Comparison of Meta-Hueristics performance in solving MCP

Elmabruk Laias
Computer Department,
Faculty of Art & science/Gamens/,
University of Benghazi, Benghazi, Libya

Abdalla M. Hanashi
Computer Engineering Department, Faculty of Engineering,
Sabratha University, Sabratha, Libya

***Abstract*- This paper evaluates the performance of three hybrid meta-heuristic algorithms in solving maximum clique problem. The graph traversal problem can be found in many applications making it one of the most studied NP-hard.**
In recent researches, the performance of each one of the three was measured separately via DIMACS benchmarks, however this study compares the results of implementing the three algorithms to find the maximum clique of 30 DIMACS graph, where it was found that Simulated Annealing and Genetic Algorithm did better than Tabu Search, in addition, results show that GA only could find the optimum of certain benchmarks, while SA only could find the optimum of another set of benchmarks. Also, the study investigates combining two distinct local optimizers with genetic algorithm.

Keywords- component; Maximum Clique Problem;Meta heurisrtics ;Local Optimization.

I. INTRODUCTION

In mathematics and computer science, graphs are used to represent the bilateral relations between objects ,and it is defined as follows:
A graph **G = (V, E)** contains set of vertices **V** and edges **E**, vertices are the objects and relation between them is represented by edges.
With regards to relations (edges) graphs can be classified into two main categories, directed and undirected graphs.
A subgraph **K** in a graph **G** is a clique (complete graph). if every two vertices in **K** are connected to each other.
Finding maximum clique of a graph are among the most studied graph search problems, as it can found in several real-life applications such as networking, social media analysis…etc.
In graph theory, a maximal clique problem is concerned with finding a clique that cannot be extended to a larger clique while maximum clique problem concerns finding the largest subgraph, such that each two vertices are connected to each other.
For instance, a graph G which contains 7 vertices and 16 edges, as shown in (figure 1.a), has a clique **k** of 4 nodes (1,2,4,5) shown in figure (1.b), and maximal clique of 4 nodes (3,4,5,7) shown in (figure 1.c), and maximum clique of 5 nodes (1,2,3,4,5) shown in (figure 1.d).
For such a problem, it is not expected to have an exact polynomial time algorithm to find the maximum clique of large graphs [3], thus, different heuristic, meta – heuristics, hybrid approaches were implemented and gave good results, many studies were conducted to evaluate algorithms’ performance and compare the different implantations, thanks to DIMACS graph benchmarks that was used in most researches to compare algorithms’ quality (maximum clique found and time needed to find it)

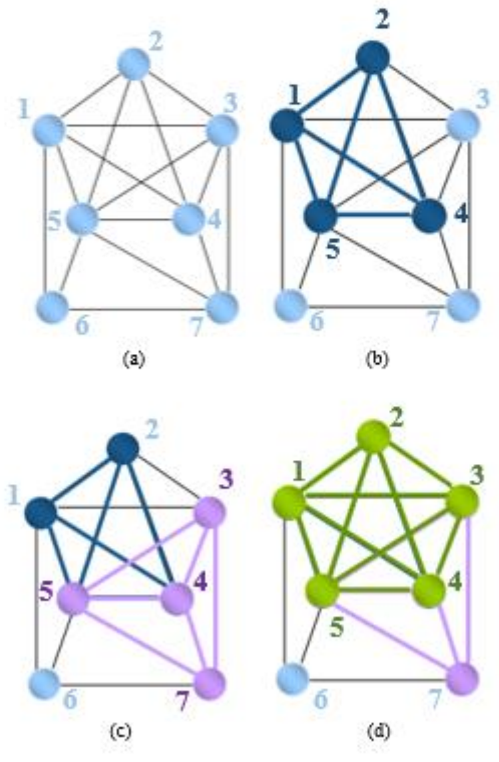


Figure 1. MCP Preliminaries

II. Meta-Hueristics Algorithms

A. *Related Work (Literature)*

Since 1970 various algorithms were introduced to solve MCP, starting with exact algorithms various branch-and-bound based algorithms guaranteed to find the optimum solution, however, due to the increased complexity of the problem, exact algorithms may require a huge computational power to find the optimum solution for a relatively large graphs, which makes such methods applicable only to graphs of limited number of nodes.

