Approximation Algorithms for Partial Covering Problems*

Rajiv Gandhi † Samir Khuller ‡ Aravind Srinivasan §

Abstract

We study a generalization of covering problems called *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-partial set cover, we wish to choose a minimum number of sets to cover at least k elements. For k-partial 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-partial vertex cover) in polynomial time. Without making any assumption about the number of sets an element is in, 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-partial 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-partial vertex cover on planar graphs, and for covering k points in R^d by disks.

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

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, 28, 31]. For vertex cover see [6, 7, 13, 23, 24, 32].

In this paper we study a generalization of "covering" to "partial covering" [29, 33]. Specifically, in k-partial 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-partial vertex cover. (We sometimes refer to k-partial set cover as "partial set cover", and k-partial vertex cover as "partial vertex cover"; the case where k equals the total

*A preliminary version of this work appeared in *Proc. International Colloquium on Automata, Languages, and Programming*, pages 225–236, 2001.

†Department of Computer Science, Rutgers University, Camden, NJ 08102. E-mail: `rajivg@camden.rutgers.edu`. Part of this work was done when the author was a student at the University of Maryland and was supported by NSF Award CCR-9820965.

‡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 CCR-0113192. E-mail: `samir@cs.umd.edu`.

§Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Part of this work was done while at Bell Labs, Lucent Technologies, 600-700 Mountain Avenue, Murray Hill, NJ 07974. Research supported in part by NSF Award CCR-0208005. E-mail: `srin@cs.umd.edu`.

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 by Charikar *et al.*[11].

We begin our discussion by focusing on vertex cover and k-partial vertex cover. A very simple approximation algorithm for unweighted vertex cover (full coverage) is attributed to Gavril and Yannakakis, (see [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-partial vertex cover was given by Bshouty and Burroughs [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 [26] 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-partial vertex cover based on the elegant "local ratio" method.

Our algorithm does not improve on the approximation factors of the previous algorithms, but we derive a natural primal-dual algorithm. Burroughs [10] studied the primal-dual algorithm and showed that applying the primal-dual algorithm as it is, gives an $O(n)$ approximation. In this work we show that the primal-dual algorithm along with a thresholding approach gives us a 2-approximation for the partial vertex cover problem.

1.1 Problem Definitions and Previous Work

- **k-Partial 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 $c : \mathcal{S} \to \mathcal{Q}^+$, 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 [28] and Lovász [31]. 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 [34] 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 [23] gives an f-approximation.

- **k-Partial Vertex Cover:** Given a graph $G = (V, E)$, a cost function $c : V \to \mathcal{Q}^+$, 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, 26, 8]).

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

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

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 [25] for additional references). For the special case of geometric covering problems, Hochbaum and Maass [27] have developed a polynomial approximation scheme.

1.2 Methods and Results

- **k-Partial Set Cover:** For the special case when each element is in at most f sets, we combine a primal-dual algorithm [13, 19] with a thresholding method to obtain an f-approximation when $f > 1$.

  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 the sets having a higher cost than the guessed set, to infinity. This is to make sure 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 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.

  For set cover where the sets have cardinality at most Δ there are results (starting from [17, 20]) 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 [34].

- **k-Partial Vertex Cover:** By switching to a probabilistic approach for rounding the LP relaxation of the problem, we obtain improved results for k-partial 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., [18]. 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., [21]. 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-partial vertex cover in the case of bounded maximum degree and arbitrary k; this is the first Property (P) result for k-partial 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-partial 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-partial vertex cover.

- **Geometric Covering:** There is a polynomial approximation scheme based on dynamic programming for the full coverage version [27]. 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 [27] more complex, although it is still a polynomial-time algorithm.

- **k-Partial 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-partial vertex cover for planar graphs. This is based on Baker's method for the full covering case [3].

2 k-Partial Set Cover

The k-partial set cover problem can be formulated as an integer program as follows. We assign a binary variable $x_j \in \{0,1\}$ to each $S_j \in \mathcal{S}$. 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, given below, is obtained by letting the domain of x_j and y_i be $0 \le x_j, y_i \le 1$. Notice that the upper bound on x_j and y_i is unnecessary and is thus dropped.

$$\begin{aligned}
\min \quad & \sum_{j=1}^{m} c(S_j) \cdot x_j \\
\text{subject to} \quad & \\
& y_i + \sum_{j: t_i \in S_j} x_j \ge 1, i = 1, 2, \ldots, n \\
& \sum_{i=1}^{n} y_i \le n - k \\
& x_j \ge 0, j = 1, 2, \ldots, m \\
& y_i \ge 0, i = 1, 2, \ldots, n
\end{aligned}$$

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 as follows.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} u_i - (n-k) \cdot z \\
\text{subject to} \quad & \\
& \sum_{i: t_i \in S_j} u_i \le c(S_j), j = 1, 2, \ldots, m \\
& u_i \le z, i = 1, 2, \ldots, n \\
& u_i \ge 0, i = 1, 2, \ldots, n \\
& z \ge 0
\end{aligned}$$

The algorithm SetCover in Figure 1 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.

Primal-Dual($\mathcal{T}', \mathcal{S}', c', k'$)

// Returns a subset C of $\mathcal{S}'$ that is feasible,
// i.e., C covers $\geq k'$ elements of $\mathcal{T}'$
// z is maintained implicitly in the algorithm. At all times $z = \max_i u_i$
$C \leftarrow \emptyset$
$E \leftarrow \mathcal{T}'$
Initialize all u_i to 0
while C is not feasible
// increase the dual variables u_i for $t_i \in E$.
// When selecting S_a, sum $\sum_{a:t_i \in S_a} u_i$
// is taken over all the $t_i \in S_a$ before the start of the while loop.
    do Increase u_i uniformly for all $t_i \in E$ until $\exists$ a set S_a s.t. $\sum_{a:t_i \in S_a} u_i = c'(S_a)$
        $E \leftarrow E \setminus S_a$
        $C \leftarrow C \cup \{S_a\}$
return C

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

if ($k \leq 0$) **return** $\emptyset$
Sort the sets in increasing order of their cost
for $j \leftarrow 1$ **to** m
    do $c'(S_j) \leftarrow \infty$
for $j \leftarrow 1$ **to** m
// create a modified instance $I_j = (\mathcal{T}^j, \mathcal{S}^j, c', k_j)$.
// run Primal-Dual on this instance.
// SC_j is the cover obtained in iteration j.
    do $c'(S_j) \leftarrow c(S_j)$ // S_j is the highest cost set in OPT
        $\mathcal{S}^j \leftarrow \mathcal{S} \setminus \{S_j\}$ // S_j is removed from the instance
        $\mathcal{T}^j \leftarrow \mathcal{T} \setminus S_j$ // all elements of S_j are removed
        $k_j \leftarrow k - |S_j|$
        $\text{cost}(SC_j) = \infty$
        if ($|S_1 \cup S_2 \cup \ldots \cup S_j| \geq k$)
            then $SC_j \leftarrow \{S_j\} \cup$ Primal-Dual($\mathcal{T}^j, \mathcal{S}^j, c', k_j$)
                    $\text{cost}(SC_j) = \sum_{S_x \in SC_j} c(S_x)$
$SC = \min\{\text{cost}(SC_1), \text{cost}(SC_2), \ldots, \text{cost}(SC_m)\}$
return SC

Figure 1: Algorithm for k-Partial Set Cover.

