Approximation Algorithms: Connected Dominating Set

Haricharan & Abhinav

October 23, 2023

Outline

We'll learn all these!

- Exploring the problem and the theoretical lower bound on α
- Goal of the Session
- Colours of Nodes
- The Algorithm
- Proving that we get a *connected* set
- Proving that we get a *dominating* set
- Terminology: singly- and doubly-counted vertices
- Defining costs of nodes
- Proving the approximation guarantee!

Connected Dominating Set

Exploring the Problem

Consider a graph $G(E, V)$. A connected dominating set S is a set of vertices that has two properties:

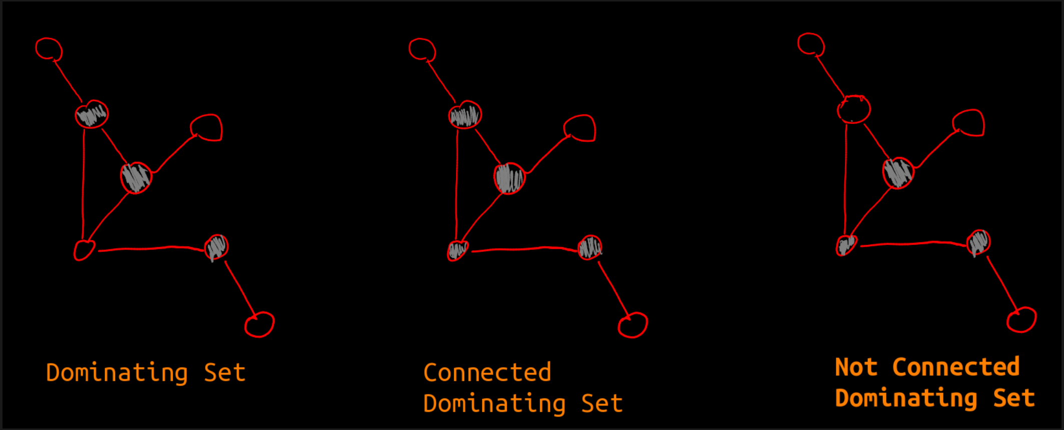
- The induced subgraph of S in G is connected
- Every node in V is either in S or adjacent to a node in S

Our goal is to discover the *minimum-cardinality* connected dominating set. Solving this problem is NP-Hard, so we try to come up with an approximation algorithm for it.

Theoretical Lower Bound on α

We can't come up with an approximation algorithm that gives us $\alpha < \log n$.
If we get an approximation factor $\boxed{\alpha = H(2\Delta) + O(1)}$, where Δ is the maximum-degree, we're doing very well!

Examples of CDS



Local Information in Graphs

What is it and why even use it?

- Local information - each step of the algorithm doesn't require information of the full graph to make a decision.
- In the CDS algorithm, we are only concerned with vertices at most 2 edges away from any vertex we picked up in our dominating set so far.
- Useful in settings where the entire graph isn't revealed to us at once and we have to load portions of it at a time - ex: a social media graph.

Historical Note

1. Guha and Khuller (1998): two algorithms
   1. Centralized Greedy Approach: $H(\Delta) + 2$
   2. Local Information Greedy Approach: $2H(\Delta) + 2$
2. There is a penalty factor of "two" above, when we limit ourselves only to local information.
3. Natural question: Can we get a $H(\Delta)$ bound using only local information?
4. This paper, *Revisiting Connected Dominating Sets: An Almost Optimal Local Information Algorithm* (2019)[1], answers it in the affirmative

[1]Khuller is one of the authors of this paper as well, returning to the problem with a better result after 20 years!

An overview of the algorithm

- At each step of the algorithm, we maintain a set S of vertices that are picked to be in the dominating set.
- When choosing the next vertex to be picked up, we consider candidates only from the two-hop neighborhood of S (denoted as $N^2(S)$) - the set of vertices v such that there exists a path of length at most 2 between v and some vertex in S. (Note that S is included in its two-hop neighborhood!)
- S need not be connected during intermediate stages of the algorithm. However, the final output will be a connected set (this will be proved later)
- Analysis will be done similar to Set Cover - Each time we add a vertex to S, we distribute a unit cost among vertices that it affects.

