Approximation Algorithms for Connected Dominating Sets*

Sudipto Guha†
Dept. of Computer Science
Stanford University
Stanford, CA 94305

Samir Khuller‡
Dept. of Computer Science and UMIACS
University of Maryland
College Park, MD 20742

Abstract

The dominating set problem in graphs asks for a minimum size subset of vertices with the following property: each vertex is required to be either in the dominating set, or adjacent to some vertex in the dominating set. We focus on the related question of finding a *connected dominating set* of minimum size, where the graph induced by vertices in the dominating set is required to be *connected* as well. This problem arises in network testing, as well as in wireless communication.

Two polynomial time algorithms that achieve approximation factors of $2H(\Delta)+2$ and $H(\Delta)+2$ are presented, where Δ is the maximum degree, and H is the harmonic function. This question also arises in relation to the traveling tourist problem, where one is looking for the shortest tour such that each vertex is either visited, or has at least one of its neighbors visited. We also consider a generalization of the problem to the weighted case, and give an algorithm with an approximation factor of $(c_n+1)\ln n$ where $c_n \ln k$ is the approximation factor for the node weighted Steiner tree problem (currently $c_n = 1.6103$). We also consider the more general problem of finding a connected dominating set of a *specified subset* of vertices and provide a polynomial time algorithm with a $(c+1)H(\Delta)+c-1$ approximation factor, where c is the Steiner approximation ratio for graphs (currently $c = 1.644$).

1 Introduction

The *connected dominating set (CDS) problem* is defined as follows. Given a graph $G=(V,E)$, find a minimum size subset S of vertices, such that the subgraph induced by S is connected and S forms a dominating set in G. This problem is known to be NP-hard [8]. Recall that a dominating set is one in which each vertex is either in the dominating set, or adjacent to some vertex in the dominating set.

A related problem is the *traveling tourist problem.* Given a graph $G=(V,E)$ find the shortest walk visiting a subset of vertices, such that each vertex is either visited, or has at

*A preliminary version of this paper appeared in the Proceedings of the Fourth Annual European Symposium on Algorithms (ESA 1996).

†This work was done while this author was at the University of Maryland and his research was supported by NSF Research Initiation Award CCR-9307462. Email addr: `sudipto@cs.stanford.edu`

‡Research supported by NSF Research Initiation Award CCR-9307462, and NSF CAREER Award CCR-9501355. Email addr: `samir@cs.umd.edu`

least one of its neighbors visited. The vertices of the graph correspond to monuments the tourist would like to see, and an edge between two vertices denotes visibility of one monument from another. The shortest such walk would guarantee that the tourist sees all monuments of interest.

A β approximation algorithm for a minimization problem runs in polynomial time and guarantees that the ratio of the cost of the solution to the optimal does not exceed β. We also refer to β as the approximation factor of the algorithm. We show that a β approximation for the connected dominating set problem yields a 2β approximation for the traveling tourist problem. Consider a spanning tree of the connected dominating set S and perform a tree traversal. This yields a walk in which exactly $2(|S|-1)$ edges are traversed. Any set of vertices visited by the tourist, form a connected dominating set. Thus $S \leq \beta \cdot |OPT_{CDS}| \leq \beta \cdot |OPT_{TT}|$, where OPT_{CDS} denotes an optimal connected dominating set and OPT_{TT} denotes an optimal traveling tourist tour, and the result follows.

We study the connected dominating set problem when the vertices have weights, and we wish to minimize the total weighted sum of the vertices that form the connected dominating set. This also yields an approximation algorithm for the weighted traveling tourist problem, where the weights denote the tourist's cost of buying a ticket to visit the monument.

We consider the Steiner CDS problem, where only a specified subset of vertices have to be dominated by a connected dominating set (defined formally in Subsection 1.2). For the unweighted Steiner CDS problem, we provide approximation algorithms that run in polynomial time. When the vertices have weights, the Steiner CDS problem is at least as hard as the notorious set TSP problem on graphs (defined in Subsection 1.2) for which no non-trivial approximation algorithm is known. The same is true for the CDS problem when the edges have weights. Recent work by Marathe, Ravi and Sundaram [18] gives bicriteria approximation algorithms for the "service-constrained network design problem". The goal is to design a tree that has low cost and is sufficiently close to each vertex in the graph (these service requirements can be non-uniform). They obtain bicriteria algorithms by relaxing the domination distance requirement, as well as the network cost.

1.1 Our Results

We present two approximation algorithms for the connected dominating set problem. The first approach is to develop a greedy algorithm for solving the problem. A naive greedy algorithm is shown to do badly. Surprisingly, with a simple modification we are able to show an approximation factor of $2(1 + H(\Delta))$ (in practice, this algorithm appears to do very well). We also provide an efficient implementation of this algorithm. This algorithm is described in Section 2.

The second algorithm is an improvement of the first algorithm. The algorithm finds a dominating set in the first phase, and in the second phase connects the dominating set. In an earlier version of this paper [9] we established a bound of $H(\Delta) + H(H(\Delta))$. Using Slavík's greedy set-cover bound [22], we were able to show that the approximation factor is $\ln n + O(1)$. Recently, Berman (personal communication) suggested a modification to our algorithm [9], which improves the approximation factor to $H(\Delta) + 2$. We describe the algorithm and give a simple proof of an approximation ratio of $\ln \Delta + 3$ (since $\ln \Delta \approx H(\Delta) - 0.7$, the difference is very small). This algorithm is described in Section 3.

