Journal of Theoretical and Applied Information Technology
30th September 2018. Vol.96. No 18
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A SOLUTION FOR TRAVELING SALESMAN PROBLEM USING GREY WOLF OPTIMIZER ALGORITHM

[1]AMEEN SHAHEEN, [1]AZZAM SLEIT, [1]SALEH AL-SHARAEH

[1] Computer Science Department, King Abdullah II School for Information Technology, The University of Jordan, Amman 11942, Jordan

E-mail: aminalshahin@gmail.com, azzam.sleit@ju.edu.jo, ssharaeh@ju.edu.jo

ABSTRACT

This paper presents an algorithm based on Grey Wolf Optimizer (GWO) for solving the Traveling Salesman Problem (TSP), which is called (GWO-TSP). Traveling Salesman Problem is a well-known NP-Hard problem in optimization which aims at finding the shortest path between cities, where each city must be visited exactly once. The GWO is a recently established meta-heuristic algorithm for solving optimization problems which has successfully solved many optimization problems. GWO-TSP has been compared with well-regarded algorithms such as: Chemical Reaction Optimization (CRO) and Genetic algorithm (GA). In addition, GWO-TSP has been evaluated analytically and by using simulations in terms of error rate and execution time. The algorithms are tested on a number of benchmark problems. Experimental results show that GWO is promising in terms of optimal cost, error rate and standard deviation in comparison with other algorithms.

Keywords: *Grey Wolf Optimizer, Traveling Salesman Problem, Optimization Problems, Meta-Heuristic.*

1. INTRODUCTION

While the number of cities has increased significantly, mobility has become one of the challenges of daily life because of the existence of many dissimilar ways to reach the same city [1]. Some Algorithms could be used to guide people who use any of transportation or movement methods like (car, walking, train, and bus) to reach its destination in the shortest path [2]. From the point of driving travel, we can discover that the driving travel problem can be classified as a kind of Travel Salesman Problem.

Traveling Salesman Problem (TSP) has received much consideration from computer researchers and mathematicians as it's so easy to describe and so difficult to solve. The centrality of the TSP is that it is illustrative of a bigger class of problems known as combinatorial optimization problems (COP) [3]. The TSP problem has a place in the class of such problems known as NP-complete as shown in Figure 1. Specifically, if one can find an efficient (polynomial-time) algorithm for the TSP, at that point efficient algorithms could be found for all others [4].

The TSP is the problem of finding the shortest path between nodes or cities [3]. The problem is to create the shortest tour in aggregation manner to visit each node exactly once and then return to the starting node.

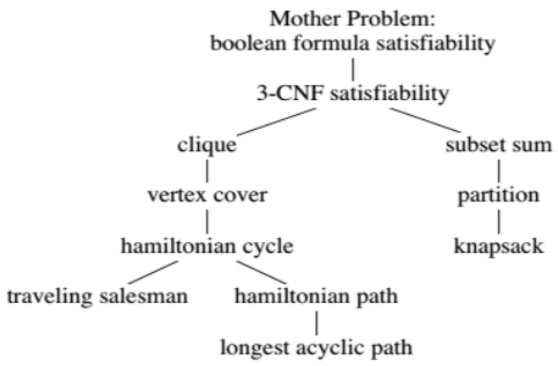
TSP is an NP-Complete problem, where it requires a considerably large amount of computational time and resources for solving it. Since it is a permutation problem. This class of problems is usually much harder to solve than subset problems as there are $n!$ different permutations of n objects [6].

1.1 TSP Formulation:

TSP is formulatized as follow:

Journal of Theoretical and Applied Information Technology
30th September 2018. Vol.96. No 18
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



- Let $V = \{1, 2, \dots, n\}$ be the vertices of graph G.
- Then a permutation $p = \{p_1, p_2, \dots, p_n\}$ of vertices in V defines a unique tour consisting of edges $(p_i, p_i + 1)$, where $i = 1, 2, \dots, n - 1$, and the edge (p_n, p_1).
- The cost of p, denoted as $C(p)$, is the sum of the cost of edges in the associated tour.The problem is to find a minimum cost permutation p, given the edge weights.

A tour is a simple cycle, which starts and ends at vertex 1. Every tour consists of an edge $(1, k)$ for some k in$V - \{1\}$ and a path from vertex k to vertex 1. The path from vertex k to vertex 1 goes through each vertex in $V - \{1, k\}$ exactly once and if the tour is optimal then the path from k to 1 must be the shortest path going through all vertices in $V - \{1, k\}$. Hence, the principle of optimality holds, so if we let $g(i, S)$ be the length of the shortest path starting at vertex i going through all vertices in S and terminating at vertex 1 then $g(1, V - \{1\})$ is the length of an optimal salesman tour [7].

