10 Distributed Dominating Set Approximation

A dominating set of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes such that for all nodes $v \in V$, either $v \in S$ or a neighbor u of v is in S. There are many distributed applications where computing a small dominating set of the network graph is important. It is well-known that computing a dominating set of minimal size is NP-hard. We therefore look for approximation algorithms, that is, algorithms which produce solutions which are optimal up to a certain factor.

10.1 Sequential Greedy Algorithm

In order to understand the problem, we start with a very simple sequential algorithm. We start with $S = \emptyset$ and greedily add 'good' nodes to S until S is a dominating set. We call nodes in S *black*, nodes which are covered (neighbors of nodes in S) *grey*, and all uncovered nodes *white*. By $w(v)$, we denote the number of white nodes among the direct neighbors of v, including v itself. We call $w(v)$ the *span* of v. The most natural greedy approach works as follows.

Greedy Algorithm:

1: $S := \emptyset$;
2: **while** $\exists$ white nodes **do**
3: $\quad v := \{v \mid w(v) = \max_u\{w(u)\}\}$;
4: $\quad S := S \cup v$;
5: **od**

Theorem 10.1. *The* Greedy Algorithm *computes a* $\ln \Delta$*-approximation, that is, for the computed dominating set* S *and an optimal dominating set* S^**, we have*

$$\frac{|S|}{|S^*|} \leq \ln \Delta.$$

Proof. We prove the theorem using amortized analysis. Each time we choose a new node of the dominating set (each greedy step), we have cost 1. Instead of assigning the whole cost to the node we have chosen, we distribute the cost equally among all newly covered nodes. Assume that node v, chosen in line 3 of the algorithm, is white itself and that its white neighbors are v_1, v_2, v_3, and v_4. In this case each of the 5 nodes v and $v_1, \ldots, v_4$ get charged $1/5$. If v is chosen as a grey node, only the nodes $v_1, \ldots, v_4$ get charged (they all get $1/4$).

Now, assume that we know an optimal dominating set S^*. By the definition of dominating sets, to each node which is not in S^*, we can assign a neighbor from S^*. By assigning each node to exactly one node of S^*, the graph is decomposed into stars, each having a dominator (node in S^*) as center and non-dominators as leaves. Clearly, the cost of an optimal dominating set is 1 for each such star. In the following, we show that the amortized cost (distributed costs) of the greedy algorithm is at most ($\mathrm{H}(\Delta) \approx \ln \Delta$) for each star. This suffices to prove the theorem.

Thus, we now look at a single star with center $v \in S^*$. Let $w(v)$ be the number of white nodes in the star of v. If some nodes in the star of v become gray, they get charged some cost. By the greedy condition of the algorithm, this weight can be at most $1/w(v)$ per newly covered node. Otherwise, the algorithm could rather have chosen v for the dominating set because v would cover at least $w(v)$ nodes. After becoming grey, nodes do not get charged any more. In the worst case, no two nodes in the star of v are covered together. In this case, the first nodes gets charged at most $1/(\delta(v) + 1)$, the

second node gets charged at most $1/\delta(v)$, and so on. Thus, the total amortized cost of a star is at most

$$\frac{1}{\delta(v)+1} + \frac{1}{\delta(v)} + \cdots + \frac{1}{2} + \frac{1}{1} = \mathrm{H}(\delta(v)+1) \leq \mathrm{H}(\Delta+1) = \ln(\Delta) + \mathrm{O}(1)$$

where Δ denotes the maximal degree of graph G. □

Remarks:

- One can show that unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\mathrm{O}(\log\log n)})$, no polynomial-time algorithm can approximate the minimum dominating set problem better than $\ln\Delta - \mathrm{o}(\ln\Delta)$. Thus, unless $\mathrm{P} \approx \mathrm{NP}$, the simple greedy algorithm is optimal.

10.2 Distributed Greedy Algorithm

Observation: The span of a node can only be reduced if any of the nodes at distance at most 2 are included in the dominating set. Therefore, if the span of node v is greater than the span of any other node at distance at most 2 from v, the greedy algorithm chooses v before any of the nodes at distance at most 2. This leads to a very simple distributed version of the greedy algorithm. Every node v executes the following algorithm.

