1st Project - 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 two solutions, the exhaustive search always return the optimal solution and has an exponential complexity, whereas the greedy algorithm has a linear order of complexity for best case, and a quadratic order of complexity for worst case, but doesn't always return the best solution.

Both algorithms where compared and analysed in this paper and it is verified that the greedy approach is a good alternative to the exhaustive one when the goal is to find the solution of this graph problem as fast as possible, without the necessity of always obtaining the optimal solution.

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

In this project were used two algorithms to solve the same problem, and both were analysed. The two approaches used were a exhaustive one and a greedy one. For the exhaustive search algorithm we calculate all the possible dominating sets (by using combinations of all the vertices) for a given undirected graph and calculating the total weight of each dominating set and finding the minimum weight dominating set. For the greedy algorithm we calculated the ratio between the weight of the vertex and the degree of the vertex for every vertex of the graph. Then is selected the vertex with the lowest ratio (because we want the maximum number of connected vertices (degree) for the lowest weight) and removed from this list, for that vertex we will add all of its adjacent vertices to a set, and repeat the process until the size of the set is the same than the number of vertices of the graph. The graph it is defined by vertices which one with an associated weigh, edges and its adjacency matrix.

III. formal computational complexity analysis

A. exhaustive search

To make all the possible subsets of vertices with the given graph, we made all the possible combinations of vertices of the given graph witch leave us with 2^n sets. But in this case we still have an empty set, which can never be a dominating set, so we must subtract one to the number of total sets. So in conclusion the total number of subsets that can be a dominating set is $2^n - 1$, considering n as the number of vertices.

$$subsets\_found(n) = \sum_{i=1}^{n} \sum_{j=1}^{\frac{n!}{i!(n-i)!}} 1 = 2^n - 1$$

In the exhaustive search, for the best and the worst case the number of comparisons will always be the same, no matter the density chosen. This happens because we will always compare the adjacency of every single vertex in the subset to all the others in the graph, and only in the end we will conclude that the given subset is a dominating set. So even if the density is at 100% and therefore all the vertices are adjacent, we still will verify all the other vertices in the subset (which isn't very efficient), and if the density is at 0% we are obligated to verify all the other vertices in the graph.

$$num\_comp(n) = \sum_{i=1}^{n} \sum_{j=1}^{\frac{n!}{i!(n-i)!}} \left(\sum_{l=1}^{i} \left(\sum_{m=1}^{n} 1 \right) + 1 \right) =$$

$$2^{n-1}n^2 + 2^n - 1$$

The overall complexity of the exhaustive search will be exponential,O(2^n), both in the best and worst case, because for each subset created we will compare the adjacency of each vertex in the subset to all the vertices in the graph and only in the end check if it is a

dominating set. Which means that we will iterate four loops, the first two to create all the possible subsets in the graph, and the last two compare the adjacency between the vertices in the subset and the ones of the graph.

B. greedy search

The best case happens when the density of the graph is 100%, because all the vertices will be adjacent. Only the first vertex will be added to the set. The final result is a subset with only one vertex, which is the one with the lowest ratio (minimum cost). This means that, we can conclude on the first iteration that the selected subset is a dominating set

$$num\_comp(n) = \sum_{i=1}^{n} 1 + 1 = n + 1$$

Considering that the graph has n vertices, in the best case scenario the while loop will only run for one iteration and the inner loop (for) will only iterate n times, which translates to a linear complexity, O(n).

The worst case happens when the density of the graph is 0%, because no vertex will be adjacent to another vertex. All the vertices will be added to the set. This means that the we have to iterate all the vertices in the graph and for each vertex check if it has some adjacent vertex.

$$num\_comp(n) = \sum_{i=1}^{n} \left(\sum_{j=0}^{n} 1 \right) = n(n+1)$$

In this case the overall complexity will be quadratic $O(n^2)$, since the inner and outer loop will have to iterate all the vertices (the +1 corresponds to comparing if the vertex is adjacent to it self which isn't efficient)

IV. Experimental results

The experimental results can be found in the "exhaustiveSearch.xlsx" and "greedy.xlsx" files. These results contains the number of vertices (until 21),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. Each file corresponds to a different approach and there is also a graphic with the number of comparisons.