SetCover then calls Primal-Dual on I_j which uses a primal dual approach [19] 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_a such that $\sum_{a:t_i \in S_a} u_i = c'(S_a)$. 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 pseudo-code (Figure 1), 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 $f > 1$ 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 $(\mathcal{T}^h, \mathcal{S}^h, c, k_h)$ (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, $\text{cost}(SC)$ is upper bounded by $\text{cost}(SC_h)$.

$$
\begin{aligned}
\text{cost}(SC) &\leq \text{cost}(SC_h) = \sum_{S_x \in SC_h} c'(S_x) \\
&= \sum_{S_x \in SC_h} c(S_x) \\
&= \sum_{S_x \in ASC} c(S_x) + c(S_l) + c(S_h) \\
&\leq \sum_{S_x \in ASC} \sum_{i: t_i \in S_x} u_i + c(S_h) + c(S_h) \\
&= \sum_{i: t_i \in T_c} u_i . |\{x | t_i \in S_x \wedge S_x \in ASC\}| + 2 \cdot c(S_h) \\
&\leq f \cdot \sum_{i: t_i \in T_c} u_i + 2 \cdot c(S_h) \\
&= f \cdot (\sum_{i=1}^{n} u_i - \sum_{i: t_i \in T_u} u_i) + 2 \cdot c(S_h) \\
&= f \cdot (\sum_{i=1}^{n} u_i - |T_u| \cdot z) + 2 \cdot c(S_h) \\
&\leq f \cdot (\sum_{i=1}^{n} u_i - (n-k) \cdot z) + 2 \cdot c(S_h) \\
&\leq f \cdot (DFS(I_h) + c(S_h)) \leq f \cdot OPT.
\end{aligned}
$$

■

Corollary 2.2 SETCOVER(E, V, c, k) *gives a 2-approximate solution for k-partial vertex cover.*

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

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

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 plus the newly added) are covered optimally using 2-sets and 1-sets. This can be solved in polynomial time using maximum matching [17]. 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. Without loss of generality, we assume that all subsets of each set are available and hence all coverings are assumed to be disjoint.

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 (such that the total number of elements covered are at least k) 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 [15] 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 cannot 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 coverage 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 to 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, since our solution may have a 1-set that covers an element that OPT may not cover. We obtain a bound on the number of 1-sets as a side effect of the proof for the lower bound on OPT.

3.1 Analysis:

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.
B: set of elements covered by 2-sets or 3-sets of S and neither covered by 2-sets nor 3-sets of OPT, i.e., B represents "bad" elements.
C: set of elements covered by 2-sets or 3-sets of S and OPT, i.e., C represents the elements common to S and OPT.
D: set of elements covered by 2-sets or 3-sets of OPT and neither covered by 2-sets nor 3-sets of S, i.e., D represents desirable elements.
$P_{CB}(T_{CB})$: a 2-set (3-set) of S that covers elements in B and C. P and T stand for pair and triple respectively.
$P_{CD}(T_{CD})$: a 2-set (3-set) of OPT that covers elements in C and D.

If S consists of 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 in S we will construct a graph G 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. The terms vertices and sets of OPT would be used interchangeably. This will be true even in expressions where symbols for vertices (lower case) may be used instead of symbols for sets. The reference will be clear from the context in which it is used. Similarly, the terms edges and sets of S will be used interchangeably. 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). Also, if an edge covers two elements of a vertex of size 3 in OPT, it is represented as a self-loop on the vertex of OPT. Figure 2(a)is an example of H. In order to adhere to the definition of graph, if an edge covers any element in B, the vertex on that end of the edge is a dummy vertex. Whenever we refer to the vertices of a graph, we do not include dummy vertices.

Lemmas 3.2, 3.3, 3.4, 3.5, 3.6, and 3.8 characterize the structure of any component H in G. In each of these lemmas, we try to prove that H has a certain structure. The proof of each of the above lemmas is based on the following approach. We assume for contradiction that H does not have the structure in question. In that case we show that a semi-local improvement is possible, a contradiction. We use figures to illustrate the proof of these lemmas. In each of the figures, we will show the result when semi-local improvement is applied to H. The scenario before the improvement is shown on the left of each figure and on the right we show the improved partial cover. The improved partial cover consists of some sets that were part of OPT and some sets that were part of S before the improvement. In the improved solution the sets of OPT that we include are marked by solid boundaries and the sets of S that we include are represented by solid edges.

We will now introduce some notation that will be used heavily in the proofs of the lemmas. For *any* vertex z in G, let z^t denote truncated z. We define z^t as follows. If z is a 2-set the $z^t = z$, otherwise z^t covers exactly two of the three elements of z. Figure 2(b) shows a 3-set that is truncated. Let P be a path in H between vertices u and w. Let I_p and E_p denote the set of *internal* vertices and set of edges in P respectively. Thus, I_p contains all vertices of P other than u and w. Hence, $|I_p| = |E_p| - 1$. Let F_p denote the elements covered by edges in E_p. Also, every 3-set, t, in I_p is truncated to contain the two elements in $t \cap F_p$. Thus I_p consists of only 2-sets. If u or w is a T_{CD}, say u, then u^t consists of an element in D and the element in $u \cap E_p$. For any cycle C in H, let V_c and E_c denote the vertices and edges respectively of C. For any vertex $w \in C$, let $I_{cw} = V_c \setminus \{w\}$, where again the 3-sets are truncated to contain its elements that are covered by the cycle. Thus, I_{cw} consists of only 2-sets. Note that $\forall w \in V_c$, $|I_{cw}| = |E_c| - 1$.

Lemma 3.1 *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 Note that $k \leq b_1 + 2b_2 + 3b_3$. 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. Recall that if $a_1 = 0$, then $a_2 > 0$ since we cannot have only 3-sets in our solution. ■

Recall that S and OPT represent our solution and an optimal solution respectively. W.l.o.g. we can modify S as follows. When we compute an optimal solution corresponding to a certain choice of 3-sets, we pick a solution that maximizes the number of 2-sets belonging to OPT. This does not affect the size of S or the number of elements covered. The following lemmas apply to the graph H corresponding to the modified S.

Lemma 3.2 *If H has a T_{CD} that covers two elements in D then the third element must be shared with a 3-set of S. If H has a triple with three elements in D then our solution is optimal.*

Proof Consider the case when H has a triple w that covers exactly two elements in D and the third element is shared with a 1-set or a 2-set of S. We will show that a $(1,0)$ optimization (inserting w) gives an improved solution, a contradiction. The new cover would be $S \cup \{w\} \setminus \{e_w\}$, where e_w is the edge incident on w in H. The new solution covers more elements and uses the same number of sets as before. Figure 2(c) illustrates this case. Now consider the case when w is a triple that covers three elements in D. In this case, our solution must contain all triples, i.e., our solution must be optimal. If not, then we can cover at least one extra element using the same number of sets by swapping w with some 2-set or 1-set in S. This is equivalent to a $(1,0)$ semi-local improvement. ■

Lemma 3.3 *H has at most one set of OPT that covers elements in C and D.*

Proof Assume otherwise. Consider a path P in H between two vertices w_1 and w_2 that cover elements in C and D. By Lemma 3.2, w_1 and w_2 each cover exactly one element in D. Let $e_i \notin E_p$ represent an edge in H that is incident on w_i. This happens only if w_i is a T_{CD}. We will consider the following three cases based on the sets represented by w_1 and w_2.

<u>Case I</u>: $w_1 = T_{CD}$ and $w_2 = T_{CD}$.

We will contradict our assumption by showing that we can obtain a better solution by performing a $(2,0)$ semi-local optimization (inserting w_1 and w_2). Let the new cover be $(S \cup I_p \cup \{w_1, w_2\}) \setminus (E_p \cup \{e_1, e_2\})$. The size of the new cover is $(|S| + |E_p| - 1 + 2) - (|E_p| + 2) = |S| - 1$. The new solution covers all the elements in F_p. Moreover, the new solution covers 2 extra elements due to w_1 and w_2 and loses 2 elements due to e_1 and e_2. All other edges of S are included in the new solution. Thus we use one less set to cover the same number of elements. Figure 3(a) illustrates this case.

<u>Case II</u>: $w_1 = T_{CD}$ and $w_2 = P_{CD}$.