Colours of Nodes

Why colour nodes suddenly? You'll see!

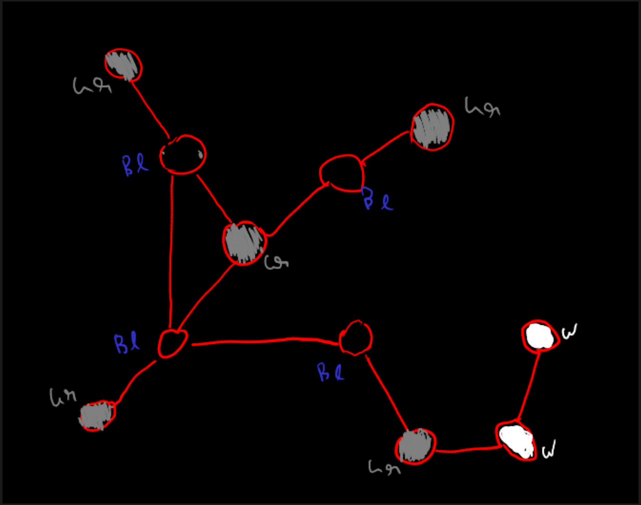
At each stage in the algorithm, every node can be:

- Black: We've picked the node in our solution set, i.e. $u \in S$
- Gray: This node is adjacent to a node that is black
- White: Every other node in our graph

Note that when we output our solution, we should have only black and gray nodes, and *no white nodes* (else we wouldn't have a dominating set)!

Also observe that every node in S is black, every node in $N^1(S) \setminus S$ is gray, and every node in $N^2(S) \setminus N^1(S)$ is white.

Example of Colours of Nodes



The Algorithm

2-Hop Local Information

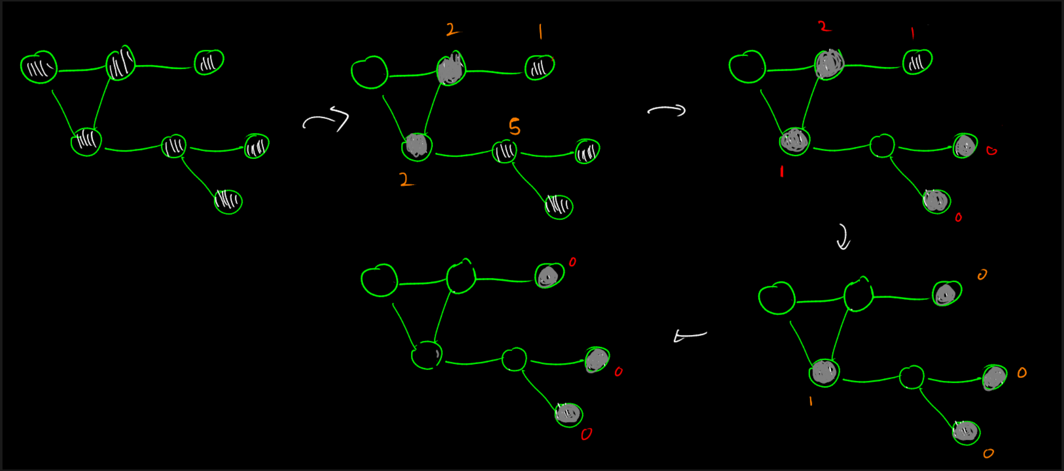
w_v: Number of white nodes in $N^1(v)$ (this includes v itself)[2]

c_v: Number of black connected components adjacent to v

- Initialise S to contain only some arbitrary vertex s. Set $\bar{V}$ to $N^2(s)$
- Find vertex v in $\bar{V}$ with the maximum value of $2w_v + c_v - 1$, breaking ties arbitrarily
- Add v to S, and update $\bar{V}$ to be $N^2(S)$
- Update c_v, w_v values for all vertices in $\bar{V}$
- Repeat from step 2 until all vertices in $\bar{V}$ have $2w_v + c_v - 1 \leq 0$, then terminate.