To find the optimum solution for larger problems in a feasible time, heuristic, meta-heuristics and hybrid approaches were devised, generally, meta-heuristic algorithms were designed to escape local optima that the algorithm may stuck in when started with a poor initial solution.

Meta- heuristic algorithms, have been used to solve the many graph traversal problems including maximum clique, however, few offered very good results.

In simulated annealing a solution will be randomly generated to be the current solution, the algorithm will then perturb that solution to get the next solution, both solution will be evaluated by a cost function, if the next solution is better, the algorithm will accept it, however if the current is better the algorithm accept it with a certain probability that drops exponentially as the algorithm proceeds.

Different simulated annealing implementations to solve graph search problem can be found,

In [1] simulated annealing was used to solve [clique problem] and it was found that [simulated annealing did better than heuristics] and in [2] simulated annealing was combined with a greedy heuristic to solve graph coloring problem, in these implementations SA outperformed greedy heuristics.

In [3] pure version of simulated annealing used to solve MCP, algorithm got the maximum clique for most benchmarks and better than best known for some benchmark, however, the algorithm didn't maintain a high rate for finding the maximum clique of many benchmarks.

To solve MCP, Genetic Algorithm has been implemented in various ways, hybrid Genetic algorithms where GA is combined with a local search (greedy algorithm) such as HGA [5]. In [6] [MARCHIORI E] used a similar technique.

In other implementations extending cliques algorithm were introduced, in [7], cliques will be found via GA and extended by a helper function.

In [9] [BHASN H and MAHAJAN R] used single point crossover and mutation, then in [10] implemented hybrid genetic algorithm and used roulette wheel section but no experimental results were given.

In [13], multiple fitness functions were applied to evaluate the chromosome, in [12] local search was used with GA and chromosome mutation process was done with the help of probability model, both [12] and [13] provided better results in terms of quality and time.

Like genetic algorithm different implementation of tabu search can be found in the literature, tabu search will take advantage of memory to store the places visited by the algorithm in search space to avoid visiting them again for a certain interval defined by the algorithm (Tabu Tenure), which will diversify the search process.

The algorithm will start with a current solution and goes to a neighbor, this neighbor will be recorded in tabu list so that for next iterations this neighbor will be prohibited for a certain interval, unless this neighbor met aspiration criteria which is defined by the algorithm

In addition to tabu list, frequency list, and other parameters can be added to algorithm to give better results
Generally, Tabu search had, good performance in solving graph problems, in [14] and [16], got maximum clique for most benchmarks. A comparison between simulated annealing and tabu search in solving graph coloring problem can be found in [15]

B. ***Implementaions :***

In this research, three meta-heuristics algorithms combined with local search algorithm were used, the local optimizer will extract clique from a given solution and try to improve it by making it larger.

1. General model:

The model consists of both meta-heuristics and heuristics algorithms (figure 2), while meta-heuristics algorithm is searching space for the global optimum, the local optimizer tries to help meta-heuristics algorithm in converging faster.
generally, meta-heuristics implementation will be as follows:

- Random Solution(s)
- Optimize Solution(s)
- Fitness Calculation
- MH: while: (Condition):
- Find neighbors via exploration and exploitation
- Optimize solution(s)
- Evaluate solution(s)
- Save the Best Solution

That means that solution will be locally optimized after each perturbation process.

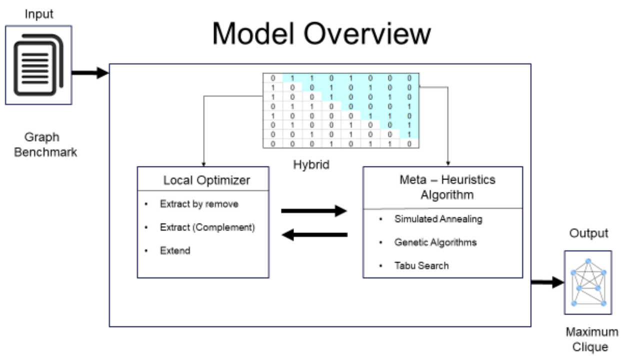


Figure 2. Hybrid Model

First of all, the graph file (benchmark) will be converted to adjacency array which will provide data for both meta-heuristic and heuristic algorithm.
The meta- heuristic algorithm will start with a solution, perturbate it to get a neighbor solution, after perturbation the solution will be locally optimized then the algorithm will continue normally with a local optimization step after each perturbation.
The local optimizer will extract a clique from a solution then try to extend it to a larger clique
Extract process can be done using various heuristic functions, in this research two extract function were applied, the first one (figure 3) will obtain a clique from a subgraph by removing nodes with lowest degree and check whether the subgraph is a clique or not.

