CSE 6140 Final Project

BOWEN LI, MIAN WU, YUCHEN WANG
GEORGIA INSTITUTE OF TECHNOLOGY, ATLANTA, GA

ABSTRACT

The Minimum Vertex Cover (MVC) problem is a classical NP-hard optimization problem that is often used in computational complexity theory as a starting point for NP-hardness proofs [1]. In this project, three algorithms: Branch and Bound, Heuristics method with approximation, and Local Search are implemented to find optimal solution to the problem. We chose 2 local search algorithms: Simulated Annealing and Hill Climbing. We evaluated the performances of all the 4 algorithms based on their execution time, result (size of returned vector set) and relative error.

1 Definition

The Minimum Vertex Cover problem can be described as: Given an undirect graph $G = (V, E)$, A vertex cover V' of G is a subset of V such that $uv \in E \Longrightarrow u \in V' \wedge v \in E'$. A minimum vertex cover is the vertex cover with smallest size of $|V'|$.

2 Related Work

For the Branch and Bound algorithm, a brief description was given in the work of Lüling and Monien [2]. Zhong mentioned some common branching rules such as vertex selection, mirror branching, satellite branching and packing branching. The pseudo-code in his article gives details on how the algorithm works [3].

A modified heuristic approximation algorithm by adding vertex with highest degree and removing its neighbor, was mentioned that performs better than greedy algorithm [4].

In terms of local search, Selman and Gomes states that Hill Climbing might be better at finding local optimum instead of global optimum, however, some strategy can be applied to improve its performance. Backtracking technique can be used to prevent the search reaching an undesirable state and by allowing a downhill step, it may help the search get out of plateau [5]. In 1983, Kirkpatrick, Gelatt and Vecchi published on solving the Traveling Salesmen Problem with a simulated annealing algorithm. His article demonstrates the feasibility of this approach and the fact that the slower the annealing temperature drops, the closer the results are optimal [6].

3 Algorithm

3.1 Branch and Bound

3.1.1 description

The branch-and-bound algorithm is essentially an evolution of the enumeration algorithm, which discards the search in a sub-tree if it can determine that it is impossible to find a better solution than the current one, which is called "pruning".

In our implementation, we applied depth-first-search techniques and kept updating the upper bound, lower bound, visited node and visitededges. Each node represents a sub-vertex which is a subproblem, each subproblem has its lower bound and we won't proceed if lower bound exceed the upper bound. Initially, we set the upper bound to be the result of our Approximation method and the lower bound to be the half of the upper bound. The time complexity of Brand and Bound is $O(2^n)$ since for each vertex, we are going to decide whether it is going to be included in the vertex cover.

3.1.2 pseudo-code

ALGORITHM 1 BRANCHANDBOUND

```
BnB(G)
If G is empty
    Return the current solution X
Else if LB + X > best solution Y
    Return X
    If X < Y do
        Y← X
G1, G2 ← branch(G)
Y←BnB(G1)
Y'←BnB(G2)
If Y' <Y
Y←Y'
Return Y
```

3.2 Heuristic

3.2.1 description

A modified greedy approach was used for our heuristic algorithm. First, the vertex cover set is initialized as an empty set. Then choose the node with the highest degree to add to the set and delete the node and its neighbor from the edge set. It is believed that this modified version is better than the classic greedy algorithm used for MVC [4].

It is still a 2-approximation algorithm since just like greedy algorithm, we need to go through every edge till there is not edges left that is unvisited. Since the algorithm select the node with the highest degree each time, we believe the time complexity of the algorithm is O(logV). The space complexity should be O(V) if we need to get the exact set of vertex cover.

Although it might not give us the most accuracy result by heuristic approximation, the result returned by this algorithm is still more reliable compared to traditional greedy algorithm. In other words, it is guaranteed to be better than greedy. Also, its time complexity is quite low among algorithms to solve MVC, which means it could be a good choice to use the algorithm to calculate the lower bound when design other algorithms.