We also show an approximation preserving reduction from the set-cover problem to the connected dominating set problem, showing that it is hard to improve the approximation guarantee of $H(\Delta)$ for any graph and asymptotically large n and Δ unless $NP \subseteq DTIME[n^{O(\log\log n)}]$ [17, 7]. This is described in Subsection 5.1.

In Subsection 4.1 we give a $(c_n + 1)\ln n$ approximation for the node weighted CDS problem, where $c_n \ln k$ is the approximation factor for the node weighted Steiner tree problem [13] and k is the number of terminals (currently $c_n = 1.6103$ [10]). We next consider the Steiner CDS problem, where only a specified subset of vertices have to be dominated by a connected dominating set. For the unweighted case, we provide an approximation algorithm that runs in polynomial time in Subsection 4.2. It has an approximation factor of $(1 + c)H(\min(\Delta, k)) + c - 1$, where c is the approximation ratio for the Steiner tree problem (currently $c = 1.644$ [15]) and k is the size of the set we want to dominate.

When the vertices have weights, the Steiner connected dominating set problem is at least as hard as the notorious set TSP problem on graphs (defined shortly) for which no non-trivial approximation algorithms are known. The same is true for the edge weighted CDS problem. These reductions are described in Subsection 5.2.

1.2 Preliminaries and Problem Definitions

In all the following discussion the **weight** of a set is the sum of the weights of the elements contained in it.

The **connected dominating set** problem is the following: given a graph $G = (V, E)$ find the smallest subset S of vertices that induce a connected subgraph and each vertex in $V - S$ is adjacent to at least one vertex in S.

The **node weighted connected dominating set** problem is the generalization of connected dominating set problem to the case where the vertices have weights, and we are looking for the smallest weighted subset S.

The **Steiner connected dominating set** problem is the following: given a graph $G = (V, E)$ and a subset R of required vertices, find the smallest subset S of vertices that induce a connected subgraph and each vertex of $R - S$ is adjacent to at least one vertex on S.

The **node weighted Steiner connected dominating set** problem is the generalization of Steiner connected dominating set problem to the case where the vertices have weight, and we are looking for the smallest weighted subset S.

The **edge weighted connected dominating set** problem is the following: given a graph $G = (V, E)$ with weights on the edges, find a smallest weight tree whose vertices form a dominating set.

The **Steiner tree** problem is defined as follows: given a graph $G = (V, E)$, and a set $R \subseteq V$ of required vertices (terminals) in an edge weighted graph, find a minimum weight tree connecting the terminals. Note that the tree may include other vertices that are not required vertices. These vertices are termed **Steiner vertices**.

The **node weighted Steiner tree** problem is the Steiner tree problem, except that the vertices of the graph have weights associated with them and the weight of the tree is the sum of the weights of its vertices. (The case when both vertices and edges have weights can easily be reduced to the case when only the vertices have weights.)

The **set cover** problem is the following: given a set of elements U, and a set $\mathcal{S}$ of subsets of U, we wish to find the smallest collection of sets $\mathcal{S}' \subset \mathcal{S}$ such that $\cup_{S \in \mathcal{S}'} S = U$.

The **set TSP** problem is defined as follows: given an edge weighted graph $G = (V, E)$ and a partition of $V = (V_1 \cup V_2 \cup \ldots \cup V_k)$, find the shortest tour that contains at least one vertex from each V_i.

Given a graph $G = (V, E)$, we use Δ to denote the maximum degree of a vertex in the graph. We use n and m to denote the numbers of vertices and edges in G, respectively. We use $N(v)$ to denote the set of neighbors of a vertex v. The degree of a vertex v is denoted by $d(v)$.

1.3 Applications and Related Work

The paper by Paul and Miller [20] discusses applications related to testing nodes in a computer network using a short "traveling tourist tour". They also consider the related question of finding a tour that visits each edge of the graph (connected vertex cover). This is needed when one requires testing the links as well as the nodes. Approximation algorithms for the latter problem were given by Arkin, Halldórsson and Hassin [1]. We observe that there is a simple algorithm for the unweighted connected vertex cover problem that gives a factor 2 approximation (the one given in [1] is more complicated). Perform a Depth First Search, and take all the non-leaf vertices as the vertex cover. This clearly induces a connected graph, and the approximation ratio is 2, as shown by Savage [21]. In practice, however this method will probably give large connected vertex covers.

Other applications for the connected dominating set problem are in doing broadcasts for wireless computers in digital battlefields. The broadcast is done to the vertices in the connected dominating set. The nodes in the connected dominating set are responsible for relaying messages. Each node not in the dominating set, is not responsible for relaying any messages [12]. Many of the ideas in our paper have been used to design a distributed algorithm for routing based on minimum connected dominating sets in ad-hoc networks [5].

Recent work by Marathe, Ravi and Sundaram [18] gives bicriteria approximation algorithms for the "service-constrained network design problem". The goal is to design a tree that has low cost and is sufficiently close to each node in the graph (these service requirements can be non-uniform). They obtain bicriteria algorithms by relaxing the domination distance requirement, as well as the network cost.