We will show how to obtain a better solution by performing a $(1,0)$ semi-local optimization (inserting w_1), a contradiction. Let the new cover be $(S \cup I_p \cup \{w_1, w_2\}) \setminus (E_p \cup \{e_1\})$. The size of the new cover is $(|S| + |E_p| - 1 + 2) - (|E_p| + 1) = |S|$. We cover an extra element since we cover 2 extra elements due to w_1 and w_2 and lose 1 elements due to e_1. Thus we have an improved solution that uses the same number of sets to cover more elements. This case is illustrated in Figure 3(b).

<u>Case III</u>: $w_1 = P_{CD}$ and $w_2 = P_{CD}$.

This cannot happen as S maximizes the number of 2-sets belonging to OPT. Figure 3(c) illustrates this case. This is an example where we use the assumption that all subsets of each set are available. ■

Lemma 3.4 *If H has a T_{CD} or P_{CD} then H is acyclic.*

Proof Assume otherwise. Let w denote the set of OPT that covers elements in C and D. Let u be the vertex of the cycle, L, that is closest to w. Consider the path P between w and u. Note that by Lemma 3.3, u can not be a T_{CD} or a P_{CD}. We will consider the following cases.

<u>Case I</u>: $w = T_{CD}$ or P_{CD} and u is a 3-set.

A $(1,0)$ semi-local optimization (inserting u) will give us an improved solution. The new cover is $(S \cup I_p \cup \{w^t, u\} \cup I_{lu}) \setminus (E_p \cup E_l)$, where recall that E_l represents the edges of the cycle L. Note that if $w = T_{CD}$ then it is truncated to cover its element in D and the element in $w \cap F_p$. The size of the cover is $(|S| + |E_p| - 1 + 2 + |E_l| - 1) - (|E_p| + |E_l|) = |S|$. The new solution covers all elements in F_p and it does not delete edges that are not in E_p. Moreover, by including w^t in the cover, the new solution covers an extra element. Thus the new solution uses the same number of sets to cover an extra element and hence it is an improvement, a contradiction. Figures 4(a) and 4(b) illustrate this case.

<u>Case II</u>: $w = T_{CD}$ or P_{CD}, u is a 2-set.

Since u is a 2-set, it must be part of a 1-cycle. We will prove that this can not happen by showing that a better 2-cover is possible. Consider the 2-cover $(S \cup I_p \cup \{w^t, u\}) \setminus (E_p \cup E_l)$. The size of the cover is $(|S| + |E_p| + 1) - (|E_p| + 1) = |S|$. Note that we manage to cover all elements in F_p and we do not delete any edge that is not in E_p. Moreover, the new solution covers an extra element by including w^t in our cover. Thus, we get an improved 2-cover that uses the same number of sets to cover an extra element, a contradiction. Figures 4(c) and 4(d) illustrate this case.

Case III: $w = u$.

We will show that a $(1,0)$ semi-local optimization (inserting w) gives an improved solution, a contradiction. The new cover will be $(S \cup \{w\} \cup I_{lw}) \setminus (E_l)$. The size of the new cover is $(|S| + 1 + |E_l| - 1) - |E_l| = |S|$. In addition to covering all the elements covered by the cycle, the new solution covers an extra element due to w. Hence the new solution uses the same number of sets to cover more elements. ■

Lemma 3.5 *H does not have more than one cycle.*

Proof By Lemma 3.4, the claim is true when H has a T_{CD} or a P_{CD}, since H is acyclic. For the rest of the proof we will assume that H does not contain a T_{CD} or a P_{CD}. Assume for contradiction that H has two cycles C_1 and C_2. We will consider the following cases.

Case I: C_1 and C_2 are disjoint.

Let $u \in C_1$ and $w \in C_2$ be the closest pair of vertices on the two cycles and let P be the path between u and w. If both C_1 and C_2 are 1-cycles and u and w are 2-sets then we can show that there exists a better 2-cover that uses fewer sets. Figure 5(a) illustrates this scenario. If both C_1 and C_2 are 1-cycles and u or w (or both) is a 3-set then by truncating the 3-sets corresponding to u and w and removing the edges in E_p, we can get an improved solution that uses fewer sets. If $C_1(C_2)$ is a 1-cycle and $C_2(C_1)$ is not then $w(u)$ is a 3-set. In this case a $(1,0)$ improvement (inserting $w(u)$) is possible. Figure 5(b) illustrates this scenario. If neither C_1 nor C_2 is a 1-cycle then u and w both are 3-sets. A $(2,0)$ improvement (inserting u and w) is possible. Figure 5(c) shows an example of this scenario. In each of the above cases the new solution would be $(S \cup \{u, w\} \cup I_{c_1 u} \cup I_{c_2 w} \cup I_p) \setminus (E_{c_1} \cup E_{c_2} \cup E_p)$. The size of the new cover is $(|S| + 2 + |E_{c_1}| - 1 + |E_{c_2}| - 1 + |E_p| - 1) - (|E_{c_1}| + |E_{c_2}| + |E_p|) = |S| - 1$. Note that the only edges deleted from S are the edges in C_1, C_2 and P. The new solution covers all the elements covered by these deleted edges. All other edges in S are part of the new cover. Thus we use one set less to cover the same number of elements as S.

Case II: C_1 and C_2 are not disjoint.

Let $u \in C_1$ and $w \in C_2$ be some vertices shared by C_1 and C_2. If both C_1 and C_2 are 1-cycles then $u = w$ is either a 2-set or a 3-set. This is a trivial case in which $(S \cup \{u^t\}) \setminus (E_{c_1} \cup E_{c_2})$ is a better cover. If either C_1 or C_2 is a 1-cycle then $u = w$ is a 3-set. In this case a $(1,0)$ improvement (inserting u) is possible. If neither C_1 nor C_2 is a 1-cycle then let u and w both be different 3-sets. A $(2,0)$ improvement (inserting u and w) is possible. This is illustrated in Figure 5(d). The new solution in each of the above scenarios is calculated as follows. Let V_{12} and E_{12} be the vertices and the edges that are shared by C_1 and C_2. Let V_{1-2} and E_{1-2} be the vertices and edges that are in C_1 and not in C_2. Let V_{2-1} and E_{2-1} be the vertices and edges that are in C_2 and not in C_1. Note that $|V_{12}| = |E_{12}| + 1$, $|V_{1-2}| = |E_{1-2}| - 1$ and $|V_{2-1}| = |E_{2-1}| - 1$. The new solution would be $(S \cup V_{1-2} \cup V_{2-1} \cup V_{12}) \setminus (E_{1-2} \cup E_{2-1} \cup E_{12})$. The size of the new cover is $(|S| + |E_{1-2}| - 1 + |E_{2-1}| - 1 + |E_{12}| + 1) - (|E_{1-2}| + |E_{2-1}| + |E_{12}|) = |S| - 1$. Again following the same argument as in Case I, the new solution covers the same number of elements as S, but using one set less. ■

Lemma 3.6 *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 that shares elements with a 3-set of S, i.e., 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 Let r be a 1-set of S. Let u denote a T_{CD} or P_{CD} in H. Consider the case when u and x denote the same set. In that case a $(0,1)$ semi-local improvement (removing the 3-set y) will cover the same number of elements using fewer 1-sets. Now we consider the case when u and x denote different sets. Consider a path P between u and x in H. In this case a $(0,1)$ semi-local improvement (removing the 3-set y) is possible, a contradiction. The new solution would be $(S \cup \{x^t, y^t, u^t\} \cup I_p) \setminus (y \cup E_p \cup r)$. The size of the cover is $(|S| + 3 + |E_p| - 1) - (1 + |E_p| + 1) = |S|$. The new solution covers all the elements covered by $y \cup E_p$. The new solution covers an element in D by inserting u^t. This accounts for the element covered by r that is not in the new solution. Thus the new cover is of the same size as the old one, however the new solution has one singleton less than in S. Hence it is an improved solution. Figure 6 illustrates some of the cases. ■

Lemma 3.7 *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 The outline of our proof is as follows. In each component H we will charge the edges (sets of S) to the vertices (sets of OPT) of H. Our charging scheme satisfies the following property. Each vertex is charged by at most one edge and an edge e is charged to a vertex v only if $|e \cap B| \geq |v \cap D|$. We then argue that each edge that is not charged to any vertex must cover an element in B. Through careful arguments we then show that these edges can be accounted for by vertices in OPT that are not yet charged.