A. exhaustive search

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

For the best and worst case we can conclude that its order of complexity is exponential, $O(2^n)$, because the ratio between num_comp(n+1) and num_comp(n) tends to the value of 2,2. Therefore, proving that the order of complexity of the exhaustive algorithm is exponential, $O(2^n)$.

vertices	comp(n)	comp(n+1)	comp(n+1) /comp(n)
1	2	11	5,5
2	11	43	3,909090909
3	43	143	3,325581395
4	143	431	3,013986014
5	431	1215	2,819025522
6	1215	3263	2,685596708
7	3263	8447	2,588722035
8	8447	21247	2,515330887
9	21247	52223	2,457899939
10	52223	125951	2,411791739
11	125951	299007	2,373994649
12	299007	700415	2,342470243
13	700415	1622015	2,315791352
14	1622015	3719167	2,29293009
15	3719167	8454143	2,273128096
16	8454143	19070975	2,255814102
17	19070975	42729471	2,240549893
18	42729471	95158271	2,226993894
19	95158271	210763775	2,214876046
20	210763775	464519167	2,203980105

TABLE I: Ratio of comparisons for the exhaustive search

B. greedy search

In the greedy algorithm the number of comparisons is much lower than in the exhaustive one, because by using this algorithm we don't search for all the possible solutions.

For the best case, it is possible to see that the ratio between num_comp(2n) and num_comp(n) tends to 2, therefore we can conclude that the order of complexity of the algorithm is linear, O(n).

vertices	comp(n)	comp(2n)	T(2n)/T(n)
1	2	3	1,5
2	3	5	1,666666667
3	4	7	1,75
4	5	9	1,8
5	6	11	1,833333333
6	7	13	1,857142857
7	8	15	1,875
8	9	17	1,888888889
9	10	19	1,9
10	11	21	1,909090909

TABLE II: Ratio of comparisons for the best case of the greedy search

In relation to the worst case, the ratio tends to the value of 4, proving that the order of complexity of the algorithm in the worst case is quadratic, $O(n^2)$.

vertices	comp(n)	comp(2n)	T(2n)/T(n)
1	2	6	3
2	6	20	3,333333333
3	12	42	3,5
4	20	72	3,6
5	30	110	3,666666667
6	42	156	3,714285714
7	56	210	3,75
8	72	272	3,777777778
9	90	342	3,8
10	110	420	3,818181818

TABLE III: Ratio of comparisons for the worst case of the greedy search

V. 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 exhaustive algorithm is exponential, O(2^n), for the best and worst case. For the greedy algorithm we also proved that the order of complexity is linear, O(n), for the best case, and for the worst case is quadratic, O(n^2)

B. Estimation of results for bigger problem instances

In the exhaustive search the density of the graph will not affect the time it takes to get the answer, because it will take the same time to compute an 0% and an 100% density graph. However, the execution time will double($\approx 2,2$) when the number of vertices in the graph is increased linearly. So to get how much time will 22 vertices with a random density take t calculate, all we need to do is take the time taken with 21 vertices and multiply by 2,2

$$time\_taken(n) = time\_taken(n-1) * 2,2$$

$$time\_taken(21) = time\_taken(20) * 2,2$$

$$\approx 88.187 seconds$$

Note: the time taken for 21 vertices in our tests was $\approx 87,158$ seconds (results of exhaustive search 0% - in the excel)

For a graph with 40 vertices it will take $\approx$ 614838242.465 seconds (or 19 years, 181 days, 4 hr. 24 min. 2 sec), because for 20 vertices took 87,158s then for 40 vertices it will take 87,158*$2,2^{(40-20)}$=614 838 242.465 seconds

It is important to take the density of the graph into account when calculating results for larger problem situations using the greedy algorithm, because this will influence the time that it takes to get an solution. An graph with higher density will take less time to compute an answer than a one with a very low density.

