2nd Project - Randomized Algorithm for the Minimum Weight Dominating Set

Nuno cunha - 98124

***Abstract* –A minimum weight dominating set in a undirected graph is set of vertices such that every vertex of the graph either belongs to it, or is adjacent to one vertex of this set and where the sum of the weights of the vertices is the small as possible. It's proven to be an NP-Hard Problem (Nondeterministic Polynomial Time Hard Problem) which means that it is solvable in polynomial time by a non-deterministic Turing machine (in this case a modern computer).**

In this paper, its presented an random algorithm to calculate a minimum weight dominating set, this algorithm not always returns the best result, but it gives a close answer to the optimal solution in a much less time than the exhaustive search done in the 1st project.

This random algorithm is a balance between quality of the answer and the time taken to give that answer. In this solution the time taken depends of the density of the graph.

I. Introduction

The goal of this project is to find a minimum weight dominating set for a given undirected graph G(V, E), whose vertices carry positive weights, with n vertices and m edges. An dominating set of G is a subset of vertices such that every vertex of the graph either belongs to it, or is adjacent to one vertex of this subset and the weight of an dominating set is the sum of its vertices' weights. A minimum weight dominating set is an dominating set whose total weight is as small as possible.

Dominating sets are extensively used in wireless networks for clustering and formation of a routing backbone, so the solution to this problem can be used in these situations to help getting a good solution in the shortest time possible.

II. Method

The approach used is to randomly generate sets until we found a approximated answer. In the beginning we will try to make subsets of vertices with half the size of the graph, if we find a subset that is a dominating set we will decrease the size of the subsets created until we cant find a dominating set within an given number of tries, or the number of vertices is 1. If in the beginning we cant find a subset with half the vertices of the graph that is a dominating set,within an given number of tries, we will create subsets with the same size of the graph. For every iteration we skip a subset if was already analysed. When we can't find a dominating set for a given size of a subset it means that the right solution is probably of that size, so we will test more random subsets of that size until we have tested a certain number of subsets(we will never analyse the same subset). In the end we will find the dominating set with lowest weight in the analysed subsets.

III. Formal computational complexity analysis

For the formal computational complexity analysis we will only analyse the best and worst case scenarios because its very difficult to analyse an average case since the algorithm is random. In this random algorithm we don't analyse all the possible combinations of vertices of the given graph $(2^n - 1)$, since it would be very time consuming, instead we take a smarter approach, first we try to find the correct size of the answer and then we try to analyse the most possible solutions for that size and not taking to much time.

The best case is when the density is 100% because we will find a dominating set in the first iteration and reduce the size of the solution until the size gets to one. When we get to one we will test several subsets for this size to find the optimal solution.

$$num\_comp(n) = \sum_{i=1}^{\frac{n}{2}} \left(\sum_{j=1}^{i} \left(\sum_{k=1}^{n} 1 \right) + 7 \right) +$$

$$\sum_{i=1}^{10000} \left(\sum_{k=1}^{n} 1 + 4 \right) =$$

$$\frac{n^3}{8} + \frac{n^2}{4} + \frac{20007}{2} + 40000$$

Considering that the graph has n vertices, in the best case scenario the overall complexity will be cubic, O(n^3).

The worst case happens when the density of the graph is 0%, because no vertex will be adjacent to another vertex. We will iterate the while loop only 3 times, but the fors loops inside the while loop will iterate many times. The first iteration we will try to find a dominating set with half the number of vertices of the graph, but because the density is 0% we can't find any. On the second iteration we will try to find a dominating set with the same the number of vertices as the graph, and we will find it on the first subset generated. Because we found a dominating set we will try to decrease the length of the solution, but it is impossible to find a dominating set at this length. So now we have concluded that the size of the answer is equal to the number of vertices of the graph, then we will generate some more subsets and in the end show the one with the minimum weight.

$$num\_comp(n) = \sum_{i=1}^{n}\left(\sum_{j=1}^{10000}\left(\sum_{k=1}^{i}\left(\sum_{l=1}^{n}1\right)+3\right)+2\right)$$

$$+\sum_{i=1}^{10000}\left(\sum_{j=1}^{\frac{n}{2}}\left(\sum_{k=1}^{n}1\right)+3\right)+1+\sum_{i=1}^{10000}\left(\sum_{k=1}^{n}1\right)+4=$$

$$5000(n^3+8n^2+17n+2)$$

In this case the overall complexity will be cubic O(n^3)

IV. Experimental results

The experimental results can be found in the "RandomSearch.xlsx" file. These results contains the number of vertices,edges,density(0%,12,5%,25%,50%,75% and 100%), number of comparisons, divisions, additions, solutions (subsets found), time taken, the answer and it's weight. In the file there is also some graphics to visualize the information.