We will now introduce some notation that will be used in the proof of this claim. Let O_c be the vertices in OPT that are charged by some edge. Let O_u be the remaining vertices in OPT. Let $o_u = |O_u|$ and $o_c = |O_c|$. Let $a_i^u, i \in \{1,2\}$, be the number of sets of size i that are not charged to any set in OPT. Let a_i^c be the number of sets of size i that are charged to some set of OPT.

Below are the details of the proof. Our proof considers the following two cases.
Case I: $a_1 > 0$.
In this case, OPT cannot contain a set with two elements in D; otherwise, we can use a 2-set to cover the two elements in D and drop a singleton thereby covering an extra element using the same number of sets. In each component H, we will charge an edge e to a vertex v using the charging scheme described below. Let H_s be the subgraph of H that consists of all the vertices of H and all edges of H that do not cover any element in B. Consider the following two cases.
Case I(a): All elements covered by the vertices of H are in C.
By Lemma 3.5, H has at most one cycle. Thus, H_s is a tree or contains just one cycle. Hence, in H_s the number of vertices is at least equal to the number of edges. This means that each edge in H_s can be charged to a vertex in H_s. The edges of H that are not charged to any vertex cover an element in B.
Case I(b): H consists of vertices that cover elements in D.
By Lemma 3.3, H has exactly one such vertex, say w. By Lemma 3.4, H can not have any cycles. In this case, H_s is a tree. We can charge all the edges in H_s to all the vertices in H_s, except w. Thus all the edges in H that do not cover any element in B are charged to some vertex in H. By Lemma 3.6, the number of edges incident on every vertex in H except w must be equal to the size of the vertex. This means that H must have at least one P_{CB}. We charge one such P_{CB} to w. As in the previous case, the edges of H that are not charged to any vertex must cover an element in B. Below we show how to account for these edges.

Note that a_1^u is the number of singleton sets of S that cover an element in B. All other 1-sets of S cover elements in C and hence are charged to some set of OPT. We have $a_1^c + a_2^c = o_c$. Recall that OPT cannot have a set with two elements in D. Also, since $a_1 > 0$, S and OPT cover exactly the same number of elements. Thus the number of uncharged vertices together with the singletons in OPT must be at least equal to the number of elements in B covered by the edges that are not charged to any vertex. Thus we have

$$\begin{aligned} a_1^u + a_2^u &\le b_1 \\ a_1^u + a_2^u + a_1^c + a_2^c &\le b_1 + o_c \\ a_1 + a_2 &\le b_1 + b_2 + b_3 \end{aligned} \tag{1}$$

Case II: $a_1 = 0$.
We analyze this case by again considering two cases.
Case II(a): H contains a 3-set w that does not share elements with any 3-set in S.
In this case there can be at most one set in OPT that covers two elements in D. Otherwise, a $(1, 0)$ improvement (inserting w) is possible as follows. By Lemma 3.2, w cannot have two elements in D. If w is a T_{CD} then we can swap w with the two edges incident on it and use a 2-set to cover two elements in D. If all elements of w are in C then we can include w in our cover along with two 2-sets to cover the four elements in D and delete the three edges incident on w. In both the cases we use the same number of sets to cover one extra element. For the remainder of the argument we will assume that there is at most one set covering two elements in D. We denote this set by t. We analyze this sub-case by using the same charging scheme as in Case I. As in Case I(a), when H consists of vertices that only cover elements in C, we can show that the edges that are not charged to any vertex contain an element in B. Now consider the case when H has a vertex, say d, that covers an element in D. In this case, one vertex of H may not be charged by any edge. Let this vertex be d. The vertex d may cover two elements in D and by our charging rule it will not be charged. Recall that a_2^u denotes the number of edges of size two that are not charged to any vertex. Note that each edge that is not charged to any vertex covers an element in B. First, we will bound a_2^u for the case when t does not exist. In this case our solution may cover $k + 1$ elements. Hence, $a_2^u \le o_u + b_1 + 1$. If t exists then S covers the same number of elements as OPT. If not, then a $(1, 0)$ improvement (inserting w) is possible. This is because if w is a T_{CD} then we can swap w for the two edges incident on it and if w is not a T_{CD} then we can include w and t and delete the three edges incident on w. In both the cases we use one set less to cover one less element. Also, note that t is not charged by any edge and belongs to the set O_u. Hence, we have $a_2^u \le (o_u + 1) + b_1 = o_u + b_1 + 1$. Thus, whether t exists or not the bound on a_2^u is the same.

$$\begin{aligned} a_2^u &\le o_u + b_1 + 1 \\ a_2^u + a_2^c &\le o_u + b_1 + 1 + o_c \\ a_2 &\le b_1 + b_2 + b_3 + 1 \\ a_1 + a_2 &\le b_1 + b_2 + b_3 + 1 \end{aligned}$$

Case II(b): H does not contain w.
In this case, if H contains a vertex of size three then it shares an element with a 3-set of S. Hence, at most two elements of any vertex can be covered by edges in H. Hence, H is either a cycle or a path. In H, if the number of edges is less than or equal to the number of vertices then each edge is charged to some vertex. Now consider the case when the number of edges in H is greater than the number of vertices in H. In this case, H has two elements in B and none in D. Since the number of edges is one more than the number of vertices, exactly one edge in H is not charged to any vertex in H. Observe that the number of elements covered by the edges is two more than the number of elements covered by vertices in OPT. Thus even if O_u contains only 2-sets with both elements in D, we have $o_u \ge a_2^u$. Taking into account singletons in OPT and that S may cover $k + 1$ elements we get

$$\begin{aligned} a_2^u &\le b_1 + o_u + 1 \\ a_2^c + a_2^u &\le b_1 + o_c + o_u + 1 \\ a_2 &\le b_1 + b_2 + b_3 + 1 \end{aligned}$$

■

Lemma 3.8 *If H contains a 1-cycle 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 Assume otherwise. Let u be the vertex in the 1-cycle, l. Let P be the path between x and u. We will show that a semi-local $(0,1)$ improvement is possible by discarding y. The resulting solution will be $(S \cup \{x^t, u^t\} \cup I_p) \setminus (E_p \cup l)$. In the new solution x^t contains an element in $x \cap y$ and an element in $x \cap F_p$ and u^t contains an element of l and an element of $u \cap F_p$. The size of the new cover will be $(|S| + 2 + |E_p| - 1) - (|E_p| + 1) = |S|$. The new solution covers all the elements covered by $E_p \cup l$. Since we retain every other set of S, the new solution covers at least as many elements as S while using one less singleton. Figure 7 illustrates this case. ■

Lemma 3.9 *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 $a_1 > 0$. We have $a_1 = a_1^u + a_1^c$. From inequality (1) we have $a_1^u + a_2^u \leq b_1$. Thus, if we prove $a_1^c \leq a_2^u$ then we are done. We will prove this by showing that if H has a 1-set then there is some edge in H that is not charged to any vertex. Consider a H that has a 1-set of S. This 1-set corresponds to a 1-cycle in H. By Lemma 3.5, H does not have a cycle other than the 1-cycle. By Lemma 3.4, H does not have a T_{CD} or P_{CD}, i.e., all elements covered by the vertices of H are in C. By Lemma 3.8, 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}, say e. Consider the subgraph of H, H_s as constructed in Lemma 3.7. H_s contains all the vertices of H and contains edges that have both its endpoints in C. Thus, e is not part of H_s. Since H has a 1-cycle, H_s has an equal number of vertices and edges. Each vertex in H_s and hence H get charged by some edge in H_s. Hence, edge e does not get charged to any vertex. This completes the proof. ■