3.2.1 pseudo-code

ALGORITHM 2 HEURISTICS APPROXIMATION

```
1   VC ← ∅
2   E' ←E[G]
3   while E' != ∅ do
4       Pick the node v with the highest degree
5       VC ←VC ∪{v}
6       remove v and all edges incident to v in E'
7   return VC
```

3.3 Local Search

3.3.1 description

Stimulated annealing. The algorithm would accept the worse result at a probability p each iteration, which is a practical way to reach the global optimum. The higher the temperature is, the larger p we are able to get. As a result, it is more likely to accept the worse case. In our code, we choose the stop criteria as the running time. The vertices of the graph are copied into not Visited and temp in order to record the undecided nodes and the temporary candidate vertex cover. A vertex v among vertices that has not been visited will be picked, and there are two neighbor solutions we can choose: a set without v and a set with v inside. If v is already inside, remove it from the set, and if not, add it to the set. Then we check if the set is a valid vertex cover, and if it is, we record this candidate solution and start to find other candidates. Otherwise, it stays in the inner while loop, modifies the temporary set by removing/adding an unvisited vertex, and it may go back to the previous result until another potential solution is found. Time complexity of this algorithm is O(E) per iteration. The worst case is that we have to check if the vertex cover is valid every time.

Although backtracking strategy is utilized in Stimulated Annealing, the algorithm is still not guaranteed to return the optimal solution. However, by setting a probability to increase the number of vertices that come back to the set, it is at least able to jump out of the local minimum and approach the optimal solution.

Hill Climbing. In this algorithm, we start with the entire vertex set as the initial vertex cover and try to remove one node at a time from the current set to form a new valid set whose size is smaller than one or more nodes until the removal of any node results in an invalid vertex cover. In this case, the neighborhood is defined as the set of all nodes that are one node less than the current node. The order of deletion is from vertex with smallest degree to those with the largest. The time complexity of hill climbing is O(1) for each iteration, and for the whole program the time complexity is dependent of the cut-off time we set.

3.2.1 pseudo-code

ALGORITHM 3 STIMULATED ANNEALING

```
1   VC ←V[G]
2   Not-Visited ←VC
3   while Running Time < t do
4       Pick a node v in G randomly
5       Not-Visited ← Not-Visited - v
6       while Not-Visited != ∅ do
7           If v is not in VC do
8               VC ←VC + v
9           Else VC ←VC – v
10          Evaluate P
11          If isValidVertexCover(VC) do
12              Break;
13  return VC
```

ALGORITHM 3 HILL CLIMBING

```
1   VC ←V[G]
2   VC←Sort (VC)
3   while Running Time < t do
4       Remove the vertex v with the least degree in VC
5       IF !IsvalidVertexCover(VC) do
6           VC←VC+v
7           Select the next v in VC
8   return VC
```

4. Evaluation for Empirical results

CPU: Apple M1

4.1 comprehensive table

Table 1 in follows shows the comprehensive information for the three algorithms respectively. The running time is in unit s, VC is the minimum vertex cover solution generated by algorithms within the running time, and the Rela.Err is the relative error which is calculated by the equation (SOLUTION-OPT)/OPT

Dataset	Branch & bound			Approximation			Simu anneal			Hill Climbing		
\\\\\\\\\\	Time	VC	Rela.Err	Time	VC	Rela.Err	Time	VC	Rela.Err	Time	VC	Rela.Err
as-22july06	130	3315	0.36%	4.56	3307	0.12%	999	3311	0.21%	9.14	3306	0.10%
delaunary-n10	47.20	737	4.83%	1.88	766	0.14%	1.61	768	9.24%	0.095	736	4.70%
email	1.25	607	2.19%	0.045	605	1.85%	6.29	677	14.00%	0.09	604	1.68%
football	0.017	95	1.06%	0.001	96	2.12%	0.0115	98	4.48%	0.0005	95	1.06%
hep-th	30.46	3948	0.05%	2.14	3947	0.03%	171.06	4375	11.43%	3.48	3943	0.04%
jazz	0.244	158	0%	0.0026	159	0.63%	0.1539	164	3.80%	0.01	158	0%
karate	0.001	14	0%	0	14	0%	0.0022	18	28.57%	0.0	13	7.14%
netscience	0.9471	899	0%	0.97	899	0%	4.001	954	6.11%	0.16	898	0.11%
power	8.67	2267	2.91%	0.7	2277	3.36%	58.92	2573	12.25%	1.29	2277	3.36%
star	142.66	7366	6.72%	4.9	7374	6.74%	987.8	7277	5.43%	9.38	7373	6.82%
star2	318	4675	2.93%	4.07	4697	3.41%	183.17	5048	11.10%	7.93	4696	3.40%