1.2 TSP Example:

From Figure 2, an optimal tour of this example (graph) has length 8 from node A, which follows the path of:

$$A \rightarrow C \rightarrow B \rightarrow D \rightarrow A$$

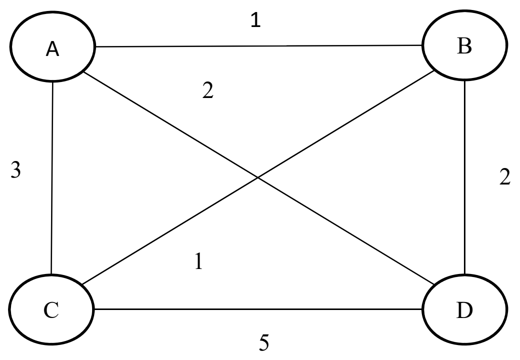


Figure 2. TSP Example.

Due to the importance of the TSP, many researchers solve it using different approaches; such as Bee Colony [8], Genetic algorithm [9] and Chemical Reaction Optimization [10]. These methodologies do not generally locate the optimal solution. Rather; they will often find the near-optimal solutions for the problem. Most of these algorithms called Swarm optimization (SO) [11] or Meta-Heuristic optimization mechanisms [12]. Swarm optimization is attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies and other animal societies [11], which can be used to solve optimization problems. As an example in natural systems of Swarm optimization are Bird Flocking [13], Bacterial Growth [14], Ant Colonies [15], Fish Schooling [16] and Firefly [17].

1.3 Grey Wolf Optimizer (GWO)

In Meta-Heuristic optimization mechanisms, the primary categories of these techniques are Single-Solution-Based and Population-Based; in the first category, the search begins with a single elect solution [18]. This unique elect solution is enhanced over several iterations. In the second category, it represents the optimization by starting with a set of random solutions; this population is enhanced over the course of iterations. The main advantage that characterizes the population-based meta-heuristic over the single-based algorithms is its' high exploration power. This power is attained as it works to find a global solution rather than local ones. These solutions are ordinarily thought to be good enough in light of the fact that they are as well as can be expected to be found in a reasonable amount of time (Polynomial time). In this way, optimization often takes the role of finding the best solution possible in a reasonable amount of time [19]. The Grey Wolf Optimizer (GWO) is also Swarm Optimization and Meta-Heuristic algorithms where there many researchers used it to solve many optimization problems, where it obtained good results, such like: Scheduling problem [21], Economic Emission Dispatch[22] and Parameter Estimation in Surface Waves [23].

The inspiration for GWO is from a species of wolves called the Grey Wolf (Canis lupus), by imitating its hunting methods and hierarchical pack distribution, which are referred to as; Alpha (α), Beta (β), Delta (δ), and Omega (ω); these are used to imitate the series of commands as shown in Figure 3 [20]. As seen, sovereignty reclines from top to bottom. The first level is Alpha (α), which is the leader, which is not necessary to be the most robust wolf but the superior to other wolves in managing the pack. Thus it is responsible for the decision making. The second level is Beta (β), which helps Alpha in decision making. Thus, it represents as the mentor to Alpha and an educator to the pack. The third level is Delta (δ) which controls Omega (ω). This category could be Scouts, sentinels, elders, hunters, and caretakers. Finally, the fourth level is Omega (ω) that acts as the scapegoat and gives up to all dominant wolves.

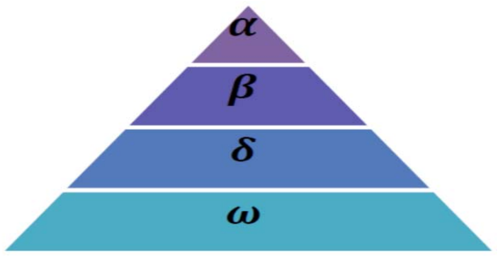


Figure 3. Hierarchy Of Grey Wolf [20].

The main contributions of this study are outlined as follows:

- This study adapted GWO to solve TSP problem and it's executed sequentially on a number of different size datasets of TSP taken from the World TSP [44] and measure performance in terms of execution time, optimal cost, error rate and standard deviation.
- In order to compare the results of GWO with other meta-heuristic algorithms, GA (Genetic algorithm) and CRO (Chemical reaction optimization) are chosen and adapted to solve TSP. The performance metrics in terms of execution time, optimal cost, error rate and standard deviation are computed.
- A comparison between GWO, CRO, and GA in terms of all performance metrics. The GWO shows better performance results than alternative algorithms in terms of optimal cost, error rate and standard deviation.

The remaining of this paper is structured as follows: in Section II, a work related to this paper is illustrated. Then in section III, we address our proposed model for GWO algorithm for solving TSP. Section IV shows the experimental results and discussions. Finally, conclusions and future work are presented in section V.

2. RELATED WORKS

Several researchers have been conducting research on solving TSP and apply their algorithm on different topologies; below are some of these very recent studies.