The number of comparisons in the random algorithm grows exponentially, in the worst and best case.

By looking at the tables we mus have into account that the algorithm changes with the size of the graph. When the number of vertices is lower than 7 we will always make at least 75 subsets, and if number of vertices is higher than 13 in which we have an cap of 10 000 subsets. Because of this we see some fluctuation between these number of vertices, this fact can be observed in the worst case (TABLE II).

For the best case we can conclude that its order of complexity is between quadratic and cubic, O(n^2) and O(n^3), because the ratio between num_comp(2n) and num_comp(n) tends to the value of 5. Therefore, proving that the order of complexity of the random algorithm is between quadratic and cubic.

vertices	comp(n)	comp(2n)	comp(2n) /comp(n)
1	20011	20032	1,001049423
2	20032	20188	1,00778754
3	20087	20588	1,024941504
4	20188	21352	1,057658015
5	20350	22600	1,110565111
6	20588	24452	1,187682145
7	20917	27028	1,292154707
8	21352	28792	1,348445111
9	21908	32501	1,483522001
10	22600	37130	1,642920354
11	23443	42781	1,824894425
12	24452	49556	2,026664486
13	25642	57557	2,244637704
14	27028	66886	2,474692911
15	28625	77645	2,712489083
16	28792	89936	3,123645457
17	30460	103861	3,409750492
18	32501	119522	3,677486847
19	34598	136943	3,958118966
20	37130	156460	4,213843253
21	39705	177511	4,470746757
22	42781	200486	4,686332718
23	45883	226403	4,934354772
24	49556	252520	5,095649366
25	53234	281735	5,292388323
26	57557	314246	5,459735566
27	61860	346071	5,59442289
28	66886	381056	5,697096552
29	71863	415009	5,775002435
30	77645	455712	5,869173804
31	83345	492095	5,904313396
32	89936	535536	5,954634407
33	96408	579187	6,007665339
34	103861	619184	5,961660296
35	111154	662023	5,95590802
36	119522	712648	5,962483894
37	127533	755981	5,927728509
38	136943	807312	5,895241086
39	145943	846623	5,801052466
40	156460	904142	5,778742171
41	166174	947947	5,704544634
42	177511	1003096	5,650894874
43	188392	1053263	5,590805342
44	200486	1099956	5,486447932
45	213147	1159237	5,438673779
46	226403	1206944	5,330954095
47	238477	1262941	5,295860817
48	252520	1312946	5,199374307
49	266568	1368587	5,134100867
50	281735	1423076	5,05111541

TABLE I: Ratio of comparisons for the random search in the best case

In relation to the worst case, the ratio tends to the value of 8, proving that the order of complexity of the algorithm in the worst case is cubic, O(n^3). The fluctuation of the values is already explained above.

vertices	comp(n)	comp(2n)	comp(2n) /comp(n)
1	20011	20110	1,004947279
2	20110	60842	3,02545997
3	40209	129786	3,227784824
4	60842	2336822	38,4080405
5	84605	9256173	109,4045624
6	129786	16334202	125,8548842
7	380475	26303237	69,13262895
8	2336822	41451092	17,73823252
9	6122876	60500338	9,881032704
10	9256173	82620414	8,925979884
11	12507809	109580498	8,760966689
12	16334202	141860590	8,684880351
13	20890059	179940690	8,613699463
14	26303237	224300798	8,527497889
15	32758024	275420914	8,407738941
16	41451092	333781038	8,052406388
17	51020165	399861170	7,837316285
18	60500338	474141310	7,837002663
19	70890375	557101458	7,858633249
20	82620414	649221614	7,857883815
21	95360455	750981778	7,875190801
22	109580498	862861950	7,874229135
23	124910543	985342130	7,888382408
24	141860590	1118902318	7,887337265
25	160020639	1264022514	7,899121775
26	179940690	1421182718	7,898061956
27	201170743	1590862930	7,908023335
28	224300798	1773543150	7,906985467
29	248840855	1969703378	7,915514428
30	275420914	2179823614	7,914517392
31	303510975	2404383858	7,921900874
32	333781038	2643864110	7,920953586
33	365661103	2898744370	7,927406952
34	399861170	3169504638	7,926512689
...	...	...	...
40	649221614	5154426414	7,93939435
41	698661695	5549786738	7,943453574
42	750981778	5964867070	7,942758726
43	805411863	6400147410	7,946428038
44	862861950	6856107758	7,945775982
45	922522039	7333228114	7,94910886
46	985342130	7831988478	7,948496506
47	1050472223	8352868850	7,951537096
48	1118902318	8896349230	7,95096148
49	1189742415	9462909618	7,953746541
50	1264022514	10053030014	7,95320487

TABLE II: Ratio of comparisons for the random search in the worst case

V. Accuracy of the obtained solutions

To evaluate the quality of the answers we decided to do the relative error using the values of the previous work.