4.1.1 Branch and Bound

The branch and bound algorithm may take a long time to complete the whole traversal search space (with exponential candidate solutions). If owing to the initial starting point, it will begin traversal with maximum degree node, which means most possible vertex to be vertex cover. In our experiment, the first time are quick which are less then running time of this algorithm is 350s. The second time took a long-time count in hours. The results all within 600 seconds. The BnB algorithms can find optimal solutions of 4 in 11 datasets which and a large relative error in 6.72% in star.graph.

4.1.2 Heuristic Approximation

The approximation method was reliable in all four algorithms. The worst 6.74% on star2 graph. This algorithm was implemented by modified search way such that in the worst case if Vertex need to spend O(maxdegree) to check if it is vertex cover. So it is quite fast.All run time are less than 5 seconds. Nice algorithm among this four.

4.1.3 Stimulated Annealing

We can only see the result may worse than approximation and hill climbing. Sometimes works well, and more evaluation in 4.2 and 4.3.

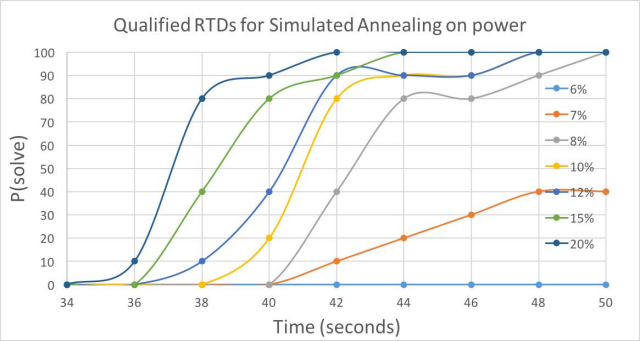
4.1.2 Hill Climbing

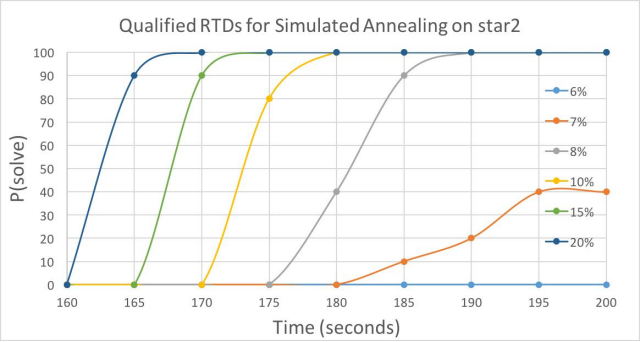
We can see this one is a good method on relative error and time. However, I think we need to do more evaluation on 4.2 and 4.3.

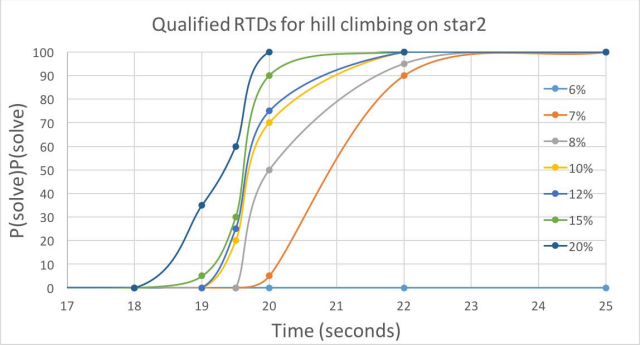
4.2 QRTDs of LS1 and LS2:

Based on multiple cut-off time experiments, we draw these graphs on power and star2.

