A LOCAL AND GLOBAL APPROACH TO THE MINIMUM VERTEX COVER OPTIMISATION PROBLEM

Ethan Leet
School of Information and Technology
Griffith University, Gold Coast campus, Australia
ethan.leet@griffithuni.edu.au

***Abstract* – The Minimum Vertex Cover (MVC) is a NP-hard optimisation problem where the objective is to determine a subset of vertices within an undirected graph which covers all edges such that the number of vertices present in the subset is minimised. This research used the compliment of eight common graphs and their respective benchmarks to provide insight into three different types of algorithms that could be used to solve this type of optimisation problem.**

***Keywords* – Minimum Vertex Cover, Local Search, Simulated Annealing**

I. INTRODUCTION

Optimisation is a prominent research area which encompasses a variety of problems, including MVC and all its variants. Previous solutions for the MVC have focused heavily on a greedy approach. The Brock compliment graphs were individually designed to outsmart the approach of a greedy algorithm [1].

This research provides two local search algorithms; a next-best greedy and randomised search, as well as a global search algorithm simulated annealing. In addition to the Brock collection four other graphs were also individually tested with each of the three aforementioned algorithms [1]. Each graph can be considered a test case where individual algorithms were measured against each other and the benchmark provided.

The remainder of this report analyses previous works carried out in relation to the MVC problem as well as its variants, a description of the techniques used, followed by their results and a summary.

II. PREVIOUS WORKS

[2] outlines a greedy approach to the MVC cover by selecting the highest degree node present in an adjacency matrix representation of a graph and adding it to a cover. The nodes and its incidents are then removed from the possible list of next nodes, and the algorithm repeats. It is stated that this type of approach obtained near optimal solutions for all graphs tested.

A Randomised Greedy approach was conducted by Gao, Friedrich, Neumann and Hercher [3]. In this approach, a random node is chosen on each iteration rather than the highest degree node chosen in the pure greedy approach. The aim of their research was to provide an alternative method to escape the local minima in which most greedy solutions were found to get stuck in. It was concluded that this type of algorithm obtained similar results compared to the pure greedy method outlined above.

[4] produced a multitude of different algorithm types for the MVC problem. The most promising algorithm is a stochastic global search and optimisation algorithm, simulated annealing. In their work, they managed to escape the local minima solutions found by the randomised and greedy approaches. Although this approach better results, the global minima was not found for any test cases.

III. TECHNICAL APPROACH

For this research three algorithms based off the above previous works were implemented; a next-best greedy search algorithm, a randomised greedy search algorithm, and simulated annealing. The compliment of the eight aforementioned DIMACS graphs are stored as a hash-map and represented as an adjacency list [1]. Each node within the adjacency list contains the vertex and the set of their edges. Each algorithm is given 100 trials with no cutoff nor restrictions and their results compared to the benchmarks for that graph.

A. Next-Best Greedy Local Search

The next-best greedy local search algorithm evaluates each node based off its degree. Where the highest degree nodes are placed into the cover and then removed from the next-best choice. This process repeats until a cover is found. Although the next-best vertex is always chosen, this is not always the same vertex upon each trial, due to the randomness of the hash-map implementation and the density of the graphs used.

Algorithm 1 Next-Best Greedy Local Search

```
populate ad jacency_list
populate node_ranks
while not valid cover do
    cover ← node_ranks.max
    node_ranks ← remove node_ranks.max
    if cover exists then
        return cover.size
    end if
end while
```

B. Randomised Local Search

This algorithm is a local search algorithm where its choice for adding the next node into the potential cover is random. Each node has the same chance at getting picked, and the random distribution of chosen nodes is normally distributed.

