International Journal of Production Management and Engineering

doi:10.4995/ijpme.2016.4618

Received 2016-02-01 Accepted: 2016-05-31

Solving the Traveling Salesman Problem Based on The Genetic Reactive Bone Route Algorithm whit Ant Colony System

Majid Yousefikhoshbakht [a]*, Nasrin Malekzadeh [b], Mohammad Sedighpour [c]

[a] Young Researchers & Elite Club, Hamedan Branch, Islamic Azad University, Hamedan, Iran.

[a] *khoshbakht@iauh.ac.ir*

[b] Young Researchers & Elite Club, Ardabil Branch, Islamic Azad University, Ardabil, Iran.

[c] Hamedan Branch, Islamic Azad University, Hamedan, Iran.

Abstract: The TSP is considered one of the most well-known combinatorial optimization tasks and researchers have paid so much attention to the TSP for many years. In this problem, a salesman starts to move from an arbitrary place called depot and after visits all of the nodes, finally comes back to the depot. The objective is to minimize the total distance traveled by the salesman. Because this problem is a non-deterministic polynomial (NP-hard) problem in nature, a hybrid meta-heuristic algorithm called REACSGA is used for solving the TSP. In REACSGA, a reactive bone route algorithm that uses the ant colony system (ACS) for generating initial diversified solutions and the genetic algorithm (GA) as an improved procedure are applied. Since the performance of the Metaheuristic algorithms is significantly influenced by their parameters, Taguchi Method is used to set the parameters of the proposed algorithm. The proposed algorithm is tested on several standard instances involving 24 to 318 nodes from the literature. The computational result shows that the results of the proposed algorithm are competitive with other metaheuristic algorithms for solving the TSP in terms of better quality of solution and computational time respectively. In addition, the proposed REACSGA is significantly efficient and finds closely the best known solutions for most of the instances in which thirteen best known solutions are also found.

Key words: Reactive Bone Route Algorithm, Genetic Algorithm, Ant Colony System, Traveling Salesman Problem, NP-hard Problems.

1. Introduction

The traveling salesman problem (TSP) is one of the most important problems in industrial problems which is concerned with searching for the minimum Hamiltonian cycle in a network of nodes by a salesman, based at one fixed node called depot, to serve a set of nodes such that:

- The total distance travelled by the salesman is minimized,
- This salesman must leave and also return to the depot.

In literature, this minimum Hamiltonian cycle means the closed walk that traverses every node once and only once in a graph traversing the minimum path in terms of the length of the edges. This problem belongs to hard combinatorial optimization problems that calls for the determination of the optimal sequence of deliveries conducted. Their importance relies upon the fact that they are difficult to be solved but are intuitively used for modelling several real world problems (Masutti e*t al.,* 2009).

In practice, the basic TSP is extended with constraints, for instance, on the allowed capacity of the salesman, the length of the route, arrival, departure and service time, the time of collection and delivery of goods. It should be noted that the main goal of all TSP problems is to obtain the minimum transportation cost. In another view, there are several reasons for choosing the TSP as the problem to examine the efficiency of the new algorithm:

1. Although the TSP is easily understandable, this problem is an important NP-hard optimization

problem like the n-queens problem (Masehian e*t al.,* 2014) and k-sat (Jaafar and Samsudin, 2013). Furthermore, TSP arises in several applications including the computer wiring, designing hardware devices and radio electronic devices, etc.

2. It is easily understandable, since the algorithm behavior is not obscured by too many technicalities.
3. Because new algorithms are easily applied in the TSP, It is a standard test bed for new algorithmic ideas. In most of the time, a good performance of the TSP is often taken as a proof of their usefulness.

For the first time, Tarantilis and Kiranoudis (2002) presents an adaptive memory-based method for solving the CVRP, called Bone Route. This concept is a population-based method producing a new solution out of components of routes of previous solutions. The components of routes used in this method are sequences of nodes, called bones. Rochat and Taillard (1995) introduced this concept to describe a pool of good solutions that is dynamically updated throughout the solution search process. Some components of these solutions are extracted from the pool periodically and combined to construct a new solution. Furthermore, Tarantilis and Kiranoudis (2007) also presented a more flexible adaptive memory-based algorithm for real-life transportation based on the Bone Route method framework for the classical vehicle routing problem.

