A New Optimization Algorithm For Combinatorial Problems

Azmi Alazzam and Harold W. Lewis III
Systems Science and Industrial Engineering Department
State University of New York at Binghamton
Binghamton, NY 13902, USA

***Abstract*—Combinatorial optimization problems are those problems that have a finite set of possible solutions. The best way to solve a combinatorial optimization problem is to check all the feasible solutions in the search space. However, checking all the feasible solutions is not always possible, especially when the search space is large. Thus, many meta-heuristic algorithms have been devised and modified to solve these problems. The meta-heuristic approaches are not guaranteed to find the optimal solution since they evaluate only a subset of the feasible solutions, but they try to explore different areas in the search space in a smart way to get a near-optimal solution in less cost and time. In this paper, we propose a new meta-heuristic algorithm that can be used for solving combinatorial optimization problems. The method introduced in this paper is named the Global Neighborhood Algorithm (GNA). The algorithm is principally based on a balance between both the global and local search. A set of random solutions are first generated from the global search space, and then the best solution will give the optimal value. After that, the algorithm will iterate, and in each iteration there will be two sets of generated solutions; one from the global search space and the other set of solutions will be generated from the neighborhood of the best solution. Throughout the paper, the algorithm will be delineated with examples. In the final phase of the research, the results of GNA will be discussed and compared with the results of Genetic Algorithm (GA) as an example of another optimization method.**

Keywords—meta-heuristic; optimization; combinatorial problems

I. INTRODUCTION

Many optimization problems have been encountered in different domains of manufacturing and industry. Usually the optimization problem that needs to be solved is first formulated and all the constraints are given. The optimization problem mainly consists of an objective function and a set of constraints. The objective function can be in mathematical form or combinatorial form. Once the objective function of the optimization problem is formulated and all the constraints are defined, then the main issue is to solve this problem.

The solution is usually the best values of the variables or the best scenarios which can also be called the optimal solution. This optimal solution should give us the best performance or best fitness in terms of the objective function.

In most optimization problems there is more than one local solution. Therefore, it becomes very important to choose a good optimization method that will not be greedy and look only in the neighborhood of the best solution; because this will mislead the search process and leave it stuck at a local solution. However, the optimization algorithm should have a mechanism to balance between local and global search. An example of a two-dimensional function that has more than one local and global solution is shown in Fig.1 [1].

There are multiple methods used to solve optimization problems of both the mathematical and combinatorial types. In fact, if the optimization problem is simple or if the search space is small, then the optimization problem can be solved using conventional analytical or numerical procedures. However, if the optimization problem is difficult or if the search space is large, it will become difficult to solve the optimization problem by using conventional mathematics or using numerical induction techniques. For this reason, many meta-heuristic optimization methods have been developed to solve such difficult optimization problems. These include Genetic algorithm (GA), simulated annealing (SA), ant colony algorithm (ACA), and particle swarm (PS). Most of these meta-heuristic optimization problems are inspired by nature, biology, or environment.

The term meta-heuristic refers to a specific class of heuristic methods. Fred Glover first used this term and defined it as follows, "A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality.

The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule." [2].

The meta-heuristic algorithms do not always guarantee an optimal solution. However, in most cases a near optimal solution can be obtained in much less time than the computational methods [3-4].

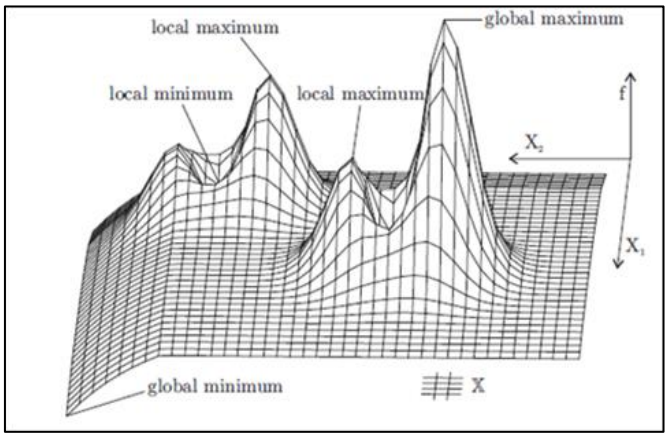

Fig. 1. Global and local optima of two-dimensional function [1].