Figure 1 to Figure 2 are LS 1 results. Figure 3 to Figure 4 are LS 2 results. Figure 5 is the box plots for randomized local search algorithms.

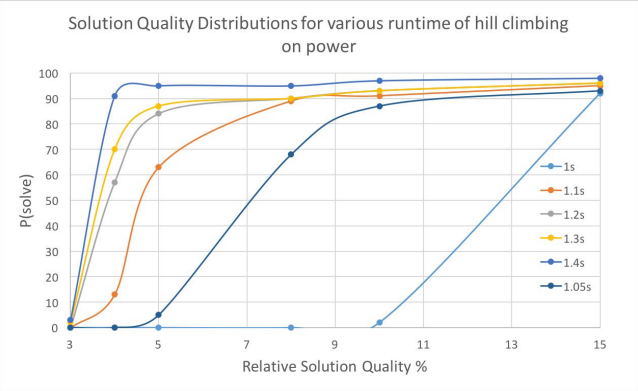




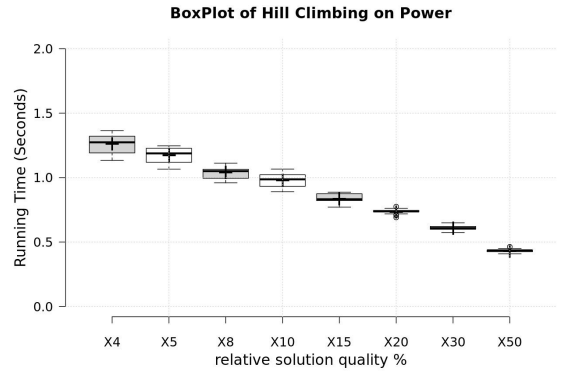


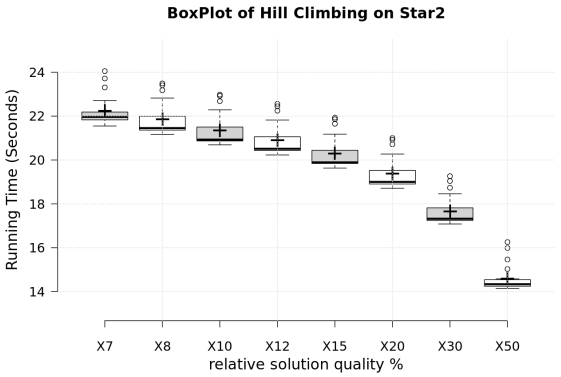
4.3 SQDs for LS 1 and LS 2

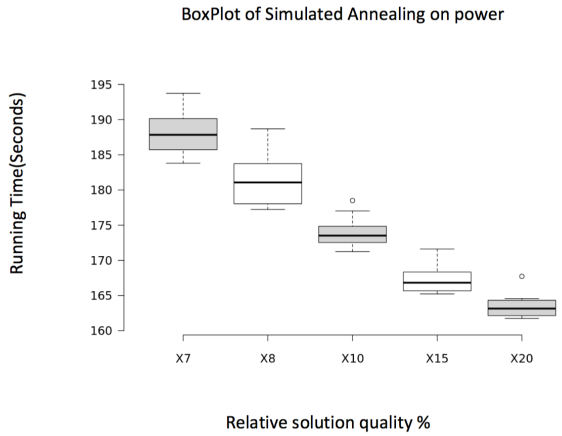
The figure 1 and figure 2 LS 2 hill climbing.

