Evaluating the Performance of Simulated Annealing Algorithms for Solving the Minimum Vertex Cover Problem

Hugues ESCOFFIER, Nicolas GODRON

1. Introduction

1.1 Minimum Vertex Cover Problem

The ‘*Minimum Vertex Cover Problem*’ (MVCP) is a classic NP-hard optimization problem. In this report, we have developed and optimised a simulated annealing algorithm to approximate the optimal solution to the problem. The algorithm was then tested on reference graphs and showed promising results in finding optimal results with high probability.

1.2 Formalisation of the MVCP

Consider $G(V, E)$ an undirected graph with V the vertex set and E the edge set. In the MCVP, we try to find V' the smallest subset of V such that for each edge (a, b), there is a included in V' and/or b included in V'. Moreover, we precise that the generated graphs in our program do not have self-loops (a, a). An example of a graph and solution is shown in Figure 1.

1.3 Simulated Annealing

Simulated Annealing is a global optimization algorithm that is inspired by the process of annealing in metallurgy, where a material is heated to a high temperature and then cooled slowly to reduce defects and increase its structural stability. The algorithm is used to solve optimization problems by simulating the physical process of annealing. The basic idea of the algorithm is to start with a random solution to a problem and then repeatedly make small random changes to the solution while gradually decreasing the "temperature" of the system. The probability of accepting a new solution increases as the temperature decreases, which allows the algorithm to explore a wider range of solutions and escape local optima. The algorithm terminates when the temperature reaches a low enough value, or the best solution is not improving anymore.

1.4 Goal of the study

The purpose of this study is to present an implementation of simulated annealing from scratch, and to further improve upon this method by utilizing existing literature, with a specific focus on the method outlined in the paper by Xinshun et Al. The goal is to demonstrate the effectiveness of the optimized method in comparison to the initial, naive implementation.

2. Materials & Methods

2.1 Random Generation of an instance of the problem

In this project, the random vertex generation is done through the function `create_vertex(nb_nodes, nb_edges)` of the `vertex.py` script. It implements the Erdos-Rényi algorithm from scratch and returns an instantiated graph g. In detail, the random generation follows this pseudocode algorithm:

```
# Generation of a graph with a specified number of nodes and edges

DEFINE FUNCTION create_vertex(nb_nodes, nb_edges):

    INITIALIZE Graph g(nodes,edges) TO an empty graph (nodes, edges =  empty sets)
    INITIALIZE nodes_list TO list of length(nb_nodes)
    INITIALIZE edges_list TO empty list

    IF nb_edges >= (nb_nodes*(nb_nodes - 1))/2:
                  THEN OUTPUT('The number of edges requested is too large. The number of
edges has been set to the maximum number of possible edges')
                  SET nb_edges TO (nb_nodes*(nb_nodes - 1))/2

    WHILE length(edges_list) < nb_edges:
                  SET edge TO list containing two random elements from nodes_list
                  ADD edge TO edges_list
                  SET edges_list TO list of unique values from edges_list

    ADD nodes_ list to g.nodes
    ADD edges_ list to g.edges

    RETURN g
```

The random generation is the first command launched by the `main(nb_nodes, nb_edges)` of the `run.py` script.

2.2 Brute Force

In this study, we proposed a brute force algorithm for solving the minimum vertex cover problem in undirected graphs. The approach involves decomposing and testing all possible subsets of vertices for a given graph. This method allows us to compute the optimal solution to the problem for small graphs with up to 20 vertices. We then used this basic approach as a benchmark to evaluate the performance of a simulated annealing program.

2.3 Simulated Annealing

a. Simulated Annealing 'Naive'

In this study, we implemented a classical method of simulated annealing with elitism (keeping the best solution) for solving the minimum vertex cover problem in undirected graphs.
The method began with a complete solution as the initial solution (*s0*). The parameters used for this method are summarized in Table 1.

