Final Course Project Report

(2017 Fall)

Name	Zizheng JIANG	**WATIAM**	z225jian
Student ID	20716960	**Email**	z225jiang@uwaterloo.ca

Introduction

1. Project input:

   As input, use the output of graphGen on ecelinux, I do this by generating graphs for |V| ∈ [5, 15] using graphGen program, in increments of 2. That is, graphs with 5, 7, 9, …, 15 vertices. It is designed for calculating the running time of the three algorithms, CNF-SAT, APPROX-VC1 and APPROX-VC2.

2. Running time and ratio measurement method:

   The running time of each algorithm is calculated by the function *pthread_getcpuclockid()*, I use three vectors to store the data after I get each algorithms' running time, which can make sure the running time of each algorithms' output in the right order.

   In order to calculate the ratio's value. I created three vectors (vector<unsigned> cnfsat, vector<unsigned> approxvc1 and vector<unsigned> approxvc2) to store the output of each algorithms. Then divide the size of approxvc1 and approxvc2 by the size of cnfsat, to get the answer ratio1 and ratio2.

   I put all the data into excel and used excel to calculate the average and standard deviation. Finally, using excel according to the data to draw line charts.

Analysis

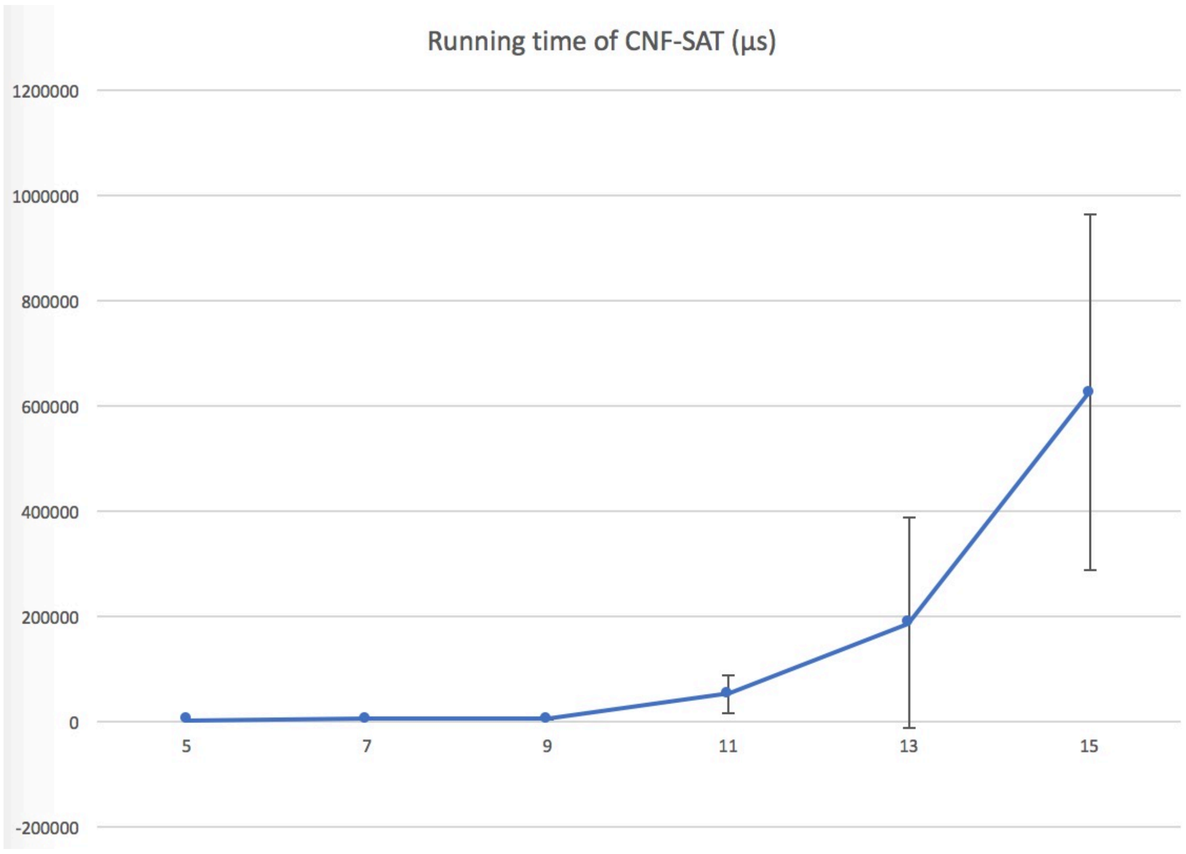