One of the approaches, which is used as a solution for the TSP, is based on dynamic programming by the study described in [24]. The authors describe how to use dynamic programming to solve the TSP, where they explained that TSP needs O(n!) to find all possible path in brute force approach, where the goal from using dynamic programming to minimize complexity to reach non-polynomial time less than O(n!). The main idea they follow is to divide TSP to sub-problems, each one presents a partial solution, then at the end of each sub-solution, the algorithm starts searching for next node to extend the tour, which is started as a sub-solution. Depends on their approach there is at most $2n \times n$ sub-problems and each one takes linear time to be solved, then the total time would be $O(n^2 * 2^n)$. Another approach for solving TSP is a genetic algorithm. The authors in [25] proposed a heuristic method to solve TSP. Based on a combination of genetic algorithm and an improvement to local tour; they used traditional GA with up to 41 sets and symmetric distance and up to 442 cities. They reported that their approach to find the optimal solution in most cases within 10 seconds, they don't use mutation step but they use immigration step, which is conduct to produce new generation from scratch. In [26], the authors proposed a novel Practical Swarm Optimization (PSO) for solving the TSP. They used a search strategy and crossover elimination technique to speed up the convergence, they reported that a large scale of nodes could be solved using their approach compared with another swarm algorithm. They also proposed another PSO-algorithm to solve the TSP by generalizing chromosome, which is used in GA. This time they used two search techniques to speed up convergence. The study presented in [27], proposed a solution for the TSP presuming that cities represent a directed graph and the cost between every two cities is cij> 0, the goal was to find G. G is a directed cycle tour includes all vertex in V, where V is set of vertices. In addition, to find G it should be the minimum cost tour. They used a matrix to point all edges between cities then calculate costs for each city to reach any other city in the matrix directly or through other connected pairs. At last, they merge costs that they calculated and find the minimum one and name it as the optimal tour. They reported an analytical result supported by example showing that the time needed to solve the TSP using his proposed approach would be O (n * 2^n). Also, they were very clear that this solution is not that efficient even for a modest number of cities. A recent meta-heuristic algorithm used to solve TSP is in [28], where the authors used the Firefly Algorithm to solve TSP. The experimental results obtained on standard TSP instances show that Firefly Algorithm (FA) provides better results than ACO, GA, and SA in most of the instances. In [29], authors used approximation algorithms to find the near-optimal solution, the approximation algorithm used for maximization or minimization based on the problem, when it comes to TSP it is minimization. They focused on a special case of TSP, which is Metric TSP (the distance between two cities is the same in each opposite direction). Then they proposed a parallel two-approximation algorithm for metric TSP. Finally, they reported that the algorithm found near optimal solution with a significant reduction in runtime. Also, authors in [30] proposed an algorithm based on Chemical Reaction Optimization and Lin-Kernighan local search for the Traveling Salesman Problem as a sequential approach. Experimental results show that the proposed algorithm is efficient. In [31], authors utilized Grey Wolf Algorithm to solve the Capacitated Vehicle Routing Problem, authors present a hybrid algorithm ‘K-GWO’ based on GWO and the traditional K-means clustering algorithm. Then they are developing two new clustering heuristics. The resulting algorithm is used in the clustering phase of the cluster-first route-second method to solve the CVR problem. The algorithm is tested on a number of benchmark problems. Better results are obtained using the hybrid model K-GWO. The study presented in [32] present a hybrid algorithm based on CRO and Greedy algorithms called Greedy Chemical Reaction Optimization (GCRO) for solving TSP problem, authors firstly solve the TSP problem using the basic version of CRO then the solution enhanced by hybridizing it with a greedy strategy. The proposed approach obtained better results than CRO and GA in term of quality of the solution.

Because of the importance of TSP and its applications [33], this study presents a solution to the TSP by using the GWO, where GWO is a recent establish meta-heuristic optimization mechanisms and its success in solving many optimization problems with good results. In order to compare the result of GWO with other meta-heuristic algorithms, GWO will be compared with GA and CRO. GA regardless of achieving great success in solving many optimization problems, it is also used for comparison in most meta-heuristic optimization research like [34, 35, 36, 37, 38]. Also, CRO used to compared with GWO because it is one of the newest meta-heuristic algorithms where it also obtained good results in solving NP problems as [39, 40, 41, 42, 43].

3. PROPOSED APPROACH “GWO-TSP ALGORITHM”

In this study, we employed the concept of GWO to solving TSP. The proposed solution is implemented sequentially using standard JAVA language.

GWO is a new nature-inspired meta-heuristic (Swarm intelligence), where this type of algorithms are inspired by natural systems [20]. GWO gets its name from the nature of the social hierarchy of wolves, as well as their hunting behaviour. The Hunting behaviour of Grey Wolves is split into four procedures: (1) Chasing, (2) Encircling, (3) Hunting and (4) attacking the victim. In chasing phase, the Algorithm considers that α is the best solution; β is the second best solution and δ is the third best solution. However, ω represents the rest of the candidate solutions. Thus the hunting is led by the dominant wolves (α, β, and δ). In other words, Grey Wolves could recognize the position of the prey through an iteration process and surround it, where in the encircling phase; Grey Wolves bounded the victim through the hunt (optimization) by calculating the distance between the locations of the prey. In the hunting phase, the hunt generally is led be the leader (α). However, sometimes β and δ contribute in hunting. In another hand, there is no idea about the position of the prey that represents the optimum. Therefore, the algorithm assumes that α, β, and δ has preferable knowledge about the position of prey. Thus, the algorithm saves the first three best solutions then update the locations of the rest wolves (ω) depending on the position of the dominant wolves (best search agent) and in attacking phase where it's the final phase, the hunting proceeding obtained the optimization solution the prey stops proceeding.

