l) \leq c(S_h)$. Let T_c represent the set of points covered by ASC. Let $T_u = \mathcal{T}^h \setminus T_c$ represent the set of uncovered elements. Since the algorithm chooses a cover, SC, of the lowest cost, cost(SC) is upper bounded by cost(SC_h).

$$\begin{aligned}
\text{cost}(SC) &\leq \text{cost}(SC_h) = \sum_{S_k \in SC_h} c'(S_k) \\
&= \sum_{S_k \in SC_h} c(S_k) \\
&= \sum_{S_k \in ASC} c(S_k) + c(S_l) + c(S_h) \\
&\leq \sum_{S_k \in ASC} \sum_{i:t_i \in S_k} u_i + c(S_h) + c(S_h) \\
&= \sum_{i:t_i \in T_c} u_i . |\{t_i\} \cap ASC| + 2 \cdot c(S_h)
\end{aligned}$$

[2] This is because we relax the constraints for the dual problem.

$$
\begin{aligned}
&\le f \cdot \sum_{i:t_i \in T_c} u_i + 2 \cdot c(S_h) \\
&= f \cdot (\sum_{i=1}^{n} u_i - \sum_{i:t_i \in T_u} u_i) + 2 \cdot c(S_h) \\
&\le f \cdot (\sum_{i=1}^{n} u_i - |T_u| \cdot z) + 2 \cdot c(S_h) \\
&\le f \cdot (\sum_{i=1}^{n} u_i - (n-k) \cdot z) + 2 \cdot c(S_h) \\
&\le f \cdot (DFS(I_h) + c(S_h)) \le f \cdot OPT.
\end{aligned}
$$

□

Corollary 2.2: SETCOVER(E, V, c, k) *gives a 2-approximate solution for k-Vertex Cover.*

2.1. Parallel Implementation of Partial Set Cover Algorithm

We assume as before that each element belongs to at most f sets. The framework for the algorithm is the same as the one we described for the primal-dual serial algorithm. The parallel algorithm runs in "rounds". In each round, we simultaneously raise all dual variables u_i corresponding to the uncovered elements. In the serial algorithm we pick one set in each iteration, namely a set S_j such that $(\sum_{i:t_i \in S_j} u_i = c'(S_j))$. (Recall that c' denotes the modified cost function.) We change this step in the algorithm to pick *all* sets such that $(c'(S_j) - \sum_{i:t_i \in S_j} u_i \le \epsilon c'(S_j))$. (This condition will let us prove that $c'(S_j) \le (\sum_{i:t_i \in S_j} u_i)/(1-\epsilon)$.) We stop as soon as we have covered at least k elements.

Suppose the algorithm covers at least k elements after ℓ rounds. The main problem is that in the last round we can include many sets simultaneously, while we can afford to include only a few. Let δ be the number of elements that we need to cover *after* round $\ell - 1$. To select an appropriate subset of the chosen sets, we need to pick a minimal collection of chosen sets that cover at least δ elements. To accomplish this, we order the sets chosen in the last iteration arbitrarily. Now compute in parallel the "effective" number of elements each set covers and choose a minimal collection based on the fixed ordering. (All these steps can be implemented in parallel using prefix computations.)

Theorem 2.3: *The parallel algorithm runs in $(1 + f\log(1/\epsilon))(1+\log n)$ rounds. The number of processors is linear in the size of the input. The parallel algorithm produces a $\frac{f}{1-\epsilon}$-approximate solution.*

3. Set Cover for Small Sets

Problem: Given a collection C of small subsets of a base set U. Each small subset in the collection has size at most Δ, and their union is U. The objective is to find a minimum size sub-collection that covers at least k elements.

Here we have the original partial set cover instance with the additional information that the sets are of "small" size, i.e., Δ is small. We obtain an approximation factor of $4/3$ for the case when $\Delta = 3$ using the the idea of (s, t) semi-local optimization [15]. This technique consists of inserting up to s 3-sets (sets of size 3) and deleting up to t 3-sets from the current cover. Then the elements that are not covered by the 3-sets (already existing ones + the newly added) are covered optimally using 2-sets and 1-sets. This can be solved in polynomial time using maximum matching [16]. The vertices are the uncovered elements of U and the edges are the admissible 2-sets. The 2-sets corresponding to the maximum matching edges and the 1-sets corresponding to the vertices not covered by the maximum matching form an optimum covering. We will order the quality of a solution by the number of sets in the cover and among two covers of the same size we choose the one with fewer 1-sets and if the covers have the same size and neither cover has a 1-set we choose the one that covers more elements.

The algorithm starts with any solution. One solution can be obtained as follows. Choose a maximal collection of disjoint 3-sets. Cover the remaining elements optimally using 2-sets and 1-sets. Perform semi-local $(2, 1)$ improvements until no improvement is possible.

The proof for the bound of $4/3$ for full coverage does not extend to the partial coverage version. For the full coverage, to prove the lower bound on the optimal solution Duh and Fürer construct a graph G in which the vertices are the sets chosen by OPT and the edges are 1-sets and 2-sets of the approximate solution. They prove that G can not have more than one cycle and hence argue that the total number of 1-sets and 2-sets in the solution is a lower bound on OPT. This works well for the full coverage version but breaks down for the partial covering problem. For the partial covering case G having at most one cycle is a necessary but not a sufficient condition to prove the lower bound.

In the full version of the problem, to bound the number of 1-sets in the solution they construct a bipartite graph with the two sets of vertices corresponding to the sets chosen by the approximate solution and OPT. If a set corresponding the approximate solution intersects a set corresponding to OPT in m elements then there are m edges between their corresponding vertices in the graph. In each component of the graph they show that the number of 1-sets of the solution in that component is at most the number of 1-sets of OPT in that component. This is clearly not the case in the partial covering case. We obtain a bound on the number of 1-sets as a side effect of the proof for the lower bound on OPT.

The detailed proof of our theorem is shown in Appendix B.

Theorem 3.1: *The semi-local $(2,1)$-optimization algorithm for 3-set partial covering problem produces a solution that is within $\frac{4}{3}OPT + 1$.*

4. Probabilistic Approaches for k-Vertex Cover

We now present a randomized rounding approach to the natural LP relaxation of k-vertex cover. Analyzed in three different ways, this leads to three new approximation results mentioned in §1: relating to vertex cover (full coverage) for expander graphs of constant average degree, k-vertex cover on bounded-degree graphs, and multi-criteria k-vertex cover problems. We first describe the basic method and prove some probabilistic properties thereof, and then consider the three applications.

The k-vertex cover problem on a graph $G = (V, E)$ can be formulated as an integer program as follows. We assign binary variables x_j for each $v_j \in V$ and $z_{i,j}$ for each $(i,j) \in E$. In this formulation, $x_j = 1$ iff vertex v_j belongs to the cover, and $z_{i,j} = 1$ iff edge (i,j) is covered. The corresponding LP relaxation can be obtained by letting each x_j and $z_{i,j}$ lie in $[0,1]$.

$$\min \quad \sum_{j=1}^{n} x_j$$

subject to

$$x_i + x_j \geq z_{i,j}, \quad (i,j) \in E \tag{1}$$

$$\sum_{(i,j)\in E} z_{i,j} \geq k \tag{2}$$

$$x_j, z_{i,j} \in [0,1], \quad \forall i,j.$$

Our basic approximation recipe will be as follows. The LP relaxation is solved optimally. Let $\{x_i^*\}$, $\{z_{i,j}^*\}$ denote an optimal LP solution, and let $\lambda = 2(1-\epsilon)$, where $\epsilon \in [0, 1/2]$ is a parameter that will be chosen based on the application. Let $S_1 = \{v_j | x_j^* \geq 1/\lambda\}$, and $S_2 = V - S_1$. Include all the vertices in S_1 as part of our cover, and mark the edges incident on vertices in S_1 as covered. Now independently for each $j \in S_2$, round x_j to 1 with a probability of λx_j^*, and to 0 with a probability of $1 - \lambda x_j^*$. Let W be the random variable denoting the number of covered edges at this point. If $W < k$, we choose any $k - W$ uncovered edges and cover them by arbitrarily choosing one end-point for each of them.

