DAPO

2020/2021

1st Fase

Min. Dominating Set

Alexandru Caramida nº 45604

Gonçalo Mateus nº 51927

Min. Dominating Set

Given an undirected graph G=(V, E) we want to compute a minimum size dominating set, a subset S ⊆ V of its vertices such that for all vertices v ∈ V, either v ∈ S or a neighbor u of v is in S.
Computing a dominating set of minimal size is NP-hard.

Greedy Algorithm

Initially, the dominant set D is empty and with each iteration of the algorithm, vertex v ∈ V is added to D until D becomes a dominant set. The vertex selected to belong to D is chosen on the condition that it covers the maximum number of vertices not covered in the previous iteration. In case of tie the vertex to be added is chosen at random among them.

Algorithm: greedy(G)
Input: an undirected graph G
Output: size of the dominating set D

D = ∅
for every $v_i \in G$
    $weight_i$ = 1 + d(v_i)
    $covered_i$ = false

do
    v = chooseVertex (weight)
    if v != −1
        add v to D
        adjustWeights (G, weight, covered, v)
until v = −1

return D.size

Method: chooseVertex (weight)
Input: the weight vector
Output: a vertex of G which covers the maximum number of vertices not yet covered

M = max $weight_{i, 1 \le i \le n}$
if M = 0
    return −1
else
    S = $\{v_i | weight_i = M\}$
    randomly return an element of S

Method: adjustWeights(G, weight, covered, v_i)
Input: the graph G, the weight vector, the covered vector, the index of the v_i

$weight_i$ = 0
for every v_j neighbour of v_i that $weight_j > 0$
    if !$covered_i$
        $weight_j$- -
    if !$covered_j$
        $covered_j$ = true
        $weight_j$- -
        for every v_k neighbor of vj
            if $weight_k > 0$
                $weight_k$ - -
$covered_i$ = true

For a graph G with v vertices and e edges, a call to choose a vertex is at most ***O***(v). The total time spent in adjustWeights is ***O***(e). If d is the size of the dominant set found, the total complexity of the algorithm is in ***O***$(v*d + e) \le$ ***O***(v^2).

Time complexity: ***O***(v^2) [(1)](#)
Space complexity: ***O***(v)

Approximation ratio: ***O***(log(Δ)), where Δ is the maximum degree of a vertex [(2)](#)

Linear Programming Algorithm

The linear programming algorithm consists of solving the following linear optimization problem:

$Min\ \ F\ = \sum xi$

$xi\ \ + \sum variables\ of\ adjacent\ vertices\ \ >=$ 1for all xi

xi >= 0 for all xi

We then count the number of variables with a value that exceeds 1/ (d+1), where d stands for the biggest degree a vertice has in the graph. The result is the size of an approximation of the minimum set cover.
The approximation ratio is d+1.

Test Instances

Social Network Samples
This is a set of anonymised social network samples from
https://davidchalupa.github.io/research/data/social.html
pokec_500.col
pokec_2000.col
pokec_10000.col
pokec_20000.col
pokec_50000.col
gplus_500.col
gplus_2000.col
gplus_10000.col
gplus_20000.col
gplus_50000.col

This is a set of Graph Coloring Instances from
https://mat.gsia.cmu.edu/COLOR/instances.html
anna.col
homer.col
david.col
huck.col

Both test instances can also be found in reference (2) page. 18-19

Results

Results sampled from 20 runs per test.
All times are in seconds.

Run time

Graph	Greedy		Linear Programming	
Samples Pokec	avg	standard deviation	avg	standard deviation
pokec 500	0.00212	0.00506	0.03330	0.04264
pokec 2000	0.00347	0.00062	0.05549	0.02067
pokec 10000	0.04762	0.00364	44.31408	0.12705
pokec 20000	0.20411	0.00277		
pokec 50000	1.65080	0.02688		
Samples Google+				
gplus 500	0.00033	-		
gplus 2000	0.00424	0.00004		
gplus 10000	0.09503	0.00063		
gplus 20000	0.37109	0.00259		
gplus 50000	2.82216	0.02871		
DIMACS Graphs				
anna	0.00006	-		
homer	0.00076	0.00001		
david	0.00002	-		
huck	0.00003	-		

Dominating Set

Graph	Greedy			Linear Programming
Samples Pokec	min	max	avg	
pokec 500, γ=16	16	16	16	16
pokec 2000, γ=75	75	75	75	75
pokec 10000, γ=413	413	414	413.05	413
pokec 20000, γ=921	924	928	925.80	
pokec 50000, γ≥2706	2767	2783	2776.25	
Samples Google+				
gplus 500, γ=42	42	42	42	
gplus 2000, γ=170	176	179	177.35	
gplus 10000, γ=861	893	899	895.85	
gplus 20000, γ≥1716	1808	1821	1813.80	
gplus 50000, γ≥4566	4830	4861	4844.90	
DIMACS Graphs				
anna, γ=12	12	12	12	
homer, γ=96	91	92	91.30	
david, γ=2	2	2	2	
huck, γ=9	9	9	9	

Conclusions

We proposed two algorithms for approximating the minimum dominating set problem, a Greedy algorithm and the Linear Programming algorithm.
In the end we had problems with the Linear Programming algorithm and couldn't make all the tests.
The evaluation was carried out with 20 runs per test.
From the few tests we managed to perform for both algorithms in terms of time the greedy algorithm is clearly much faster. About the MDS we can't conclude anything because 3 tests are not enough for conclusions.
In the end we couldn't finish the 1 phase of the assignment as intended.

References

1. Algorithm and time complexity
   Bouali Zakariae - Comparaison d'algorithmes pour le problème d'ensemble dominant minimum dans un graphe, thesis.pdf *page. 58-62*
   https://github.com/JavaZakariae/MinDominatingSet
2. Approximation ratio
   David Chalupa - An Order-based Algorithm for MinimumDominating Set with Application in GraphMining, *page. 4-5*.
   https://arxiv.org/pdf/1705.00318.pdf