In GWO, there are four types of the wolf: Alpha (α), Beta (β), Delta (δ), and Omega (ω) and the prey. Wolfs applied the hunting methods to hunt prey, this is being implemented in a hierarchical way until Alpha wolf take the decision to attack.

TSP contains a number of cities. While there is a cost of traveling between each pair of cities, the objective is to find the shortest path going through all cities. This means a simple cycle tour, which starts and ends at city 1. By applying GWO to find the possible solution for TSP, Figure 4 presents the pseudo-code for the proposed "GWO_TSP" algorithm. Table 1 shows the main attributes and their meaning related to the proposed algorithm "GWO_TSP" in comparison with wolfs meaning in GWO.

Figure 4. The Pseudo Code Of GWO For Solving TSP.

```
Algorithm1: "GWO-TSP"
Input: TSP problem.
Output : Shortest tour among all cities
1//Initialization phase
2 Population size [].
3 Preys : Initial and next preys selection size = 3;
4 Population [ ].
5 Select 3 random preys from city map and added as initial population.
6  While Population.size < Population [ ] and
`  FullTour (Population [ ])// Iteration phase
7  {
8   For each Population [i]
9     Calculate the destination of all wolfs
10      $X_\alpha$ = next best wolf from city map.
11      $X_\beta$ = next second wolf from city map.
12      $X_\delta$= next third wolf from city map.
13      Update Population as:
14      Population [i]  = Population [i] + $X_\alpha$.
15      Population [i+1] + $X_\beta$ // New population
16    Population [i+2] + $X_\delta$//  New population
17      Calculate fitness for each Population.
18    End for
19  If Population> Population.size
20      Remove most costly tour.
21  End if
22  }
23  return X alpha // final stage
```

Table 1: The PROFILE of GWO-TSP

GOW meaning	GWO-TSP meaning
Grey wolf individual	Candidate solution: a tour
Prey	Start city
Grey wolfs	The remaining cities
Alpha (α) wolf	Nearest city to the start city.
Beta (β) wolf	Second nearest city to the start city.
Delta (δ) wolf	Third nearest city to the start city.
Number of iterations	Number of solutions.
Fitness function	Current optimal TSP solution.

As in [20], GWO has three main stages:

1) Initialization stage.

2) Iterations stage.

3) Final stage.

The GWO-TSP algorithm, as shown in Figure 4 (see lines 1, 6 and 23, present these three stages). First, the initialization stage can be shown in Figure 4 (see lines 1-5) to assign initial values for the algorithm parameters. Each individual in the Population is an array, which represents the maximum number of candidate solutions. While each of them consists a full tour as in Table 1. Next, initialize and assign the value of three preys, this is because as the concept of GWO, its assumes that first three wolves (α, β, and δ) has preferable knowledge about the position of prey and as Table 1 each prey represent a city from the city map. Population (see line 4) is an empty array used to constructing the candidate solutions. In order to start building solutions, three preys (or cities) will selected randomly from the city map (data-set) and added as an initial population where all towns that surrounding them are considered the grey wolves which is the final step in the initialization stage.

The goal of the Iteration stage is to generate and build the candidate solutions (or tours) until reaching the best solution. Iteration stage is shown in Figure 4 (see lines 6-22). After generating the three required population as in the initialization stage, each of them will contain a prey that surrounded by a group of wolves. The function (see line 9) used to calculate the destination of all wolfs from the prey then returns the nearest three wolfs from the prey and added to X_α, X_β, and Xδ respectively. Now, based on the positions of the three best wolves, the X_α will be added to the original population and two new population will be created and added the prey in each one plus X_β and X_δ respectively. That's mean, each population will generate two new populations and each of them will contain two preys. In other words, after the first iteration, the number of population will be equal nine and each of them will contain two preys. Next, the algorithm will calculate the fitness for each population, which means, the cost of traveling between cities. While increasing of number of population, the function (see line 20) used to remove the most costly tour. This will happen when the number of the population is larger than the allowed population size. Iteration stage keeps working until reaches the stopping criteria, two stopping criteria used to stop the iteration stage. Firstly, the number of the population must be larger than the allowed population size, and each population should be contained a full solution. Which means, as equal as the number of cities, this is done through FullTour function.

In the final stage, the best solution found will be retrieved.

3.1 GWO-TSP Example

To understand how this study uses GOW to optimize TSP problem, by referring to Figure 2. It shows a simple TSP problem as in Figure 5.a which is an undirected graph of four cities and six edges, each edge has its own traveling cost. This example will be used to represent the possible solutions by GWO-TSP. Based on TABLE 1, each element in GWO represents what its mean in GWO-TSP. For instance, Grey wolf population in GWO represented in GWO-TSP as a candidate solution of TSP. By adapting the elements in GWO to our proposed algorithm, The example that was displayed in Figure 2 and Figure 5.a will become as shown in Figure 5.b.

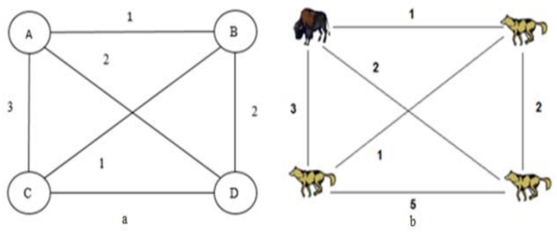


Figure 5. Solving TSP problem using GWO. a) TSP example. b) TSP-GWO example.