The meta-heuristic algorithms can be classified based into different categories[5-6]:

1) Constructive and Improvement:

A constructive heuristic (also known as a greedy approach) usually constructs a solution from scratch based on a certain criteria. Some of the common constructive heuristics are nearest neighbor, multiple fragment and insertion heuristics [7]. An improvement or neighborhood search, which is usually known as a local search, attempts to improve the solution by exploring the neighborhood of the current solution [8]. The neighborhood of a solution is the set of solutions that are close to the current solution. The local optimal solution is the best solution in each neighborhood and the global optimum is the best solution with respect to the whole search space. An improvement or local search begins the search from a given solution, and then iteratively attempts to improve the solution quality by using move operators, the move operator is usually determined based on the neighborhood structure, and it aims to change (move) the solution to a newer solution in the same neighborhood but with a better fitness.

2) Single Solution and Population based approaches:

In the single based solution, a unique solution is first generated and then based on a certain move criteria, other solutions are generated. Some of the meta-heuristic methods that start with a single solution are: Tabu Search (TS) and Simulated Annealing (SA). Population based algorithms on the other hand start by generating a set of multiple initial solutions. Examples of those methods would be Genetic Algorithm (GA) and Ant Colony Algorithm (ACA).

The computational drawbacks of mathematical techniques and methods (i.e., complex derivatives, sensitivity to initial values, and the large amount of enumeration memory required) have forced researchers to rely on meta-heuristic algorithms based on simulations and some degree of randomness to solve optimization problems [9]. Although, these meta-heuristic approaches are not very accurate and they do not always give the optimal solution, in most cases they give a near optimal solution with less effort and time than the mathematical methods [10].

The meta-heuristic algorithms are general purpose stochastic search methods simulating natural selection and biological or natural evolution [11]. Different meta-heuristic algorithms have been developed in the last few decades simulating and emulating different processes. Some of these meta-heuristic algorithms were inspired by the biological evolutionary processes; such as the evolutionary strategy (ES) [12], evolutionary programing [13-15], and the genetic algorithm (GA) proposed by Holland [16-17].

Some meta-heuristic algorithms emulate different animal behaviors; like the tabu search (TS) proposed by Glover [18], the ant colony algorithm (ACA) by Dorigo et al [19], Particle Swarm Optimization (PSO) [20], Harmony Search (HS) algorithm [21], Bee Colony Optimization (BCO) [22]. Other meta-heuristic algorithms were inspired by different physical and natural phenomena like the simulated annealing (SA) [23], and the Gravitational Search Algorithm (GSA) [24].

The distribution of publications which applied the meta-heuristics methods to solve the optimization problem in the past decade is shown in Fig.2. [25].

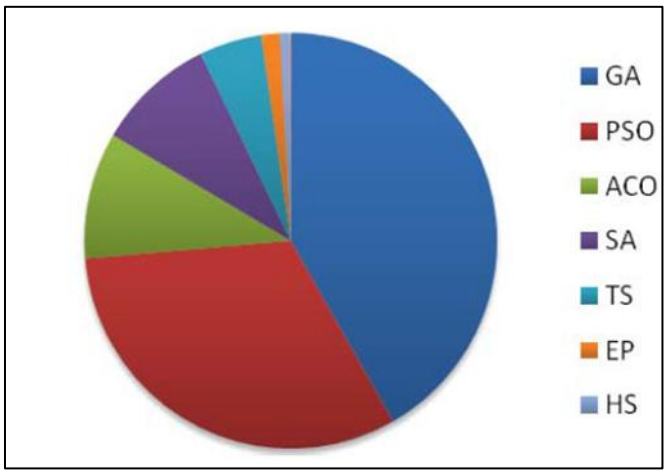

Fig. 2. Pie chart showing the publication distribution of the meta-heuristics algorithms [25].