We now introduce some notation to analyze the above process. Throughout, we let $\Pr[\cdot]$ and $\mathbf{E}[\cdot]$ denote probability and expectation, respectively. Let y^* represent the optimal objective function value of the LP, and define $S_0 \subseteq S_1$ by $S_0 = \{v_j : x_j^* = 1\}$. Let y_F^* and y_P^* be the contribution to y^* of the vertices in S_0 and $V - S_0$ respectively. Denote by $U_{i,j}$ the event that edge (i,j) is uncovered. Let C_1 be the cost of the solution produced by our randomized scheme *before* the step of covering $k - W$ edges if necessary, and let C_2 be the cost incurred in covering these $k - W$ edges, if any. The total cost C is of course $C_1 + C_2$; thus, $\mathbf{E}[C] = \mathbf{E}[C_1] + \mathbf{E}[C_2]$. Now, it is easy to check that $\mathbf{E}[C_1] \leq y_F^* + \lambda y_P^*$, and that $\mathbf{E}[C_2] \leq \mathbf{E}[\max\{k - W, 0\}]$. So we have

$$\mathbf{E}[C] \leq y_F^* + \lambda y_P^* + \mathbf{E}[\max\{k - W, 0\}]. \tag{3}$$

The following lemma on the statistics of W will be useful. As usual, let $\overline{\mathcal{E}}$ denote the complement of an event $\mathcal{E}$.

Lemma 4.1: *(i)* $\mathbf{E}[W] \geq k(1-\epsilon^2)$. *(ii) Suppose the graph G has maximum degree d. Then, the variance Var$[W]$ of W is at most $d\mathbf{E}[W]$.*

Proof: (i) Consider any edge (i,j). Now if $x_i^* \geq 1/\lambda$ or $x_j^* \geq 1/\lambda$, $\Pr[U_{i,j}] = 0$; otherwise, $\Pr[U_{i,j}] = (1-\lambda x_i^*)(1-\lambda x_j^*)$. Consider the latter case. Since $x_i^* + x_j^* \geq z_{i,j}^*$, we can check that for any given $z_{i,j}^* \in [0,1]$, $(1-\lambda x_i^*)(1-\lambda x_j^*)$ is maximized when $x_i^* = x_j^* = z_{i,j}^*/2$. Hence,

$$\begin{aligned} \Pr[U_{i,j}] &\leq (1-\lambda z_{i,j}^*/2)^2 \\ &= (1-(1-\epsilon)z_{i,j}^*)^2 \\ &\leq 1 - z_{i,j}^*(1-\epsilon^2). \end{aligned}$$

Thus, since $\mathbf{E}[W] = \sum_{(i,j)\in E} \Pr[\overline{U_{i,j}}]$, we get

$$\mathbf{E}[W] \geq \sum_{(i,j)\in E} z_{i,j}^*(1-\epsilon^2) \geq k(1-\epsilon^2).$$

(ii) We have $W = \sum_{(i,j)\in E} \overline{U_{i,j}}$. It is also an easy calculation to see that if a random variable W' is the sum of *pairwise independent* random variables each of which lies in $[0,1]$, then $\mathrm{Var}[W'] \leq \mathbf{E}[W']$. However, the terms $U_{i,j}$ that constitute W do have some dependent pairs: if edges (i,j) and (i',j') share an endpoint, then $\overline{U_{i,j}}$ and $\overline{U_{i',j'}}$ are dependent (positively correlated). Define γ to be the sum, over all unordered pairs of distinct edges (i,j) and (i',j') that share an end-point, of $\Pr[\overline{U_{i,j}} \wedge \overline{U_{i',j'}}]$. Using the above observations and the definition of variance, a moment's reflection shows that $\mathrm{Var}[W]$ is upper-bounded by $\mathbf{E}[W] + \gamma$. Now, for any events A and B,

$$\begin{aligned} \Pr[A \wedge B] &\leq \min\{\Pr[A], \Pr[B]\} \\ &\leq (\Pr[A] + \Pr[B])/2. \end{aligned}$$

Thus, the term "$\Pr[\overline{U_{i,j}} \wedge \overline{U_{i',j'}}]$" in γ is at most $(\Pr[\overline{U_{i,j}}] + \Pr[\overline{U_{i',j'}}])/2$. Finally, since each edge has at most $2(d-1)$ other edges that share an end-point with it, we get that

$$\gamma \leq \sum_{(i,j)\in E} (2(d-1)/2)\cdot \Pr[\overline{U_{i,j}}] = (d-1)\mathbf{E}[W].$$

So, $\mathrm{Var}[W] \leq \mathbf{E}[W] + \gamma \leq d\mathbf{E}[W]$. □

4.1. Vertex Cover on Expanders

Suppose we have a vertex cover problem; i.e., k-vertex cover with $k = m$. The LP relaxation here has "1" in place of "$z_{i,j}$" in (1), and does not require the variables $z_{i,j}$ and the constraint (2). We focus here on the case of expander graphs of constant average degree. That is, for some constants c and d, we are studying graphs where: (i) the number of edges m is at most nd, and (ii) for any set X of vertices with $|X| \leq n/2$, at least $c|X|$ vertices outside X have a neighbor in X.

Since $k = m$, it is well-known that we can efficiently compute an optimal solution x^* to the LP with all entries lying in $\{0, 1/2, 1\}$. Let $H = \{v_j | x_j^* = 1/2\}$ and $F = \{v_j | x_j^* = 1\}$. Also, since $W \leq k = m$ always holds, $\mathbf{E}[\max\{k-W, 0\}] = \mathbf{E}[k-W] \leq m\epsilon^2$, by Lemma 4.1(i). Thus, (3) shows that $\mathbf{E}[C]$ is at most $y_F^* + 2(1-\epsilon)y_H^* + m\epsilon^2$. (The overall approach of: (i) conducting a randomized rounding and then doing a greedy fixing of violated constraints, and (ii) using an equality such as our "$\mathbf{E}[\max\{k-W,0\}] = \mathbf{E}[k-W]$" here, is suggested in [33]. We next show how the expansion property is useful in bounding $\mathbf{E}[C]$ well. However, in the context of *partial* covering, an equality such as "$\mathbf{E}[\max\{k-W,0\}] = \mathbf{E}[k-W]$" does not hold; so, as discussed in §4.2 and §4.3, new analysis approaches are employed there.) Choosing $\epsilon = y_H^*/m$ to minimize $y_F^* + 2(1-\epsilon)y_H^* + m\epsilon^2$, we get

$$\mathbf{E}[C] \leq y_H^*(2 - y_H^*/m) + y_F^*. \tag{4}$$

Case I: $|H| \leq n/2$. Note that the edges incident on vertices in H must have their other end-point in F; otherwise the LP constraint on such edges will be violated. Since G is an expander, $|F| \geq c \cdot |H|$. Also, $y_F^* = |F|$ and $y_H^* = |H|/2$. So, since $y^* = y_H^* + y_F^*$, we have $y_H^* = y^*/(1+a)$ for some $a \geq 2c$. We can now use (4) to get

$$\mathbf{E}[C] \leq 2y_H^* + y_F^* = (2 - a/(1+a))y^*;$$

i.e., at most $(2 - 2c/(1+2c))y^*$ since $a \geq 2c$.

Case II: $|H| > n/2$. So, we have $y_H^* \geq n/4$. Bound (4) shows that $\mathbf{E}[C] \leq (2 - y_H^*/m)y^*$; we have $m \leq nd$ by assumption. So, $\mathbf{E}[C] \leq (2 - 1/(4d))y^*$ in this case.