Theorem 3.10 *The semi-local $(2,1)$-optimization algorithm for the 3-set partial covering problem produces a solution that is within $\frac{4}{3}OPT + 1$.*

Proof Adding up the inequalities in Lemmas 3.1,3.7 and 3.9, we get

$$3(a_1 + a_2 + a_3) \leq 4(b_1 + b_2 + b_3) - b_1 - b_2 + 2$$
$$c(S) = a_1 + a_2 + a_3 \leq \frac{4}{3}OPT + \frac{2}{3}$$

■

4 Probabilistic Approaches for Partial Vertex Cover

We now present a randomized rounding approach to the natural LP relaxation of k-partial vertex cover. Analyzed in three different ways, this leads to three new approximation results mentioned in Section 1: relating to vertex cover (full coverage) for expander graphs of constant average degree, k-partial vertex

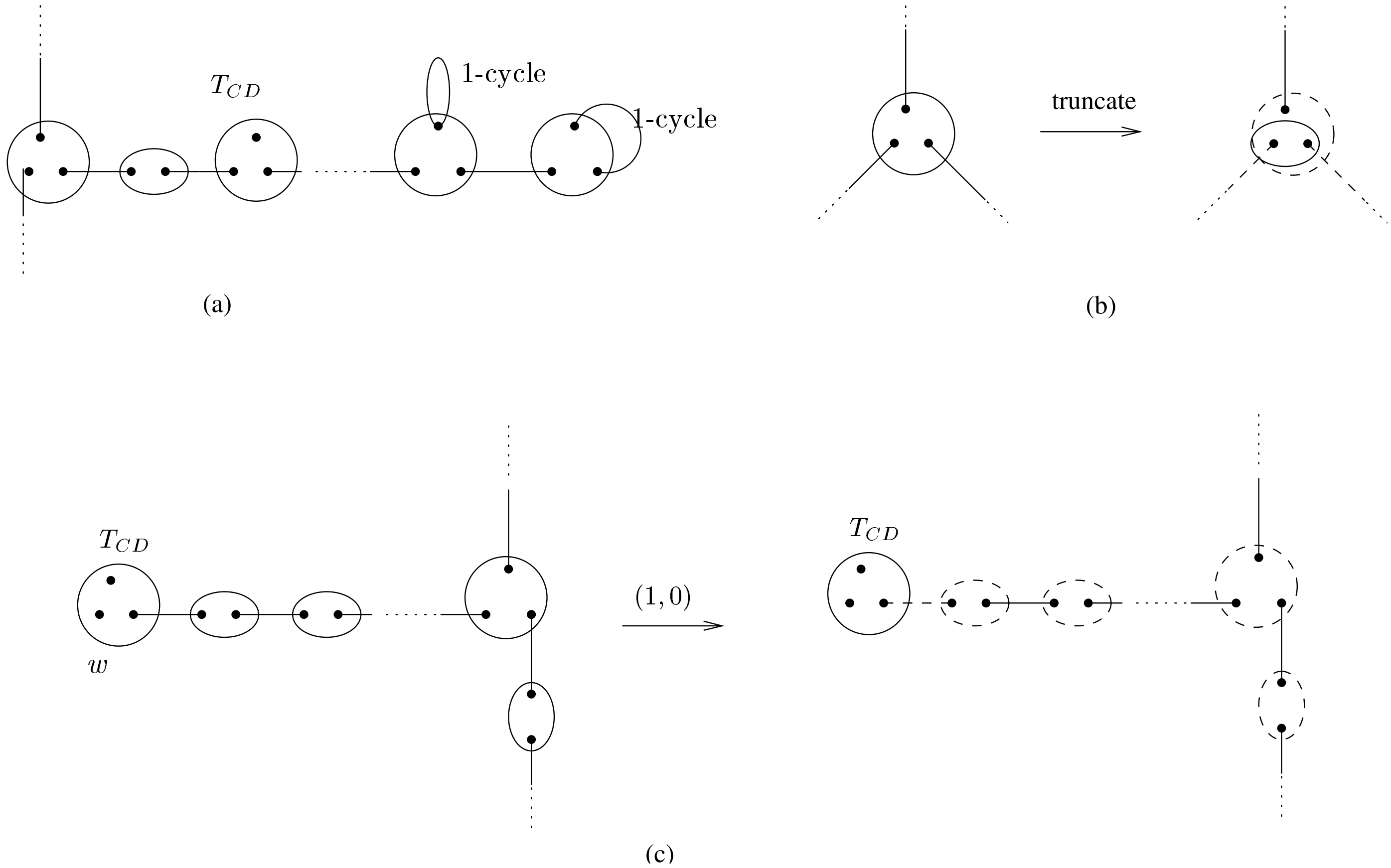

Figure 2: (a) Example of H. (b) Truncating a 3-set. (c) H can not have a T_{CD} covering two elements in D (Lemma 3.2).

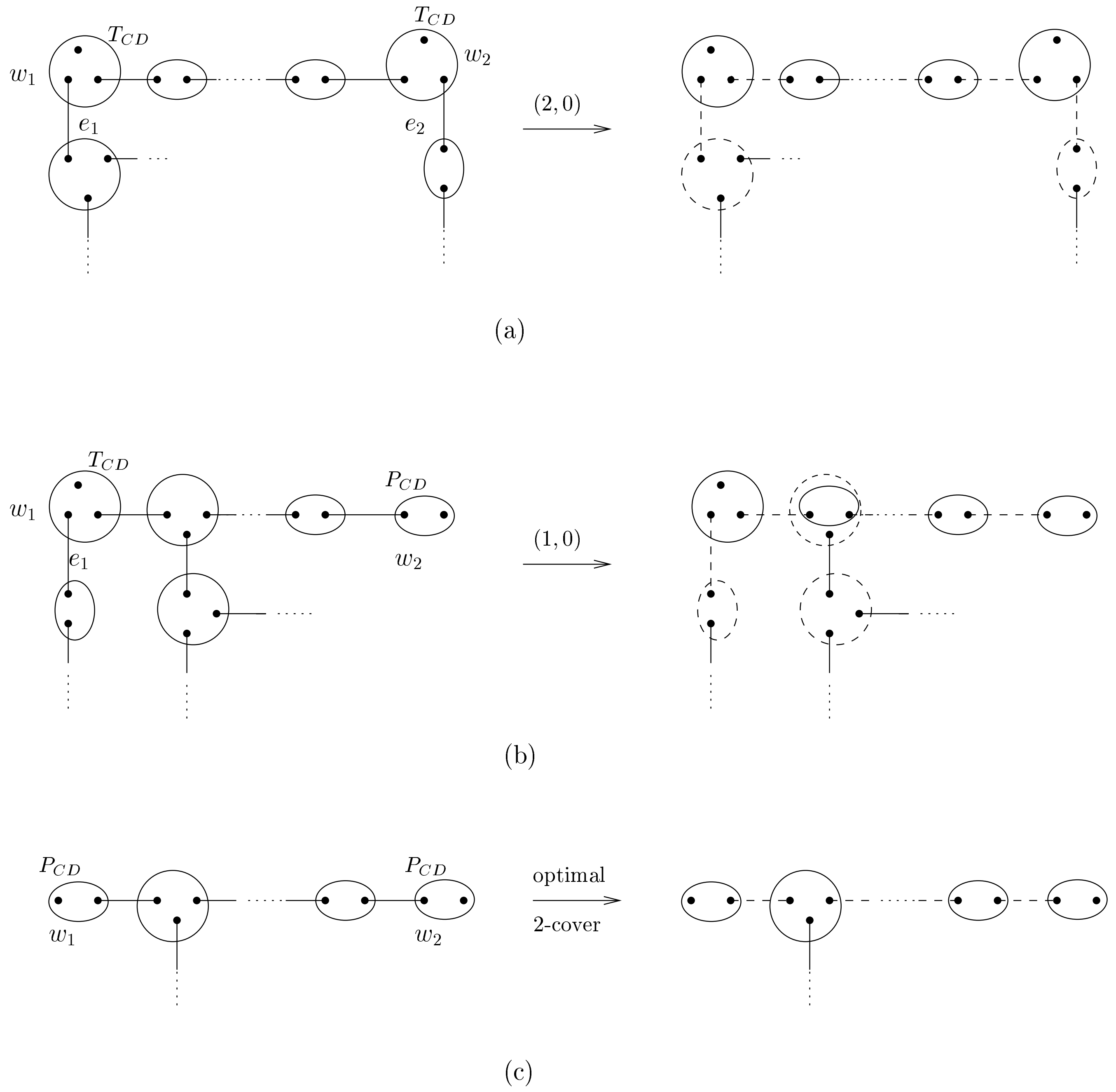

Figure 3: Examples for Lemma 3.3. In each of the following cases an improved partial cover (represented by figures on the right) contains the sets of OPT marked by solid boundaries and the sets of S corresponding to the solid edges. (a) Two T_{CD} sets in H lead to a $(2, 0)$ semi-local improvement. (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, w.l.o.g., our algorithm finds a solution maximizing the number of 2-sets belonging to OPT.