In this paper we introduce a new optimization algorithm that can be applied to combinatorial problems. The new optimization problem is named Global Neighborhood Algorithm (GNA), and it is a population based and derivative free algorithm like other evolutionary optimization algorithms including Genetic Algorithm (GA), Ant Colony (ACA) and Evolutionary Strategy (ES). A set of randomly generated solutions from the entire search space are first generated and then the best of these solutions is chosen. After that, the algorithm will iterate, and in each iteration there will be two sets of generated solutions; one from the global search space and the other set of solutions will be generated from the neighborhood of the best solution. This paper starts with a background about optimization problems, then the methodology of the GNA algorithm is explained, and after that results for using this algorithm to solve the well-known Traveling Salesman(TSP) problem are also discussed.

II. METHOLODOLGY

The algorithm proposed in this paper is used to optimize combinatorial problems. The combinatorial problems could have more than one local and global optimal value within the

search space values. The proposed methodology will work to find the optimal value among these local optima by switching between exploration and exploitation. Exploration allows for exploring the whole search space. Exploitation allows focusing the search in the neighborhood of the best solution of generated solutions.

In order to explain the methodology of the GNA algorithm, assume we have discrete function that we need to optimize and let us say that we need to minimize this function (without loss of generality).

So the objective function we have is:

$$min \quad g = f(x_1, x_2, \dots, x_n) \tag{1}$$

Where:

$x_1, \ x_2, \ \dots, x_n$ are the different combinations of the solution sequence; we can think of these combinations as the city sequence in the TSP problem.

We need to find the optimal combination or solution $(x_1, x_2, \dots, x_n)$ that will give the optimal (minimum) value for the objective function (g).In general, if each of the variables $(x_1, x_2, \dots, x_n)$ can be chosen in $(n_1, n_2, \dots, n_n)$ ways respectively, then if we want to enumerate all the possible solutions this will yield $(n_1 \times n_2 \times \dots \times n_n)$ solutions. However, this process could take several hours or days depending on the size of the problem. Thus, using a meta-heuristic approach is better even if does not always give the optimal solution, but in most cases it will give a solution that is close to the optimal solution with less computational power.

According to the GNA algorithm, a set of (m) random solutions are first randomly generated from the set of all possible solution, where: $(x_1, x_2, \dots, x_n)$ can be chosen in $(n_1, n_2, \dots, n_n)$ ways.

The generated solutions will then look like:

$(x_1^q, x_2^q, \dots, x_n^q)$ where $q = 1,2, \dots, m$

The fitness for the above solution will be evaluated and this can be done by substituting them in the objective function (g).

The solutions are then sorted according to their fitness obtained from the objective function:

$f(s_1) < f(s_2) < f(s_3) < \dots < f(s_m)$

$s_1 = (x_1', x_2', \dots, x_n')$ is the solution sequence with best fitness.

The best combination (s_1) is then used as a good measure for the local optimal solution and it is also initially set as the best known solution.

In the next iteration, 50% of the (m) generated solutions will be generated near the best solution neighborhood by using a suitable move operator.

The other 50% of the (m) generated solutions will be still generated from the whole search space, and the reason for that is to allow for the exploration of the search space, because if we just choose the solutions close to the best solution we will only be able to find the local solution around this point, and since the function that need to be optimized could have more than one local optima, which might lead us to get stuck at one of these local optima.

Next, the best solutions from the above (m) solutions (50%, 50%) is calculated. The new value for the best solution is compared to best known solution and if it was found to be better it will replace it.