Thus we see that $\mathbf{E}[C] \leq [2 - \min\{2c/(1+2c), 1/(4d)\}] \cdot y^*$. In other words, for the family of expanders of constant average degree, we can get a constant-factor approximation that is strictly better than 2.

4.2. k-Vertex Cover: Bounded-Degree Graphs

We now show that any constant d, k-vertex cover on graphs of maximum degree at most d can be approximated to within $2(1-\Omega(1/d))$, for any value of the parameter k. We also demonstrate that the integrality gap in this case is at most $2(1-\Omega(1/d))$. We start with a couple of tail bounds that will be of use now, as well as in §4.3. First, suppose X is a sum of independent random variables X_i each of which lies in $[0,1]$; let $\mathbf{E}[X] = \mu$. Then for any $\delta \in [0,1]$, the Chernoff bound shows that $\Pr[X \geq \mu(1+\delta)]$ is at most $e^{-\mu\delta^2/3}$. We will also need tail bounds for certain non-independent situations. Suppose X is a random variable with mean μ and variance σ^2; suppose $a > 0$. Then, the well-known Chebyshev's inequality states that $\Pr[|X-\mu| \geq a]$ is at most σ^2/a^2. We will need stronger tail bounds than this, but only on X's *one-sided* deviations (say, below its mean). We will use the Chebyshev-Cantelli inequality (see, e.g., [1]), which shows that $\Pr[X - \mu \leq -a] \leq \sigma^2/(\sigma^2 + a^2)$.

We now analyze the performance of our basic algorithm (of randomized rounding of the LP solution followed by a simple covering of a sufficient number of edges), for the k-vertex cover problem on graphs with maximum degree bounded by some given constant d. The notation remains the same. The main problem in adopting the method of §4.1 here is as follows. Since k equaled m there, we could use the equality $\mathbf{E}[\max\{k - W, 0\}] = \mathbf{E}[k - W]$, thus substantially simplifying the analysis. Here, however, such an equality is not true; furthermore, $\mathbf{E}[\max\{X,Y\}] \geq \max\{\mathbf{E}[X], \mathbf{E}[Y]\}$ for any pair of random variables X, Y. (In fact, the two sides of this inequality may differ a lot. For instance, suppose X is the sum of n independent random variables, each of which is uniformly distributed on $\{-1, 1\}$; let Y be the constant 0. Then the r.h.s. is zero, while the l.h.s. is $\Theta(\sqrt{n})$.) Instead, we take recourse to the Chebyshev-Cantelli inequality, and use Lemma 4.1(ii).

We now claim that

$$\Pr[W \leq (k(1-\epsilon^2) - 2\sqrt{kd})] \leq 1/5. \qquad (5)$$

This is trivially true if $k < 4d$, since $\Pr[W \geq 0] = 1$. So suppose $k \geq 4d$. Lemma 4.1 and the Chebyshev-Cantelli inequality show that $\mu \doteq \mathbf{E}[W] \geq k(1-\epsilon^2)$, and that $\Pr[W \leq \mu - 2\sqrt{d\mu}] \leq 1/5$. Subject to $\mu \geq k(1-\epsilon^2) \geq 4d(1-\epsilon^2)$, $\mu - 2\sqrt{d\mu}$ is minimized when $\mu = k(1-\epsilon^2)$. Thus we have (5).

Next, for a suitably large constant c_0, we can assume that $k \geq c_0 d^5$. (Any optimal solution has size at most k, since in an optimal solution, every vertex should cover at least one new edge. So if k is bounded by a constant–such as $c_0 d^5$–then we can find an optimal solution in polynomial time by exhaustive search.) Also, by adding all the constraints of the LP and simplifying, we get that $y^* \geq k/d$. Thus, letting $\delta = 1/(3d)$, a Chernoff bound shows that immediately after the randomized rounding, the probability of having more than $2y^*(1-\epsilon)(1+\delta)$ vertices in our initial cover is at most 1/5 (if the constant c_0 is chosen large enough). Recall (5). So, with probability at least $1 - (1/5 + 1/5) = 3/5$, the final cover we produce is of size at most

$$2y^*(1-\epsilon)(1+\delta) + k\epsilon^2 + 2\sqrt{kd}.$$

We now choose $\epsilon = y^*(1+\delta)/k$; since $y^* \geq k/d \geq c_0 d^4$ with c_0 sufficiently large, some simplification shows that the final cover size is at most $2y^*(1-\Omega(1/d))$.

4.3. k-Vertex Cover: Multiple Criteria

We now briefly consider multi-criteria k-vertex cover problems on arbitrary graphs. Here, we are given a graph G and, as usual, have to cover at least k edges. We are also given ℓ "weight functions" w_i, and want a cover that is "good" w.r.t. all of these. More precisely, suppose we are given vectors

$$w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,n}) \in [0,1]^n, \ i = 1, 2, \ldots, \ell$$

and a fractional solution x^* to the k-cover problem on G. Define $y_i^* = \sum_j w_{i,j} x_j^*$ for $1 \leq i \leq \ell$. We aim for an integral solution z such that for *each* i, $y_i = \sum_j w_{i,j} z_j$ is not "much above"

y_i^*. Multi-criteria optimization has recently received much attention, since participating individuals/organizations may have differing objective functions, and we may wish to (reasonably) simultaneously satisfy all of them if possible. The result we show here is that if $y_i^* \geq c_1 \log^2(\ell + n)$ for all i (where c is a sufficiently large constant), then we can efficiently find an integral solution z with $y_i \leq 2(1 + 1/\sqrt{\log(\ell + n)})y_i^*$ for each i. Please see Appendix C for a short description of the analysis.

5. Geometric Packing and Covering

Problem: Given n points in a plane, find the smallest number of (identical) disks of diameter D that would cover at least k points.

A polynomial time approximation scheme exists for the case when $k = n$ (full covering). The algorithm uses a strategy, called the *shifting strategy*. The strategy is based on a divide and conquer approach. The area, I, enclosing the set of given points is divided into strips of width D. Let l be the shifting parameter. Groups of l consecutive strips, resulting in strips of width lD are considered. For any fixed subdivision of I into strips of width D, there are l different ways of partitioning I into strips of width lD. The l partitions are denoted by $S_1, S_2, \ldots, S_l$.

The solution to cover all the points is obtained by finding the solution to cover the points for each partition, $S_j, 1 \leq j \leq l$, and then choosing a minimum cost solution. A solution for each partition is obtained by finding a solution to cover the points in each strip (of width lD) of that partition and then taking the union of all such solutions. To obtain a solution for each strip, the shifting strategy is reapplied to each strip. This results in the partition of each strip into "squares" of side length lD. As will be shown later, there exists an optimal covering for such squares.

We modify the use of shifting strategy for the case when $k \leq n$ (partial covering). The obstacle in directly using the shifting strategy for the partial covering case is that we do not know the number of points that an optimal solution covers in each strip of a partition. This is not a problem with the full covering case because we know that any optimal solution would have to cover all the points within each strip of a partition. For the partial covering, this problem is overcome by "guessing" the number of points covered by an optimal solution in each strip. This is done by finding a solution for every possible value for the number of points that can be covered in each strip and storing each solution. A formal presentation is given below.

Let A be any algorithm that delivers a solution to cover the points in any strip of width lD. Let $A(S_i)$ be the algorithm that applies A to each strip of the partition S_i and outputs the union of all disks in a feasible solution. We will find such a solution for each of the l partitions and output the minimum.

Consider a partition S_i containing p strips of width lD. Let n_j be the number of points in strip j. Let n_j^{OPT} be the number of points covered by OPT in strip j. Since we do not know n_j^{OPT}, we will find feasible solutions to cover points for all possible values of n_j^{OPT}. Note that $0 \leq n_j^{OPT} \leq k'_j = \min(k, n_j)$. We use *dynamic programming* to solve our problem. The recursive formulation is as follows:

$$C(x, y) = \min_{0 \leq i \leq k'_x} (D_i^x + C(x-1, y-i))$$

where $C(x, y)$ denotes the number of disks needed to cover y points in strips $1..x$ and D_i^x is the number of disks needed to cover i points in strip x. Computing $C(p, k)$ gives us the desired answer.