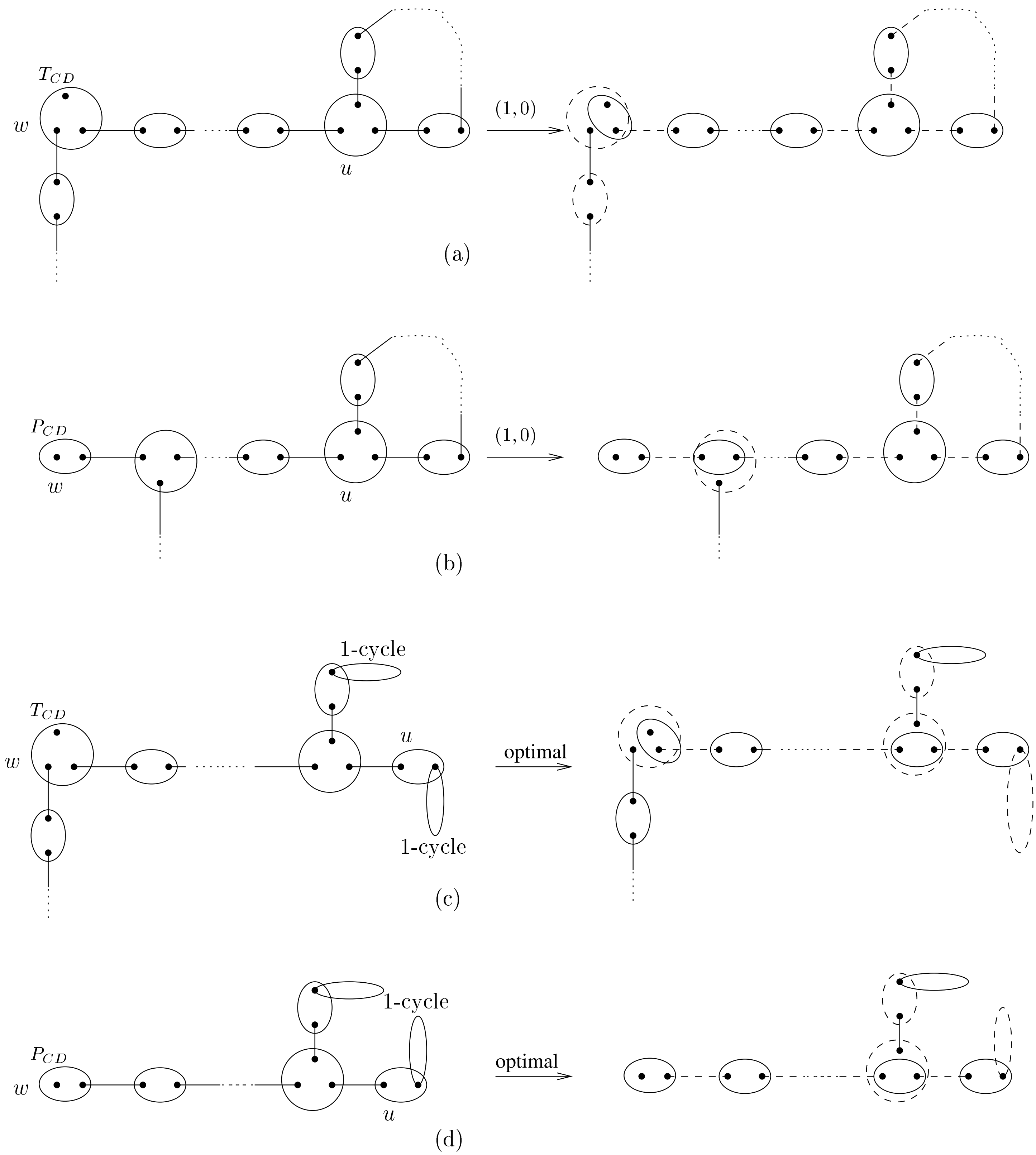

Figure 4: Examples for Lemma 3.4. In each of the following cases an improved partial cover (represented by figures on the right) contains the sets of OPT marked by solid boundaries and the sets of S corresponding to the solid edges. (a),(b) A (1, 0) optimization (inserting u) gives us an improved solution. (c),(d) Our algorithm finds an optimal 2-cover. The figure on the left shows a sub-optimal 2-cover.