Fig 1. Running time of CNF-SAT

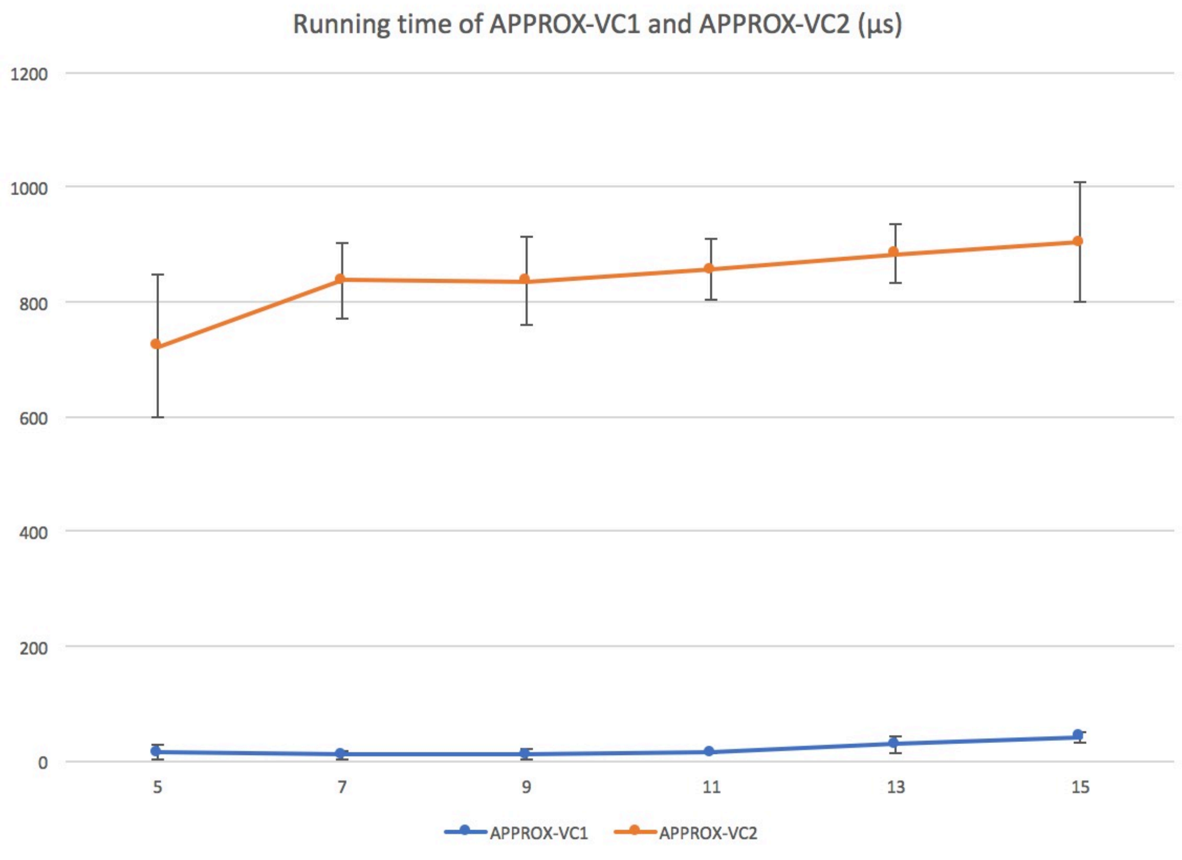

Fig 2. Running time of APPROX-VC1 and APPROX-VC2

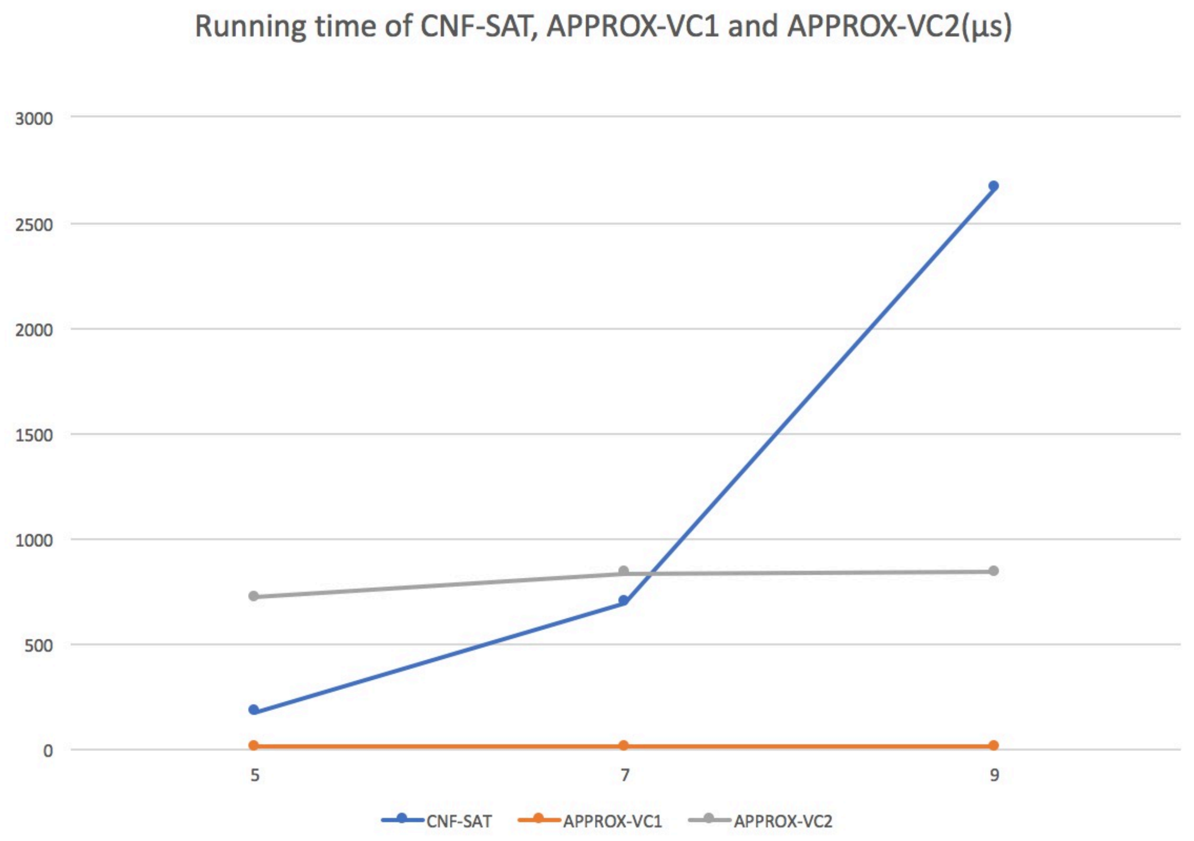

Fig 3. Running time of CNF-SAT, APPROX-VC1 and APPROX-VC2

Figure 1 is the running time of algorithm CNF-SAT, as we can see from the figure, as the number of vertices increases, the algorithm's running time growth is exponential. Based on a4_encoding.pdf, we know that the number of clauses is (k + n * ($_kC_2$) + k * ($_nC_2$) + |E|), which means the number of clauses increase in an exponential manner with vertex cover size. That's why the running time of CNF-SAT algorithm grows exponentially.