For each strip s, for $0 \leq i \leq k'_s$, D_i^s can be calculated by recursive application of the algorithm to the strip s. We partition the strip into squares of side length lD. We can find optimal coverings of points in such a square by exhaustive search. With $O(l^2)$ disks of diameter D we can cover $lD \times lD$ square compactly, thus we never need to consider more disks for one square. Further, we can assume that any disk that covers at least two of the given points has two of these points on its border. Since there are only two ways to draw a circle of given diameter through two given points, we only have to consider $2\begin{pmatrix} n' \\ 2 \end{pmatrix}$ possible disk positions where n' is the number of given points in the considered square. Thus, we have to check for at most $O(n'^{2(l\sqrt{2})^2})$ arrangements of disks.

Let Z^A be the value of the solution delivered by algorithm A. The shift algorithm S_A is defined for a local algorithm A. Let r_B denote the performance ratio of an algorithm B; that is, r_B is defined as the supremum of $Z^B / |OPT|$ over all problem instances.

Lemma 5.1: *$r_{S_A} \leq r_A(1+\frac{1}{l})$ where A is the local algorithm and l is the shifting parameter.*

Proof: Consider a partition S_i with p strips of width lD. We have that $r_A \geq \frac{Z_j^A}{|OPT_j|}$, where j runs over all strips in partition S_i and $|OPT_j|$ is the number of disks in an optimal cover of n_j^{OPT} points in strip j. It follows that $Z^{A(S_i)} \leq r_A \sum_{j \in S_i} |OPT_j|$

Let OPT be the set of disks in an optimal solution and $OPT^{(1)}, \ldots, OPT^{(l)}$ the set of disks in OPT covering points in two adjacent lD strips in $1, 2, \ldots, l$ shifts respectively. Thus we have

$$\begin{aligned}
\sum_{j \in S_i} |OPT_j| &\leq |OPT| + \left|OPT^{(i)}\right| \\
Z^{S_A} &= \min_{i=1..l} Z^{A(S_i)} = \frac{1}{l}\sum_{i=1}^{l} Z^{A(S_i)} \\
&\leq \frac{1}{l} r_A (\sum_{i=1}^{l} \sum_{j \in S_i} |OPT_j|) \\
&\leq \frac{1}{l} r_A (\sum_{i=1}^{l} |OPT| + \left|OPT^{(i)}\right|).
\end{aligned}$$

There can be no disk in the set OPT that covers points in two adjacent strips in more than one shift partition. Therefore, the sets $OPT^{(1)}, \ldots, OPT^{(l)}$ are disjoint and can add up to at most OPT. It follows that $\sum_{i=1}^{l}(|OPT| + \left|OPT^{(i)}\right|) \leq (l+1)|OPT|$. Substituting this in the bound above for Z^{S_A} we get that Z^{S_A} is at most $\frac{1}{l} r_A.(l+1)|OPT| = r_A.(1+\frac{1}{l})|OPT|$. □

Theorem 5.2: *The above algorithm yields a PTAS with performance ratio at most $(1+\frac{1}{l})^2$.*

Proof: We use two nested applications of the shifting strategy to solve the problem. The above lemma applied to the first application of the shifting strategy would relate the performance ratio of the final solution, r_{S_A}, to that of the solution for each strip, r_A.

$$r_{S_A} \leq r_A(1+1/l) \tag{6}$$

The lemma when applied to the second application of shifting strategy relates r_A to the performance ratio of the solution to each square, say $r_{A'}$. Thus, $r_A \leq r_{A'}(1+1/l)$. But since we obtain an optimal solution for each square, $r_{A'} = 1$. Bound (6) shows that $r_{S_A} \leq (1+1/l)^2$. □

6. k-Vertex Cover for Planar Graphs

Full vertex cover for planar graphs of bounded tree-width can be computed optimally in linear time.This immediately leads to a PTAS for planar graphs by a combination of results of Baker and Bodlaender [3, 4]. Baker gives a general framework that constructs a PTAS for any problem which can be solved optimally for l-outerplanar graphs — planar graphs where all nodes have a path of length $\leq l$ to a node on the outermost face [3]. This method is based on the *shifting strategy* that is similar to the method used for geometric covering in the previous section. Bodlaender [4] proves that any l-outerplanar graph has tree-width at most $3l-1$. Vertex cover for graphs of bounded tree-width can be solved optimally in polynomial time, thus implying such a solution for graphs that are l-outerplanar for a fixed constant l.

First we describe how to create a collection of decompositions of a planar graph G into a set of l-outerplanar graphs. Let $d(v)$ = shortest path length from v to any node on the outer face of G. For each value of $\delta = 0, 1 \ldots, (l-1)$, we generate a decomposition as follows. Let $G_i = (V_i, E_i)$ be the i^{th} l-outerplanar graph for a fixed δ. $V_i = \{v | li + \delta \leq d(v) \leq l(i+1) + \delta$ and $E_i = \{(u,v) | u \in V_i$ and $v \in V_i\}$. There are l different ways of creating these decompositions, one for each δ. These correspond to the l partitions $S_1, S_2, \ldots, S_l$ in the geometric covering case. In the full covering case, the algorithm is to find a vertex cover for each of the l decompositions and then to take the best solution. The vertex cover for each decomposition is the union of the solutions to each l-outerplanar graph in the decomposition. As in the case of geometric covering the obstacle in directly using the above algorithm for the partial covering case is that we do not know the number of edges covered by OPT in each outerplanar graph. As in the previous section, we overcome this obstacle by "guessing" the number of points covered by an optimal solution in each l-outerplanar graph. The dynamic programming formulation in the previous section can be used once the following correspondence between the various entities is noted. The vertices in our case correspond to the disks and the edges correspond to the points to be covered. An l-outerplanar graph corresponds to the strip of width lD. As in the previous case, we still have l such decompositions. In the geometric covering problem the solution to each strip is calculated by recursively applying the shifting strategy to each strip. In this case, an optimal solution for the partial vertex cover for l-outerplanar graphs is computed as shown in the next section.

We now give a linear-time algorithm for bounded tree-width graphs (if the graph has tree-width l, then the time required for the algorithm to run will be exponential in l but linear in the size of the graph). The following definition is standard (see, e.g., [4]).

Definition 1: *Let $G = (V, E)$ be a graph. A* **tree-decomposition** *of G is a pair $(\{X_i \mid i \in I\}, T = (I, F))$, where $\{X_i \mid i \in I\}$ is a family of subsets of V and $T = (I, F)$ is a tree with the following properties:*

1. $\bigcup_{i \in I} X_i = V$.
2. *For every edge $e = (v, w) \in E$, there is a subset X_i, $i \in I$, with $v \in X_i$ and $w \in X_i$.*
3. *For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \bigcap X_k \subseteq X_j$.*

The **tree-width** *of a tree-decomposition $(\{X_i \mid i \in I\}, T)$ is $\max_{i \in I}\{|X_i| - 1\}$. The tree-width of a graph is the smallest value k such that the graph has a tree-decomposition with tree-width k.*

Many problems are known to have linear time algorithms on graphs with constant tree-width, and there are frameworks for automatically generating a linear time algorithm, given a problem specification in a particular format [2, 5]. The partial vertex cover problem can be solved by successively using solutions to the problem of finding the maximum number of edges that can be covered using p vertices. The value of p can be selected by doing a binary search on the set of vertices which reduces in half with every successive solution. This problem can be expressed in the formalism of [5] as: $\max |E_1| \, [V_1 \subset V \wedge |V_1| \leq p \wedge E_1 = \mathrm{IncE}(V_1)]$, which states that we want to maximize the set of edges that can be covered by any subset V_1 of V such that the size of V_1 is at most p.