Distributed Greedy Algorithm:

```
1: while v has white neighbors do
2:    compute span w(v);
3:    send w(v) to nodes at distance at most 2;
4:    if w(v) largest within distance 2 (ties are broken by unique IDs) then
5:       join dominating set
6:    fi
7: od
```

Theorem 10.2. *The* Distributed Greedy Algorithm *computes a* $\ln\Delta$*-approximation for the minimum dominating set problem in* $\mathrm{O}(n)$ *rounds.*

Proof. The approximation quality follows directly from the above observation and the analysis of the *greedy algorithm*. The time complexity is at most linear because in every other round, at least one node is added to the dominating set. □

The approximation ratio of the above distributed algorithm is optimal. However, the time complexity is very bad. In fact, there are graphs on which in each iteration of the while loop, only one node is added to the dominating set. As an example, consider a graph as in Figure 1. An optimal dominating set consists of all nodes on the center axis. The *distributed greedy algorithm* computes an optimal dominating set, however, the nodes are chosen sequentially from left to right. Hence, the running time of the algorithm is $\Omega(\sqrt{n})$.

The problem of the graph of Figure 1 is that there is a long path of descending degrees (spans). Every node has to wait for the neighbor to the left. Therefore, we want to change the algorithm in such a way that there are no long paths of descending spans. Allowing for an additional factor 2 in the approximation ratio, we can round all spans to the next power of 2 and let the greedy algorithm take a node with a maximal rounded span. In this case, a path of strictly descending rounded spans has at most length $\log n$. For the distributed version, this means that nodes whose rounded span in maximal

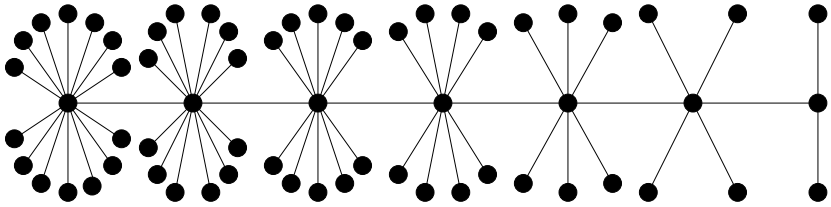

Figure 1: Distributed greedy algorithm: Bad example

within distance 2 are added to the dominating set. Ties are again broken by unique node IDs. If node IDs are chosen at random, the time complexity for the graph of Figure 1 is reduced from $\Omega(\sqrt{n})$ to $\mathrm{O}(\log n)$.

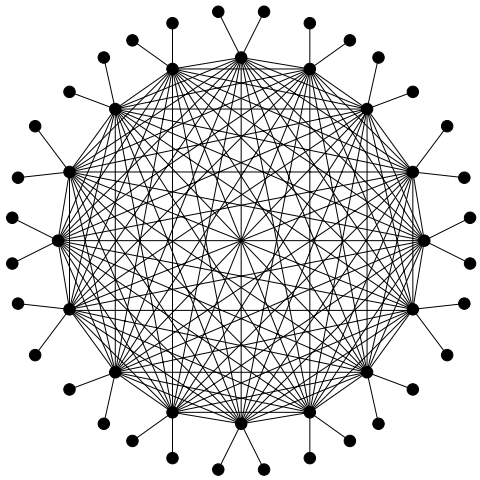

Figure 2: Bad example for distributed greedy with rounded spans

Unfortunately, there still is a problem remaining. To see this, we consider Figure 2. The graph of Figure 2 consists of a clique with $n/3$ nodes and two leaves per node of the clique. In an optimal dominating set, there are all the $n/3$ nodes of the clique. Because they all have distance 1 from each other, the described distributed algorithm only selects one in each while iteration (the one with the largest ID). Note that as soon as one of the nodes is in the dominating set, the span of all remaining nodes of the clique is 2. They do not have common neighbors and therefore there is no reason not to choose all of them in parallel. However, the time complexity of the simple algorithm is $\Omega(n)$. In order to improve this example, we need an algorithm which also chooses nodes based on the number of common white neighbors and not only based on having a large enough ID. This is accomplished by the following probabilistic algorithm which is described in [JRS01].