[2]This includes white nodes that are not in $N^2(S)$, so technically we have partial information of $N^3(S)$

Algorithm Running

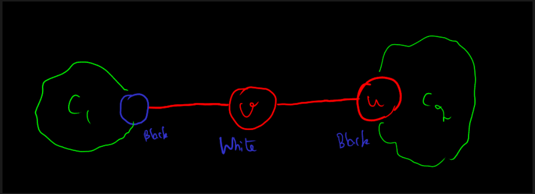


Proof that we get a connected set

Proof by Contradiction: Assume that when our algorithm terminates the set is not connected. Consider there are two components, C_1, and C_2.

When the algorithm terminates, there is at least one node in C_2 that is at a distance of 2 from C_1 (*why?*[3]). Let this node be u. Let the node between u and C_1 be v.

For the node v, $c_v \geq 2, \implies 2w_v + c_v - 1 > 0$, so our algorithm would not have terminated. This is clearly a *contradiction*.



[3]We start with a black connected component, and only pick up vertices (turning them black!) in the 2-hop neighborhood.

Proof that we get a dominating set

Proof by Contradiction: Assume that the set is not dominating.

$\implies \exists$ Some white node $v \in N^2(S)$.

For this node, $v \in N^1(v), w_v \geq 1, \implies 2w_v + c_v - 1 > 0$, so our algorithm would not have terminated.

This is a *contradiction*, so our assumption is wrong!

Proof Outline

Interesting

- Define a quantity called *cost*
- Each time we pick a vertex, we'll share one unit of cost among its neighbours
- We'll then bound the total cost each vertex holds at the end of the algorithm
- Finally, we'll use the fact that $\boxed{\text{Sum of Costs of Vertices} = \text{Number of Vertices Picked}}$, so if we've bounded the costs, we've bounded the number of vertices also

Terminology

We need this for analysis!

We call each vertex in the final solution either singly- or doubly-counted[4], depending on how many times the vertex receives shares in the solutions. We share costs such that:

- All white nodes are not counted
- All gray nodes are doubly-counted
- Black nodes can be singly- or doubly-counted. But, each black component has **exactly one** singly-counted black node.

Note

When our algorithm is done, we'll have only one component, so every node will be double-counted except for one node!

[4]The paper calls this singly- or doubly-charged, which can be confusing since they call the cost charge as well

How to assign cost?

This would help us in the bound

Every time we add a vertex to the solution set, we pay a cost[5] of 1 unit. This cost is then distributed among the neighbors of the vertex as such:

When we pick up vertex v, define one **share** as a cost of $\frac{1}{2w_v+c_v-1}$.

- **White Node:** $c_v = 0$, so we have $2w_v - 1$ shares in total. Distribute 1 share to v itself and 2 shares to all the remaining $w_v - 1$ white nodes adjacent to it.
- **Grey Node:** For each of the w_v white nodes around v, distribute 2 shares. For each black components around this node, except one (which can be chosen arbitrarily), distribute 1 share to each of them. This one share is given to the singly counted vertex in that component, thus ensuring that the final connected black component has only one singly counted vertex.