At each step, we evolved the solution using the function `move(solution)` which randomly removed a vertex from the solution. The energy function of this approach was calculated as follows:

$$Energy = Nb_{nodes} + Verify$$

where $Verify = 0$ if the solution is valid and $Verify = 1000$ otherwise.

b. Simulated Annealing 'Xinshun'

In this study, we implemented a second approach for solving the minimum vertex cover problem in undirected graphs. This approach was developed in light of the results obtained from the literature, and we have computationally implemented from scratch a mathematical

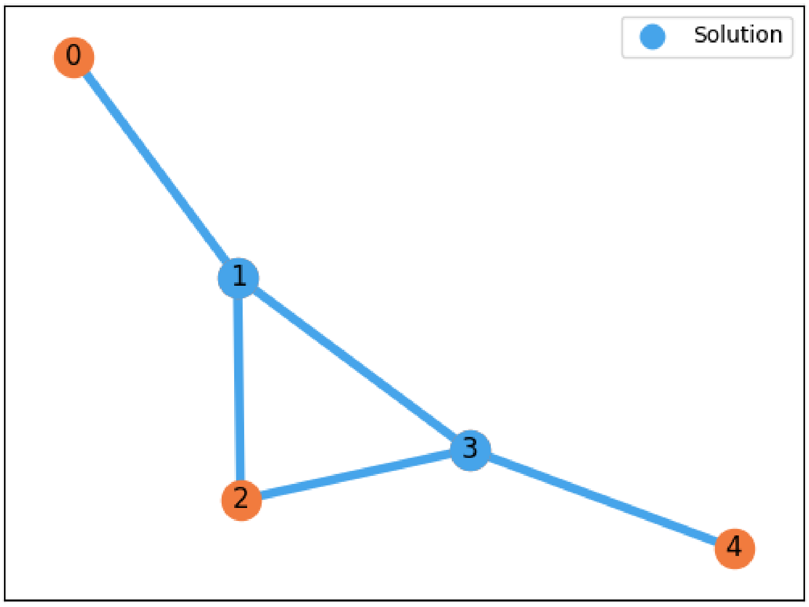

Figure 1: A simple graph showing the Minimum Vertex Cover Problem

Parameters	Naive Methods	Xinshun Methods
Alpha	0.95	0.95
T0	50	50
Max Fails	10000	10000

Table 1: Parameters used to tune the algorithm. Alpha is the constant used to update the temperature (i.e. $t_1 = \alpha * t_0$ under the geometric updating rule). T0 is the initial temperature of the annealer. Max Fails is the maximum number of failures until an iteration stops.

Graph Size	Naive Solution Good Solution Rate	Xinshun Solution Good Solution Rate
5 Vertices, 7 Nodes	100%	100%
10 Vertices, 20 Nodes	92%	100%
20 Vertices, 30 Nodes	70%	98%

Table 2: Summary of Results for Different Graph Sizes. The good solution rate is calculated as (number of optimal solution founded) / (total number of solutions founded)

solution proposed by the Xinshun et al. team in 2005. Like the first approach, we also began with an initial solution containing all the vertices of the graph, and the improvement of the solution was achieved by randomly removing a vertex.

However, the energy function used in this approach was different. It was defined as:

$$Energy = 0.99 * Nb_{vertex\ in\ solution} + Nb_{non-covered\ edges}$$

where $Nb_{vertex\ in\ solution}$ is the number of vertices in the solution and $Nb_{non-covered\ edges}$ is the number of edges not covered by the solution.

We found that introducing this 0.99 constant improved the algorithm's solution when presented with a complete graph.

The results of our simulation show that this approach improves upon the first approach by effectively reducing the number of vertices in the solution

2.4 Stopping criterion

We proposed a modified approach for solving the minimum vertex cover problem in undirected graphs using simulated annealing. Instead of working with a maximum number of steps, our approach was uniquely based on a maximum number of fails for each instance of the problem (set to 10,000). Additionally, we implemented a stopping criterion in which each iteration of the simulated annealing algorithm would be terminated after 10 iterations without improvement.

3. Results

3.1 Results of the ‘Naive’ and ‘Xinshun’ methods compared to the results of the brute force method

We evaluated two methods for solving the minimum vertex cover problem in undirected graphs. The first method is a naive solution and the second method is based on a solution proposed by Xinshun et al. in 2005. We ran both methods on 50 different graphs containing 5, 10, and 20 vertices. The results are summarized in Table 2.