Algorithm 2 Randomised Local Search

```
populate ad jacency_list
while not valid cover do
    cover ← random node
    if cover exists then
        return cover.size
    end if
end while
```

C. Simulated Annealing

Simulated Annealing is a stochastic global optimisation algorithm. This algorithm is initialised with the total vertices present and with $temperate = 100,000$ which cools at a rate of $alpha = 0.80$ per iteration. The algorithm repeatedly improves its solution by firstly picking a random vertex. If this solutions is a better solution than the current solution it will accept and decrease the temperature. Otherwise, the solution will be compared to a probabilistic calculation on *rho*, where $rho = e^{\frac{solution}{temperature}}$. If *rho* is greater than the probability of the random number, the new solution is accepted, otherwise the old solution is kept and the temperature is decreased. Due to the nature of this calculation, the probability of choosing a worse temperature decreases as the temperature cools meaning the solution can be seen to approach the optima as temperature decreases.

Algorithm 3 Simulated Annealing

```
while temperature > minimum temperature do
    solution ← random vertex
    if current cover < solution cover then
        cover ← solution
    else
        probability ← random probability
        if rho > probability then
            cover ← solution
        end if
    end if
    temperate ← cool
end while
```

IV. COMPLEXITY

The MVC problem belongs to the set of NP-hard problems. Each algorithm has several helper methods which are called multiple times on each trial.

A. Generate Adjacency List

This method constructs an adjacency list for the graph being tested. Its execution time is V where V is the amount of vertices present. A total complexity of $O(V)$ can be expected for this method.

B. Remove Edge

To remove an edge the adjacency list is traversed and the edge is removed from each vertices set of edges. This operation is carried out on $V = vertices$ and $E = edges$ amount of times giving a total complexity of $O(VE)$.

C. Minimise Cover

To minimise a solutions cover the adjacency list is traversed $V = vertex$ times. Upon each traversal random edges and their incidents are removed from the potential solution. A total complexity of $V \times VE$ can be expected for this method and is of order $O(V^2E)$.

D. Visited

This method traverses the set of edges on some vertex to ensure we aren't adding a vertex to the cover that already exists. It's runtime is dependant on the amount of edges present for that vertex and is of order $O(E)$.

E. Check Cover

A validation method was required to ensure a found cover is in fact a cover. This method removes all edges that are incident in the cover from the adjacency list. It then checks the adjacency list to see if there are any edges present. If a cover exists, all nodes left will have no edges. This method is executed $C = cover\ size$ amount of times on all $v = vertices$ present, giving a bounded complexity of $O(CV)$.

F. Simulated Annealing

The Simulated Annealing algorithm is bounded by $T = temperature$ and its cooling factor α. Upon each temperature traversal it calls the minimise cover and visited methods. The complexity for this algorithm can be described as $T\alpha(V^2E + E)$ and is bounded by $O(T\alpha V^2)$.

G. Helper Function

This function executes the above methods for each algorithm. It is executed $t = trial$ amount of times. Its complexity is bounded by the most complex method it calls, being Simulated Annealing. Therefore, a total complexity in polynomial time can be expected with a run-time of order $O(tT\alpha V^2)$.

The space complexity for each algorithm is comparable. An adjacency list holding $V = vertices$ and $E = edges$ is initially created. Each algorithm then holds a potential cover which is of size $n < V$. Finally, a set of ranks is also created which holds $V + V$ elements. A total space complexity of $3VE + n$ can be expected an is of order $O(VE)$.

V. EXPERIMENTAL RESULTS

The following results were obtained using Apple silicon M1 Max chip with stage four compiler optimisations and no cut-off restrictions.

Through experimentation it was confirmed that the next-best greedy algorithm was susceptible to local minima of 785, 786 and 787 for the Brock collection. There were a few outliers namely a local minima of 783 and 791.