The nodes in the connected dominating set together with the remaining nodes of the graph form a spanning tree with *many* leaves. The maximum leaf spanning tree problem is another related problem that has been studied and factor 3 approximations are known for it [16].

Recently, Harary and Raghavachari [11] have shown that the email gossip number of a graph is exactly $n-1+OPT_{CDS}$, where OPT_{CDS} is the size of the optimal connected dominating set. This indicates that the connected dominating set problem has many interesting applications. Polynomial algorithms for the connected dominating set problem for special classes of graphs were given by White, Farber and Pulleyblank [23].

2 Algorithm I

We introduce an algorithm that finds a connected dominating set, by "growing" a tree.

The idea behind the algorithm is the following: grow a tree T, starting from the vertex of maximum degree. At each step we will pick a vertex v in T and "scan it". Scanning a vertex, adds edges to T from v to *all* its neighbors not in T. In the end we will find a spanning tree T, and will pick the non-leaf nodes as the connected dominating set.

Initially all vertices are unmarked (white). When we scan a vertex (color it black), we mark all its neighbors that are not in T and add them to T (color them gray). Thus marked nodes that have not been scanned are leaves in T (gray nodes). All unmarked nodes are white. The algorithm continues scanning marked nodes, until all the vertices are marked (gray or black). The set of scanned nodes (black nodes) will form the connected dominating set (CDS) in the end.

The main question is the following: what rule should we use for picking a vertex to be scanned? A natural choice is to pick the vertex that has the maximum number of unmarked (white) neighbors. We call this the "yield" of the scan step. Unfortunately, as the following example shows this may not work well (see Fig. 1).

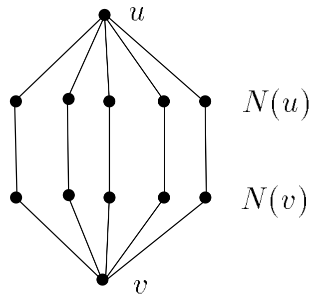

Figure 1: Example to show that the scanning rule fails for $d = 5$.

Let u and v be vertices of degree d. There is a solution of size four, by picking a path from u to v as the CDS. The algorithm begins by marking and scanning u. This adds all of u's neighbors to T. We pick a vertex from $N(u)$ and scan it, adding its only unmarked neighbor (from $N(v)$) to T. At this point, each vertex in T has exactly one unmarked neighbor. We could pick a vertex from $N(u)$ again, and scan it, adding its only unmarked neighbor to T. This continues until all the vertices from $N(u)$ have been scanned. Finally we scan a vertex from $N(v)$ and mark v. At this point, the algorithm has picked $d+2$ vertices.

The entire algorithm can be implemented in $O(m)$ steps [9]. This implementation is useful because it leads to a heuristic for the *maximum leaf spanning tree* problem as well [14].

Modified Greedy Algorithm: We now modify the scanning rule to prove a good approximation ratio for this class of algorithms (that maintain a connected set that eventually becomes a CDS). We define a new operation of scanning a pair of adjacent vertices u and v. Let u be gray and v be white. *Scanning the pair* means, first making u black (this makes v along with some other nodes, gray) and then coloring v black (makes more nodes gray). The total number of nodes that are colored gray is called the "yield" of the scan step. *At each step, we will either scan a single vertex, or a pair of vertices, whichever gives the higher yield.* (In some sense we are doing a "look-ahead" by one extra vertex, and are willing to scan a pair of vertices if this has a higher yield.)

It is clear that this algorithm finds the optimal solution in the example shown in Fig. 1. What is perhaps a little surprising, is that this simple modification lets us prove the following theorem.

Theorem 2.1 *Using the scanning rule described above yields a connected dominating set of size at most* $2(1 + H(\Delta)) \cdot |OPT_{DS}|$*, where* OPT_{DS} *is an optimal dominating set*[1] *in the graph.*

Proof: Let OPT_{DS} be the set of vertices in an optimal dominating set. The sets of vertices of G dominated by vertex $i \in OPT_{DS}$ is called S_i (we assume that i also belongs to S_i). If a vertex is dominated by more than one vertex, we arbitrarily put it in one of the sets. The proof will be based on a charging scheme. Each time we color a vertex black, we add a new vertex to our connected dominating set. We will "charge" each new vertex marked (colored gray) in this step. Since each vertex in the graph gets marked exactly once, it is charged exactly once (the first time it is marked). We will then prove that the total charge on the vertices belonging to a set S_i (for any i) is at most $2(1 + H(\Delta))$. Since there are $|OPT_{DS}|$ sets in the optimal solution, the theorem follows.

Assume that when we color a vertex black, we mark x new vertices. We will charge each such newly marked vertex $\frac{1}{x}$. In some steps we scan two vertices, and charge each newly marked vertex $\frac{2}{x}$. The main advantage of the "look-ahead" is the following. *The instant we mark some nodes in set* S_i*, even if vertex* i *has not been marked, since it is adjacent to a marked vertex, it becomes eligible to be scanned as part of a pair.* Without the "look-ahead", only marked vertices were candidates to be scanned.

We now prove the upper bound on the total charges to vertices belonging to a single set S_i. At each step, some vertices may get marked. The number of unmarked vertices in S_i is initially u_0, and finally drops to 0. Let u_j denote the number of unmarked vertices in S_i after step j. For simplicity, let us assume that at each step some vertices of S_i are marked, so the number of unmarked vertices in S_i decreases at each step.