[5] The paper calls this charge

Example Cost Sharing

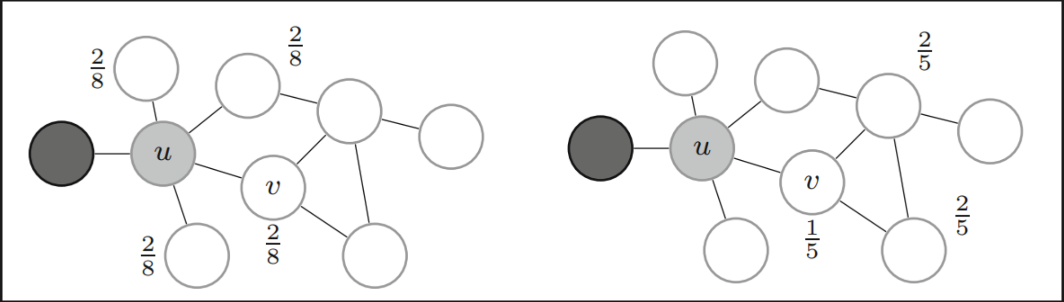
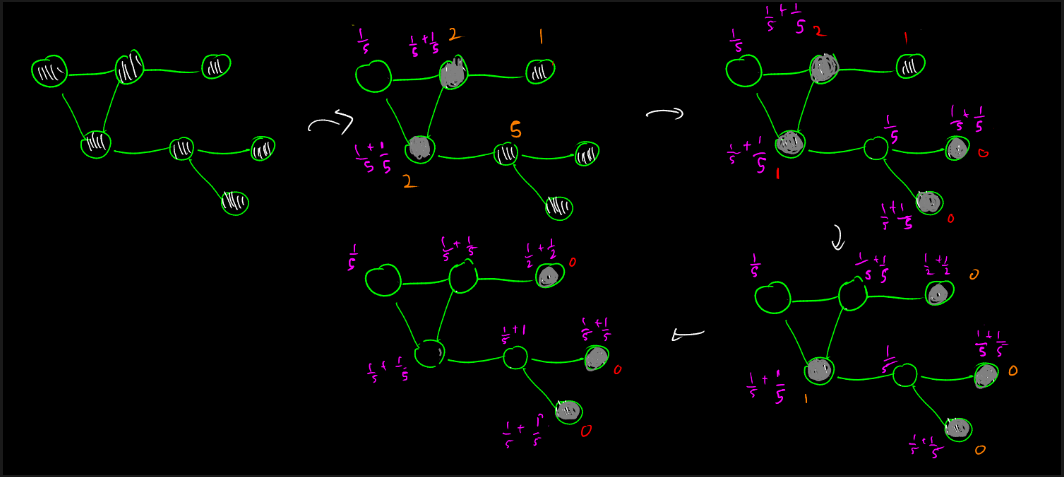


Figure: Charging *u* and *v* respectively

Cost Sharing Example



Some Definitions and Observations

- The cost of any vertex at any stage is the sum of shares it has received in all steps so far. (Note that shares received in different steps may have different values)
- The cost of any set of vertices (which in our case will be a *partition*) at any stage is the sum of shares of all the vertices in it.
- Note: The sum of costs over all vertices at any stage is the number of vertices we picked up.

Proof of the Bound

Partitioning V

Consider a partition of V into $S_1, S_2, \ldots, S_{|OPT|}$, where S_i contains a vertex v_i and its immediate neighbours. Assume that we've picked the vertex v_i in OPT. Break ties arbitrarily.

Also define, in Step j:

1. w_i^j: Number of white nodes in S_i
2. b_i^j: Number of *singly-counted* black nodes in S_i
3. c_i^j: Number of black components connected to v_i
4. $p_i^j = 2w_i^j + b_i^j - 1$

$$\boxed{b_i^j \leq c_i^j \forall i, j}$$

Some bounds

We will use these in our proof!

Observe that p_i^j can be thought of as the potential amount of shares we can give to vertices in S_i. Assume the cost in S_i changes in step j. Then, $p_i^j - p_i^{j+1} \geq 1$.

If the countedness of a node did change, we have three possibilities:

- White to Black: changes by one unit or two units
- White to Gray: changes by two units
- Singly-counted black to doubly-counted black: changes by one unit

More Bounds:

We will use these in our proof!

Consider all the steps j before vertex v_i is selected. Let some other vertex v be picked in such a step. Now, we know that:

$$\boxed{2w_v^j + c_v^j - 1 \geq 2w_i^j + c_i^j - 1 \geq 2w_i^j + b_i^j - 1 = p_i^j}$$

Total change in cost in S_i in the jth step is: $\frac{p_i^j - p_i^{j+1}}{2w_v^j + c_v^j - 1}$

Now, as long as p_i^j is positive, we can say that:

$$\frac{p_i^j - p_i^{j+1}}{2w_v^j + c_v^j - 1} \leq \frac{p_i^j - p_i^{j+1}}{p_i^j}$$

Starting the proof

But of course it is not always positive, it might be negative also. There is some step k_i such that $p_i^{k_i}$ is positive but $p_i^{k_i+1}$ is negative.