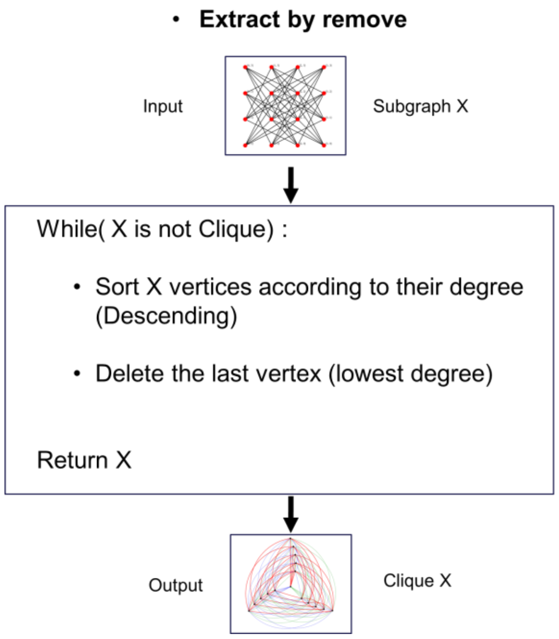
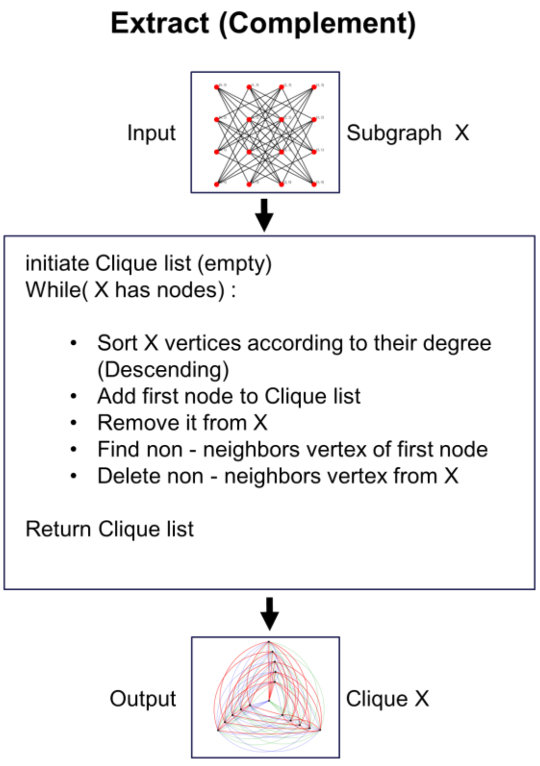

Extract by remove
Input
Subgraph X
While(X is not Clique) :
Sort X vertices according to their degree (Descending)
Delete the last vertex (lowest degree)
Return X
Output
Clique X

Figure 3. Local Optimizer (Extract by Remove)

The other one is based on the complement graph, the process is shown in (figure 4)



```
initiate Clique list (empty)
While( X has nodes) :
    • Sort X vertices according to their degree
      (Descending)
    • Add first node to Clique list
    • Remove it from X
    • Find non - neighbors vertex of first node
    • Delete non - neighbors vertex from X
Return Clique list
```

Figure 4. Local Optimizer (Extract via Complement)

2. Solution Representation & Evaluation:

A solution is represented by a list of ones and zeros with (n) length (number of nodes), where one at index (i) of the list, means that subgraph has node (i), while zero means node(i) is not included:

solution _x = [1 0 1 0 0 0 0 1 0 1 1 0 1 ………… n]