In this example, city A selected as the start city, which means the prey. In the Initialization stage, and as the pseudo-code shown in Figure 4 , the population size equals three where the prey is city A. So, city A will be added to population array as:

Population [0] = [A].

Next, for each population, the destination of all wolfs will be calculated. Which will be: The cost of traveling from A to B, the cost of traveling from A to C and the cost of traveling from A to D.

From this, X_α is city B, X_β is city D and X_δ is city C. After that, the population will be updated based on these new values and new two populations will be created. The total will be three populations as :

- Population [0] = [A,B].
- Population [1] = [A,D].
- Population [2] = [A,C].

In this example, city A selected as the start city, which means the prey. In the Initialization stage, and as the pseudo-code shown in Figure 4, the population size equals three where the prey is city A. So, city A will be added to the population array as:

Population [0] = [A].

Next, for each population, the destination of all wolfs will be calculated. This will be: The cost of traveling from A to B, the cost of traveling from A to C and the cost of traveling from A to D.

From this, X_α is city B, X_β is city D and X_δ is city C. After that, the population will be updated based on these new values and new two populations will be created. The total will be three populations as:

- Population [0] = [A,B].
- Population [1] = [A,D].
- Population [2] = [A,C].

That's mean, in the next iteration, for each population, new two populations will be created. Then the number of population will equal nine. For that, before the next iteration start, the most costly tour are removed from the population array. This is in case of the population array is larger than the population size which is equal to five in this example. So, population in the next iteration will be as :

- Population [0] = [A,B,D] → which has a cost of 3.
- Population [3] = [A,B,C] → which has a cost of 2.

- Population [1] = [A,D,B] → which has a cost of 4.
- Population [4] = [A,D,C] → which has a cost of 7.
- Population [2] = [A,C,B] → which has a cost of 4.
- Population [6] = [A,C,B] → which has a cost of 8.

While the population size is equal 5 and the population array are contained 6 Populations, then the most costly Population which is Population [6] will be removed. Iteration will keep work until reach the stopping criteria. Population at the end of Iteration stage will be as :

- Population [0] = [A,B,D,C,A] → which has a cost of 11.
- Population [3] = [A,B,C,D,A] → which has a cost of 9.
- Population [1] = [A,D,B,C,A] → which has a cost of 8.
- Population [4] = [A,D,C,B,A] → which has a cost of 9.
- Population [2] = [A,C,B,D,A] → which has a cost of 8.

In the final stage, the minimum tour cost solution will be returned, in this example, Population [1] and Population [2] provide the best tour solutions, which have a cost of eight to visit each city exactly once, and then return to the starting city. So, one of them will be returned where it's the optimal tour for this example. Figure 6 presents the best solutions steps provided by the GWO-TSP algorithm.

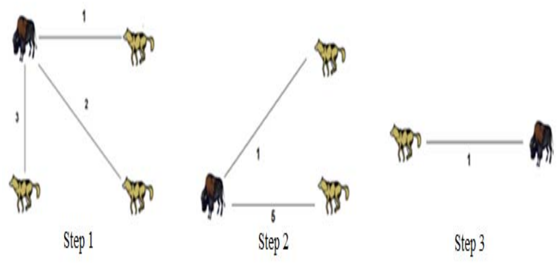


Figure 6. Steps of Solving TSP using TSP-GWO algorithm.

3.2 Analytical Evaluation of GWO-TSP Algorithm

As it is described before, GWO-TSP consists of multiple steps, where initially creates initial population then creates a new generation by calculating the destination between cities.

All the terms that precede (see line 6) are constants. As shown in Figure 4 (lines 1-5), the outer while loop is expected to run until reach the population size where each population must contain a Full solution, as shown in Figure 4 (lines 2-8). In the worst case, the number of population is equal to the number of cities which means O (n). Inside the main loop, another loop runs equal to population size as shown in Figure 4 (lines 8-18), where in each iteration, three cities are picked and updated the population O(n). The function in line 9 which used to Calculate the destination of all wolfs is require O(n) while line 10 to 16 are constants. In line 17, the time complexity for the function of Calculate the fitness value for each Population is O(n). Variables in lines 17 to 19 are constants.

The total time complexity of sequential GWO-TSP is shown in Equation 1 where T is the time complexity, N is the number of population and C is constants:

$$T(N) = O(C + N * (N + C + N + C + N) + C \ldots\ldots (1)$$

Equation 1 can be reduced to Equation 2

$$T(N) = O(2C + 3N^2 + 2NC) \ldots\ldots(2)$$

The largest term of equation 2 is n^2, Thus, the final time complexity will be O (n^2).

In order to give the efficiency of the GWO-TSP algorithm, a comparison is performed with other meta-heuristic algorithms. GA and CRO are chosen because as we mentioned in the related work section. GA is used for comparison in most meta-heuristic optimization where CRO is one of the recently published meta-heuristic algorithms and it obtained good results in solving NP problems.