Theorem 6.1 follows from Lemma 5.1 and the fact that $r_A = 1$.

Theorem 6.1: *The above algorithm gives a PTAS with a performance ratio $\leq (1 + \frac{1}{l})$.*

References

[1] N. Alon, R. Boppana and J. H. Spencer. An asymptotic isoperimetric inequality. *Geometric and Functional Analysis*, 8:411–436, 1998.

[2] S. Arnborg, J. Lagergren, D. Seese. Easy Problems for Tree-Decomposable Graphs. *Journal of Algorithms* Vol 12(2), (1991), pp. 308–340.

[3] B. Baker. Approximation Algorithms for NP-Complete Problems on Planar Graphs. *JACM*, Vol 41 (1), (1994), pp. 153–190.

[4] H. L. Bodlaender. Some classes of graphs with bounded tree width. *Bulletin of the European Association for Theoretical Computer Science*, (1988), pp. 116–126.

[5] R. B. Borie, R. G. Parker, and C. A. Tovey. Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. *Algorithmica*, Vol 7, (1992), pp. 555–581.

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

[7] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27-45, 1985.

[8] R. Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. In *Proc. Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 71-75, 1999.

[9] N. Bshouty, and L. Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem.

The Proceedings of the Fifteenth Annual Symposium on the Theoretical Aspects of Computer Science 298-308, 1998.

[10] L. Burroughs. Approximation algorithms for covering problems. Master's thesis, University of Calgary, February 1998.

[11] M. Charikar, S. Khuller, D. Mount, and G. Narasimhan. Algorithms for Facility Location Problems with Outliers. In *Proc. Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, 642-651, 2001.

[12] V.Chvátal. A greedy heuristic for the set-covering problem. *Math. of Oper. Res.* Vol. 4, 3, 233-235, 1979.

[13] K. L. Clarkson. A modification of the greedy algorithm for the vertex cover. *Information Processing Letters* 16:23-25, 1983.

[14] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", *MIT Press*, 1989.

[15] R. Duh and M. Fürer. Approximating k-set cover by semi-local optimization. In *Proc. 29th STOC*, May 1997, pages 256–264.

[16] O. Goldschmidt, D. Hochbaum, and G. Yu. A modified greedy heuristic for the set covering problem with improved worst case bound. *Information Processing Letters* 48(1993), 305-310.

[17] M. X. Goemans and J. Kleinberg. The Lovász theta function and a semidefinite programming relaxation of vertex cover. *SIAM Journal on Discrete Mathematics*, 11:196–204, 1998.

[18] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296-317, 1995.

[19] M. Halldórsson. Approximating k-set cover and complementary graph coloring. In *Proc. Fifth Conference on Integer Programming and Combinatorial Optimization*, June 1996, LNCS 1084, pages 118–131.

[20] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. In *Proc. Eleventh ACM-SIAM Symposium on Discrete Algorithms*, January 2000, pages 329–337.

[21] D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. W.P.#64-79-80, GSIA, Carnegie-Mellon University, April 1980. Also: *SIAM J. Comput.* 11(3) 1982.

[22] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics* 6:243-254, 1983.

[23] D. S. Hochbaum (editor). Approximation Algorithms for NP-hard problems. *PWS Publishing Company,* 1996.

[24] D. S. Hochbaum. The t-vertex cover problem: Extending the half integrality framework with budget constraints. In *Proc. First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems* 111-122, 1998.

[25] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of ACM*, 32(1):130-136, 1985.

[26] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256-278, 1974.

[27] M. Kearns. The computational complexity of machine learning. *M.I.T. Press*, 1990.

[28] S. Khuller, U. Vishkin, and N. Young. A Primal Dual Parallel Approximation Technique Applied to Weighted Set and Vertex Cover. *Journal of Algorithms*, 17(2):280–289, 1994.

[29] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.* 13:383-390, 1975.

[30] G. L. Nemhauser and L. E. Trotter, Jr. Vertex packings: Structural properties and algorithms. *Mathematical Programming* 8:232-248, 1975.

[31] E. Petrank. The hardness of approximation: Gap location. *Computational Complexity* 4:133-157, 1994.

[32] P. Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters* 64:251-254, 1997.

[33] A. Srinivasan. New Approaches to Covering and Packing Problems. In *Proc. Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, 567-576, 2001.

Appendix

A. Pseudo-Code for k-Set Cover

PRIMAL-DUAL$(\mathcal{T}', \mathcal{S}', c', k')$

```
    /* Returns a subset C of S' that is feasible; */
    /* i.e., C covers ≥ k' elements of T' */
    C ← ∅
    E ← T'
    while C is not feasible
    /* increase the dual variables u_i for t_i ∈ E. */
    /* When selecting S_i, sum ∑_{i:t_i ∈ S_i} u_i */
    /* is taken over all the t_i ∈ S_j before the start of the while loop. */
            do Increase u_i for all t_i ∈ E until ∃ a set S_j s.t. ∑_{i:t_i ∈ S_i} u_i = c'(S_i)
                E ← E \ S_i
                C ← C ∪ {S_i}
    return C
```

SETCOVER$(\mathcal{T}, \mathcal{S}, c, k)$

```
    if (k ≤ 0) return ∅
    Sort the sets in increasing order of their cost
    for j ← 1 to m
        do c'(S_j) ← ∞
    for j ← 1 to m
    /* create a modified instance I_j = (T^j, S^j, c', k_j). */
    /* run PRIMAL-DUAL on this instance. */
    /* SC_j is the cover obtained in iteration j. */
        do c'(S_j) ← c(S_j) /* S_j is the highest cost set in OPT */
           S^j ← S \ {S_j} /* S_j is removed from the instance */
           T^j ← T \ S_j /* all elements of S_j are removed */
           k_j ← k − |S_j|
           cost(SC_j) = ∞
           if (|S_1 ∪ S_2 ∪ ... ∪ S_j)| ≥ k_j)
             then SC_j ← {S_j} ∪ PRIMAL-DUAL(T^j, S^j, c', k_j)
                  cost(SC_j) = ∑_{S_k ∈ SC_j} c(S_k)
    SC = min{cost(SC_1), cost(SC_2), ..., cost(SC_m)}
    return SC
```

Figure 1: Algorithm for k-Set Cover (§2).

B. Proofs for §3

B.1. Notation:

S: our solution.
OPT: optimal solution.
a_i: number of sets of size i $(i = 1, 2, 3)$ in S.
b_i: number of sets of size i $(i = 1, 2, 3)$ in OPT.
D: set of elements that are covered by 2-sets and 3-sets of OPT and not covered by 2-sets and 3-sets of S.
B: set of elements that are covered by 2-sets and 3-sets of S and not covered by 2-sets and 3-sets of OPT.
C: set of elements that are covered by 2-sets and 3-sets of S and OPT.
$P_{CD}(T_{CD})$: a 2-set (3-set) of OPT that covers elements in C and D.
$P_{CB}(T_{CB})$: a 2-set (3-set) of S that covers elements in B and C.

If S consists only 3-sets then our solution is optimal, hence we will not consider this case. In order to upper bound the number of 1-sets and 2-sets we will construct a graph in which the vertices correspond to 2-sets and 3-sets of OPT and the edges correspond to 1-sets and 2-sets of S. Let H be a component of G. Note that in H a 1-set of S would be represented as a 1-cycle (self loop). Figures 2, 3, 4, 5, and 6 that are referenced in the lemmas below can be found in the appendix.