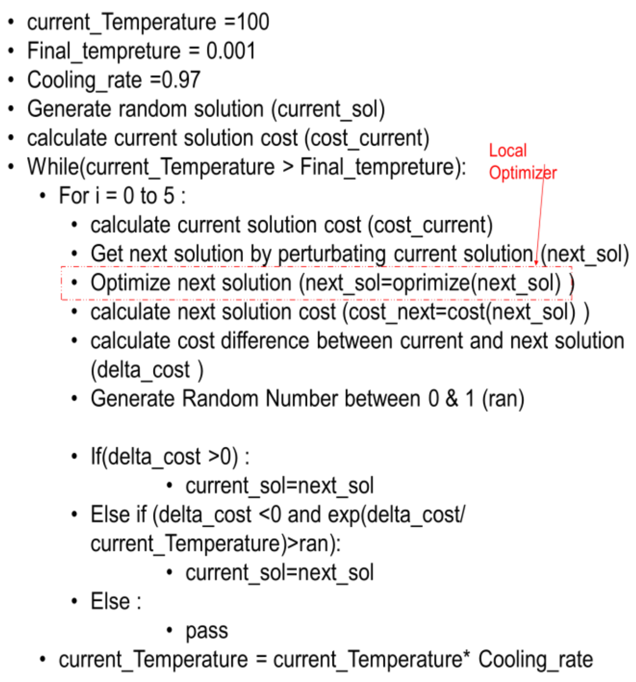
Because solutions are guaranteed to be a clique after applying the local optimizer, solution evaluation can be done by just counting the number of nodes (number of ones in the list):

fitness(solution _x)= $\sum_{k=0}^{n}$ solution _x(k)

3. Simulated Annealing

In simulated annealing, swapping two vertices were used as exploitation (local search) element, while flipping was used as exploration element (global search).

local search algorithm (local optimizer) will optimize the solution after exploitation or exploring process (figure 5).


Local
Optimizer

```
• current_Temperature =100
• Final_tempreture = 0.001
• Cooling_rate =0.97
• Generate random solution (current_sol)
• calculate current solution cost (cost_current)
• While(current_Temperature > Final_tempreture):
    • For i = 0 to 5 :
        • calculate current solution cost (cost_current)
        • Get next solution by perturbating current solution (next_sol)
        • Optimize next solution (next_sol=oprimize(next_sol) )
        • calculate next solution cost (cost_next=cost(next_sol) )
        • calculate cost difference between current and next solution
          (delta_cost )
        • Generate Random Number between 0 & 1 (ran)

        • If(delta_cost >0) :
                • current_sol=next_sol
        • Else if (delta_cost <0 and exp(delta_cost/
          current_Temperature)>ran):
                • current_sol=next_sol
        • Else :
                • pass
    • current_Temperature = current_Temperature* Cooling_rate
```

Figure 5. Simulated Annealing Pseudo Code

4. Genetic Algorithm

Like simulated annealing, heuristic algorithm will be applied when a new chromosome is generated to guarantee that this new chromosome is a clique, however, in GA implementation two local optimizers were used (extract by remove) and (extract by complement) as follows:

- Generate random variable between 0 and 1
- If random variable < predefined rate:

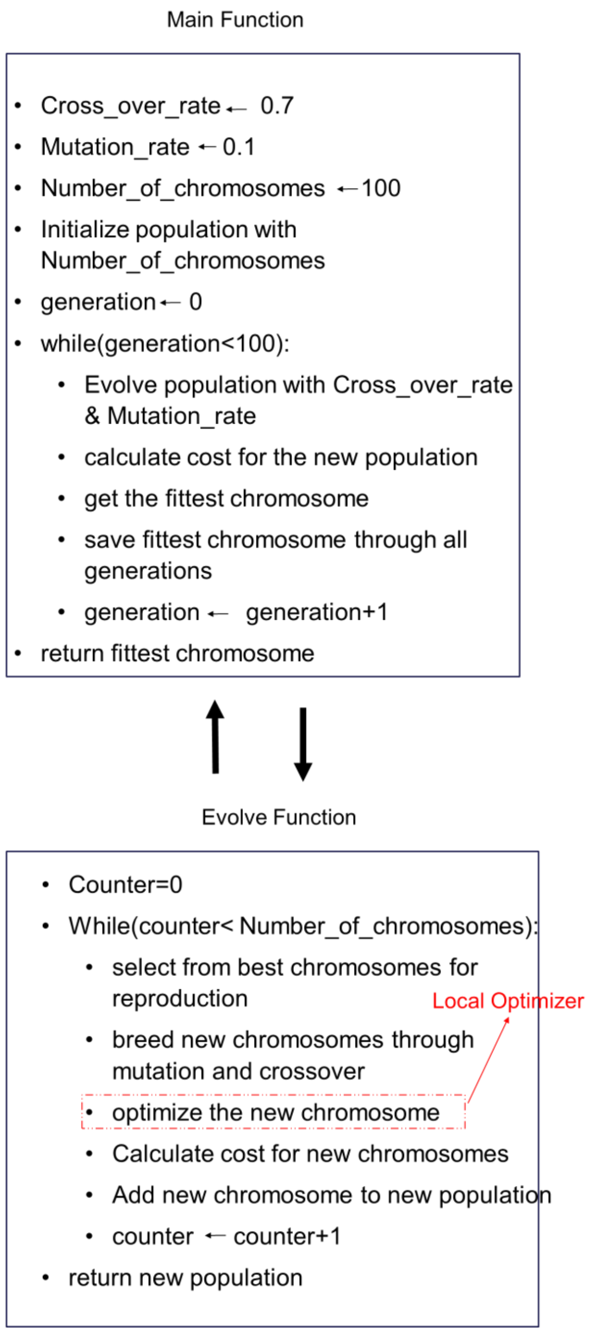
Use extract by remove

- Else:

Use extract by complement

both single point and two-point crossover were applied, in addition both flip bit and inversion mutation were tested.

In this implementation k-tournament selection was used to get parents from a current generation to breed new chromosomes for the next generation (figure 6)



```
Main Function

• Cross_over_rate ← 0.7
• Mutation_rate ← 0.1
• Number_of_chromosomes ←100
• Initialize population with
  Number_of_chromosomes
• generation ← 0
• while(generation<100):
    • Evolve population with Cross_over_rate
      & Mutation_rate
    • calculate cost for the new population
    • get the fittest chromosome
    • save fittest chromosome through all
      generations
    • generation ← generation+1
• return fittest chromosome

Evolve Function

• Counter=0
• While(counter< Number_of_chromosomes):
    • select from best chromosomes for
      reproduction
    • breed new chromosomes through
      mutation and crossover
    • optimize the new chromosome
    • Calculate cost for new chromosomes
    • Add new chromosome to new population
    • counter ← counter+1
• return new population
```

Figure 6. Genetic Algorithm Pseudo Code

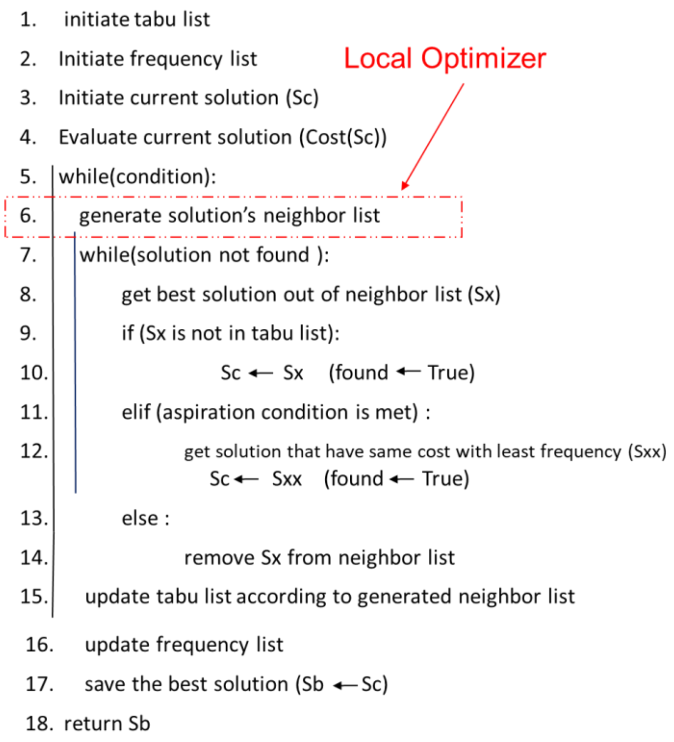
In some experiments two mutation were applied with a certain rate, which means that some of the chromosome will be mutated by the first mutation operator, the others with the second one, depending on a random variable.