For 12.5% density the results where very good the relative error is 0% at its lowest and 24% at its highest.

random	exhaustive	relative error (%)
58	58	0
74	74	0
99	99	0
131	131	0
127	127	0
159	159	0
116	116	0
215	215	0
215	199	8,040201005
221	221	0
219	219	0
221	221	0
262	262	0
224	224	0
185	185	0
225	225	0
275	266	3,383458647
241	241	0
342	276	23,91304348
343	343	0
214	206	3,883495146

TABLE III: Relative error at 12.5% density

For 25% density the results where also very good the relative error is 0% at its lowest and 22% at its highest.

random	exhaustive	relative error (%)
58	58	0
74	74	0
99	99	0
106	106	0
118	118	0
140	140	0
112	98	14,28571429
130	130	0
120	111	8,108108108
138	123	12,19512195
136	136	0
145	119	21,8487395
172	172	0
159	159	0
138	138	0
130	115	13,04347826
160	160	0
195	172	13,37209302
163	163	0
198	190	4,210526316
129	129	0

TABLE IV: Relative error at 25% density

For 50% density the results where not so good, the relative error is 0% at its lowest and 184% at its highest.

random	exhaustive	relative error (%)
58	58	0
74	74	0
74	74	0
74	74	0
41	41	0
82	69	18,84057971
70	56	25
43	43	0
41	41	0
41	41	0
69	49	40,81632653
40	40	0
84	49	71,42857143
40	40	0
117	71	64,78873239
122	43	183,7209302
143	103	38,83495146
40	40	0
49	49	0
76	58	31,03448276
28	28	0

TABLE V: Relative error at 50% density

For 75% density the results where good the relative error is 0% at its lowest and 183% at its highest.

random	exhaustive	relative error (%)
58	58	0
74	74	0
16	16	0
16	16	0
41	41	0
54	41	31,70731707
25	25	0
28	28	0
24	24	0
24	24	0
24	24	0
16	16	0
28	28	0
31	31	0
25	25	0
25	25	0
25	25	0
68	24	183,3333333
33	33	0
34	34	0
19	19	0

TABLE VI: Relative error at 75% density

We can conclude that as the density increases the random algorithm gives less accurate answers. This happens because with more density we can have much more dominating sets therefore it is more probable that we don't find the best one.

VI. Discussion

A. *Comparison of the results of the experimental and the formal analysis*

We can conclude that the formulas obtained in the formal analysis were correct, because all the results that we got from the empirical Analysis coincide with the formulas that we got. Therefore proving that the order of complexity of the random algorithm is is between quadratic and cubic, O(n^2) and O(n^3), for the best case. For the worst we also proved that the order of complexity is cubic, O(n^3).

B. *Estimation of results for bigger problem instances*

In the random search the density of the graph will significantly affect the time it takes to get the answer, because it will take much more time to compute an 0% and density graph, than an an 100% density graph.
For the worst case, the execution time will increase by ($\approx 1,03$) when the number of vertices in the graph is increased linearly.
So to get how much time will 101 vertices with a 0% density graph take t calculate, all we need to do is take the time taken with 100 vertices and multiply by 1,03

$$time\_taken(n) = time\_taken(n-1) * 1,03$$

$$time\_taken(101) = time\_taken(100) * 1,03$$

$$\approx 1732.457 seconds$$

where n is the number of vertices in the graph. Note: 1,03 is the value that we got by comp(n+1)/comp(n)

For a graph with 200 vertices it will take $\approx$ 32325.681 seconds (or 8 hrs 58 min 45 sec), because for 100 vertices took 1681.997s then for 200 vertices it will take 1681.997*1.03$^{(200-100)}$=32325.681 seconds

For the best case, the execution time will increase approximately linearly with the increase of the vertices. The factor between comp(n+1)/comp(n) for the best case is 1,02.

So for a graph with 750 vertices it will take $\approx$ 174335.256 seconds (or 48 hrs 25 min 37 sec), because for 100 vertices took 0,448s then for 750 vertices it will take 0,448*1.02$^{(750-100)}$=174337.036 seconds

Also to denote that the value 1,02 was still decreasing (not stabilized) so probably in reality it will take less time than estimated

VII. Conclusion

In conclusion, we can say that the random search gives a balanced answer between the quality of the answer and the time taken.