The TSP is a problem of high computational complexity (NP-hard). This means that a polynomial time algorithm does not exist for it and the computational attempt required to solve this problem increases exponentially with the size of the problem. In more details, large-scale TSPs (involving usually more than 100 nodes) are unlikely to be solvable in a reasonable amount of time by exact algorithms. This has led researchers to develop metaheuristics that manage to find high quality solutions in a reasonable amount of computer time. Therefore, an efficient reactive bone route algorithm is combined with a modified genetic algorithm and ant colony system (ACS) is proposed. In this work, a new solution from a component of the other solutions is produced while using new diversification and intensification mechanisms. The REACSGA employs the ACS for generating initial diversified solutions and modified GA improvement procedure for improving the initial solutions. Some test problems of TSPLIB are considered and the results of REACSGA are compared to the several metaheuristic algorithms. The results show that the proposed algorithm in comparison with these algorithms can provide better solutions.

In the following parts of this paper, background and literature review is presented in Section 2. In Section 3, the proposed algorithm is described in more details. In Section 4, the proposed algorithm is compared with some of the famous metaheuristic algorithms on standard TSP problems. Finally in Section 5, the conclusions are presented.

2. Background and literature review

The traveling salesman problem (TSP) is a well-known optimization problem in operations research that has nowadays received much attention because of its practical applications in industrial and service problems. For these reasons, different algorithms including exact, heuristic and metaheuristic algorithms have been explored during the several decades ago. For example, the exact optimization methods which can guarantee optimality based on different techniques have been proposed for solving small-size problems with relatively simple constraints. These techniques use algorithms that generate both a lower and an upper bound on the true minimum value of the problem instance. If the upper and lower bound coincide, a proof of optimality is achieved (Yadlapalli e*t al.,* 2009; Cordeau e*t al.,* 2010).

Although optimal solutions can be obtained using exact methods for small size of TSP problems, the computational time required to solve adequately large problem instances is still prohibitive. So, the heuristic and meta-heuristic algorithms have been proposed by researchers and scientists in order to provide near-optimal solutions with reasonable computational time for large-size problems. The proposed heuristics can be separated into three categories including tour construction heuristics, tour improvement heuristics, and composite heuristics. The most powerful algorithms in this group are composite algorithms consisting of the construction algorithm to produce an initial solution and the improvement procedure for finding the best solutions (Renaud e*t al.,* 1998).

As the TSP is an NP-complete problem, most of the heuristic algorithms do not have high quality

for solving the TSP and then great effort has been devoted to metaheuristics that produce a good tour, if not optimal. Johnson and McGeoch (2002) conclude that these algorithms can provide remarkably good results in reasonable amounts of time for practical instances. Since the meta-heuristic approaches are very efficient for escaping from local optimum, they are one of the best algorithms for solving combinatorial optimization problems. That is why the recent publications are more based on meta-heuristic approaches such as gravitational emulation search (Balachandar and Kannan, 2007), neural network (Thiago *et al.,* 2009), ant colony optimization (ACO) (Yousefikhoshbakht *et al.,* 2013), imperialist competitive algorithm (Yousefikhoshbakht and Sedighpour, 2012) and particle Swarm optimization (Zhong *et al.,* 2007).

Although heuristic and metaheuristic algorithms can obtain better quality solution compared with the exact algorithm, many researchers have found that the employment of hybridization in optimization algorithms can improve the quality of the problem in comparison with these algorithms. For examples, ant colony optimization (ACO) and beam algorithm (López-Ibáñez e*t al.,* 2010), GA with a local search (Créput and Koukam, 2009), ACO with Sweep algorithm (Yousefikhoshbakht and Sedighpour, 2013), threshold accepting and edge recombination (Liu, 2007), particle swarm optimization and local search (Shi e*t al.,* 2007), variable neighborhood descent search and GRASP (Hernandez-Perez e*t al.,* 2009) have greater ability for finding an optimal solution to solve the complex problems.