The procedure is then repeated until a certain stop criterion is met. This stop criterion can be a pre-specified number of iterations (t), or when there is no further improvement on the final value of the optimal solution we obtained.

The pseudo code for the GNA algorithm is shown in Fig.3.

```
Define objective function (g)

Initialize the values for all parameters: m,t

Generate (m) feasible solutions from the search space

Evaluate the fitness from the objective function (g)

Optimal solution= the best solution.

i=1

Do while i<t,++
    Generate 50% × m solutions from the
    neighborhood of the best solution

    Generate 50% × m solutions from the
    search space

    Find the best solution from the (m)
    generated solution

    If best solution is less (better) than
    optimal solution

            Optimal solution=best solution

    End If

End DO
```

Fig. 3. Psuedo Code For GNA Algorithm

The flowchart for the GNA algorithm is shown in Fig.4.

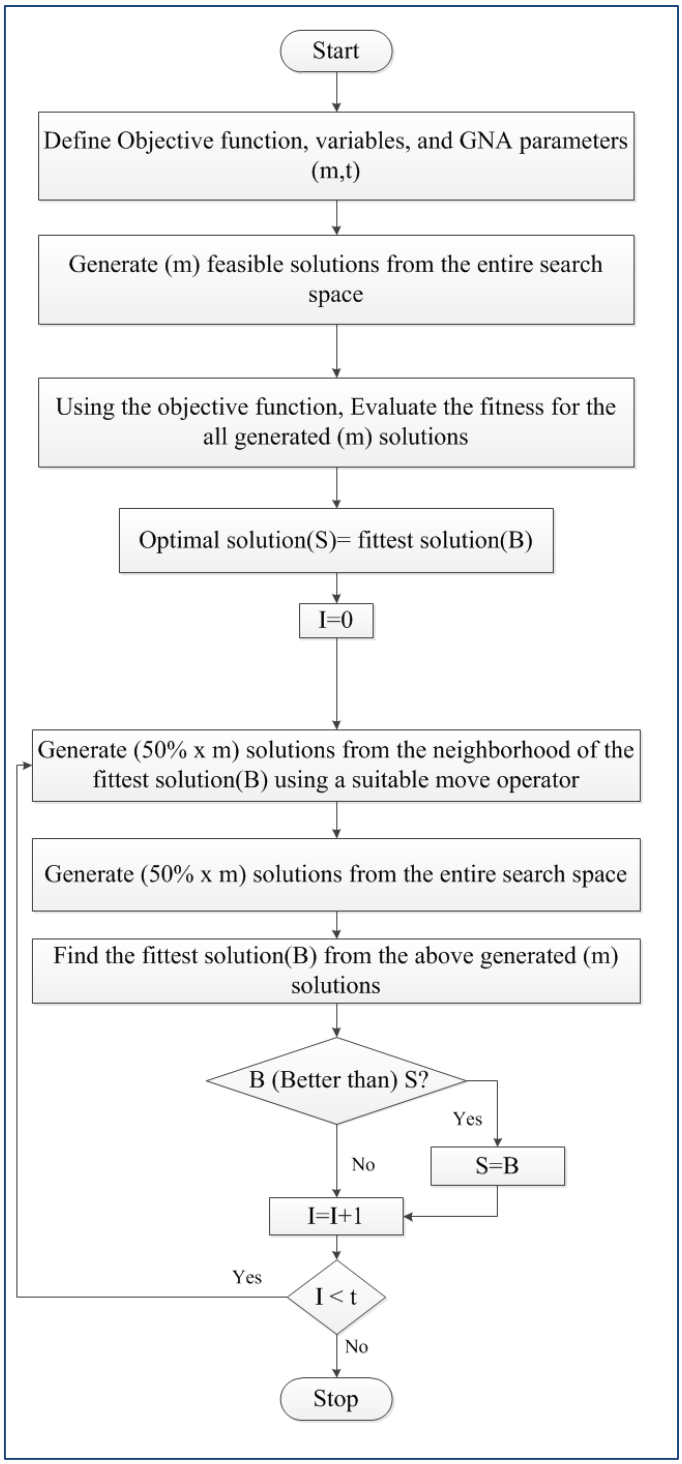

Fig. 4. Flow Chart For GNA Algorithm.