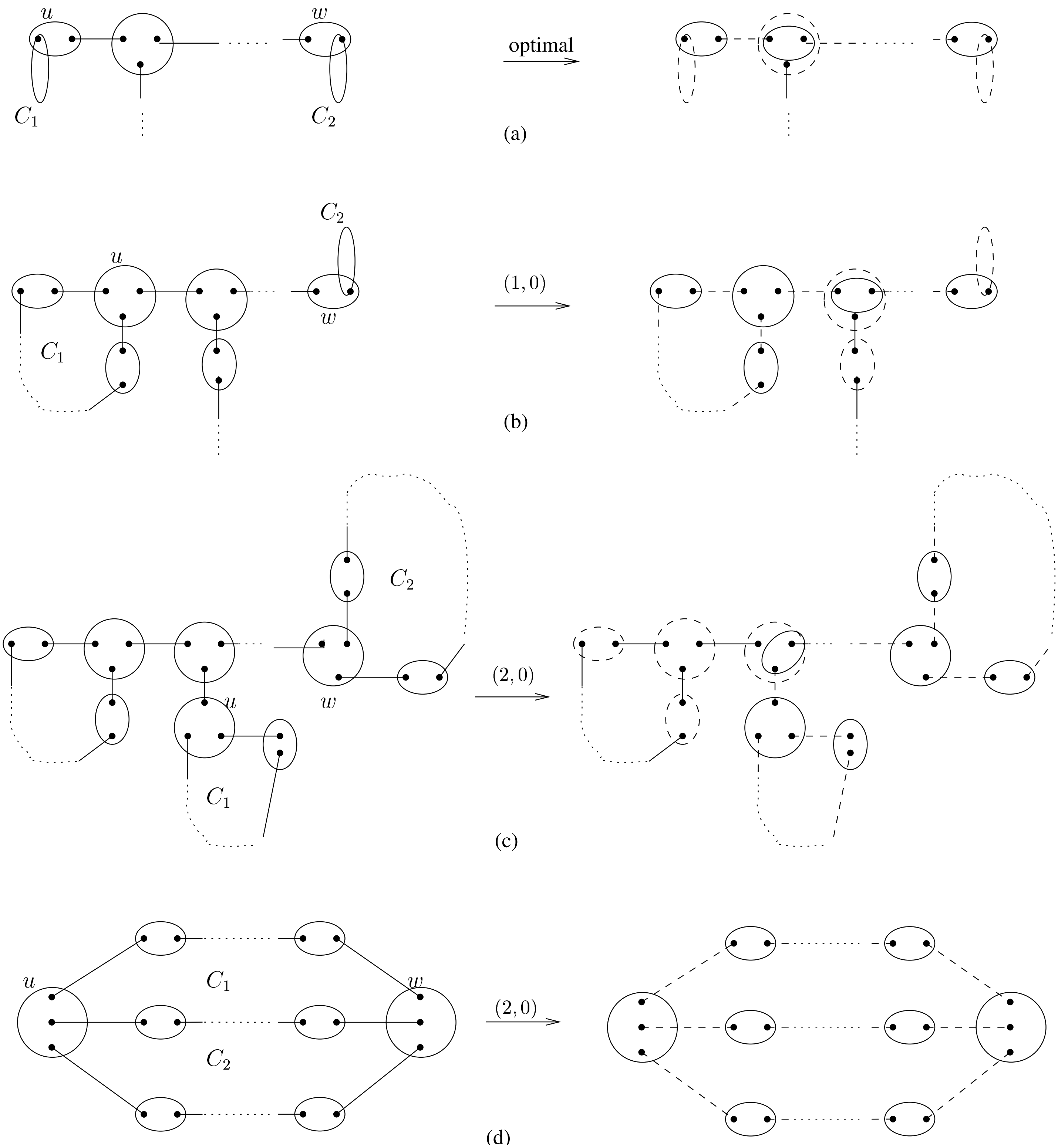

Figure 5: Examples for Lemma 3.5. (a),(b),(c) Semi-local improvement in the case when H has two disjoint cycles.(d) A $(2,0)$ improvement leads to a better solution when neither C_1 and C_2 are not disjoint and neither of them are 1-cycles.

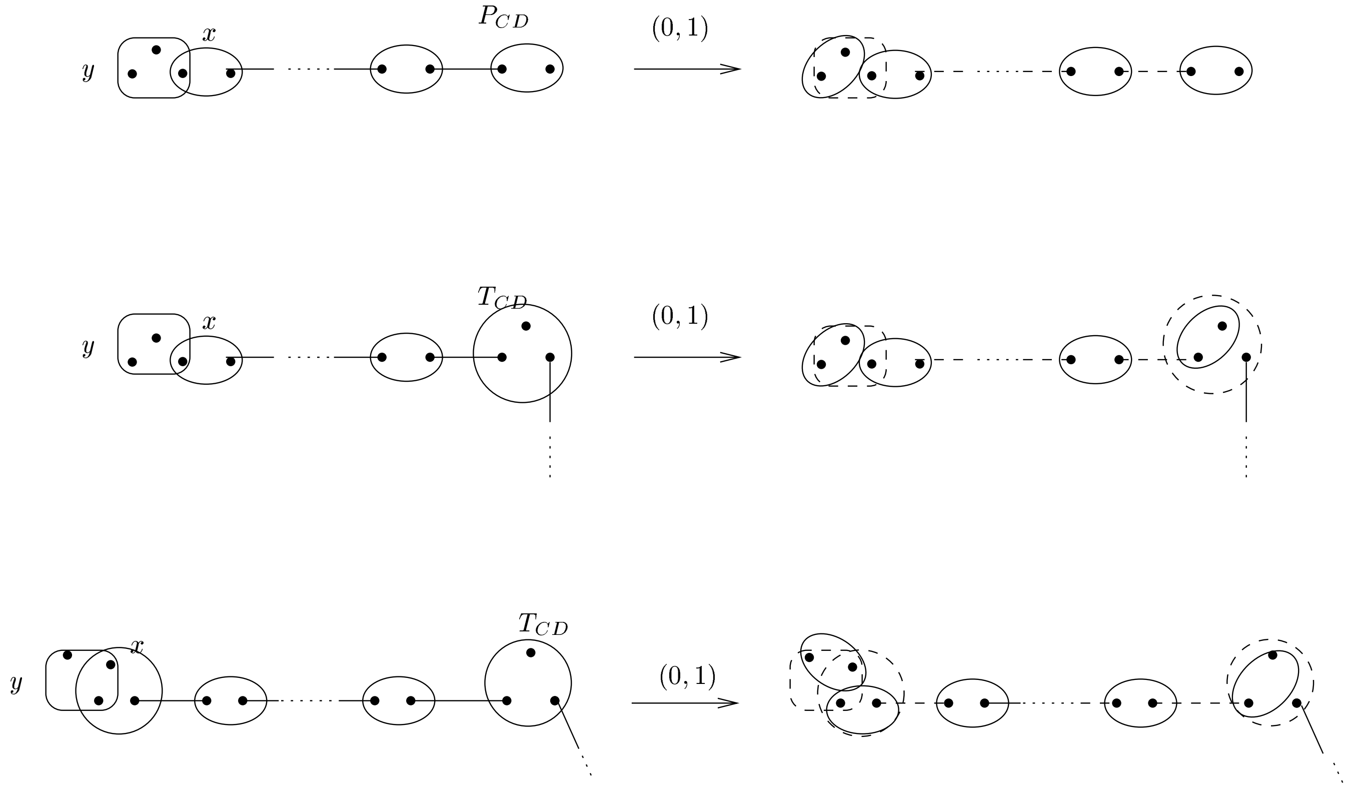

Figure 6: Examples for Lemma 3.6. In the above instances $(0,1)$ semi-local improvement yields an improved solution.

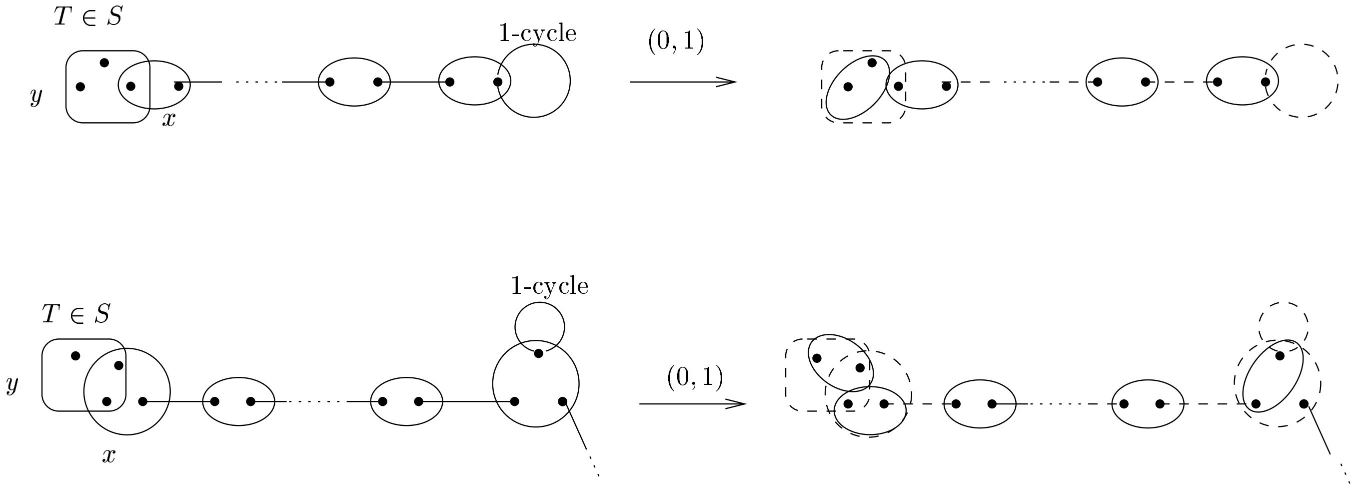

Figure 7: Examples for Lemma 3.8. In the above instances $(0,1)$ semi-local improvement yields a cover whose size is same as before but has one less singleton.

cover on bounded-degree graphs, and multi-criteria k-partial vertex cover problems. We first describe the basic method and prove some probabilistic properties thereof, and then consider the three applications.

The k-partial 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 $[0,1]$.