Wang (2010) presented a hybrid algorithm in which GA, ACO and a new strategy called GSA were proposed aiming at the key link in the algorithm. This algorithm converts the genetic solution from GA into information pheromone to distribute in ACO. Furthermore, GSA takes a new matrix formed by the combination of the former 90% of individuals from genetic solution and 10% of an individual by random generation as the basis of the transformation of pheromone value. The best combination of genetic operators in GA was also discussed. Besides, Weber (2006) proposed a distributed algorithm in which ant colony and genetic algorithms work independently of each other and only communicate when better solutions are discovered. In this algorithm, ant colonies are used to explore the solution space, while genetic algorithms are used to improve the convergence rate of the search.

3. The proposed algorithm

In this section, our algorithm in the name of REACSGA is presented. It should be noted that the proposed Bone Route algorithm is a method producing a new solution out of components of routes called the bones of previous solutions. This component of used routes is sequences of nodes. In other words, the fact behind the Bone Route is to extract bone nodes with predefined size and frequency of the Adaptive Memory (AM). The size and frequency specified by the algorithm-designer restricts the number of nodes in a bone (bone-size) and the minimum number of stored routes in the AM that must include a bone (bone-freq-min). it is noted that the value of the bone-freq-max and bone-size express the degree of similarity among the new constructed solution and the previously stored solutions in the AM. Therefore, if these values are considered high, the new solution is more similar to other solutions in AM.

In the proposed REACSGA, *n* different diversified initial solutions are generated by the modified ACS. The goal of building multiple initial solutions is to spread the search in order to explore different regions of the solution space of the problem (diversification strategy). At each iteration of the proposed ACS, each ant builds a solution of the TSP step by step. At each step, the ant makes a move in order to complete the actual solution by following a probability function. The probability of ant *k* which moves from city *i* to city *j* which has not been visited yet is presented in formula (1).

$$P_{ij}^{k}(t)=\begin{cases}1 & if\ q\le q_0\ \&\ j=j^{*}\\ 0 & if\ q\le q_0\ \&\ j\neq j^{*}\\ \dfrac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{r\in J_i^k}\tau_{ir}^{\alpha}(t)\eta_{ir}^{\beta}(t)} & Otherwise\end{cases} \quad (1)$$

Where

$j^{*}=\arg\max_{r\in J_i^k}\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)$ identifies the unvisited node in J_i^k that maximizes $P_{ij}^{k}(t)$.

$\tau_{ij}(t)$: The value of pheromone on the arc (i,j)

$\eta_{ij}(t)$: The heuristic information the arc (i,j) defined here as the reciprocal of the distance between node i and node j.

q: A uniformly distributed random number between 0 to 1.

q_0: A variable assumed 0.2 at the beginning of the algorithm. In every iteration of the algorithm, this variable is increased 0.01 until q_0=0.9. The smaller q_0, the higher the probability to make a random choice ($0 \leq q_0 \leq 1$).

α,β: The controlling parameters by the user.

In order to improve future solutions, the pheromone trails of the ants must be updated to reflect the ant's performance and the quality of the solutions found. This updating is a key element of adaptive learning technique of ACS and helps to ensure the improvement of subsequent solutions. Pheromone trail is updated in global updating. After all ants have completed their schedule, the pheromone level is updated by applying the global updating rule only on the paths that belong to the best found schedule since the beginning as follows:

$$\tau_{ij}(t+1)=(1-\rho)\cdot \tau_{ij}(t)+\rho(1/C_b) \quad if \{edge(i,j)\in T_b\} \quad (2)$$

Where

C_b: The cost of the best tour

ρ: A parameter in the range [0, 1] that regulates the reduction of pheromone on the edges.

T_b: The best tour found by ants.

This rule is intended to provide a greater amount of pheromone on the paths of the best schedule, thus intensifying the search around this schedule. In other words, only the best ant that took the shortest route is allowed to deposit pheromone.

After n solutions of the TSP are built by the modified ACS, they are considered as a group of initial population for GA. Then crossover and mutation operations are applied to them for improving the obtained solution of the modified ACS. Order is still one of the best crossovers in terms of quality a speed. Therefore, this method that is simple to implement has been considered here and the modified order crossover is proposed. In order crossover, a randomly chosen crossover point divides the parent strings into left and right substrings. The right substrings of the parents are selected and replace these genes based on the arrangement in other chromosomes. The only difference between the proposed crossovers with the order crossover is that instead of all the positions to the right of the selected chromosome, several random positions of all the genes in the parent are selected (Figure 1). Clearly this method allows only the generation of valid strings.