Fast Distributed Dominating Set Algorithm:

1. Node v is a candidate for joining the dominating set if its span $w(v)$ rounded to the next power of 2 is maximal within distance 2.
2. Each white node u computes its support $s(u)$, which is the number of candidates that cover u.
3. For a candidate v, let $m(v)$ be the median support of all white neighbors. v joins the dominating set with probability $1/m(v)$.

The above three steps are carried out until all nodes are covered. In [JRS01], the following theorem is proven.

Theorem 10.3. *[JRS01] The* fast distributed dominating set algorithm *computes a dominating set of expected size* $\mathrm{O}(\log \Delta)$ *in time* $\mathrm{O}(\log n \log \Delta)$ *w.h.p.*

Remarks:

- Taking the average support instead of the median support in step 3, gives an algorithm with approximation ratio and time complexity $\mathrm{O}(\log n \log \Delta)$.
- Based on different techniques (LP relaxation), there is another fast distributed dominating set algorithm with approximation ratio $\mathrm{O}(\Delta^{1/\sqrt{k}} \log \Delta)$ and time complexity $\mathrm{O}(k)$ [KW03, KMW05]. Choosing $k = \mathrm{O}(\log^2 \Delta)$, the approximation ratio is $\mathrm{O}(\log \Delta)$.
- It is not known whether there is a local (fast) good deterministic approximation algorithm. This is an interesting and important open problem.

10.3 Simple One Round Lower Bound

In the second part of this chapter, we try to give lower bounds for the distributed approximation of dominating sets. To start, we show that in a single communication round, dominating set cannot be approximated better than by a factor $\Omega(\sqrt{\Delta})$ where Δ is the maximum degree of the graph. We assume that initially, all nodes know the identity of all their neighbors. They do not know anything else about the topology of the network graph. In one communication round, each node can inform all adjacent nodes about its neighborhood. Therefore, after one round, each node knows the graph up to distance two. It does not know, however, how nodes at distance two are interconnected to each other.

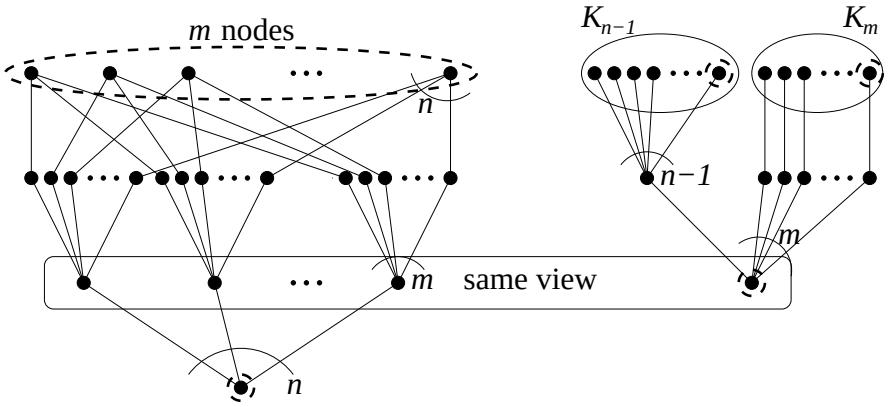

Figure 3: Simple lower bound graphs for one-round algorithms

Theorem 10.4. *In one communication round (as described above), minimum dominating set cannot be approximated better than* $\Omega(\sqrt{\Delta})$.

Proof. To prove this theorem, we have a look at the two graphs of Figure 3. In the left graph, an optimal dominating set consists of the m nodes at the top and the bottom node, that is, it contains $m + 1$ nodes. In the right graph, we have a minimum dominating set consisting of only the 3 nodes

which are marked by a dashed circle. As indicated in the figure, after only one communication round, all the nodes on the third level from the top have the same view. They all have $m+1$ neighbors, m of which have degree 2 and one has degree n. Because all these nodes see the same topology, the probability p that they go to the dominating set is equal for all of them. In the left graph, we have n such nodes, resulting in a dominating set of expected size at least pn. In the right graph, not choosing the bottom node forces us to choose m of the nodes of the top right part in order to get a valid dominating set. Therefore, in expectation, there are at least $(1-p)m$ nodes in the dominating set. For the approximation ratio, we then get

$$\max\left\{\frac{pn}{m+1}, \frac{(1-p)m}{3}\right\} \underset{(m=\sqrt{n})}{=} \max\left\{\frac{pn}{\sqrt{n}+1}, \frac{(1-p)\sqrt{n}}{3}\right\}.$$