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} x_j \\ \text{subject to} \quad & \\ x_i + x_j \ &\geq \ z_{i,j}, \quad (i,j) \in E & (2) \\ \sum_{(i,j)\in E} z_{i,j} \ &\geq \ k & (3) \\ x_j, z_{i,j} \ &\in \ [0,1], \quad \forall i,j. \end{aligned}$$

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 $\epsilon \in [0,1]$ 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

$$\mathbf{E}[C] \leq y_F^* + \lambda y_P^* + \mathbf{E}[\max\{k - W, 0\}]. \tag{4}$$

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* $(2d-1) \cdot \mathbf{E}[W]$.

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 $\mathbf{E}[W] = \sum_{(i,j)\in E} \Pr\left[\overline{U_{i,j}}\right]$, 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 $\overline{U_{i,j}}$ 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 $\Pr\left[\overline{U_{i,j}} \wedge \overline{U_{i',j'}}\right]$. Using the above observations and the definition of variance, a moment's reflection shows that $\mathrm{Var}[W]$ is upper-bounded by $\mathbf{E}[W] + 2\gamma$. 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 "$\Pr\left[\overline{U_{i,j}} \wedge \overline{U_{i',j'}}\right]$" in γ is at most $(\Pr\left[\overline{U_{i,j}}\right] + \Pr\left[\overline{U_{i',j'}}\right])/2$. Finally, since each edge has at most $2(d-1)$ other edges that share an end-point with it, we get that

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

So, $\mathrm{Var}[W] \leq \mathbf{E}[W] + 2\gamma \leq (2d-1) \cdot \mathbf{E}[W]$. ■

4.1 Vertex Cover on Expanders

Suppose we have a vertex cover problem; i.e., k-partial vertex cover with $k = m$. The LP relaxation here has "1" in place of "$z_{i,j}$" in (2), and does not require the variables $z_{i,j}$ and the constraint (3). 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, (4) shows that $\mathbf{E}[C]$ is at most $y^*_F + 2(1-\epsilon)y^*_H + m\epsilon^2$; following the notation underlying (4), y^*_H is the total contribution to y^* from the vertices in H. (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 [35]. 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 Sections 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

$$\mathbf{E}[C] \leq y^*_H(2 - y^*_H/m) + y^*_F. \tag{5}$$

Case I: $|H| \leq n/2$. Note that edges with exactly one end-point 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 (5) 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 (5) 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 Partial Vertex Cover: Bounded-Degree Graphs

We now show that for any constant d, k-partial 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 Section 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-partial 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 Section 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

$$\Pr\left[W \leq (k(1-\epsilon^2) - 2\sqrt{kd})\right] \leq 1/3. \tag{6}$$

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 $\Pr\left[W \leq \mu - 2\sqrt{d\mu}\right] \leq 1/3$. 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 (6).

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 $y^* \geq k/d$. Let $\delta = 1/(3d)$. Assuming

$$\epsilon \leq 0.7, \text{ say}, \tag{7}$$

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 (6). So, with probability at least $1-(1/3+1/5)=7/15$, the final cover we produce is of size at most

$$2y^*(1-\epsilon)(1+\delta)+k\epsilon^2+2\sqrt{kd}. \tag{8}$$

We now choose $\epsilon = y^*(1+\delta)/k$. (We may assume that this choice satisfies (7) without loss of generality. Indeed, if this choice of ϵ violates (7), then we have $y^* \geq 0.7k(1+1/(3d))^{-1} \geq 21k/40$. However, since any partial cover problem can be trivially solved using k vertices, we can get a (40/21)–approximation – i.e., a constant-factor approximation strictly better than two – in this case.) Recalling that $y^* \geq k/d \geq c_0 d^4$ with c_0 sufficiently large, let us see now that (8) is at most $2y^*(1-\Omega(1/d))$. First, since $\epsilon = y^*(1+\delta)/k$, we have

$$2y^*(1-\epsilon)(1+\delta)+k\epsilon^2 = 2y^*(1-y^*(1+\delta)^2/(2k)) \leq 2y^*(1-(1+\delta)^2/(2d)) \leq 2y^*(1-1/(2d)).$$

Next,

$$2\sqrt{kd} = 2\sqrt{k/d}\cdot d \leq 2\sqrt{y^*}\cdot\sqrt{y^*}/(\sqrt{c_0}d) = (2/\sqrt{c_0})\cdot y^*/d.$$

So, if the constant c_0 is chosen large enough, the size of the cover is at most $2y^*(1-\Omega(1/d))$ with high probability.

4.3 Partial Vertex Cover: Multiple Criteria

We now briefly consider multi-criteria k-partial 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-partial vertex 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

$$\forall i,\ y_i^* \geq c_1 \log^2(\ell+n) \tag{9}$$

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.

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

$$\Pr[W \leq (k-1)] \leq 2nm/(2nm+1) = 1-1/(2nm+1),$$

which, though large, is $1-\Omega(1/n^{O(1)})$. Also, for each i, a Chernoff bound easily helps show the following. using the property (9): if the constant c is sufficiently large, then

$$\Pr\left[y_i > 2(1+1/\sqrt{\log(\ell+n)})y_i^*\right] \leq (\ell(4nm+2))^{-1}.$$

Therefore, the probability of existence of an i for which $y_i > 2(1+1/\sqrt{\log(\ell+n)})y_i^*$ holds, is at most $1/(4nm+2)$. Thus, with probability at least $1/(2nm+1) - 1/(4nm+2) = 1/(4nm+2)$ we will have our desired solution; this can be boosted to a high probability by repeating this basic algorithm $O(nm)$ times.

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 re-applied 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 the 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 $1 \ldots x$ 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 $2\binom{n'}{2}$ 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.

For any algorithm A, 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. Let Z_j^A be the number of disks chosen by Algorithm A in strip j. 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

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

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.

$$r_{S_A} \leq r_A(1+1/l) \tag{10}$$

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 (10) shows that $r_{S_A} \leq (1+1/l)^2$. ■

6 k-Partial 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 below.

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 6.1 *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. Note that $\mathrm{IncE}(V_1)$ is the set of edges incident to a vertex in V_1.

Theorem 6.2 follows from Lemma 5.1 and the fact that $r_A = 1$.

Theorem 6.2 *The above algorithm gives a PTAS with a performance ratio* $\leq (1 + \frac{1}{l})$.

7 Concluding Remarks

We have presented improved approximation algorithms for a family of partial covering problems. Since the publication of the preliminary version of this work [16], our results of Section 4.2 have been built upon and improved. First, the work of [36] showed how to obtain faster algorithms for the main result of Section 4.2, by applying a "level-sets" distribution of [36] along with some ideas from Section 4.2. By building further upon these ideas and by employing semidefinite programming, it has been shown in [22] that partial vertex cover in graphs with maximum degree d, can be approximated to within $(2 - \Theta((\log \log d)/\log d))$.

Acknowledgments. We thank Madhav Marathe for useful discussions. We thank the ICALP 2001 referees and the journal referees for their helpful comments. Our thanks also to Daniel Khodorkovsky for pointing out an error in an earlier version of Lemma 4.1. Part of this work was done when the second and third authors attended the DIMACS Workshop on *Multimedia Streaming on the Internet* at the DIMACS Center, Rutgers University, Piscataway, NJ, on June 12–13, 2000.

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. In *Proc. 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] R. Gandhi, S. Khuller, and A. Srinivasan. Approximation Algorithms for Partial Covering Problems. In *Proc. International Colloquium on Automata, Languages, and Programming*, 225-236, 2001.

[17] 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.

[18] 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.

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

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

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

[22] E. Halperin and A. Srinivasan. Improved Approximation Algorithms for the Partial Vertex Cover Problem. In *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 161-174, 2002.

[23] 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.

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

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

[26] 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.

[27] 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.

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

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

[30] 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.

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

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

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

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

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

[36] A. Srinivasan. Distributions on level-sets with applications to approximation algorithms. In *Proc. IEEE Symposium on Foundations of Computer Science*, 588-597, 2001.