Figure 2 is the running time of algorithm APPROX-VC1 and APPROX-VC2, as we can see from the figure, as the number of vertices increases, APPROX-VC1's running time grows linearly, APPROX-VC2's running time grows linearly too. This is because with the vertex value grows linearly, the edges size grows linearly, which means the input grows linearly, APPROX-VC1 and APPROX-VC2 don't need to add clauses, APPROX-VC1 just need to pick the vertex with the highest degree and vertex2 need to pick an edge "randomly" and then remove all edges touch the two vertices. APPROX-VC2's running time is much higher than APPROX-VC1's running time. This is because I use "dev/random" in APPROX-VC2 to let the result of this algorithm is different with the same input.

As we can see in Figure 1, with the number of vertex grows, the running time's standard deviation is also growing. Maybe because the minisat based on some kind of way leads to this result.

As we can see in Figure 2, we can also see that the standard deviation of APPROX-VC2 is higher than APPROX-VC1, this is because APPROX-VC2 pick edges randomly, if it happens to pick an edge with the largest degree every time, the running time of APPROX-VC2 may very small, but if it happens to pick an edge with the smallest degree every time, the running time can be very large.

That's why APPROX-VC2's running time standard deviation is bigger than APPROX-VC1's running time standard deviation.

As we can see in Figure 3, when the number of vertices is 9, CNF-SAT's running time is much higher than the other two algorithms, but when the number of vertices is 5 or 7, CNF-SAT's running time is between the other two algorithm, which means we can choose which algorithms to use based on the number of vertices.