The results show that both algorithms give excellent results for graphs containing 5 vertices. However, as the number of vertices in the graph increases, the performance of the naive solution starts to decline, with an error rate of 8% for graphs with 10 vertices and 30% for graphs with 20 vertices. On the other hand, the Xinshun solution gives almost perfect results for all the tested graph sizes. These results demonstrate that the Xinshun methods is more robust and effective in solving the minimum vertex cover problem for a wide range of graph sizes.

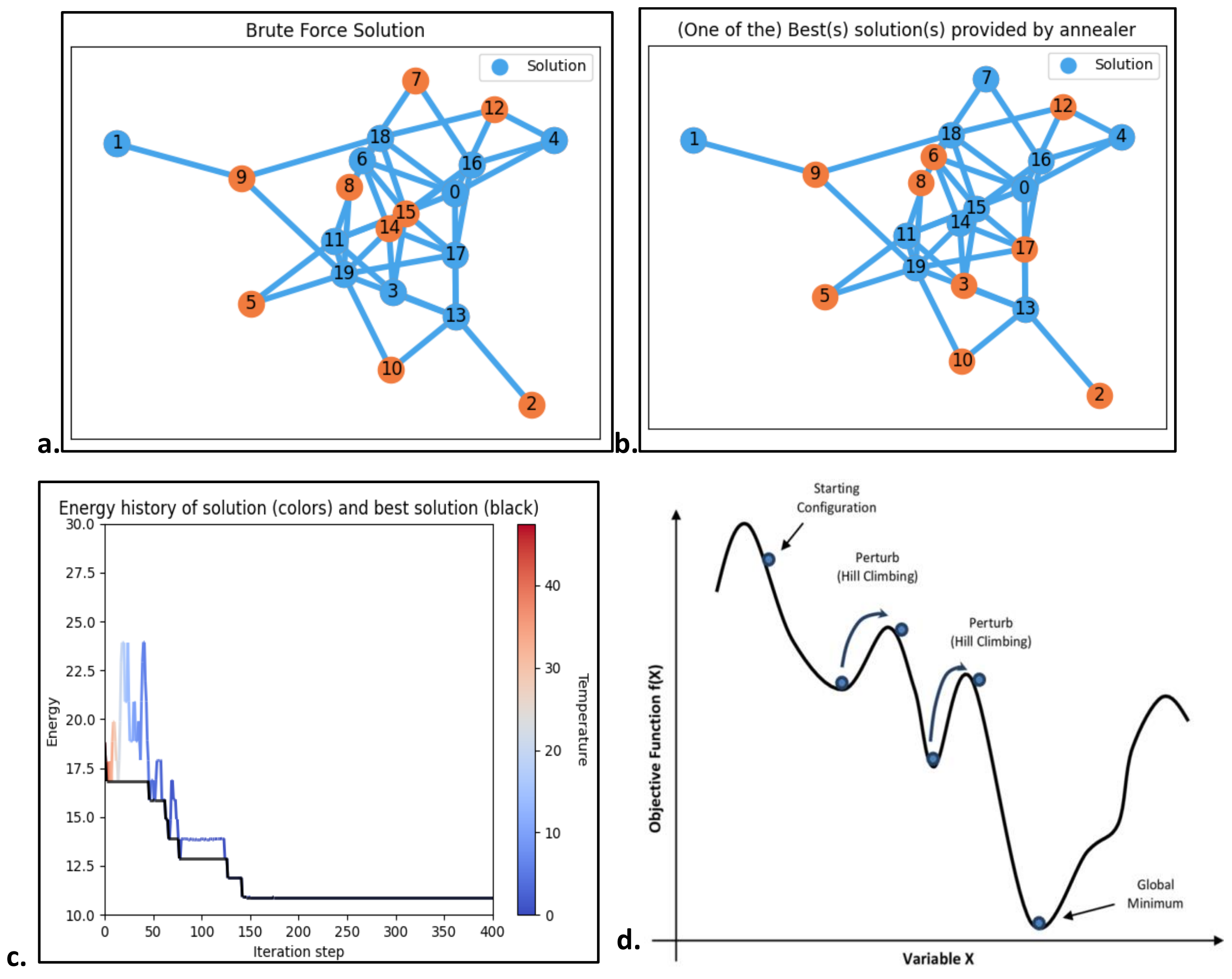

Figure 2: a. Brute Force and **b.** Simulated annealing solutions in blue to the same graph. **c.** Energy curves as a function of the iteration step. The multicolor curve corresponds to the best solution of this iteration step. Its color ranges from red for high temperatures to blue for low ones. The black curve corresponds to the best solution since the beginning of the annealing process. *NB:* The plot has been cropped at the 400th iteration step to allow for better visualisation. **d.** Visual aid to understand the annealing process, by Omid Ghasemalizadeh.

3.2 Visual Result

As seen on **Fig. 2a.** and **2b.**, the solution from the annealer is optimal as it has the same number of nodes (11) included as one of the brute force solutions. We can also note that these two solutions differ from one another:

Solution(Brute Force) = {0, 1, 3, 4, 6, 11, 13, 16, 17, 18, 19}

Solution(Annealer) = {0, 1, 4, 7, 11, 13, 14, 15, 16, 18, 19}

As for the energy history, on **Fig. 2c.**, it shows the best solution at each iteration (multicolor oscillating curve) as well as the best solution encountered yet (black curve). At the beginning of the annealing process, the temperature is high (red on the color scale), the solution oscillates with great amplitude. As illustrated with a one-dimension objective function on **Fig. 2d.**, this allows the exploration of the solution space. Indeed, it is then possible for the solution to move up the gradient (towards 'worse' solutions). This property is important to avoid the local minimum pitfall, in which an energy barrier could keep the metaheuristic algorithm from reaching the global minimum of an objective function.

As the annealing progresses the temperature lowers, to allow for smaller scale improvement of the solution. The improvement of the best solution per iteration lowers until it reaches its minimum, with no improvement for 10,000 steps. The SA algorithm then outputs this solution.

4. Discussion

4.1 Simulated Annealing 'Naive'