The number of marked vertices in S_i after the first step is $u_0 - u_1$. Each vertex gets a charge of at most $\frac{2}{u_0 - u_1}$ (the actual charge may be a lot smaller, if only one vertex was scanned at this step, or if we marked many other vertices as well). Once some vertex in S_i is marked, vertex i

[1] It is easy to see that $|OPT_{DS}| \leq |OPT_{CDS}|$.

becomes an "eligible" vertex to be scanned as a part of a pair, since it is *adjacent* to a marked vertex. In the j^{th} step, the number of vertices of set S_i that get marked is $u_j - u_{j+1}$, and the charge to each vertex in S_i is at most $\frac{2}{u_j}$ as vertex i was an eligible vertex to be scanned. Let $u_k = 0$.

Adding up all the charges we get

$$\frac{2}{u_0 - u_1}(u_0 - u_1) + \sum_{j=1}^{k-1} \frac{2}{u_j}(u_j - u_{j+1})$$

$$\leq 2 + 2\sum_{j=1}^{k-1} \frac{(u_j - u_{j+1})}{u_j} \ .$$

(With some algebraic manipulation (see [6, page 977]) and using the fact that $u_0 \leq \Delta + 1$, one can show that this is at most $2(1 + H(\Delta))$. □

Remark: We could modify the algorithm and at each step scan either one or two vertices, whichever results in a smaller charge to each vertex. In practice, this should give better solutions. Thus we only pick a pair of nodes if its yield is at least twice as that of a single node.

Implementation Issues: A naive implementation appears to give a worst case running time of $O(mn^2)$. In each iteration we choose either one vertex, or a pair of vertices, and color them black. It is clear that we may have $\Theta(n)$ iterations, since the optimal solution may have $\Theta(n)$ vertices. In each iteration, we wish to identify a pair of nodes with the highest yield. For each gray vertex u, we scan its adjacency list and consider all its white neighbors. For *each* white neighbor v of u, we wish to determine the number of vertices that would get marked if we scanned the pair (u, v). Since u and v have common white neighbors, we cannot simply add up the number of white neighbors of each vertex to obtain the "yield" of this pair. We need to identify the number of white neighbors of v that are *not* adjacent to u (since those will not be colored gray by u). The number of steps in a single iteration can be computed as follows.

Let GR be the gray nodes in T. Let W be the white vertices that are adjacent to gray vertices. We can upper bound the total work done in a single iteration as follows:

$$S = \sum_{u \in GR} \ \sum_{v \in N(u) \wedge v \in W} d(v) \ .$$

In the double summation each vertex in W is counted as many times as the number of its gray neighbors, we obtain the following.

$$S \leq \sum_{v \in W} d(v)^2 \leq \sum_{v \in W} n \cdot d(v) \leq O(mn) \ .$$

This yields a bound of $O(mn^2)$. We now show that the total number of steps over all iterations is $O(mn)$ by a more careful analysis.

For each vertex we can maintain two adjacency lists, one of its gray neighbors and one of its white neighbors. We use $d_W(u)$ to denote the number of white neighbors of u and $d_{GR}(u)$ to denote the number of gray neighbors of u. The work done in a single iteration is as follows:

$$S = \sum_{u \in GR} \sum_{v \in W \wedge v \in N(u)} d_W(v)$$

$$= \sum_{v \in W} d_W(v) \cdot d_{GR}(v) \ .$$

(In the double summation, each vertex v is counted as many times as the number of its gray neighbors.) Observe that at this step, we will make a subset of white vertices gray.

Lemma 2.2 *The number of white vertices that are made gray in this iteration is at least*

$$\max_{v \in W} d_W(v) \ .$$

Proof: We pick the pair of vertices that give the highest "yield"; we certainly consider all such vertices v, and color their white neighbors gray. □

At this step, we can "charge" the vertices whose color changed from white to gray. The charge to each such vertex is at most

$$\frac{\sum_{v \in W} d_W(v) \cdot d_{GR}(v)}{\max_{v \in W} d_W(v)}$$

$$\leq \sum_{v \in W} d_{GR}(v) \leq 2m \ .$$

Since each vertex changes color from white to gray exactly once over the entire algorithm, and there are n such vertices the total number of steps is $O(mn)$.

The only remaining issue is maintaining the required adjacency lists. This can be done each time we change the color of a vertex from white to gray by scanning its adjacency list, and updating the structures for its neighbors.

3 Algorithm II

An alternate approach to growing one connected tree is to grow separate components that form a dominating set and to then connect them together. One straightforward approach is to find a dominating set using a greedy heuristic, and to use a Steiner tree algorithm to connect it. Since members of the optimum connected dominating set along with the members of the dominating set we found, induce a connected subgraph, we can prove an approximation ratio of $c(1 + H(\Delta))$, where c is the approximation ratio for the unweighted Steiner tree problem (currently $c = 1.644$ [15]).

For the special case when the required vertices form a dominating set in a graph and all edges have unit weight, Berman and Fürer [3] have announced a new algorithm with $c = \frac{4}{3}$.