III. Experimental Results and Analysis

The GNA algorithm was used to solve the Traveling Sales man Problem (TSP). The TSP problem consists of a number of cities that need to be visited one time for each, starting from one city and ending at the same city. In order to optimize the TSP problem, the optimal sequence of the different cities that gives the minimum cost (distance) of the tour length has to be found. Thus, the objective (distance) function for the TSP is given by:

$$D(S) = d(C_N, C_1) + \sum_{n=1}^{N-1} d(C_n, C_{n+1}) \qquad (2)$$

Where:
$D(S)$: The total distance for a sequence of N cities.
$d(C_n, C_{n+1})$: The Euclidean distance between the current city and the next city to be visited.
$d(C_N, C_1)$: The Euclidean distance between the last visited city and the first visited city.

To solve the TSP problem, we have to find the optimal sequence (S) that will give the shortest distance. If all the possible solutions are to be checked, then a total number of the combinations will be (N!) for asymmetric TSP or (N!/2) for the symmetric TSP. Obviously, if the number of the cities (N) is small then all the combination can be tried and a deterministic optimal solution can be found. However, if the number of the cities is large, then checking all the possible solution will take very long time and the complexity of the TSP problem will grow exponentially. For this reason, different meta-heuristic algorithms have been widely used to solve TSP problems.

In this paper, the GNA algorithm is used to solve a 29 cities TSP problem. The data were obtained from a real world problem that contained 29 cities in Bavaria, Germany; the source of these data is Zuse Institute Berlin [26]. The optimal solution for this problem is known and documented (2020). The GNA algorithm was implemented using MATLAB software, and the total number of solutions (m) generated at each iteration was 50.

At, each iteration 25 feasible solutions were generated from the whole search space and the other 25 solutions were generated from the neighborhood of the fittest solution. The neighborhood move operator that was used in our case is the two-opt swap; where two cities were randomly chosen and swapped. The code was run for different times and at each time the obtained optimal solution and the run time were recorded. The stopping criteria used was 10000 iterations. The results for the GNA algorithm from the MATLAB code are shown in TABLE.1.

As it can be seen from TABLE.1, in the 10 different run times, we obtained a near-optimal solution that is very close to the known optimal solution.

TABLE I. MATLAB CODE OUTPUT FOR USING GNA TO SOLVE THE TSP PROBLEM.

RUN	Fitness of Optimal Solution	Number of iterations	Run Time
1	2026	10000	28.64
2	2022	10000	28.31
3	2026	10000	28.39

4	2033	10000	27.94
5	2046	10000	28.11
6	2033	10000	27.77
7	2022	10000	28.24
8	2020	10000	28.09
9	2047	10000	27.68
10	2026	10000	28.05

The results of the GNA algorithm were also compared to the Genetic algorithm (GA).

The parameters for the genetic algorithm were as the following:

Generation size: 50

Crossover probability: 90%

Mutation probability: 10%

Number of iterations: 10000

TABLE II. MATLAB CODE OUTPUT FOR USING GA TO SOLVE THE TSP PROBLEM.

RUN	Fitness of Optimal Solution	Number of iterations	Run Time
1	2132	10000	66.90
2	2295	10000	72.48
3	2066	10000	71.23
4	2191	10000	69.35
5	2084	10000	74.12
6	2097	10000	67.97
7	2178	10000	70.59
8	2226	10000	68.24
9	2320	10000	67.38
10	2218	10000	73.82

TABLE. II shows that the run time for the GA is more than twice the run time for the GNA, and the solution obtained by the GA is not always close to the known optimal solution. MINTAB software was used to conduct a statistical analysis between the means of the two optimal solutions obtained by both GA and GNA.