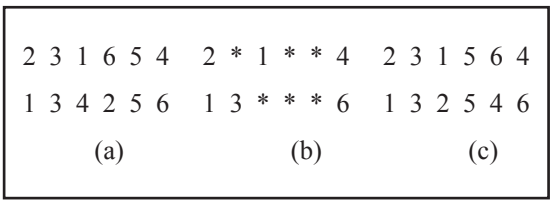

Figure 1. (*a*). Randomized selection number of genes on each chromosome. (*b*). Finding arrangement these genes in another chromosome. (*c*). Replacement these genes on based new arrangement.

Moreover, two mutations are used in the proposed algorithm. These operators randomly select two points in the string, and it replaces together or reverses the substring between these two cut points (Figure 2).

Figure 2. Two used mutations in a chromosome.

In this step, the AM is Constructed using improved solutions of previous step in which their routes are sorted by increasing costs of relative solutions. In each iteration, the s randomly detects bone-like sequences of a predetermined number of nodes using the nearest neighborhood heuristic. Then bone sequences are used to construct a new solution. It is noted that selected bones must not contain common nodes between them. In order to avoid this, the bones are selected that belong to the high quality solution. Furthermore, if those solutions have the same cost, that one is choosing with the highest frequency. Then, combining the extracted bones by applying the modified ACS generates new solutions.

In this step, three different neighborhood operators including insert, swap and 2-opt algorithms are used to improve the new constructed solution generated in the previous step and set as the best solution, if it is better than the previous elite solution. In insert move a node from its position is changed to another position. In the swap move, two nodes from the route are changed. In 2-opt move, two non-adjacent edges

are replaced by two other edges. It should be noted that there are several routes for connecting nodes in order to produce the tour again, but a state that satisfies the problem's constraints is acceptable. These operators are shown in Figure 3.

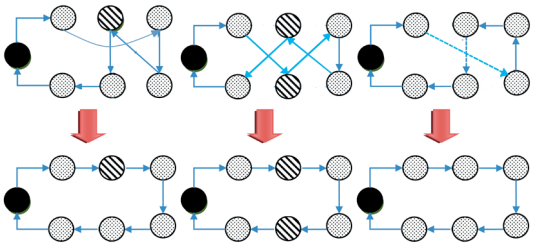

Figure 3. Insert (left), swap (middle) and 2-opt exchanges (right).

After, if AM is not full, the AM is updated and the new improved solution is added, Otherwise AM is updated by inserting the routes of new improved solution and removing routes that belong to the worst solution when the new improved solution is better than the low quality solution in AM. If the best solution of algorithm in current iteration has been improved, the search process is intensified in neighborhood of current solutions by applying these local search algorithms again. Then replace the new solution in this procedure instead of the best solution in AM if it has higher quality. This technique leads to increase the convergence of the algorithm to the best solution.

To effectively implement a metaheuristic, one of the most important elements is how to employ the search memory to get a compromise between the diversification and intensification in the search space of the problems. The intensification forces the search to check the neighborhood of some good solutions. Such procedure is a kind of utilization and learning from the accumulated experience. Nevertheless, the diversification is to explore the unexplored regions of the search space. In this step, if the meta-heuristic has not updated the best solution for a prespecified number of consecutive iterations, we must drive the search towards a part of the solution space that has not explored yet (diversification policy). So, we decrease the similarity among the solutions in AM with decreasing the value of bone-size in a number of consecutive iterations. After the diversification policy, the search process is intensified by increasing the value of bone-size for a number of consecutive iterations. This reactive behavior of the adaptive memory based metaheuristic provides a diversified solution to explore the solution space more precisely.

Therefore, at this stage, at first the parameters are updated and then the stop condition is checked. If this condition is met, the algorithm ends. Otherwise, if stop conditions are not satisfied, the algorithm is iterated. In the proposed algorithm, if the best known solution is iterated 20 times, the algorithms ends and the obtained results and values up to now are considered as the best values and the results of the algorithm. Figure 4 shows the main steps of the proposed algorithm.