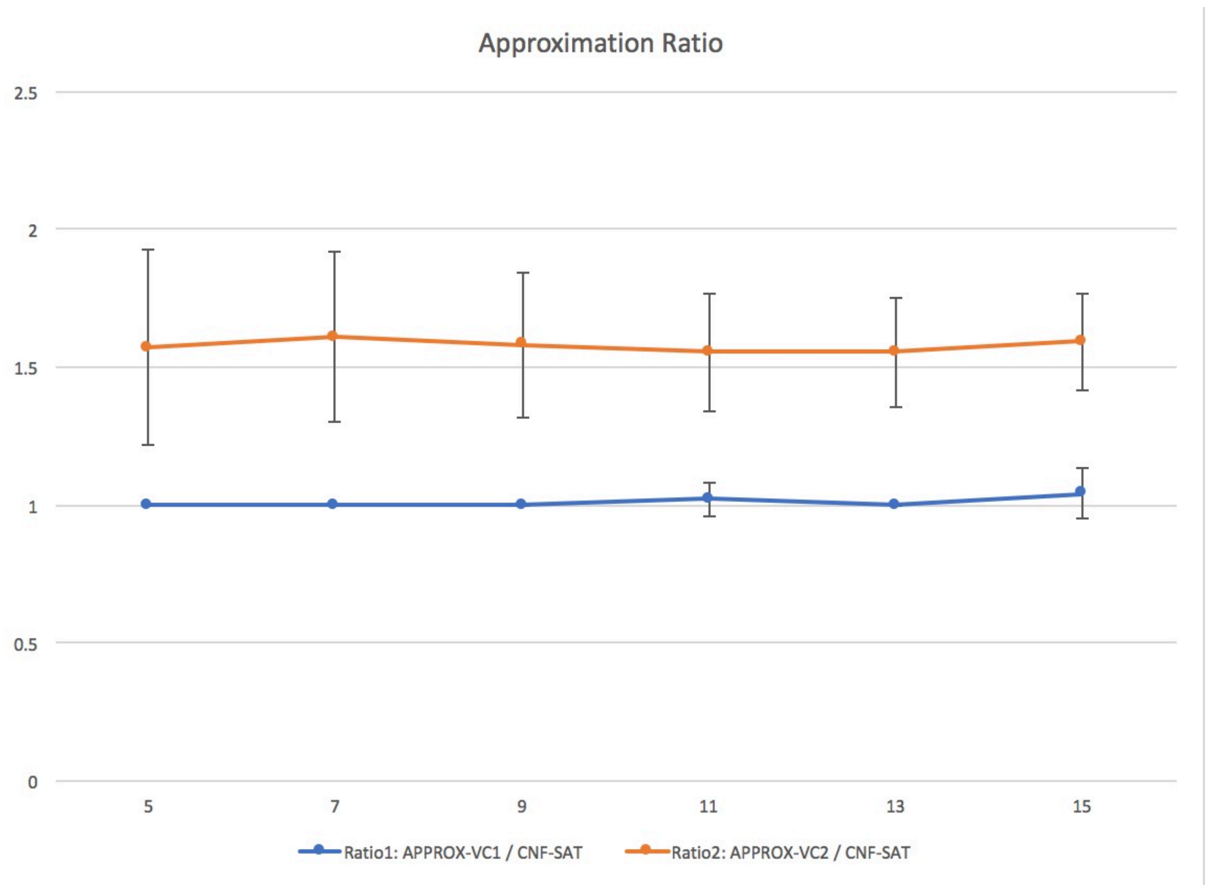

Fig 4. Approximation Ratio

Figure 4 shows the approximation ratio. Ratio1 is the APPROX-VC1's size divided by CNF-SAT's size. Ratio2 is the APPROX-VC2's size divided by CNF-SAT's size.

Based on a4_encoding, we can make sure that CNF-SAT's answer is the right answer.

Ratio 1 is very close to 1. Actually, APPROX-VC1 is a kind of greedy algorithm to solve the vertex cover problem. So, it can't always get the minimum vertex cover. Because after pick the edge with largest degree, the rest graph may not the perfect sub-structure. Ratio 1 is larger than 1 when the number of vertices is 11 and 15, this is because the answer of APPROX-VC1 is not the minimum vertex cover. As for the standard deviation, APPROX-VC1's standard deviation is 0 when ratio 1 equal to 1 (which means the answer of APPROX-VC1 is the minimum vertex cover answer).

APPROX-VC1's standard deviation is lower than APPROX-VC2 because although greedy algorithm can't always get the best answer, it can get an answer very close to the right answer.

Ratio 2 is higher than ratio1 but it largest value is 2. Because if we want to get an answer from APPROX-VC2, APPROX-VC2's answer must cover all the vertices in minimum vertex cover. Suppose minimum vertex cover of a graph is [0,1], APPROX-VC2's answer may be [0, 1] or [0, 1, x, y] (x, y are vertices in the graph). APPROX-VC2's standard deviation is also very high because I randomly pick an edge each time until edge's set is empty. Randomly pick edges may get minimum vertex cover's answer or may get an answer twice size of minimum vertex cover's size.

Conclusion

According to the running time and approximate ratio, we can decide when to use what algorithm. All three algorithms have good performance depend on the specific graph.

According to running time, when the number of vertices is smaller than 9, CNF-SAT algorithm has good, but when the number of vertices is larger than 9, CNF-SAT's running time is very high compare to other two algorithms. APPROX-VC1 algorithm is really fast even if the input is very large. APPROX-VC2's running time is between CNF-SAT and APPROX-VC1, when the number of vertices becomes large, it's good than CNF-SAT.

According to approximation ratio, CNF-SAT is guaranteed to have the minimum vertex cover answer, but it need more time to calculate the answer when vertices larger than 9. APPROX-VC1's answer is very close to CNF-SAT. APPROX-VC2's answer would never bigger than 2 times of CNF-SAT.

Based on above, we know that when we solve large scale vertex cover problem, APPROX-VC1 is a good option if we don't need to get minimum vertex cover answer. When we solve small scale vertex cover problem, CNF-SAT is the best solution.