Thus we can improve the approximation ratio to $\frac{4}{3}(1 + H(\Delta))$ by using their algorithm. By applying a simple greedy strategy to connect the vertices in the dominating set, we proved a bound of $H(\Delta) + H(H(\Delta))$ [9]. Here we present a modification of the above algorithm, as suggested by Berman, and prove an approximation ratio of $\ln \Delta + 3$. (Berman has an alternate proof for an approximation ratio of $H(\Delta) + 2$.)

The algorithm runs in two phases. At the start of the first phase all nodes are colored white. Each time we include a vertex in the dominating set, we color it black. Nodes that are dominated are colored gray (once they are adjacent to a black node). In the first phase the algorithm picks a node at each step and colors it black, coloring all adjacent white nodes gray. A *piece* is defined as a white node or a black connected component. At each step we pick a node to color black that gives the maximum (non-zero) reduction in the number of pieces.

We show that at the end of this phase if no vertex gives a non-zero reduction to the number of pieces, then there are no white nodes left.

In the second phase, we have a collection of black connected components that we need to connect. Recursively connect pairs of black components by choosing a chain of vertices, until there is one black connected component. Our final solution is the set of black vertices that form the connected component.

Lemma 3.1 *At the end of the first phase there are no white vertices left.*

Proof: Suppose there is a white node v at the end of the phase. We will show that there is a vertex that strictly reduces the number of pieces.

All neighbors of v are white or gray. If v has a white neighbor then coloring v black, reduces the number of white nodes by two, and increases the number of black components by one, thus picking v would reduce the number of pieces. Otherwise, v has a gray neighbor u. Coloring u black would reduce the number of white nodes, and not increase the number of black components since u is adjacent to a black node. Thus picking u reduces the number of pieces. □

Lemma 3.2 *At the end of the first phase if there is more than one black component, then there is always a pair of black components that can be connected by choosing a chain of two vertices.*

Proof: Consider the shortest path connecting two black components. Assume this path consists of vertices $u_0, u_1, u_2, u_3, \ldots, u_k$ where u_0 and u_k belong to black components i and j respectively. Vertex u_1 is dominated by vertex u_0. If u_2 is black then we can reduce the number of pieces by making u_1 black. By Lemma 3.1, u_2 must be gray. Vertex u_2 is adjacent to a black component ℓ, distinct from i. Components i and ℓ can be connected by choosing a chain of two vertices. □

Theorem 3.3 *The connected dominating set found by the algorithm is of size at most* $(\ln \Delta + 3) \cdot |OPT_{CDS}|$.

Proof: Define a_i to be the number of pieces left after the i^{th} iteration, and $a_0 = n$. Since a node can connect up to Δ pieces, $|OPT_{CDS}| \geq \frac{a_0}{\Delta}$. (This is true if the optimal solution has at least two nodes.) Consider the $i+1^{th}$ iteration. The optimal solution can connect a_i pieces. Hence the greedy procedure is guaranteed to pick a node which connects at least $\left\lceil \frac{a_i}{|OPT_{CDS}|} \right\rceil$ pieces. Thus the number of pieces will reduce by at least $\left\lceil \frac{a_i}{|OPT_{CDS}|} \right\rceil - 1$ This gives us the recurrence relation,

$$a_{i+1} \leq a_i - \left\lceil \frac{a_i}{|OPT_{CDS}|} \right\rceil + 1 \leq a_i(1 - \frac{1}{|OPT_{CDS}|}) + 1 \ .$$

Its solution is,

$$a_{i+1} \leq a_0(1 - \frac{1}{|OPT_{CDS}|})^i + \sum_{j=i}^{i-1}(1 - \frac{1}{|OPT_{CDS}|})^j \ .$$

Notice after $|OPT_{CDS}| \cdot \ln \frac{a_0}{|OPT_{CDS}|}$ iterations, the number of pieces left is less than $2 \cdot |OPT_{CDS}|$. After this, for each node we choose, we will decrease the number of pieces by at least one until the number of black components is at most $|OPT_{CDS}|$, thus at most $|OPT_{CDS}|$ more vertices are picked. So after $|OPT_{CDS}| \cdot \ln \frac{a_0}{|OPT_{CDS}|} + |OPT_{CDS}|$ iterations at most $|OPT_{CDS}|$ pieces are left to connect.

Assume from this point onwards, we stop after choosing a_f more nodes. The number of pieces left to connect is at most $|OPT_{CDS}| - a_f$. We connect the remaining pieces choosing chains of two vertices in the second phase. The total number of nodes chosen is at most $|OPT_{CDS}| \cdot \ln \frac{a_0}{|OPT_{CDS}|} + |OPT_{CDS}| + a_f + 2(|OPT_{CDS}| - a_f)$, and since $\Delta \geq \frac{a_0}{|OPT_{CDS}|}$, the solution found has at most $|OPT_{CDS}| \cdot (\ln \Delta + 3)$ nodes. $\square$

Remark: Berman, [2], has an alternate proof of $H(\Delta) + 2$ of the same algorithm. However, since $\ln \Delta \approx H(\Delta) - 0.7$, the difference is very small.

4 Generalizations

4.1 Node Weighted Connected Dominating Sets

An approximation factor of $(c_n + 1) \ln n$ is possible for the node weighted connected dominating set problem. The algorithm first finds a dominating set, and then connects the nodes in the dominating set.