Statistical Analysis was conducted to test if there is a statistical difference between the average for each algorithm. A two- Sample T-Test was used for this purpose. The output from MINITAB is shown in Fig.5.

Two-Sample T-Test and CI: GNA, GA

```
Two-sample T for GNA vs GA

      N     Mean  StDev  SE Mean
GNA  10  2030.10   9.65      3.1
GA   10   2180.7   86.9       27


Difference = mu (GNA) - mu (GA)
Estimate for difference:  -150.6
95% CI for difference:  (-213.1, -
88.1)
T-Test of difference = 0 (vs not =)
T-Value = -5.45  P-Value = 0.000  DF
= 9
```

Fig. 5. MINITAB output for GNA and GA statistical analysis.

A 2-Sample t test showed that there is a significant difference between the optimal solution obtained from both GNA and GA , P-value= 0.000. The 95% CI for difference was (-213.1, -88.1).

Since the difference is always negative as indicated by the confidence interval, This shows us that on average the optimal solution is always higher for the GNA.

The difference in the means between the solutions obtained by the two algorithms is also clear in the Box plot, as shown in Fig.6. It can be seen the GNA outperformed the GA in terms of obtaining a near optimal solutions, and the run time to get this solution was also less. The reason for that is the selection process in GA is more complicated, and it requires sorting the solutions in each generation, whereas in our GNA, the best solution is always selected. Also, the method by which the solutions evolve in each iteration is much simpler in the GNA, unlike the GA that uses Crossover and mutation at each iteration; which makes it take longer time.

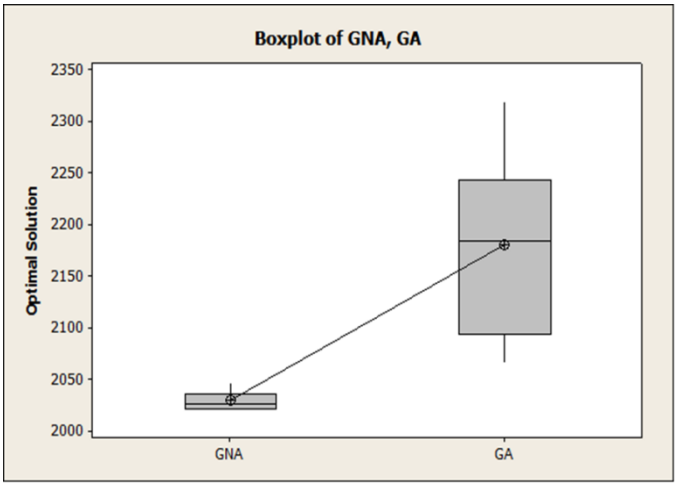

Fig. 6. Boxplot for GNA and GA output.

IV. CONCLUSION

In this paper, a new meta-heuristic optimization method was introduced and named Global Neighborhood algorithm (GNA). This optimization method is a population based algorithm; since it starts with generating a set of random solutions from the search space for the optimization problem. The proposed algorithm can be used to solve combinatorial optimization problems. These combinatorial problems are usually more difficult to solve than other continuous optimization problems. The methodology of this algorithm was elaborated and 29-cities TSP optimization problem was solved using the GNA. The TSP optimization problem was also solved using genetic algorithm (GA) and the results were compared to the GNA. Statistical analysis was conducted using MINITAB software, and it was found that the GNA showed better performance, and the results obtained were very close to the known optimal solution. Future studies can include different variants for the basic GNA algorithm to enhance the search power.

REFERENCES

[1] T. Weise, Global Optimization Algorithms – Theory and Application, Germany: it-weise.de (self-published), [Online]. Available: http://www.it-weise.de/ , 2009

[2] F. Glover and M. Laguna. Tabu search. Kluwer Academic Publishers, 1997.