4. EXPERIMENTAL RESULTS

For our experiments, we used a computer with Intel Core i5-3317U CPU 1.70GHz with 8 GB of RAM. The simulation for GWO, CRO, and GA has implemented in Java JDK 8 programming language. The algorithms were tested by 6 different size TSP problems taken from the World TSP [44]; XQF131, XQG237, PMA343, PKA379, PBL395, and PBN423. The parameters are fixed on follows: number of wolves equals the number of cities in each TSP instance and the maximum number of solutions equals 70% from the number of cities in the dataset, this value is selected to make the algorithm more scalable and to reduce

Table 2. The experimental results of GWO, CRO, and GA in terms of fitness value, quality of solution and the execution time.

Instance name	Optimal	GWO				CRO				GA			
		Best optimal	Mean optimal	Error rate(%)	Time(Sec)	Best optimal	Mean optimal	Error rate(%)	Time(Sec)	Best optimal	Mean optimal	Error rate(%)	Time(Sec)
XQF131	564	569	575	0.886	22.254	573	580	1.595	17.784	574	591	1.773	18.854
XQG237	1019	1030	1033	1.079	54.985	1033	1037	1.668	44.624	1036	1038	1.766	42.241
PMA343	1368	1385	1387	1.242	68.854	1385	1398	2.119	52.325	1400	1399	2.339	56.745
PKA379	1332	1347	1349	1.126	82.325	1347	1360	1.726	74.365	1358	1361	1.951	72.251
PBL395	1281	1296	1300	1.170	95.254	1296	1312	2.107	83.521	1311	1315	2.341	84.214
PBN423	1365	1383	1386	1.318	112.542	1383	1395	1.831	88.124	1398	1405	2.417	92.248

both of computation time and the required space. For fairness, the same specifications and same stopping criteria are used in our simulations for all algorithms.

Since GWO, CRO, and GA are meta-heuristics, the results obtained in different runs may be different. We repeat the simulation 25 times and we record the results as shown in Table 2.

In Table 2, the first column shows the name of the instance (the numbers in the names denote the nodes of each instance). The second column shows the known optimal solution for each instance taken from the World TSP [44]. For each algorithm there are four columns, Best column shows the best fitness value of the best run. The mean column shows the average quality of the 25 runs of the algorithm. The error column shows the error value of the fitness function (minimum) of the best individual that algorithm provide and the TSPLIB optimum. The error is calculated as in equation 3, finally, the time column shows the time taken to run the entire program in seconds.

$$Error = \left(\frac{BestSolution - OptimalSolution}{OptimalSolution}\right) * 100 \ldots (3)$$

Where Error is the relative value of the difference from the optimum tour, Best Solution is the tour length obtained by the experiment and Optimal Solution is the tour length of the optimum solution.

From Figure 7, it is clear that GOW always gives the highest solution quality (minimum traveling cost) for all TSP instances tested. This is followed by CRO and GA algorithms. However, the quality of solution reduces as the size of instance increase with an increasing in execution time for all algorithms.

Figure 8 shows the percentage of deviation from the known optimal solution concerning problems

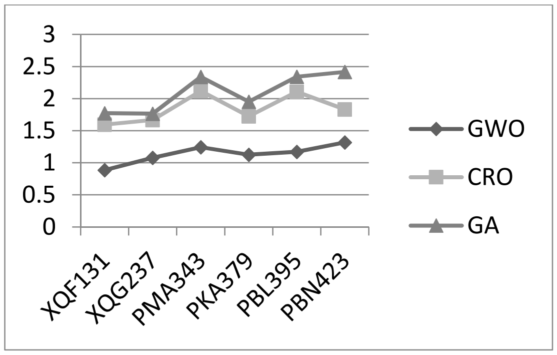


Figure 7.Quality of solutions for GWO, CRO, and GA.

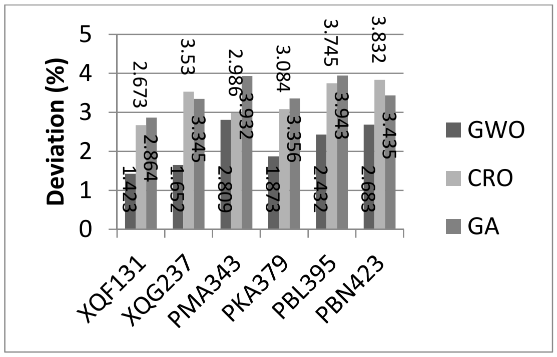


Figure 8. Deviation From Known Optimal Solution.

in the world TSP. It shows that GWO is superior than CRO and GA where the results of GWO always gives deviation less than (3%) from the optimal solution for all instances. CRO gives better results than GA for all size of problems tested.

From Figure 9, we can observe that the runtime for all algorithms is almost the same with a slightly different but the best runtime comes from CRO for some data instance such like PMA343 and PBN423. Also, it is clear that the experimental and theoretical time converge.