5. Tabu Search

In tabu search, the algorithm will start with a random solution, then tries to find neighbors for this solution.

After generating a neighbor list, local search algorithm maintains having a clique through extracting and extending processes.

The best solution of the neighbor list will be considered as the next solution however some conditions will be applied. If this best solution is recorded as Tabu, will not selected as a next solution
If it is in tabu list but its cost is the best so far, aspiration will take the solution that have the best cost and least frequency (frequency list).



```
1.  initiate tabu list
2.  Initiate frequency list
3.  Initiate current solution (Sc)
4.  Evaluate current solution (Cost(Sc))
5.  while(condition):
6.      generate solution's neighbor list
7.      while(solution not found ):
8.          get best solution out of neighbor list (Sx)
9.          if (Sx is not in tabu list):
10.                 Sc ← Sx    (found ← True)
11.         elif (aspiration condition is met) :
12.                 get solution that have same cost with least frequency (Sxx)
                    Sc ← Sxx   (found ← True)
13.         else :
14.                 remove Sx from neighbor list
15.     update tabu list according to generated neighbor list
16.     update frequency list
17.     save the best solution (Sb ← Sc)
18. return Sb
```

Figure 7. Tabu Search Pseudo Code

III. RESULTS AND COMPARISONS

A. Results:

The algorithms were implemented on python and run on an Intel® Core™ i7-6600U CPU 2.66 GHz, experiments included 30 DIMACS graph benchmark ranging from 64 to 776 nodes.
Table (1) shows the experimental results of the three meta – heuristics algorithm, simulated annealing implementation tests were done using these factors:

- initial temperature =100,
- final temperature = 0.001
- geometric cooling rate = 0.997

Simulated annealing found the optimum for (25) out of the (30) which accounts for (83%) of graphs, and near optimum for another (3) graphs (10%).
Genetic algorithm implementation included using two local optimizers, one is based on (extract by remove) algorithm and the other uses (extract by complement), this means that some chromosomes will be converted to clique by the first local optimizer and other chromosome will be converted by the second local optimizer depending on a generated random variable.
These parameters were used in most GA experiments:

- population size =100,
- maximum number of generations = 100
- nutation rate = 0.1
- cross over rate = 0.7

In K-tournament parent selection process, k value can be determined by population size, for instance if the size of population is (100), k will be (5).
Like simulated annealing GA found the maximum clique for (25) out of (30) graph benchmark, however, GA found the optimum for a different (25) benchmarks, which means that some benchmarks' maximum clique can be found either by GA or SA. For instance, GA only can find the optimum of (p_hat300-3) while SA only can find the optimum of (brock200_1).
In GA experiments showed that tuning the coefficient that controls the local optimizers had an effect on result. GA found the optimum of (san200_0.9_3.clq) when this coefficient was (0.77), and it found the optimum for (p_hat300-3.clq) when this coefficient was (0.9).
In Tabu search implementation, frequency list and aspiration criteria were used. if the best solution of neighbor list is Tabu, TS algorithm will check aspiration criteria, the aspiration used in this implementation is as follows:
(if solution has the best cost so far):
↳ get solution that have same cost with least frequency

In each cycle both Tabu list and frequency list will be updated, Tabu Tenure value can be determined by the root square of graph’s nodes (number of nodes)
TS algorithm results was not as good as SA and GA .TS found maximum clique for only (20) out of (30) which accounts for (66%). Overall, the three algorithms found the optimum for 90 % of tested benchmarks and near optimism for the rest. GA found better than best known solution for (san200_0.9_3) benchmark.

B. Comparisons:

Compared to Simple Simulated Annealing Algorithm [3], the proposed simulated annealing gave better results in most benchmarks and maintained higher rates, however, it took more time to find the optimum due to using a local optimizer.
In terms of finding the optimum, the proposed genetic algorithm had similar results to FGA [17], however, we didn’t find the maximum clique of (keller5) which is (27), though, the algorithm finds near optimum solution (26).
TS results was not as good as SA and GA, TS found the optimum of 66% only of the benchmarks, and compared to other implementations, proposed TS algorithm had lower performance [16].