Hence, if we set $m=\sqrt{n}$, independent of the choice of p, the approximation ratio is at least $\Omega(\sqrt{n}) = \Omega(\sqrt{\Delta})$. □

10.4 Vertex Cover Lower Bound

Extending the proof of the last section to more communication rounds turns out to be very difficult. Looking at a simpler but related problem, it is however possible. Instead of minimum dominating set we are now considering the minimum vertex cover problem. A vertex cover is a set of nodes S such that for each edge, at least one of the end points is in S. That is, instead of covering nodes with nodes, we cover edges with nodes.

The basic idea is to construct a graph $G_k = (V, E)$, for each positive integer k, which contains a bipartite subgraph S with node set $C_0 \cup C_1$ and edges in $C_0 \times C_1$ as shown in Figure 4. Set C_0 consists of n_0 nodes each of which has δ_0 neighbors in C_1. Each of the $n_0 \cdot \frac{\delta_0}{\delta_1}$ nodes in C_1 has δ_1, $\delta_1 > \delta_0$, neighbors in C_0. The goal is to construct G_k in such a way that all nodes in $v \in S$ see the same topology within distance k. In a globally optimal solution, all edges of S may be covered by nodes in C_1 and hence, no node in C_0 needs to join the vertex cover. In a local algorithm, however, the decision of whether or not a node joins the vertex cover depends only on its local view. We show that because adjacent nodes in S see the same topology, every algorithm adds a large portion of nodes in C_0 to its vertex cover in order to end up with a valid solution.

10.4.1 The Cluster Tree

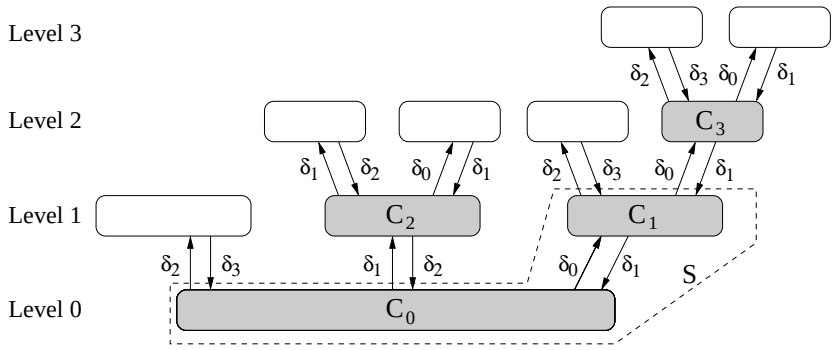

Figure 4: Cluster-Tree CT_2.

The nodes of graph $G_k = (V, E)$ can be grouped into disjoint sets which are linked to each other as bipartite graphs. We call these disjoint sets of nodes *clusters*.

We define the structure of G_k using a directed tree $CT_k = (\mathcal{C}, \mathcal{A})$ with doubly labelled arcs $\ell : \mathcal{A} \to \mathbb{N} \times \mathbb{N}$. We refer to CT_k as the *cluster tree*, because each vertex $C \in \mathcal{C}$ represents a cluster of nodes in G_k. The *size* of a cluster $|C|$ is the number of nodes the cluster contains. An arc $a = (C, D) \in \mathcal{A}$ with $\ell(a) = (\delta_C, \delta_D)$ denotes that the clusters C and D are linked as a bipartite graph, such that each node $u \in C$ has δ_C neighbors in D and each node $v \in D$ has δ_D neighbors in C. It follows that $|C| \cdot \delta_C = |D| \cdot \delta_D$. We call a cluster *leaf-cluster* if it is adjacent to only one other cluster, and we call it *inner-cluster* otherwise.