In the random search we don't always found the best solution, but one close to it. Using this heuristic we take much less time to get an answer when compared to the exhaustive search. The random search takes much less time because of it's order of complexity which is smaller than the exhaustive one, for the best case it has a order of complexity between quadratic and cubic, O(n^2) and O(n^3), while in the worst case has a cubic complexity order, O(n^3).

The exhaustive search takes substantially longer than the random search as the size of the problem's input increases. Due to the fact that finding a minimum weight dominating set is an NP-Complete problem and no effective polynomial time algorithm has been discovered, it can be inferred that the random search strategy will be the most effective one if the ultimate objective is to quickly arrive at a solution. The solution may not be the optimal one but is close to it.

Since this is an NP hard problem there is no efficient polynomial time algorithm, we should consider using an faster algorithm that is close to optimal than a one that is the optimal but it takes too much time. Considering this we should chose the random search as the best approach. (The random search is better than the exhaustive one, but worst than the gredddy one)

We can also conclude that when the number of vertices increases the number of comparisons increases as well, but when the density increases, the number of comparisons will decrease, because the probability of finding adjacent vertices will be much higher and thus finding the solution much quicker.

References

[1] "Hybrid metaheuristic algorithms for minimum weight dominating set", Nov 2022 URL: https://www.sciencedirect.com/science/article/pii/S1568494612003092

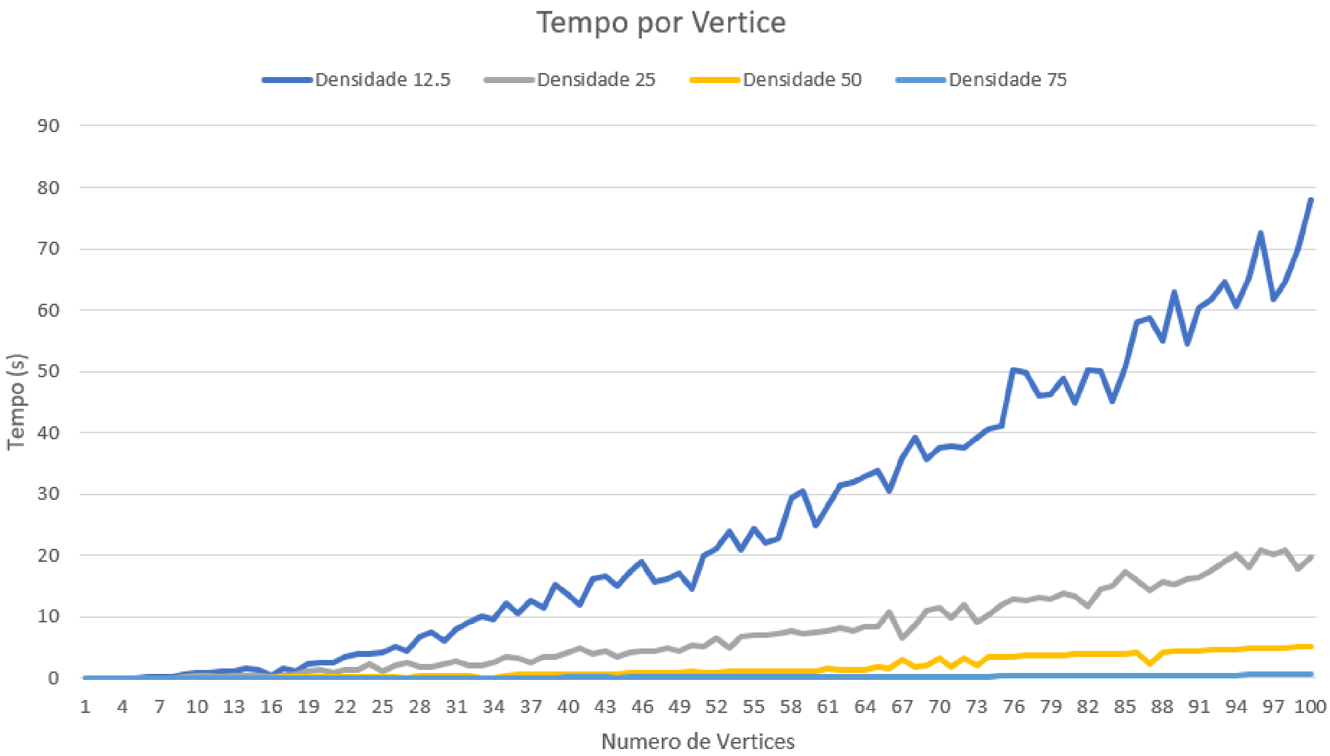

Fig. 1: Evolution of the time taken of the random search