TABLE I. Experimental Results

Benchmark	Nodes	BR	SA	GA	TS
brock200_	200	21	21	20	20
brock200_	200	12	12	11	10
brock200_	200	15	15	15	13
brock200_	200	17	16	16	15
brock400_	400	27	24	26	24
c-fat200-1	200	12	12	12	12
c-fat200-2	200	24	25	24	24
c-fat200-5	200	58	60	58	58
c-fat500-1	500	14	14	14	14
c-fat500-2	500	26	26	26	26
hamming6-	64	32	32	32	32
hamming6-	64	4	4	4	4
hamming8-	256	128	128	128	128
hamming8-	256	16	16	16	16
johnson8-	28	4	4	4	4
johnson8-	70	14	14	14	14
johnson16-	120	8	8	8	8
johnson32-	496	16	16	16	16
keller4	171	11	11	11	11
keller5	776	27	25	26	23
p_hat300-1	300	8	8	8	8
p_hat300-2	300	25	25	25	25
p_hat300-3	300	36	34	36	35
p_hat500-1	500	9	9	9	9
p_hat500-2	500	36	36	36	35
san200_0.7	200	30	30	30	30
san200_0.7	200	18	15	18	15
san200_0.9	200	70	70	70	47
san200_0.9	200	60	60	60	42
san200_0.9	200	40	36	44	35

IV. CONCLUSION

Overall, the three algorithms found the optimum of 90% of the benchmarks, both GA and SA found the optimum of 83%, however, each one of them found the optimum for diffrernt set of graphs.
In genetic algorithm implementation, two local optimizers were used such that some chromosomes will be optimized via the first local optimizer (extract by remove) while others with the second local optimizer (extract by complement).
It was found that, using both local optimizers with a certain rate, helped in finding some benchmark’s optimum.
Using the local optimizer with Tabu search algorithm didn’t give good results, other Tabu search implementation can obtain better results

REFERENCES:

1. S. Homer and M. Peinado. Experiments with polynomial-time clique approximation algorithms on very large graphs.
2. D.S. Johnson, C.R. Aragon, L.A. McGeoh, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, part ii, graph coloring and number partitioning. *Operations Research*, 39:378–406, 1991.
3. A simple simulated annealing algorithm for the maximum clique problem
4. Wu Qinghua, Hao Jinkao. An adaptive multistart tabu search approach to solve the maximum clique problem. Journal of Combinatorial Optimization, 2013, 26(1): 86-108.
5. MARCHIORI E. A simple heuristic based genetic algorithm for the maximum clique problem//SAC. 1998: 366-373.
6. MARCHIORI E. Genetic, iterated and multistart local search for the maximum clique problem//Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2002: 112-121.
7. SINGH A and GUPT A K. A hybrid heuristic for the maximum clique problem. Journal of Heuristics, 2006, 12(1-2): 5-22.
8. ZHANG Q, SUN J and Tsang E. An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE transactions on evolutionary computation, 2005, 9(2): 192-200.
9. BHASN H and MAHAJAN R. Genetic algorithms based solution to maximum clique problem. International Journal on Computer Science and Engineering (IJCSE) , 2012, 4(8): 1443- 1448.
10. BHASIN H, KUMAR N and MUNJAL D. Hybrid genetic algorithm for maximum clique problem. International Journal.2013, 2(4): 2319-4847.
11. WU Donghui, MA Liang. Improved genetic algorithm for maximum clique problem. Journal of Computer Application, 2008, 28(12): 3072-3073.
12. ZHOU Benda, YUE Qin, CHEN Minghua. Immune genetic algo-rithm based on uniform design sampling for solving maximum clique problem. Computer Engineering, 2010, 36(18): 229-231.
13. HU Nengfa, TANG Weiping. Study on genetic algorithm for solving the maximum clique problem. Journal of Hubei University (natural science), 2011, 33(2): 256-259.
14. R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Algorithmica*, 29:610–637, 2001.
15. D. Klimowicz and M. Kubale. Graph coloring by tabu search and simulated annealing. *Archives of Control Sciences*, 2:41–54, 1993.
16. Qinghua Wu and Jin-Kao Hao. Multi-neighborhood tabu search for the maximum weight clique problem
17. Suqi Zhang, Jing Wang, Qing Wu, Jin Zhan. A Fast Genetic Algorithm for Solving the Maximum Clique Problem