Advanced Algorithms
2nd Project - Randomized Algorithms

José Pedro Marta Trigo
Departamento de Eletrónica, Telecomunicações e Informática
Universidade de Aveiro

***Abstract*—This paper presents the work done for the second assignment of the Advanced Algorithms course. The paper will analyse the use of randomized algorithms to solve a well known combinatorial problem: the minimum weight vertex cover problem.**

I. Introduction

In graph theory, a vertex cover of a graph is a set of vertices that includes at least one endpoint of every edge of the graph [1].
This paper aims to present and analyze the results of the Monte Carlo method (a randomized algorithmic approach) to solve the minimum weight vertex cover problem:

> Find a minimum weight vertex cover for a given undirected graph G(V, E), whose vertices carry positive weights, with n vertices and m edges. A vertex cover of G is a set C of vertices, such that each edge of G is incident to, at least, one vertex in C. The weight of a vertex cover is the sum of its vertices' weights. A minimum weight vertex cover is a vertex cover whose total weight is as small as possible.

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle [2].

II. Formal Computational Analysis

For each singular experiment of the randomized algorithmic approach, a random vertex is selected and added to the vertex cover to begin the experiment. Then, the algorithm loops through all edges of the graph in a random order and checks if one of the edge's vertex is in the vertex cover. If not, it randomly selects one of the vertexes and adds it to the vertex cover. After iterating through all edges of the graph, there is a verification process in order to ensure that the resulting vertex cover is valid (it is valid if each edge of the graph is incident to, at least, one vertex in the edge) and to determine if it has minimum weight when compared with previous experiments. The simulation of new experiments stops when the ratio between already tested samples counter (duplicate samples) and the total number of samples tested is greater than 50%. The reason is that if most of the randomly generated samples are duplicates, then there is no point in continuing the simulation. Pseudo code for the algorithm:

Algorithm 1 Min weight vertex cover using Monte Carlo

```
1:  minWeight ← ∞
2:  minVertexCover ← {}
3:  samples ← {}
4:  while dupSamplesCounter/totalTested < 0.5 do
5:      vertexCover ← {randomChoice(vertexes)}
6:      for each edge (u, v) in randomShuffle(edges) do
7:          if u & v ∋ vertexCover then
8:              vertexCover.insert(randomChoice(u, v))
9:          end if
10:     end for
11:     if isValidVertexCover(edges, vertexCover) &
        isNotAlreadyTested(vertexCover) &
        vertexCoverWeight < minWeight then
12:         minVertexCover ← vertexCover
13:         minWeight ← vertexCoverWeight
14:     end if
return minWeight, minVertexCover
```

The computational complexity of iterating through all edges is O(n). The while loop's complexity is also O(n). Therefore, the computational complexity of this algorithm is O(n^2).

III. Graph Generator

Since the the minimum weight vertex cover problem is a graph related problem, a custom graph generator was built in order to randomly generate graphs with the predefined seed "98597". In total, 100 different graphs where generated with a vertex count ranging between 2 and 26 and an edge density of 12.5%, 25%, 50% and 75% for each vertex count. The vertexes's weights were derived from it's distance to the origin of the plane, the pair (0,0).

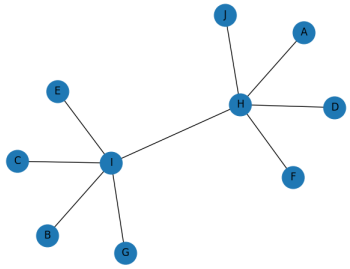

Fig. 1. Graph with 10 vertexes and 12.5% edge density

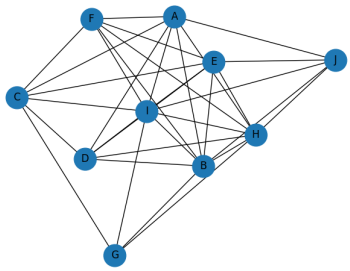

Fig. 2. Graph with 10 vertexes and 75% edge density

IV. Experiments

A. Weight Results

Unlike an exhaustive search algorithm, the Monte Carlo method does not guarantee an optimal solution to the minimum weight vertex cover problem. Instead, it yields a fairly close or accurate solution in a very reasonable amount of time. When comparing it to the greedy heuristic approach, we can notice that the Monte Carlo's results are much closer to the optimal solution.