1: Generate n initial solutions by employing the ACS.

2: Improve the quality of solutions produced in Step 1 by using the GA and determine the best solution.

3: Construct the Adaptive Memory (AM) and sort the routes by increasing the costs of their relative solutions.

Repeat

4: Extract the promising bone sequences of nodes according to some selection criteria.

5: Extract the selected bones from the AM, according to the values of the bone-size and bone-freq-min.

6: Employ the ACS to generate n new solutions using the extracted bones.

7: Improve the quality of the newly constructed solution generated in Step 6 by using GA and obtain the best current solution.

8: update AM.

9: if the best solution until now is improved, apply several local search algorithms and then update AM (If necessary)

10: update parameters.

***Until** the best solution has not been changed for 20 iterations.*

Figure 4. Outline of the REACSGA.

4. Experiments and computational results

The whole algorithmic approach was implemented in C and implemented on a Pentium 4, 3 GHz (2 Gb RAM) PC with windows XP. Like every meta-heuristic algorithm, the quality solutions produced by the proposed algorithm have been dependent on the seed used to generate the sequence of pseudo-random numbers and on the different values of the parameters. Therefore, a number of different alternative values were tested and the Taguchi

method is implemented to tune the parameters of the proposed algorithm. Finally, the ones selected are those that gave the best computational results concerning the quality of the solution. For achieving this goal, orthogonal arrays provided in this method are designed to perform examinations with varied numbers of levels. It should be noted that, increase of levels numbers results the increase of experiments numbers. So, because 13 parameters are in the proposed algorithm, the most suitable design is three-echelon experiments in this paper. In other words, according to the standard orthogonal arrays of Taguchi, *L*27 array is considered as the suitable experimental design to tune the parameters of the proposed algorithm. *L*27 Array is an experimental design with 27 times runs. If the experiments were run completely, it was necessary to run 3^7=2187 experiments to run experiments to tune the parameters of the ACS and 3^6=729 experiments to tune the parameters of Tabu Search and GA. The proposed GA and Tabu Search Algorithm run for each Taguchi experiment. Then the ration of S/N is calculated by Minitab Software v.16 and the optimum combinations of the parameters are shown in Table 1 for each algorithm.

Although the results confirm that our parameters setting worked well, it is also possible that the better solutions may exist. Thus, the selected parameters are given in Table 1. All of the parameter values have been determined on the Eil51 by the numerical experiments. Furthermore, the proposed algorithm stops after no improvements are found for 20 iterations.

The algorithm was tested on a set of 19 problems with sizes ranging from 24 to 318 nodes. The selected test problems are symmetric and Euclidean TSP instances. The word ‘‘symmetric’’ means that the travel cost from city A to city B is the same as the travel cost from city B to city A. Each instance is described by its TSPLIB name and size, e.g., in Table 2 the instance named Gr48 has size equal to 48 nodes. The sets of data used for the experiment are TSP instances available on the TSPLIB (Gutin and Punnen, 2002). In Table 2, the first and second columns show the number and the name of each instance. The third column specifies the instance's size referenced. Moreover, the fourth, fifth, sixth, seventh, eighth, ninth and tenth columns show the CPU time and the best results of seven meta-heuristic algorithms and the eleventh column present the best solutions of the proposed algorithm through 10 independent runs. Additionally, in order to recognize the performance of the method, the best solutions published (BKS) in the literature, are presented in twelfth column. The last column of Table 2 shows the gap value of the REACSGA, where the gap is defined as the percentage of deviation from the best known solution in the literature. The gap is equal to:

$$100[c(s^{**})-c(s^{*})]/c(s^{*}) \quad (3)$$

Where s^{**} is the best solution found by the algorithm for a given instance, and s^{*} is the overall best known solution for the same instance on the web. A zero gap indicates that the best known solution is found by the algorithm.

As can be seen in this column in Table 2, the REACSGA finds the optimal solution for thirteen out of nineteen problems that are published in the literature. For instance Lin318, the gap is relatively as high as 1 percent. However, in other instances, the proposed algorithm finds nearly the best known

Table 1. Optimum Parameter values.