Definition 10.5. *The cluster tree CT_k is recursively defined as follows:*

$$\begin{aligned} CT_1 &:= (\mathcal{C}_1, \mathcal{A}_1), \quad \mathcal{C}_1 := \{C_0, C_1, C_2, C_3\} \\ \mathcal{A}_1 &:= \{(C_0, C_1), (C_0, C_2), (C_1, C_3)\} \\ \ell(C_0, C_1) &:= (\delta_0, \delta_1), \quad \ell(C_0, C_2) := (\delta_1, \delta_2), \\ \ell(C_1, C_3) &:= (\delta_0, \delta_1) \end{aligned}$$

Given CT_{k-1}, we obtain CT_k in two steps:

- *For each inner-cluster C_i, add a new leaf-cluster C_i' with $\ell(C_i, C_i') := (\delta_k, \delta_{k+1})$.*
- *For each leaf-cluster C_i of CT_{k-1} with $(C_{i'}, C_i) \in \mathcal{A}$ and $\ell(C_{i'}, C_i) = (\delta_p, \delta_{p+1})$, add $k-1$ new leaf-clusters C_j' with $\ell(C_i, C_j') := (\delta_j, \delta_{j+1})$ for $j = 0 \dots k, j \neq p+1$.*

Further, we define $|C_0| = n_0$ for all CT_k.

Figure 4 shows CT_2. The shaded subgraph corresponds to CT_1. The labels of each arc $a \in \mathcal{A}$ are of the form $\ell(a) = (\delta_l, \delta_{l+1})$ for some $l \in \{0, \dots, k\}$. Further, setting $|C_0| = n_0$ uniquely determines the size of all other clusters. Note that CT_k describes the general structure of G_k, that is, it defines for each node the number of neighbors in each cluster. However, CT_k does not specify the actual adjacencies. In [KMW04], it is shown that G_k can be constructed in such a way that each node's view is a tree up to distance k, that is, there are no short cycles. Choosing $\delta_i = \delta^i$, we obtain a realization of G_k with $n_0 \leq 4^{2k}\delta^{4k^2}$.

10.4.2 Proving the Lower Bound

Having constructed G_k, it remains to prove that every k-round vertex cover algorithm behaves bad on G_k. It is possible to prove that the nodes in C_0 and the nodes in C_1 see the same topology up to distance k. Therefore, all nodes of C_0 and C_1 have to join the vertex cover with the same probability p. Because the edges connecting the C_0 with C_1 have to be covered, $p \geq 1/2$. Otherwise there is a non-zero probability that two adjacent nodes are both not in the vertex cover. Therefore, in expectation, each algorithm chooses at least half of the nodes of C_0. An optimal vertex cover consists at most $|V \setminus C_0| = n - n_0$ nodes. One can show that $n - n_0 \leq \frac{n_0(k+1)}{\delta-(k+1)}$. Therefore if we choose $\delta \geq 2(k+1)$, the approximation ratio α is at least

$$\alpha \geq \frac{n_0/2}{n - n_0} \geq \frac{n_0/2 \cdot \delta/2}{n_0 \cdot (k+1)} = \frac{\delta}{4(k+1)} \geq \frac{(n/2)^{1/(4k^2)}}{4^{1+1/(2k)}(k+1)} \in \Omega\left(\frac{n^{1/(4k^2)}}{k}\right).$$

Hence, we obtain the following theorem (note that $\Delta = \delta^{k+1}$).

Theorem 10.6. *In k communication rounds, every distributed algorithm for the minimum vertex cover problem has approximation ratio at least*

$$\Omega\left(\frac{n^{c/k^2}}{k}\right) \text{ and } \Omega\left(\frac{\Delta^{1/k}}{k}\right)$$

for some constant $c \geq 1/4$, where n and Δ denote the number of nodes and the maximum degree of the network graph.

Remarks:

- It can be shown that unique node IDs do not help, that is, the lower bounds also hold in this case.
- As a consequence of Theorem 10.6, at least $\Omega(\sqrt{\log n / \log\log n})$ and $\Omega(\log\Delta / \log\log\Delta)$ rounds are required for a constant or polylogarithmic approximation.
- By simple reductions, it can be shown that the lower bounds also hold for the minimum dominating set problem and for the construction of maximal matchings and maximal independent sets.

References

[JRS01] L. Jia, R. Rajaraman, and R. Suel. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. In *Proc. of the 20^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, pages 33–42, 2001.

[KMW04] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In *Proc. of the $23rd$ ACM Symp. on Principles on Distributed Computing (PODC)*, 2004.

[KMW05] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The Price of Being Near-Sighted, 2005. submitted.

[KW03] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22^{nd} Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.