Total cost in the first k_i steps in partition i (defined as TC) is:

$$\boxed{TC = 1 + \sum_{j=2}^{k_i} \frac{p_i^j - p_i^{j+1}}{2w_v^j + c_v^j - 1}}$$

$$\leq 1 + \sum_{j=2}^{k_i} \frac{p_i^j - p_i^{j+1}}{p_i^j} = 1 + \sum_{j=2}^{k_i} \sum_{p_i^{j+1}+1}^{p_i^j} \frac{1}{p_i^j}$$

Continuing the proof

$$TC \leq 1 + \sum_{j=2}^{k_i} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{p_i^j} = 1 + \sum_{j=2}^{k_i-1} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{p_i^j} + \sum_{t=p_i^{k_i+1}+1}^{p_i^{k_i}} \frac{1}{p_i^{k_i}}$$

(Splitting the summation)

Define $l = \max\{p_i^{k_i+1} + 1, 1\}$

$$\leq 1 + \sum_{j=2}^{k_i-1} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{p_i^j} + \sum_{t=p_i^{k_i+1}+1}^{l-1} \frac{1}{p_i^{k_i}} + \sum_{t=l}^{p_i^{k_i}} \frac{1}{p_i^{k_i}}$$

$$\leq 1 + \sum_{j=2}^{k_i-1} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{t} + \sum_{t=p_i^{k_i+1}+1}^{l-1} 1 + \sum_{t=l}^{p_i^{k_i}} \frac{1}{t}$$

(Using $p_i^{j+1} \leq p_i^j$)

Continuing the proof

$$TC \leq 1 + \sum_{j=2}^{k_i-1} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{t} + \sum_{t=p_i^{k_i+1}+1}^{l-1} 1 + \sum_{t=l}^{p_i^{k_i}} \frac{1}{t}$$

$$\leq 1 + \sum_{j=2}^{k_i-1} \sum_{t=p_i^{j+1}+1}^{p_i^j} \frac{1}{t} + \sum_{t=l}^{p_i^{k_i}} \frac{1}{t} + \sum_{t=p_i^{k_i+1}+1}^{l-1} 1$$

$$= 1 + \sum_{j=1}^{p_i^1} \frac{1}{t} + \sum_{t=p_i^{k_i+1}+1}^{l-1} 1$$

$$= 1 + H(p^1) + ((l-1) - (p_i^{k_i+1} + 1) + 1) = \boxed{1 + H(p_i^1) - (p_i^{k_i+1})}$$

The last term

We forgot the last term

Since $p_i^{k_i+1} \leq 0$, we can have at most one more step.

Change in total cost in $(k+1)$th step is upper-bounded by:

$$(p_i^{k_i+1} - p_i^{k_i+2}) \cdot 1 \leq p_i^{k_i+1} + 1$$

Adding the previous equation and this equation:

$$\text{Total cost in partition } i \text{ over all iterations} \leq \boxed{H(p_i^1) + 2}$$

Note that $p_i^1 = 2w_i^1 + b_i^1 - 1 \leq 2w_i^1 + 1 \leq 2\Delta + 1$

Final Bound

We know that there are $|OPT|$ many partitions.

$$\text{Total cost in all partitions over all iterations} = |OPT|(H(2\Delta + 1) + 2)$$

But, we know that Total cost in all partitions over all iterations $=$ Number of vertices picked by our algorithm

This gives us a $\alpha = \boxed{(H(2\Delta + 1) + 2)}$ approximation!

Some footnotes:

- The above algorithm is a 2-Hop algorithm, because we look at all the nodes in $N^2(S)$. If we look at nodes only in $N^1(S)$, it would be a 1-Hop algorithm, which gives us a poorer bound.
- The algorithm is a local-information algorithm. Why is this good? If we don't know about the entire graph, like say a Social Media Connection Graph, this is ideal.
- It is possible to get a $H(\Delta) + 2\sqrt{H(\Delta)} + 1$ (expected) approximation factor using a randomized algorithm, suggested in the same paper.