Parameter	Candidate	Optimum Value
Alpha	1 3 5	1
Beta	1.5 2.5 3.5	2.5
Rho	0.7 0.8 0.9	0.9
q_0	0.10 0.15 0.20	0.20
Number of ants	n/3 n/2 n	n
Number of population created by the ACS	n/3 n/2 n	n
Number of iterations as stop condition which the ACS is stopped after no improvements	5 7 9	7
Number of iterations as stop condition which the GA is stopped after no improvements	5 7 9	7
AMsize	5 6 7	7
Bonefreq_min	2 3 4	2
Bonefreq_max	5 6 7	7
Number of consecutive iterations. Diversification policy is done	2 3 4	4
Number of consecutive iterations. Intensification policy is done	2 3 4	4

Table 2. Comparison of algorithms for standard problems of TSP.

			ACS		GA		NN		GSAP		PSO		BCO		ACO		REACSGA			
Number	Instance	n	Best	Time	Best	Time	Best	Time	Best	Time	Best	Time	Best	Time	Best	Time	Best	Time	BKS	Gap
1	GR24	24	1272	-	1272	-	-	-	-	-	-	-	-	-	-	-	1272	2.5	1272	0
2	Bayg29	29	1610	-	1610	-	-	-	-	-	-	-	-	-	-	-	1610	2.36	1610	0
3	GR48	48	5046	-	5047	-	-	-	-	-	-	-	-	-	-	-	5046	3.87	5046	0
4	ATT48	48	10628	-	10643	-	-	-	-	-	-	-	10661	-	-	-	10628	3.99	10628	0
5	Eil51	51	427	-	429	-	427	7.95	427	-	427	4.06	428	-	426	5.77	426	4.65	426	0
6	Berlin52	52	7542	-	7548	-	7542	8.91	7542	-	7542	4.12	-	-	7542	5.90	7542	5.13	7542	0
7	Eil76	76	542	-	549	-	541	15.07	538	-	540	11.59	539	-	543	18.23	538	9.19	538	0
8	KroA100	100	21309	-	21540	-	21333	15.87	21282	-	21296	23.95	21763	-	21341	41.64	21282	18.21	21282	0
9	KroB100	100	22183	-	22431	-	22343	22.06	22141	-	-	-	22637	-	-	-	22141	21.72	22141	0
10	KroC100	100	20787	-	22993	-	20915	21.65	20749	-	-	-	20853	-	-	-	20754	25.53	20749	0.02
11	KroD100	100	21341	-	21591	-	21374	21.92	21309	-	-	-	21643	-	-	-	21335	21.15	21294	0.19
12	KroE100	100	22109	-	22198	-	22395	21.49	22068	-	-	-	22450	-	-	-	22068	19.99	22068	0
13	Eil101	101	636	-	643	-	638	21.72	630	-	-	-	635	-	-	-	629	20.93	629	0
14	Lin105	105	14534	-	14703	-	14379	20.62	14379	-	-	-	15288	-	-	-	14379	25.77	14379	0
15	KroA150	150	26749	-	27054	-	26678	39.89	26524	-	-	-	27858	-	-	-	26611	55.19	26524	0.33
16	KroB150	150	26431	-	26659	-	26264	40.61	26130	-	-	-	26535	-	-	-	26202	54.32	26130	0.28
17	KroA200	200	29762	-	30276	-	29600	61.88	29383	-	29563	198.55	29961	-	30083	338.98	29368	67.16	29368	0
18	KroB200	200	29653	-	31980	-	29637	57.91	29541	-	-	-	30350	-	-	-	29509	90.10	29437	0.25
19	Lin318	318	-	-	-	-	42834	110.28	42487	-	-	-	44685	-	-	-	42543	164.13	42029	1.22

solution, i.e., the gap is below 0.35, and overall, the average difference is 0.13.