Step 1. Use a weighted set cover approximation algorithm to find a dominating set DS. (A set cover instance is created by making each vertex an element, and each vertex corresponds to a set that contains the vertex itself, together with its neighbors. The greedy algorithm picks sets based on the ratio of their weight to the number of new elements they cover.)

Step 2. To connect the vertices in DS we use a node-weighted Steiner tree approximation algorithm [13, 10] to find a Steiner tree that includes all the vertices in DS, after making the weights of all vertices in DS equal to zero. This yields a connected dominating set CDS.

Theorem 4.1 *The weight of vertices in CDS is at most $(c_n + 1) \ln n \cdot w(OPT_{CDS})$ where OPT_{CDS} is the minimum weight connected dominating set in G, and $c_n \ln k$ is the approximation factor for the node weighted Steiner tree problem for k terminals (currently $c_n = 1.6103$).*

Proof: The weight of the vertices in DS is at most $\ln \Delta \cdot w(OPT_{CDS})$. We now run the node weighted Steiner tree algorithm [10] to find a node-weighted Steiner tree which connects the vertices in DS. The approximation factor of the algorithm is $1.6103 \ln k$, where k is the number of terminals (the paper by Klein and Ravi [13] gives a $2 \ln k$ approximation factor). Consider the vertices in OPT_{CDS}; these together with the vertices in DS induce a connected subgraph. Hence there exists a node weighted Steiner tree of weight $w(OPT_{CDS})$. Thus the tree found in Step 2 weighs at most $1.6103 \ln n \cdot w(OPT_{CDS})$. The total weight of the vertices in the connected dominating set is the weight of DS together with the weights of optional vertices chosen from G in the Steiner instance. Adding the weight of the two sets gives the required bound. □

4.2 Steiner Connected Dominating Sets

We now address the Steiner connected dominating set problem when we are required to dominate only a specified subset R of the vertices. The cost of the solution is the size of the smallest connected dominating set that dominates the vertices in R. (Notice that the objective function is slightly different from the node weighted Steiner tree problem, where required vertices have zero cost. In the Steiner CDS problem, we are charged for all vertices in the final solution that are not leaf nodes in the tree that connects R.)

Let $|R| = k$, and $OPT_{CDS}(R)$ denote the optimal solution. We present an algorithm that solves this problem. A straightforward strategy is to first find a small dominating set A, of the vertices in R, and to then connect these nodes.

Algorithm

Step 1. Modify the greedy set cover algorithm to run on the set of elements R, with the vertices in V and the nodes in R that they dominate, corresponding to sets; until no vertex covers strictly more than one uncovered vertex of R. We call the set of vertices chosen as B.

Step 2. Choose the uncovered vertices of R, call this set B'.

Step 3. For each member of B, choose a representative element of R that it dominates. Let this set be $\mathcal{R}(B)$. Apply an (edge weighted) Steiner tree approximation, with the set of required nodes $\mathcal{R}(B) \cup B'$. The final solution is the nodes of this tree and the nodes of B.

Theorem 4.2 *The connected dominating set for the subset R, is at most $(c+1)H(\delta) + c - 1$ times the optimal (where c is the Steiner approximation ratio). We define δ as the size of the largest subset of R, adjacent to a node in the graph ($\delta \leq \min(\Delta, k)$).*

Proof: By a slight modification to the proof given in [6, page 977] we can prove, $|B| \leq (H(\delta) - 1) \cdot |OPT_{CDS}(R)|$. (Since the first step reduces to finding a set cover with the size of the largest set being δ.) Since $OPT_{CDS}(R)$ cannot dominate any two vertices of B' by one vertex, $|B'| \leq |OPT_{CDS}(R)|$. Note that $B \cup B'$ dominates the set R.

Consider the set $\mathcal{R}(B)$; there is a Steiner tree with $|\mathcal{R}(B)| + |B'| + |OPT_{CDS}(R)|$ edges that connects the nodes of $\mathcal{R}(B) \cup B'$.

Apply an (edge weighted) Steiner tree approximation, with all edges having unit weight, and find a tree of size $c \cdot (|\mathcal{R}(B)| + |B'| + |OPT_{CDS}(R)|)$, where c is the Steiner approximation ratio [15]. Since this tree is edge weighted, it has essentially the same number of nodes, including those of $\mathcal{R}(B) \cup B'$. Since we have to add the vertices of B as well, we get an upper bound of $c \cdot (|\mathcal{R}(B)| + |B'| + |OPT_{CDS}(R)|) + |B|$. Notice that $|\mathcal{R}(B)| \leq |B| \leq (H(\delta) - 1) \cdot |OPT_{CDS}(R)|$, and $|B'| \leq |OPT_{CDS}(R)|$. This gives us a solution of cost at most $((c+1) \cdot H(\delta) + c - 1) \cdot |OPT_{CDS}(R)|$. □

5 Lower Bounds on Approximation Factors

5.1 Hardness result for Connected Dominating Set

We can prove that the set-cover problem can be reduced to the connected dominating set problem by an approximation preserving reduction, thus showing that the approximation factor $H(\Delta)$ will be hard to improve for general graphs. This is based on the hardness results for set cover proven by Lund and Yannakakis [17] and Feige [7].

Given a set cover instance we reduce it to a connected dominating set problem as follows:

Let the set cover instance be to cover set U, with minimum number of sets from the collection $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$.