[3] C. R. Reeves and J. E. Beasley, Modern heuristic techniques for combinatorial problems, McGraw-Hill, 1995

[4] W. Wang, P. C. Nelson, and T. M. Tirpak, "Optimization of high-speed multistation SMT placement machines using evolutionary algorithms," IEEE Transactions on Electronics Packaging Manufacturing, 22(2), 137-146, 1995.

[5] E. Silver, R. V. Vidal, and D. de Werra, "A tutorial on heuristic methods," European Journal of Operational Research, 5, 153-162, 1980.

[6] S. H. Zanakis, J. R. Evans, and A. A. Vazacopoulos, " Heuristic methods and applications: a categorized survey," European Journal of Operational Research, 43, 88-110, 1989.

[7] D. S. Johnson, "Local optimization and the traveling salesman problem," In Goos, G. and Hartmanis, J. (eds) Automata, Languages and Programming, Lecture Notes in Computer Science, 442, Springer, Heidelberg, 446-461, 1990.

[8] E. Aarts and J. K Lenstra, "Local search in combinatorial optimization," Wiley,1997.

[9] K. S. Lee, and Z. W. Geem, " A new structural optimization method based on the harmony search algorithm," Computers and Structures 82(2 004) 781–798, 2004.

[10] K. S. Lee , Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering Optimization," Comput. Methods Appl. Mech. Engrg. 194 (2005) 3902–3933, 2005.

[11] M. G. Omran and M. Mahdavi, "Global-best harmony search," Applied Mathematics and Computation 198 ,643–656, 2008.

[12] I. Rechenberg, Cybernetic solution path of anexperimental problem, Royal Aircraft Establishment, Library Translation no. 1122, 1965.

[13] L. J. Fogel, A. J. Owens and M. J. Walsh, Artificial intelligence through simulated evolution, Chichester, UK: John Wiley,1966.

[14] K. De Jong, "Analysis of the behavior of a class of genetic adaptive systems", Ph.D. Thesis, Ann Arbor, MI: University of Michigan, 1975.

[15] J. R. Koza, "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve Problems," Report No. STAN-CS-90-1314, Stanford, CA:Stanford University, 1990.

[16] D. E. Goldberg, Genetic algorithms in search optimization and machine learning, Boston, MA: Addision-Wesley, 1989.

[17] J. H. Holland, Adaptation in natural and artificial systems, Ann Arbor, MI: University of Michigan Press, 1975.

[18] F. Glover, "Heuristic for integer programming using surrogate constraints," Decision Science, 8(1):156–66, 1977.

[19] M. Dorigo, V. Maniezzo and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," IEEE Trans Systems Man Cybernet, 26(1), 29–41, 1996.

[20] J. Kennedy and R. Eberhart, "Particle swarm optimization," IEEE International Conference on Neural Networks Perth, Australia, pp: 1942-1948, 1995.

[21] Z. W. Geem, J. H. Kim and G. Loganathan, "A new heuristic optimization algorithm: Harmony search," Simulation, 76(2), 60, 2001.

[22] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," Adapt. Behav., 12(3-4), 223, 2004.

[23] S. Kirkpatrick, C. Gelatt and M. Vecchi, "Optimization by simulated annealing," Science 1983, 220(4598), 671–80, 1983.

[24] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, "GSA: A gravitational search algorithm," Inform. Sci., 179(13): 2232-2248, 2009.

[25] M. Khajehzadeh, M. R. Taha, A. El-Shafie and M. Eslami "A Survey on Meta-Heuristic Global Optimization Algorithms," Research Journal of Applied Sciences, Engineering and Technology 3(6), 569-578, 2011.

[26] Z. I. Berlin, "MP-TESTDATA- the TSPLIB Symmetric Traveling Salesman Problem Instances," vol. 2010: Zuse Institute Berlin, 2010.