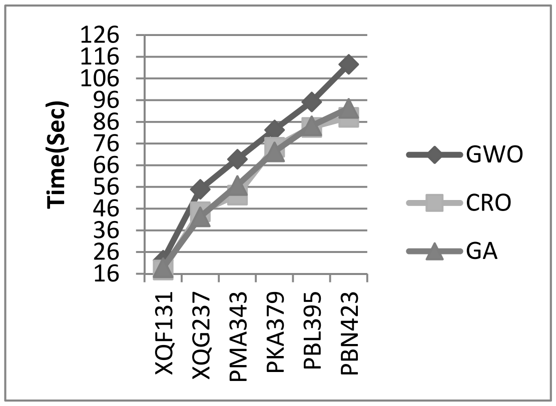


Figure 9. Runtime chart for GWO, CRO and GA.

5. CONCLUSION AND FUTURE WORK

This paper introduces a Grey wolf optimization algorithm (GWO), which is called GWO-TSP to solve the Traveling salesman problem (TSP). GWO-TSP is presented, implemented, and tested on different size of datasets. The theoretical time complexity of GWO-TSP is O (n^2). GWO-TSP algorithm is compared with CRO and GA algorithm. For fairness, same stopping criteria are used in our simulations for all algorithms. The experiments show that the GWO-TSP algorithm performed well in solving the TSP problem in term of the quality of solutions and the computational effort when compared to the alternative algorithms.

For future work, GWO-TSP can be improved to acquire better performance by implementing it using the parallel approach. Also, a comparison between GWO-TSP algorithm and other meta-heuristics which are used to solve the TSP problem could be investigated.

REFERENCES

[1] Vukmirović Si, Pupavac D (2013), The Travelling Salesman Problem in the Function of Transport Network Optimization, Osijek:Interdisciplinary Management Research IX, University in Osijek, Faculty of Economics.

[2] Zhan F, Noon C (1996) "Shortest Path Algorithms: An Evaluation Using Real Road Networks", Transportation Science.

[3] Gutin G, Yeo A, and Zverovich A (2002) "Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP," Discrete Applied Mathematics, vol. 117, pp. 81–86.

[4] Karla L. Hoffman , Manfred P and Giovanni R (2016)"Traveling Salesman Problem"Encyclopedia of Operations Research and Management Science ,Springer ,pp 1573-1578.

[5] A. Al-Shaikh, H. Khattab, A. Shariehand A. Sleit, "Resource Utilization in Cloud Computing as an Optimization Problem," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 7, no. 6, pp. 336-342, 2016.

[6] Shaheen A, Al-Sayyed R, and Sleit A (2017). Improving visual analyses and communications of ontology by dynamic tree (case study: computer system). International Journal of Advanced and Applied Sciences, 4(5): 62-66.

[7] Jonathan Francis O"Connell (2017)"A Dynamic Programming Model To Solve Optimization Problems Using GPUs" Doctor of Philosophy thesis.

[8]Wong, L., Chong, C.S.: An Efficient Bee Colony Optimization Algorithm for Travelling Salesman Problem Using Frequency-Based Pruning. In: Proceedings of 7th IEEE International Conference on Industrial Informatics, pp. 775–782 (2009)

[9] Alok Singh, Anurag Singh Baghel. (2009) A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Computing 13:1, 95-101. Online publication date: 1-Jan-2009.

[10]5. A.Y.S. Lam, V.O.K. Li, "Chemical reaction optimization: a tutorial", Memetic Computing 4, 2012, pp. 3–17.

[11] R. Poli, J. Kennedy, T. Blackwell, "Particle swarm optimization. An overview", *Swarm Intell.*, vol. 1, no. 1, pp. 33-57, 2007. [12] Bianchi, Leonora; Marco Dorigo; Luca Maria Gambardella; Walter J. Gutjahr (2009). "A survey on metaheuristics for

stochastic combinatorial optimization". Natural Computing: an international journal. 8 (2): 239–287.

[13] Reynolds, Craig W. (1987). "Flocks, herds and schools: A distributed behavioral model.". ACM SIGGRAPH Computer Graphics. 21. pp. 25–34.

[14] Zwietering MH, Jongenburger I, Rombouts FM, Van „T Riet K. (1990). Modeling of the bacterial growth curve. Appl Environ Microbiol 56(6):1875–81.

[15] Colorni, Alberto & Dorigo, Marco & Maniezzo, Vittorio. (1991). Distributed Optimization by Ant Colonies. Proceedings of the First European Conference on Artificial Life.

[16]Bastos-Filho C, Lima N, Lins A, Nascimento A, Lima M (2008) "A Novel Search Algorithm based on Fish School Behavior", Proceedings of the IEEE International Conference on Systems Man and Cybernetics, pp. 682-687

[17] Yang X.S (2010), "Firefly algorithm, stochastic test functions and design optimization," International Journal of Bio-Inspired Computation, vol. 2, no. 2, pp. 78–84.

[18] Kirkpatrick S, Jr D, and Vecchi M (1983) "Optimization by imitated annealing," science, vol. 220, pp. 671-680.

[19] Cook, Damon. (2000). Evolved and Timed Ants: Optimizing the Parameters of a Time-Based Ant System Approach to the Traveling Salesman Problem Using a Genetic Algorithm. Honors Undergraduate Thesis, Computer Science Department, New Mexico State University,USA.