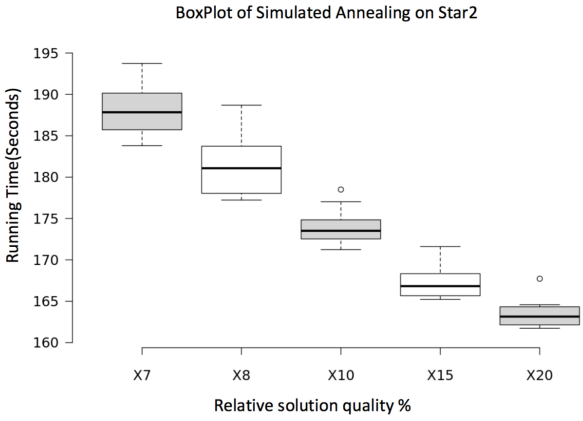


4.4 Box plots of Hill Climbing









The lower bound on the optimal solution quality of our approximation algorithm should be greater than the ½ of true optimum since it is a 2-approximation approach. As for Branch and Bound, the result return by should be the true optimum if the run time is less than the cut off time. However, in our tests, since it the running time it needs to finish the search is often much more than the cutoff time we set, the solution should be greater or equal to the true optimal.

5 Discussion

5.1 Branch and Bound

From the table, we find that the Branch and Bound algorithm gives the most accurate results although the running time grows exponentially with the increase of graph size, which indicates that the time complexity is $O(2^n)$. The running time for the largest graph is quite high, and we believe that is the nature of algorithm. To improve it, we might need to consider using a tighter lower bound. Other techniques such as parallel BnB may improve the speed as well.

5.2 Heuristic Approximation

The performance of our approximation algorithm stands out among those algorithms. It is a very fast algorithm with acceptable solution, as it only takes 4.56s to find the vertex cover of the largest graph with an relative error 0.12%. The result is surprising to us and we believe this is exactly why it is widely used in dealing with NP problems like minimum vertex coverage, traveling salesmen, etc. In terms of the time complexity, it should be O(logV) still since the running time only increase 3 times when the size of graph becomes 20 times of the previous data.

5.3 Local Search

Based on the results, the performance of two local search methods is not satisfying. Especially the Simulated Annealing method need much more time analyzing the largest graph and produces relatively high error. The result of hill climbing is good, with the largest test graph solved in 9s. Errors are relatively controlled, with less than 7% for all test graphs and average 3% error. This algorithm is an efficient way to find the minimum vertex cover, when accuracy is not strictly required.

6 Conclusion

In the project, we have applied four algorithms in order to find the optimal solution of the NP-complete problem Minimum Vertex Cover. They are Branch and Bound, Heuristic approximation, Simulated Annealing, and Hill climbing.

Among those algorithms, we can see that branch and bound algorithm gives us exact solution but also costs us lots of time. However, when the data is large, we don't get the optimal solution because getting the optimal solution will take too much time. We had to stop at cut-off time and get what we get. Approximation algorithm costs the least amount of time and generate solution with good quality. The two local search algorithms are also fast, the execution time depends on the cut-off time that we set beforehand, which will also influence their performances.

REFERENCES

[1] Richard M. Karp. 1972. Reducibility among combinatorial problems. Complexity of Computer Computations (1972), 85–103. DOI:http://dx.doi.org/10.1007/978-1-4684-2001-2_9

[2] R. Lüling and B. Monien. 1989. Two strategies for solving the vertex cover problem on a transputer network. Distributed Algorithms (1989), 160–170. DOI:http://dx.doi.org/10.1007/3-540-51687-5_40

[3] Zhong, Haonan. 2017 Engineering an Efficient Branch-and-Reduce Algorithm for the Minimum Vertex Cover Problem. *Senior Honors Theses*. 17 (2017) http://commons.colgate.edu/theses/17

[4] Sharad Singh, Gaurav Singh, Neeraj Kushwah. 2018. Optimal Algorithm for Solving Vertex Cover Problem in Polynomial Time *International Journal of Scientific & Engineering Research* 9, 8 (2018)

[5] Bart Selman and Carla P. Gomes. 2006. Hill-Climbing Search. *Encyclopedia of Cognitive Science* (2006). DOI:http://dx.doi.org/10.1002/0470018860.s00015

[6] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680. DOI:http://dx.doi.org/10.1126/science.220.4598.671