Lemma B.1: *The semi-local $(2,1)$-optimization algorithm produces a solution in which $a_1 + 2a_2 + 3a_3 \leq b_1 + 2b_2 + 3b_3 + 1$.*

Proof: If $a_1 > 0$ then S covers exactly k elements. If $a_1 = 0$ then it may cover an extra element and hence the 1 on the right hand side of the above inequality. □

Lemma B.2: *H has at most one set of OPT that covers elements in C and D.*

Proof: In H, (i) T_{CD} cannot co-exist with another T_{CD} otherwise a semi-local $(2,0)$ improvement that uses one set less to cover the same number of elements is possible. Figure 2(a) illustrates this case. In the figure it is shown that T_{CD} covers only one element in D. The case in which T_{CD} covers two elements in D is only easier. (ii) T_{CD} can not co-exist with P_{CD}, otherwise a $(1,0)$ semi-local improvement that uses the same number of sets to cover an extra element is possible. This is shown in Figure 2(b). (iii) P_{CD} can not co-exist with another P_{CD} as this would mean that in H, there is a better 2-cover than the one used by S. This is not possible as we find a 2-cover optimally. Figure 2(c) illustrates this case.

□

Lemma B.3: *If H has a T_{CD} or P_{CD} then H is acyclic.*

Proof: If T_{CD} is part of the cycle then a $(1,0)$ semi-local improvement is possible. If H has a T_{CD} that is not part of the cycle then a $(2,0)$ semi-local improvement is possible. If H has a P_{CD} then a $(1,0)$ semi-local improvement is possible. All these cases are illustrated in Figure 3.

□

Lemma B.4: *H does not have more than one cycle.*

Proof: By Lemma B.3 this is true when H has a T_{CD} or a P_{CD}. Assume that H has no such set. In that case a semi-local $2,0)$ improvement is possible. Figure 5 shows this case. □

Lemma B.5: *If $a_1 > 0$ and if H contains a T_{CD} or P_{CD} then H does not have a 2-set or a 3-set of OPT, say X, such that $X \cap Y \neq \emptyset$, where Y is a 3-set of S.*

Proof: If otherwise, a $(0,1)$ semi-local improvement is possible. The improved solution would have fewer 1-sets. Figure 4 illustrates this case.

□

Lemma B.6: *The $(2,1)$ semi-local optimization technique produces a solution in which $a_1 + a_2 \leq b_1 + b_2 + b_3 + 1$.*

Proof: Consider the case when $a_1 > 0$. From Lemmas B.2, B.3, B.4 and B.5 we conclude that if H contains a T_{CD} or P_{CD} then there also exists at least one P_{CB} in H. In each component we will charge an edge to a vertex. In H which either has a T_{CD} or a P_{CD} we can charge P_{CB} to the T_{CD} or P_{CD} and the edges whose both ends are covered can be charged to the other vertices. In H, let e_c^H be the edges that are charged and e_u^H be the edges that are uncharged. Let a_1^c be the 1-sets that are charged to some set of OPT and let a_1^u be the remaining 1-sets. $a_1^c + a_2 = \sum_H e_c^H + \sum_H e_u^H$. Each uncharged edge covers an element in B. Since S and OPT cover exactly the same number of elements, the number of elements covered by the 1-sets of OPT is at least equal to the number of elements in $a_1^u \cup B$. Thus we have

$$a_1^u + \sum_H e_u^H \leq b_1 \tag{7}$$

$$a_1^u + \sum_H e_c^H + \sum_H e_u^H \leq b_1 + b_2 + b_3$$

$$a_1^u + a_1^c + a_2 \leq b_1 + b_2 + b_3$$

$$a_1 + a_2 \leq b_1 + b_2 + b_3$$

Consider the case when $a_1 = 0$. In this case Lemma B.5 does not hold. Hence it is not necessary that if H contains a T_{CD} or P_{CD} then there also exists at least one P_{CB} in H. In such components there exists exactly one set of OPT that does not get charged by an edge in S. Let this set be the set that covers an element in D. In H, let q_c^H denote all the sets of OPT that are charged by the edges of S and q_u^H denote the uncharged sets of OPT. $b_2 + b_3 = q_c^H + q_u^H$. Since $a_1 = 0$, S may cover $k+1$ elements. Thus we have

$$a_1 + \sum_H e_u^H \leq \sum_H q_u^H + b_1 + 1$$

$$a_1 + \sum_H e_c^H + \sum_H e_u^H \leq \sum_H q_c^H + \sum_H q_u^H + b_1 + 1$$

$$a_1 + a_2 \leq b_1 + b_2 + b_3 + 1$$

□

Lemma B.7: *If H contains 1-set of S then H does not have a 2-set or a 3-set of OPT, say X, such that $X \cap Y \neq \emptyset$, where Y is a 3-set of S.*

Proof: If otherwise then a semi-local $(0,1)$ improvement is possible by discarding Y. The resulting solution will have one less singleton. Figure 6 illustrates this case. □

Lemma B.8: *The semi-local $(2,1)$-optimization technique produces a solution in which $a_1 \leq b_1$*

Proof: If $a_1 = 0$ the condition holds trivially. Hence assume $a_1 > 0$. From equation (7) we have $a_1^u \leq b_1$. Let $b_1' = a_1^u$ be the 1-sets of OPT. Let $b_1'' = b_1 - b_1'$ be remaining 1-sets of OPT. We want to prove that $a_1^c \leq b_1''$. Consider a H that has a 1-set of S. This 1-set corresponds to a 1-cycle in H. By Lemma B.4 H does not have a cycle other than the 1-cycle. By Lemma B.3 H does not have a T_{CD} or P_{CD}. By Lemma B.7 there can not be a 3-set, Y, of S such that $X \cap Y \neq \emptyset$, where X is a set of OPT in H. Hence H must have a P_{CB}. The edge corresponding to P_{CB} can not be charged to any set of OPT in H. Hence it is charged to some 1-set of OPT. Thus we have $a_1^c \leq \sum_H e_u^H \leq b_1''$. □

Proof of Theorem 3.1. Adding up the inequalities in Lemmas B.1,B.6 and B.8, we get

$$3(a_1 + a_2 + a_3) \leq 4(b_1 + b_2 + b_3) - b_1 - b_2 + 2$$

$$c(S) = a_1 + a_2 + a_3 \leq \frac{4}{3} OPT + \frac{2}{3}$$

This completes the proof of Theorem 3.1.

C. Brief description related to §4.3

Briefly, we run our algorithm with $\epsilon = 0$. Lemma 4.1 and the Chebyshev-Cantelli inequality show that

$$\Pr[W \leq (k-1)] \leq nm/(nm+1) = 1 - 1/(nm+1),$$

which, though large, is $1 - \Omega(1/n^{O(1)})$. Also, a Chernoff bound and "union bound" argument can be used to show that the probability of existence of an i for which $y_i > 2(1 + 1/\sqrt{\log(\ell + n)})y_i^*$ holds after our randomized rounding, is at most $1/(2nm+2)$. Thus, with probability at least $1/(nm+1) - 1/(2nm+2) = 1/(2nm+2)$ we will have our desired solution; this can be boosted to a high probability by repeating this basic algorithm $O(nm)$ times. Complete details will be presented in the full version.