[20] Mirjalili S, Mirjalili S, Lewis A (2014) "Grey wolf optimizer", Adv. Eng. Softw., vol. 69, pp. 46-61.

[21] Komaki GM, Kayvanfar. (2015) "Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time".Journal of Computational Science. 8(8):109–20.

[22] Song HM, Sulaiman MH, Mohamed MR (2014) "An application of Grey wolf optimizer for solving combined economic emission dispatch problems".International Review on Modelling and Simulations; 7(5):838–44

[23] Song X, Tang L, Zhao S, Zhang X, Li L, Huang J, Cai W. Grey Wolf Optimizer for parameter estimation in surface waves. Soil Dynamics and Earthquake Engineering. 2015 Aug; 75(5):147–57.

[24] Sleit A, Al-Nsour E. Corner-based splitting: An improved node splitting algorithm for R-tree. Journal of Information Science. 2014 Apr;40(2):222-36.

[25] Snyder, L.V. and Daskin, M.S., (2006). "A random-key genetic algorithm for the generalized traveling salesman problem".European Journal of Operational Research, 174(1), pp.38-53.

[26] Shi, X.H., Liang, Y.C., Lee, H.P., Lu, C. and Wang, Q.X., (2007). "Particle swarm optimization-based algorithms for TSP and generalized TSP". Information Processing Letters, 103(5), pp.169- 176.

[27] Baase S., (2009)."Computer algorithms: introduction to design and analysis". Pearson Education India.

[28] S.N. Kumbharana, G. M. Pandey. "Solving Travelling Salesman Problem using Firefly Algorithm". International Journal for Research in science & advanced Technologies. Issue-2, Volume-2, 053-057. ISSN 2319-2690, 2013.

[29] Anjaneyulu, G. S. G. N., Dashora, R., Vijayabarathi, A., &Rathore, B. S. (2014) "Improving the performance of approximation algorithm to solve travelling salesman problem using parallel algorithm" International Journal of Scientific Engineering and Technology,3(4), 334-337.

[30] J. Sun, Y. Wang, J. Li, and K. Gao. Hybrid algorithm based on chemical reaction optimization and lin-kernighan local search for the traveling salesman problem. In Natural Computation (ICNC), 2011 Seventh International Conference on, volume 3, pages 1518-1521, july 2011.

[31] L. Korayem, M. Khorsid, and S. S. Kassem, "Using Grey Wolf Algorithm to Solve the Capacitated Vehicle Routing Problem," IOP Conf. Ser. Mater. Sci. Eng., vol. 83, no. 1, p. 12014, 2015.

[32] Shaheen, Ameen & Sleit, Azzam & Al-Sharaeh, Saleh. (2018). An improved chemical reaction optimization algorithm for solving traveling salesman problem. 37-42. 10.1109/IACS.2018.8355438.

[33] Applegate D. L., Bixby R. E., Chvátal V., Cook W. J.The Traveling Salesman Problem. A Computational Study (2007) (Princeton University Press, Princeton, NJ).

[34] Shaheen, Ameen & Sleit, Azzam. (2016). Comparing between different approaches to solve the 0/1 Knapsack problem. International Journal of Network Security. 16. 1-10.

[35] P.M. Ross and D. Corne, Comparing genetic algorithms, stochastic hillclimbing and simulated annealing. *In* T.C. Fogarty (ed), *Evolutionary computing*, Springer-Verlag, 94–102 (1995).

[36] Ingber, L. and Rosen, B. (1992), Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, *J. of Mathematical and Computer* Modeling16(11), 87–100.

[37] Sleit A, Salah I, Jabay R. Approximating images using minimum bounding rectangles. InApplications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the 2008 Aug 4 (pp. 394-396). IEEE.

[38] Sleit A. On using B+-tree for efficient processing for the boundary neighborhood problem. WSEAS transactions on systems. 2008 Jul 1;11(11):711-20.

[39] Barham, Reham & Sharieh, Ahmad & Sleit, Azzam. (2016). Chemical Reaction Optimization for Max Flow Problem. International Journal of Advanced Computer Science and Applications. 7. 189-196. 10.14569/IJACSA.2016.070826.

[40] Shaheen, Ameen & Sleit, Azzam & Al-Sharaeh, Saleh. (2018). Chemical Reaction Optimization for Traveling Salesman Problem Over a Hypercube Interconnection Network. 432-442. 10.1007/978-3-319-91192-2_43.

[41] Sleit, Azzam Salah, Imad Jabay, Rahmeh, "Approximating images using minimum bounding rectangles" ICADIWT 2008, pp 394-396, 10.1109/ICADIWT.2008.4664379

[42] Yi Sun, Albert Y.S. Lam, Victor O.K. Li, Jin Xu, James J.Q. Yu, "Chemical Reaction Optimization for the optimal power flow problem", Evolutionary Computation (CEC) 2012 IEEE Congress on, pp. 1-8, 2012.

[43] Sleit A, Al-Akhras M, Juma I, Alian M. Applying ordinal association rules for cleansing data with missing values. Journal of American Science. 2009;5(3):52-62.

[44] TSP website (2009), a collection of worldwide benchmark datasets, “http://www.math.uwaterloo.ca/tsp/world/countries.html” [Access date 05/13/2018].