Table I displays the results of the Monte Carlo algorithm compared to exhaustive search and greedy heuristic algorithms for the selected vertex count and edge densities:

TABLE I
Comparison of the three algorithm's weight results

n	Monte Carlo				Exhaustive Search				Greedy Heuristic			
	12.5%	25%	50%	75%	12.5%	25%	50%	75%	12.5%	25%	50%	75%
5	15	14	33	47	15	14	33	47	15	14	33	47
10	32	42	92	121	32	42	89	121	32	42	89	142
15	29	105	177	204	29	105	177	204	29	161	244	245
20	52	151	235	261	52	151	235	261	56	187	266	305
25	128	209	289	317	128	209	289	307	196	321	375	356

Table II displays the vertex cover of both the three algorithms at 75% edge density:

TABLE II
Comparison of the three algorithm's vertex cover

n	Monte Carlo	Exhaustive Search	Greedy Heuristic
	75%	75%	75%
2	{A}	{B}	{B}
3	{B}	{B}	{B}
4	{A,D}	{A,D}	{C,D}
5	{A,C,D}	{C,D,E}	{A,C,D}
6	{A,B,C,F}	{B,C,E,F}	{C,D,E,F}
7	{A,B,D,F,G}	{A,B,D,F,G}	{B,C,D,F,G}
8	{A,B,C,D,H}	{A,B,C,D,H}	{A,B,D,F,G,H}
9	{A,B,C,D,E,H}	{A,B,C,D,E,H}	{A,B,C,D,F,G,H,I}
10	{A,B,C,D,F,H,J}	{A,B,C,D,F,H,J}	{A,B,C,D,F,H,I,J}

B. Basic Operations

Regarding Basic operations, every algorithms's basic operations count increases as the number of vertexes also increases. The increase is very steep in the exhaustive search algorithm, and very light for the greedy heuristic approach. Monte Carlo stands in the middle of the other two.

Table III displays the number of basic operations performed by the Monte Carlo algorithm compared to the exhaustive search and greedy heuristic algorithms for the selected vertex count and edge densities:

TABLE III
Comparison of the three algorithm's basic operations results

n	Monte Carlo				Exhaustive Search				Greedy Heuristic			
	12.5%	25%	50%	75%	12.5%	25%	50%	75%	12.5%	25%	50%	75%
5	198	267	200	246	780	780	815	920	64	64	102	166
10	2133	2829	2406	1742	48730	51138	57182	60081	161	241	560	1265
15	6917	26018	13039	5225	2.7e6	2.2e6	2.4e6	2.3e6	241	1137	4221	3665
20	51160	141285	43367	10489	9.9e7	9.3e7	8.6e7	8.2e7	581	2292	3484	10044
25	544818	1185652	114157	20612	3.8e9	3.2e9	2.9e9	2.9e9	2040	11398	10138	16835

C. Tested solutions

In the greedy heuristic approach, the number of tested solutions is always one since the solution is progressively built by always choosing the local optimum result during each iteration.

The Exhaustive Search method computes all possible vertex combinations of size n, hence why we see such a sharp increase in the number of tested solutions. Since edge density does not influence the total number of combinations to test, the result remains the same across all edge density values.

The Monte Carlo method's number of tested solutions varies according to vertex count and edge density since the stop condition is based on the ratio of duplicate samples to total samples tried.

Table IV displays the number of tested solutions by the Monte Carlo algorithm compared to the exhaustive search and greedy heuristic algorithms for the selected vertex count and edge densities:

TABLE IV
COMPARISON OF THE THREE ALGORITHM'S NUMBER OF TESTED SOLUTIONS

n	Monte Carlo				Exhaustive Search				Greedy Heuristic			
	12.5%	25%	50%	75%	12.5%	25%	50%	75%	12.5%	25%	50%	75%
5	9	12	7	7	31	31	31	31	1	1	1	1
10	58	65	34	21	1023	1023	1023	1023	1	1	1	1
15	177	329	118	39	32767	32767	32767	32767	1	1	1	1
20	871	1242	281	63	1e6	1e6	1e6	1e6	1	1	1	1
25	5481	7195	574	95	3.4e7	3.4e7	3.4e7	3.4e7	1	1	1	1

D. Execution Time

Regarding execution times, the greedy heuristic has a massive advantage over the exhaustive search algorithm because it doesn't need to generate and iterate over all vertex combinations and test all possible solutions to check their result. Another important observation is that edge density does not influence execution times.