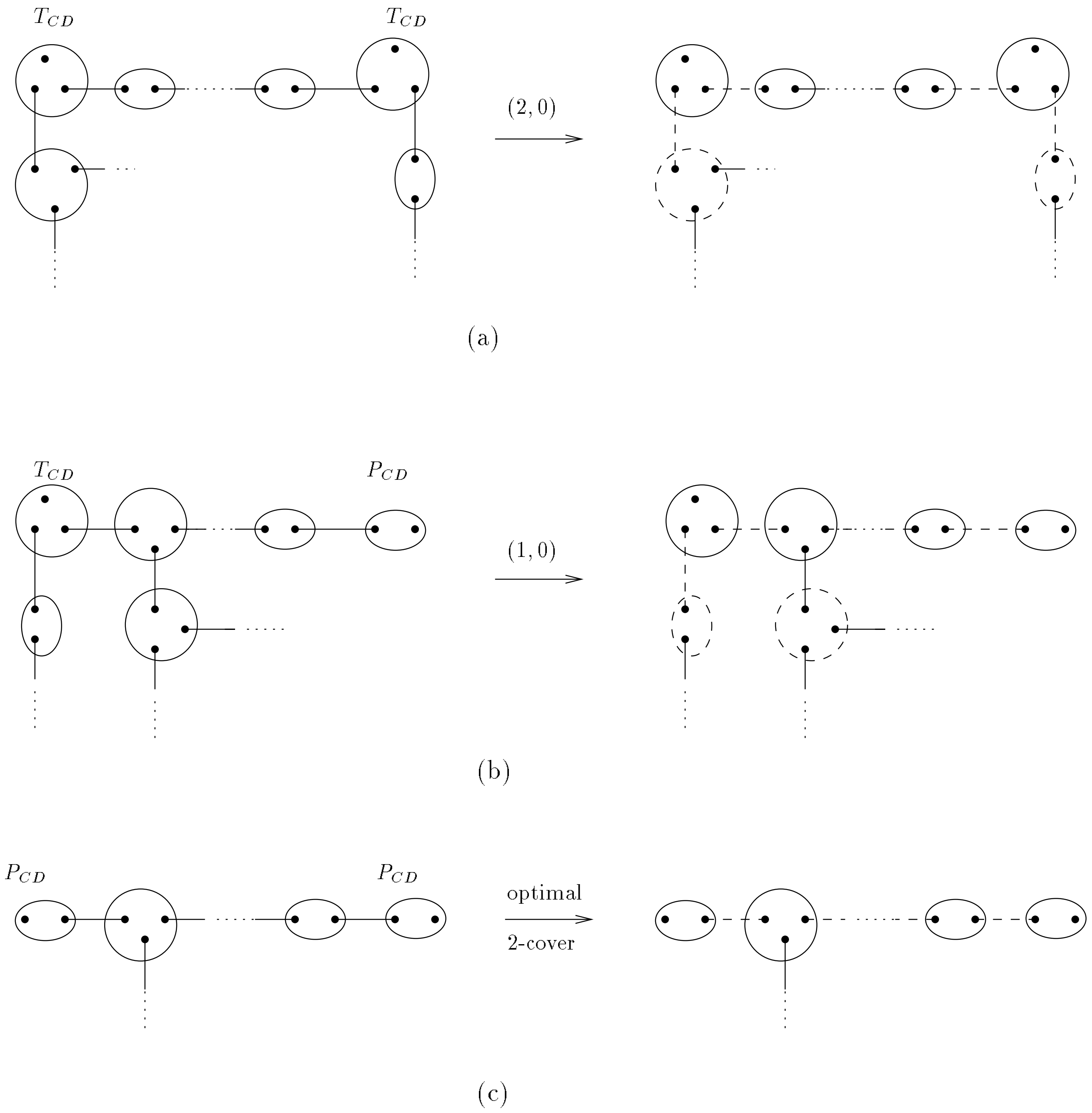

Figure 2: In each of the following cases an improved partial cover (represented by figures on the right) contains the sets of OPT marked by solid boundaries and the sets of S corresponding to the solid edges. (a) Two T_{CD} sets in H lead to a $(2,0)$ semi-local improvement. (b) A T_{CD} and P_{CD} in H leads to a $(1,0)$ semi-local improvement. (c) Two P_{CD} sets in OPT is not possible as our algorithm finds an optimal 2-cover. The figure on the left is not an optimal 2-cover.

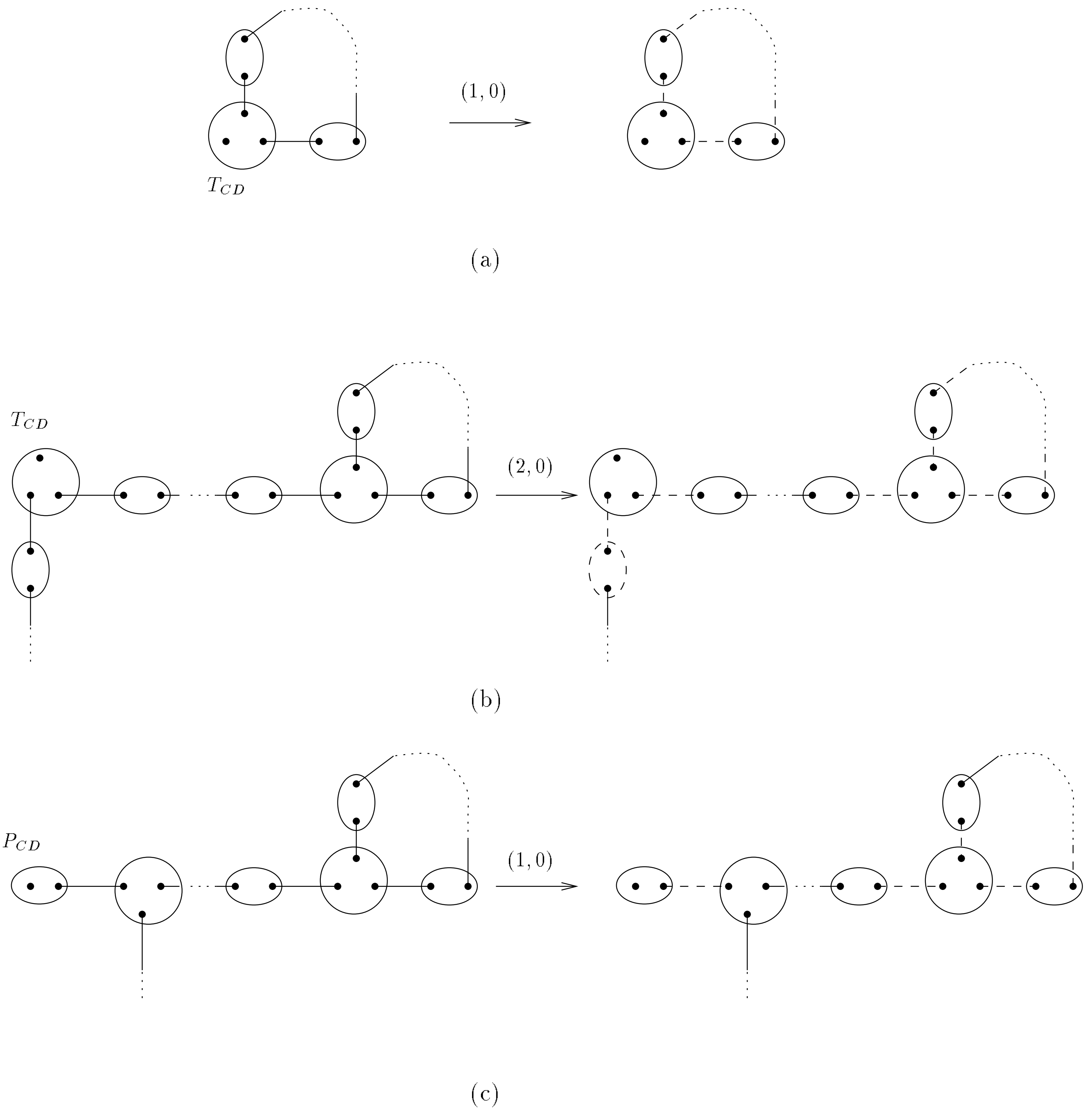

Figure 3: In each of the following cases an improved partial cover (represented by figures on the right) contains the sets of OPT marked by solid boundaries and the sets of S corresponding to the solid edges. (a) A T_{CD} set in H as part of the cycle leads to a $(1,0)$ semi-local improvement. (b) A T_{CD} in H that is not part of the cycle leads to a $(2,0)$ semi-local improvement. (c) A P_{CD} set in H containing a cycle leads to a $(1,0)$ semi-local improvement.