Construct a graph G, that has vertex set $U \bigcup \{u, v, v_1, v_2, \ldots, v_m\}$. An element $e \in U$, and v_i has an edge joining them if and only if $e \in S_i$. Each v_i has an edge to v. Vertex u has an edge only to v (see Fig. 2).

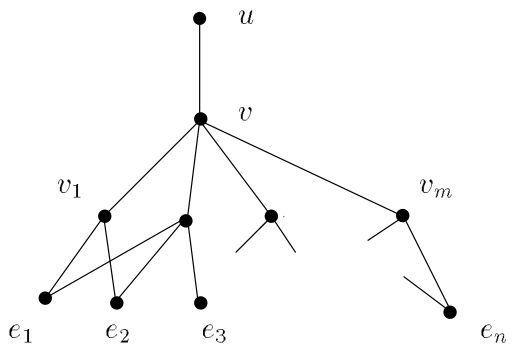

Figure 2: Reduction of set cover to connected dominating sets.

Consider a minimum connected dominating set of G. Vertex v belongs to any connected dominating set, and hence u does not belong to any minimal connected dominating set. No vertex e_j is chosen in a minimal connected dominating set, since any node that it might potentially dominate, is already dominated by v, which also provides the connectivity. Hence we will only have v and some v_i's. These v_i's will correspond to the minimum cover for the given instance of set cover.

The size of the connected dominating set is one more than the minimum set cover. Thus approximating the connected dominating set with a factor of $(1-\epsilon)H(\delta)$ (for all Δ) would imply approximating minimum set cover within the same factor. This would imply that $NP \subseteq DTIME[n^{O(\log\log n)}]$ [7].

5.2 Hardness results for Generalizations

We show two simple reductions that demonstrate that other generalizations of the CDS problem may be as hard to approximate as the "set TSP" problem for which no non-trivial approximation algorithms are known. (For the Euclidean case, Mata and Mitchell [19] have given approximation algorithms for this problem.)

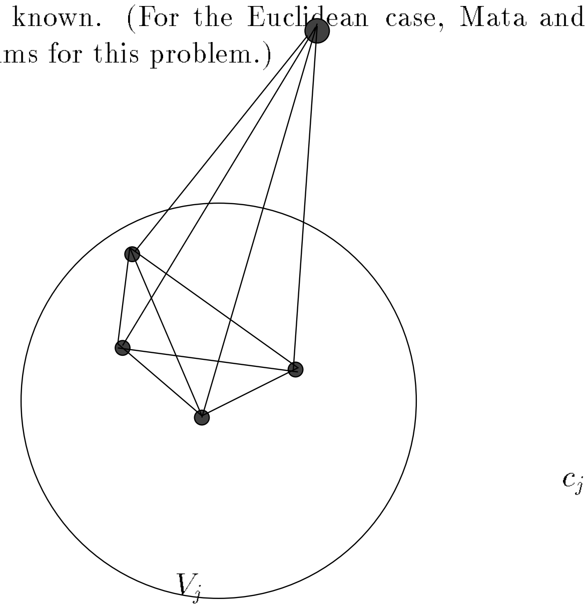

Figure 3: Reduction of set TSP problem to edge weighted CDS.

Theorem 5.1 *A polynomial approximation algorithm for the edge weighted connected dominating set problem with factor $f(n)$ would imply a polynomial approximation algorithm for the set TSP problem with factor $2f(n)$.*

Proof: We show how to reduce the set TSP problem to the edge weighted connected dominating set problem. Consider a set TSP instance $G=(V,E)$ where $V=(V_1 \cup V_2 \cup \ldots \cup V_k)$. For each subset V_j, introduce a special vertex c_j, and add edges from c_j to all $v \in V_j$, with very high cost edges. For $u, v \in V_j$, if $(u,v) \notin E$, add the edge (u,v) with very high cost. Call this new graph G'.

Any set TSP tour in G chooses at least one vertex of V_j to visit. Thus all nodes of $V_j \cup \{c_j\}$ will be dominated by the corresponding node in the tour. Since every node of G occurs in some V_j, this yields a dominating set. Since these are nodes on a tour, they also form a connected set. Hence $OPT_{CDS}(G') \leq OPT_{TOUR}(G)$.

If we have a connected dominating set of G', then it must have a vertex of V_j to dominate c_j. Hence the dominating set must have at least one vertex from each set V_j. If the cost of this connected dominating set is small ($\leq f(n)OPT_{CDS}(G')$), since we are not using the high cost edges in G', we are using only the edges of the graph G. By traversing this tree twice, we can produce a tour in G, with cost at most $2f(n)OPT_{CDS}(G') \leq 2f(n)OPT_{TOUR}(G)$. Thus, if we can approximate the connected dominating set with edge weights to a factor $f(n)$, we can approximate set TSP within a factor $2f(n)$. □

Theorem 5.2 *A polynomial approximation algorithm for the node weighted Steiner connected dominating set problem with factor $f(n)$ would imply a polynomial approximation algorithm for the set TSP problem with factor $2f(n)$.*

Proof: The proof is similar to the proof of the previous theorem. Given a set TSP instance $G = (V, E)$ where $V = (V_1 \cup V_2 \cup \ldots \cup V_k)$ we construct a graph G'. First convert the edge weights of the set TSP problem into node weights. For every edge $e = (v_p, v_q) \in E$, create an extra node v_{pq} of the same cost, connected to v_p and v_q. All other nodes are given 0 cost. For every subset V_j, introduce a special vertex c_j (of very high cost), and connect it to all $v \in V_j$. We show that the problem reduces to finding a node weighted Steiner CDS of the subset $R = \{c_j | j = 1 \ldots k\}$ of nodes of G'.