Graph	Target	Average CPU Time (seconds)	Best Cover	Average Cover
brock800_1.clq	777	0.0669	784	786.31
brock800_2.clq	776	0.06638	784	786.16
brock800_3.clq	775	0.06655	785	786.3
brock800_4.clq	774	0.06662	783	786.1
C2000.9.clq	1922	0.16573	1943	1948.85
C4000.5.clq	3982	5.22589	3986	3988.15
MANN_a45.clq	691	0.00216	700	702.93
p_hat1500.clq	1488	0.70672	1491	1493.4

TABLE I
Next-Best Greedy Local Search Results

Similar to the next-best greedy approach, the randomised local search algorithm also got stuck in the same local minima. It results were almost on par with the next-best greedy algorithm however it was outperformed in all test cases except for brock_2.

Graph	Target	Average CPU Time (seconds)	Best Cover	Average Cover
brock800_1.clq	777	0.06338	784	786.3
brock800_2.clq	776	0.06283	783	786.14
brock800_3.clq	775	0.06346	784	786.18
brock800_4.clq	774	0.06271	784	786.43
C2000.9.clq	1922	0.15088	1943	1948.58
C4000.5.clq	3982	4.8254	3986	3988.22
MANN_a45.clq	691	0.00217	704	794.98
p_hat1500.clq	1488	0.69113	1491	1493.61

TABLE II
Randomised Local Search Results

It can be seen the both the randomised and next-best greedy approaches did not perform very well compared to the DIMACS benchmarks[1]. Both algorithms are comparable in terms of speed and accuracy, with the next-best greedy approach being slightly more reliable. It is speculated that this is due to the design of the graphs used. They are made to break greedy type algorithms and ensure they get stuck in local minima, rather than find a global minima. It also confirms the hypothesis of stated previous works [2] [3].

Graph	Target	Average CPU Time (seconds)	Best Cover	Average Cover
brock800_1.clq	777	2.14652	782	784.04
brock800_2.clq	776	2.13537	782	784.15
brock800_3.clq	775	2.16887	781	784
brock800_4.clq	774	2.14215	783	784.14
C2000.9.clq	1922	8.97447	1941	1944.01
C4000.5.clq	3982	240.951	3985	3986.59
MANN_a45.clq	691	0.186	703	704.44
p_hat1500.clq	1488	37.5072	1490	1491.6

TABLE III
Simulated Annealing Results

The Simulated Annealing algorithm out-performed both the local search algorithms in all test cases other than the MANN graph. In all other cases it found tighter average results confounded to one or two local minima, rather than a spread of several as seen in the local algorithms. The lowest covers found were also lower than the covers found by the local search algorithms.

VI. CONCLUSION

It can be seen through the above results that a greedy local approach is insufficient when searching for a MVC. The Brock compliment graphs were individually designed to break greedy optimisation algorithms. As such, non optimal results were seen and can be expected for any greedy type algorithm for this graph collection.

The Global approach provided much more accurate results at the cost of computational complexity. This increase in computational complexity can be expected when optimisation algorithms start to approach the benchmark or global minima due to the NP-hardness of the problem. Further experimentation into this algorithm would prove beneficial. It was noted that as maximum temperature increased as well as a cooling factor increase the algorithm began to find lower local minima.

When conducting research into the Minimum Vertex Cover optimisation problem, or any of its variants, a greedy local approach should be avoided and an extension of the global stochastic approach used in this research should instead be considered.

While the Simulated Annealing algorithm provided close to benchmark results the final few minimum covers are always the hardest to get. When finding these final minimal covers advanced techniques such as Phase Local Search (PLS) should be instead considered [5].

VII. REFERENCES

[1] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015.

[2] S. C. Dimri, K. C. Purohit, and D. Pant, "A greedy approach based algorithm for the vertex cover problem," *International Journal of Scientific Engineering Research*, vol. 4, no. 3, 2013.

[3] W. Gao, T. Friedrich, F. Neumann, and C. Hercher, "Randomized greedy algorithms for covering problems," 2018.

[4] K. Kumar, N. Maaroju, and D. D. Garg, "Complete algorithms on minimum vertex - cover,"

[5] W. Pullan, "Optimisation of unweighted/weighted maximum independant sets and minimum vertex covers,"