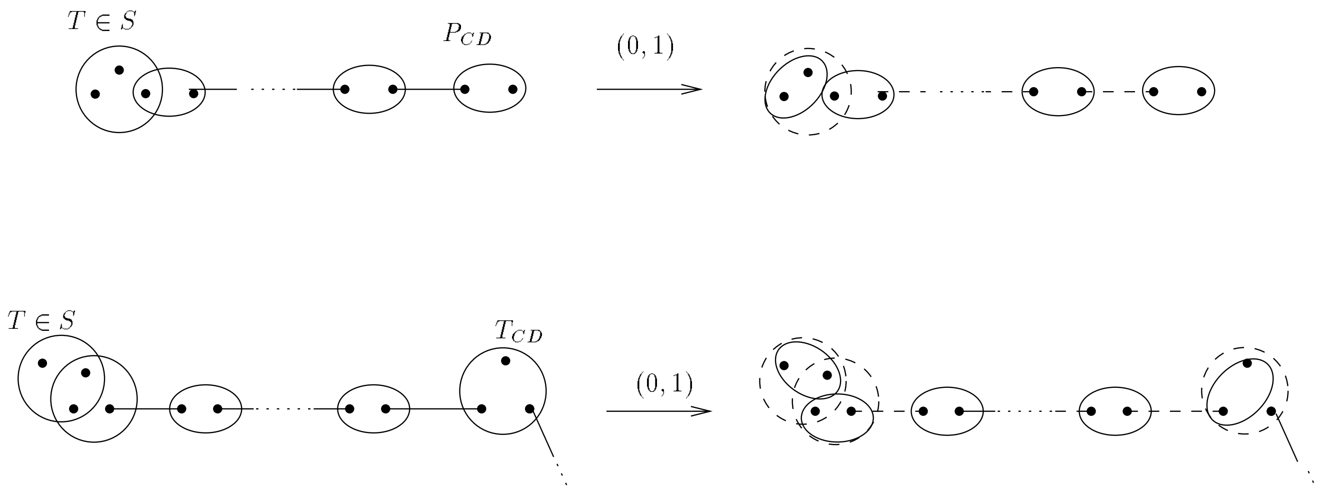

Figure 4: In the above instances $(0,1)$ semi-local improvement yield a solution which uses an extra set to cover an extra element. Thus a singleton can be discarded from our solution giving us a better solution.

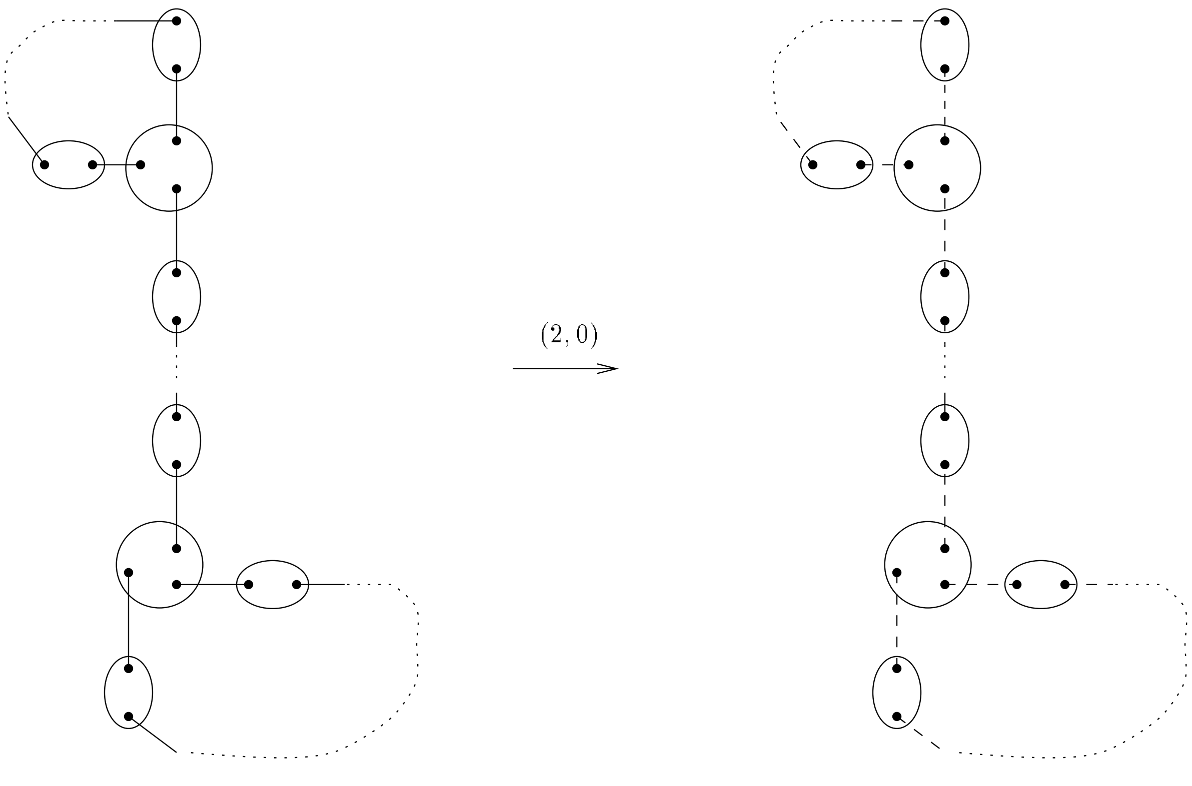

Figure 5: In the above instance $(2,0)$ semi-local improvement yields a solution which uses one set less to cover the same number of elements.

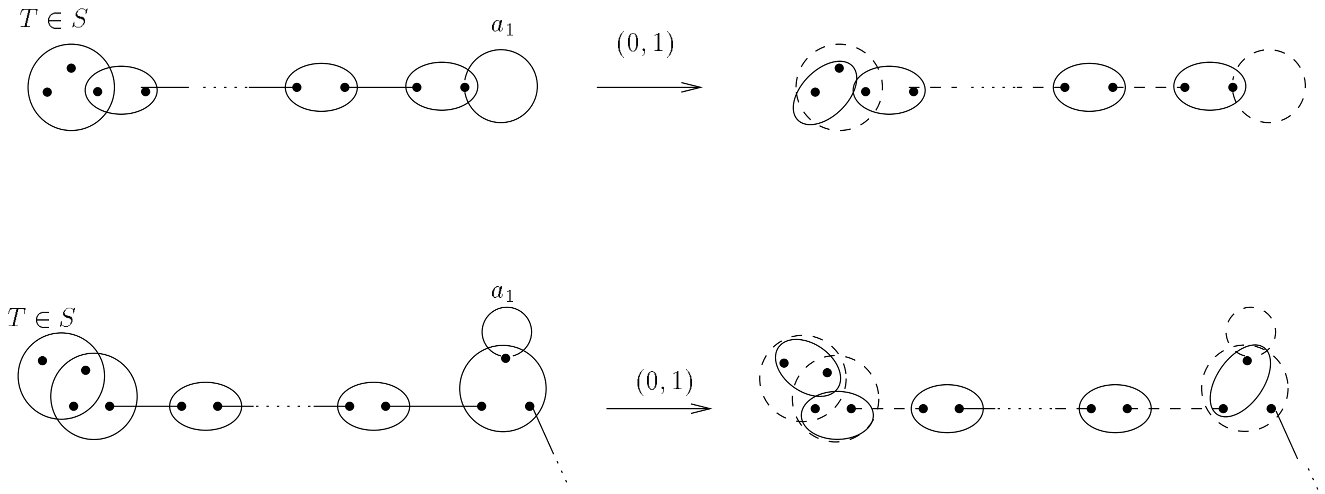

Figure 6: In the above instances $(0, 1)$ semi-local improvement yields a cover whose size is same as before but has one less singleton.