For the best case using the gready search the running time will increase approximately linearly, and the in worst case the running time will increase approximately quadratically. Unfortunately the maximum number of vertices that we can have in the graph is 81, the simulations doesn't support more, and for this value the running time was still very close to zero (≈ 0.004 s).

As for the comparisons in the best case they grow linearly, so we can calculate them easily,

$$num\_comp(n) = n + 1$$

where n is the number of vertices in the graph. For a graph with 100 vertices it will take 101 comparisons to get an answer.

Now considering the worst case the number of comparisons grow quadratically, so we can calculate them with the formula that we got from the formal analysis,

$$num\_comp(n) = n^2 + n$$

where n is the number of vertices in the graph. So for a graph with 100 vertices it will take 10 100 comparisons to get an answer.

VI. Conclusion

In conclusion, we can say that the exhaustive search always gives the best answer possible, but it takes too much time due to its order of complexity, which is exponential, O(2^n), for the best and worst case. This algorithm shouldn't be used in very big instances of this problem.

In the greedy 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 one. The greedy 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 linear complexity order, O(n), while in the worst case has a quadratic complexity order, O(n^2).

The exhaustive search takes substantially longer than the greedy 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 greedy 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 greedy search as the best approach.

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

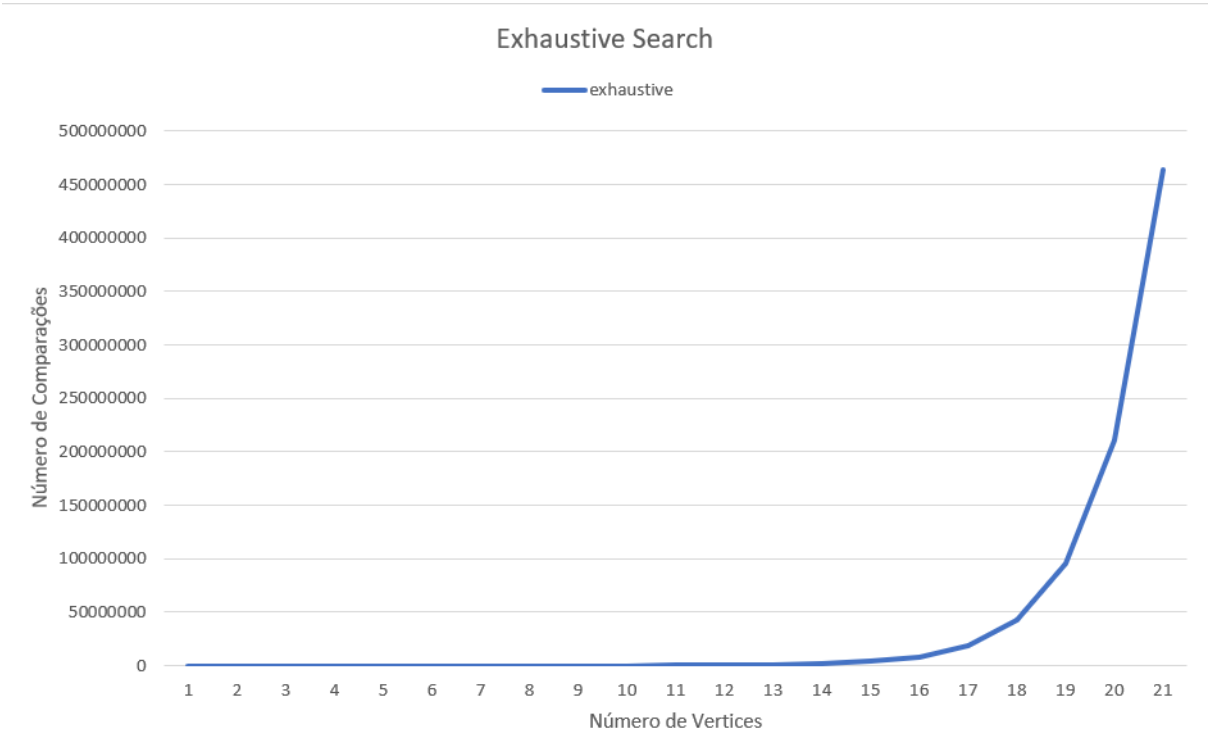

Fig. 1: Evolution of the number of comparisons of the exhaustive search

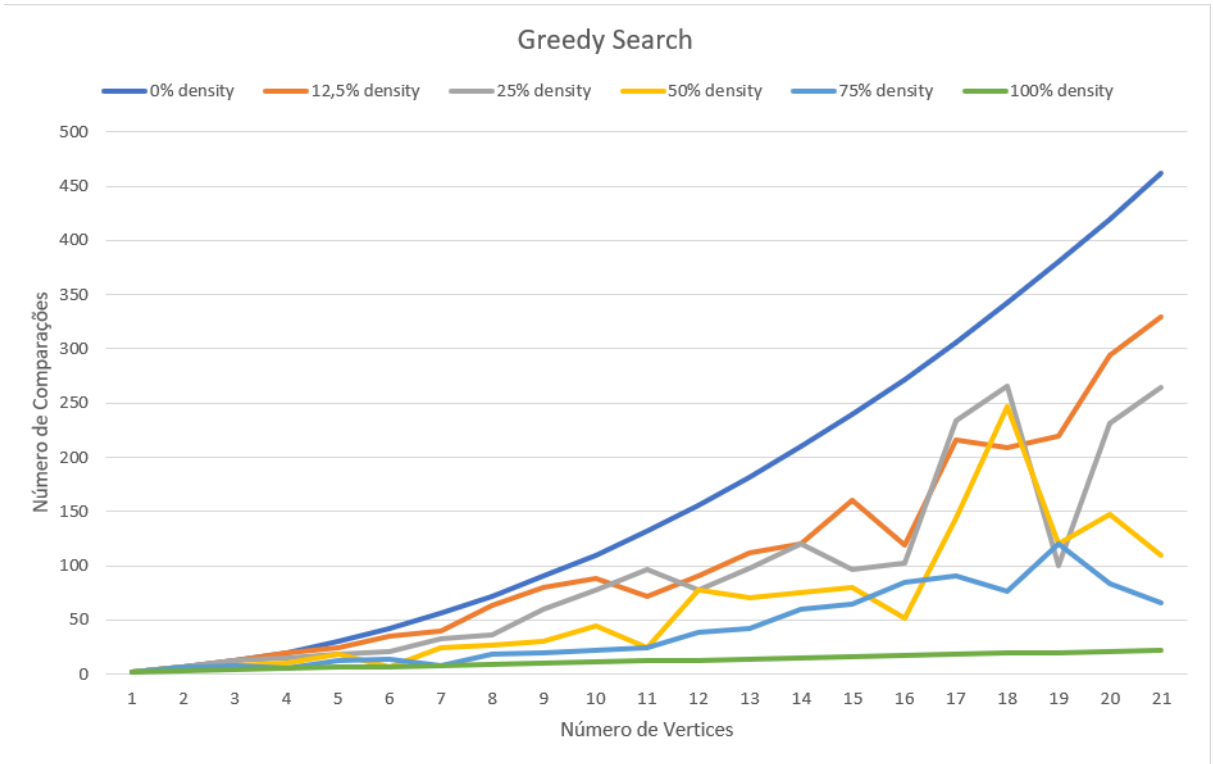

Fig. 2: Evolution of the number of comparisons of the greedy search, for different values of vertices and density