Any set TSP tour in G chooses at least one vertex of V_j to visit. Thus each c_j will be dominated. The weight of the edges $e = (v_p, v_q)$ translates to the weight of the corresponding vertices v_{pq}. Since the nodes form a tour, they also form a connected set in G', together with the new nodes that subdivide edges. Thus $OPT_{CDS}(G') \leq OPT_{TOUR}(G)$.

Consider a connected dominating set that dominates R. To dominate c_j, it must pick a vertex from V_j. (W.l.o.g, the connected dominating set does not contain c_j.) If the cost of this connected dominating set is small ($\leq f(n)OPT_{CDS}(G')$), since we are not using the high cost nodes in G', we are using only the nodes of the graph G along with nodes that correspond to the subdivided edges. Thus the dominating set chooses vertices that are also in G, and the corresponding vertices for each edge of G that it includes. This yields a tree that connects at least one element from each V_j using edges of G. By traversing this tree twice, we can produce a tour in G, of cost at most $2f(n)OPT_{CDS}(G') \leq 2f(n)OPT_{TOUR}(G)$. Thus, if we are able to approximate the connected dominating set of a subset with node weights to a factor $f(n)$, we can approximate set TSP within a factor $2f(n)$. □

Acknowledgments: We thank Estie Arkin, Randeep Bhatia, Ray Miller, Serge Plotkin, Balaji Raghavachari and Azriel Rosenfeld for useful discussions. We thank Vaduvur Bharghavan for re-igniting our interest in the connected dominating sets problem. We thank Piotr Berman for allowing us to include his modifications to the algorithm presented in [9] that led to Algorithm II. We also thank the diligent reviewer for detailed comments on an earlier draft of this paper.

References

[1] E. Arkin, M. Halldórsson and R. Hassin, "Approximating the tree and tour covers of a graph", *Information Processing Letters*, 47: 275–282, (1993).

[2] P. Berman, personal communication, May (1996).

[3] P. Berman and M. Fürer, personal communication, May (1996).

[4] P. Berman and V. Ramaiyer, "Improved approximation algorithms for the Steiner tree problem", *J. Algorithms*, 17:381–408, (1994).

[5] V. Bharghavan and B. Das, "Ad-hoc routing using minimum connected dominating sets", to appear *IEEE International Conference on Communications*, (1997).

[6] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, The MIT Press, 1989.

[7] U. Feige, "A threshold of $\ln n$ for approximating set-cover", *28th ACM Symposium on Theory of Computing*, pages 314–318, (1996).

[8] M. R. Garey and D. S. Johnson, "Computers and Intractability: A guide to the theory of NP-completeness", *Freeman, San Francisco* (1978).

[9] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets", *Proc. of 4th Annual European Symposium on Algorithms*, pages 179–193, (1996).

[10] S. Guha and S. Khuller, "Improved approximation algorithms for node weighted Steiner trees", *manuscript* (1997).

[11] F. Harary and B. Raghavachari, "The E-mail gossip number and the connected domination number", submitted to *Applied Math. Letters* (1996).

[12] A. Kothari and V. Bharghavan, "Algorithms for unicast and multicast routing in ad-hoc networks", *manuscript* (1996).

[13] P. N. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner trees", *J. Algorithms*, 19(1):104–114, (1995).

[14] D. Kleitman and D. West, "Spanning trees with many leaves", *SIAM Journal on Discrete Mathematics*, 4(1):99–106, (1991).

[15] M. Karpinsky and A. Zelikovsky, "New approximation algorithms for the Steiner tree problem", *Technical Report, Electronic Colloquium on Computational Complexity* (ECCC): TR95-030, (1995).

[16] H. I. Lu and R. Ravi, "The power of local optimization: approximation algorithms for maximum-leaf spanning tree", *Proc. of the* 30^{th} *Annual Allerton Conf. on Communication, Control and Computing* pages 533–542, (1992).

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

[18] M. Marathe, R. Ravi and R. Sundaram, "Service-constrained network design problems", *Proc. of 5th Scandinavian Workshop on Algorithm Theory*, pages 28–40, (1996).

[19] C. S. Mata and J. S. B. Mitchell "Approximation algorithms for geometric tour and network design problems", *Proc. of the 11th Annual Symp. on Computational Geometry*, pages 360–369, (1995).

[20] S. Paul and R. Miller, "Locating faults in a systematic manner in a large heterogeneous network", *IEEE INFOCOM*, pages 522–529, (1995).

[21] C. Savage, "Depth-First search and the vertex cover problem", *Information Processing Letters*, 14(5): 233–235, (1982).

[22] P. Slavík "A tight analysis of the greedy algorithm for set cover", *28th ACM Symposium on Theory of Computing*, pages 435–441, (1996).

[23] K. White, M. Farber and W. Pulleyblank, "Steiner trees, connected domination and strongly chordal graphs", *Networks*, 15: 109–124, (1985).