This algorithm accepting only valid solutions allowing a limited exploration of the universe has the problem of falling quickly into a local minimum and not being able to escape from it. This kind of algorithm has good results on small samples but becomes extremely bad as soon as the number of vertices increases. We see in Table 2 a significant degradation for n = 20.

4.2 Simulated Annealing 'Xinshun'

The method for solving the minimum vertex cover problem in undirected graphs. The results of our simulation show that this method is able to provide optimal solutions for almost all the graphs tested so far (Table 2). However, we observed one instance where the algorithm made a bad prediction error. The graph in question contained only one optimal solution with a cardinality of 10, and the algorithm found only one solution with a cardinality of 11. It is worth noting that this is an isolated case, as the algorithm performed successfully in 7 out of 8 instances where the situation of a graph with only one optimal solution was presented. This suggests that the algorithm is generally able to find the optimal solution, even in cases where the graph contains a unique solution.

However, we were unable to test the performance of this method on larger graphs due to limitations in the computation of brute force solutions. Despite this limitation, the results obtained with this method are extremely encouraging and suggest that we have developed a near-optimal solution for solving the problem. Further research is needed to confirm the scalability of this method on larger graphs and to improve the performance of the algorithm.

4.3 Conclusion

In conclusion, this study presented the development and implementation of two approaches for solving the minimum vertex cover problem in undirected graphs. The first approach was developed in a naive way in order to better understand the problem. The second approach, which was developed with the help of literature, was found to be more optimal and effective in solving the problem. Further research is needed to confirm the scalability and robustness of this method in larger graphs and to study its performance in other specific types of graphs.

5. Bibliography

Xinshun Xu, Jun Ma, An efficient simulated annealing algorithm for the minimum vertex cover problem, Neurocomputing, Volume 69, Issues 7–9, 2006, Pages 913-916, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2005.12.01.

6. Appendix: Program

The program as well as example outputs are available on Github: https://github.com/h-escoffier/Anneal-ize_the_Vertex