The Monte Carlo method has very reasonable execution times paired with very accurate minimum weight vertex cover solutions.

Table V displays the execution time in seconds by the Monte Carlo algorithm compared to the exhaustive search and greedy heuristic algorithms for the selected vertex count and edge densities:

TABLE V
COMPARISON OF THE THREE ALGORITHM'S EXECUTION TIMES

n	Monte Carlo				Exhaustive Search				Greedy Heuristic			
	12.5%	25%	50%	75%	12.5%	25%	50%	75%	12.5%	25%	50%	75%
5	1.9e-4	2.7e-4	2.2e-4	2.5e-4	9.7e-5	9.6e-5	1.6e-4	1.4e-4	5.6e-5	4.1e-5	5.5e-5	4.3e-5
10	1.7e-3	3.7e-3	2.4e-3	2e-3	2.8e-3	2.9e-3	3.2e-3	3.2e-3	8e-5	9.3e-5	7e-5	9.4e-5
15	0.008	0.036	0.017	0.009	0.12	0.11	0.11	0.09	5.2e-5	1.8e-4	1.6e-4	1.4e-4
20	0.06	0.19	0.07	0.024	3.8	3.4	3	3.1	7.5e-5	1.3e-4	1.7e-4	4.6e-4
25	0.66	1.63	0.25	0.056	139.3	102.6	102.3	102	2.1e-4	2.1e-4	2.9e-4	4.1e-4

V. EXPERIMENTAL AND FORMAL ANALYSIS COMPARISON

Figure 3 show the execution time growth for the Monte Carlo algorithm. As observed, the results validate the formal analysis.

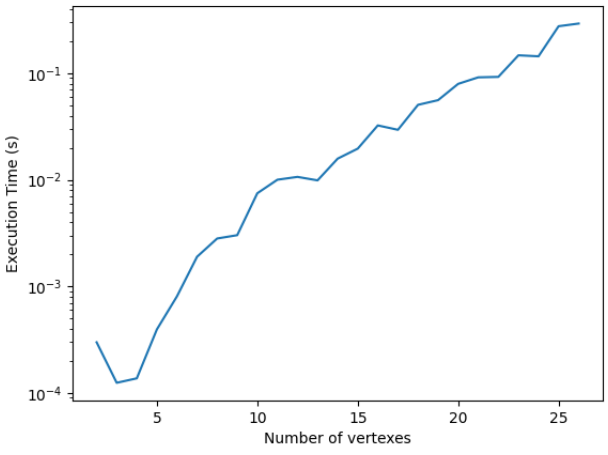

Fig. 3. Comparison of Monte Carlo algorithm execution time with formal analysis

VI. LARGER PROBLEM INSTANCES

The Monte Carlo algorithm's execution time for 20 vertexes at 50% edge density is approximately 0.07s. By knowing this we can approximate the execution time of larger instances with the following formula:

$$T(n) = \frac{n^2}{20^2} * 0.07$$

The execution time for 25 vertexes at 50% edge density was 0.25 seconds. We can now compare this value to our approximation, using the formula:

$$T(26) = \frac{25^2}{20^2} * 0.07 = 0.11s$$

The relative approximation error was 127% which indicates us that we cannot feasibly estimate the execution time of this strategy, since the stop condition has a high variance.

VII. CONCLUSION

The exhaustive search algorithm is a simple implementation approach to find the optimal solution for smaller computational problems. However, the exponential increase in execution time means that they are unsuitable for large problems. Randomized algorithms such as the Monte Carlo method are a better choice when a fairly accurate solution is acceptable when traded for a much lower execution time.

REFERENCES

[1] "Vertex cover" https://en.wikipedia.org/wiki/Vertex_cover
[2] "Monte Carlo method" https://en.wikipedia.org/wiki/Monte_Carlo_method