A computational experiment has been conducted to compare the performance of the proposed algorithm for TSP with some of the best techniques designed including ACO, GA, and particle swarm optimization (PSO) (Zhong e*t al.,* 2007), ACS, bee colony optimization (BCO) (Wong e*t al.,* 2008), self-organizing neural network (NN) (Masutti and Castro, 2009) and genetic algorithm combined by simulated annealing and ant colony system and particle swarm optimization (GSAP) (Chen and Chien, 2011) in Table 2. The results also show that the REACSGA has the ability to escape from local optimum and find the best solutions for most of the instances. The results of this comparison show that the proposed algorithm gains equal solutions to the GA in GR24 and Bayg29, and it gains better solutions than the GA in other problems from Gr48 to KroB200.

Furthermore, the results indicate that although the ACS gives an equal solution to the proposed algorithm for 5 instances including GR24, Bayg29, GR48, ATT48 and Berlin52, this algorithm cannot gain optimal solutions for others and yields worse solutions than the proposed REACSGA algorithm. Besides, in general the proposed algorithm gives better results compared to other three algorithms including PSO, ACO and BCO algorithms in terms of the solution's quality. The performance comparison of results shows that the proposed algorithm method clearly yields better solutions than the BCO. Moreover, the computational results of the REACSGA and PSO shows that these algorithms have a close competition and the proposed algorithm gives better 4 solutions than PSO. In other words, the performance of the proposed algorithm is better in reaching the sub-optimal solution than the PSO.

Furthermore, the results indicate that although the NN yields solutions equal to the proposed algorithm for Lin105, this algorithm cannot maintain this advantage in the other examples. The proposed algorithm yields better solutions than the NN for other 18 instances. Moreover, computational results of the proposed algorithm and GSAP show that these algorithms have a close competition, but the proposed algorithm produces seven better solutions more than GSAP. In other words, the REACSGA performs better in reaching the suboptimal solution than the GSAP. As a result, the proposed algorithm yields better solutions than the GA, ACS, GSAP, NN, PSO, ACS, and BCO.

Although direct comparisons of the required computational times cannot be exactly compared, as they closely depend on various factors such as the processing power of the computers, the programming languages, the coding abilities of the programmers and the running processes on the computers, the CPU times of common instances obtained by the proposed algorithm are compared with NN, PSO and ACO in Figure 5. It is noted that because the CPU time of all instances were not reported, only five instances Eil51, Berlin52, Eil76, KroA100 and KroA200 are considered here. By comparing the obtained results in this figure and Table 2, it is concluded that the proposed algorithm can obtain high quality solutions in acceptable time.

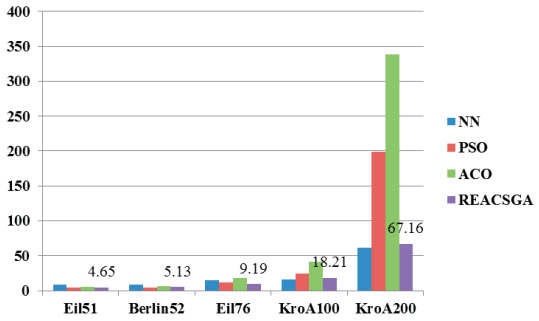

Figure 5. Comparison of CPU time of algorithms for standard problems of TSP.

5. Conclusions

In this paper, a reactive algorithm which uses the modified ACS for generating initial diversified solutions and GA for intensification mechanisms was proposed for solving the TSP as a non-deterministic polynomial (NP-hard) problem. Furthermore, Taguchi Method is used to set the parameters of the proposed algorithm in order to obtain high quality solution both in CPU time and cost. Experiments are implemented to evaluate the algorithm's performance on some test instances of TSPLIB. Computational results demonstrate that our algorithm is effective for solving TSP. As shown, in almost 19 cases of instances, the results of mentioned algorithms can be improved by our algorithm and the Gap between the BKS reported and the solution found by the proposed algorithm for 13 instances is zero. Using this proposed algorithm for other versions of the TSP and also applying this method in other combinational optimization problems including the vehicle routing problem, School bus routing problem and the sequencing of jobs are suggested for future research.

References

Balachandar, S. R., Kannan, K. (2007). Randomized gravitational emulation search algorithm for symmetric traveling salesman problem, *Applied Mathematics and Computation*, 192(2), 413-421. http://dx.doi.org/10.1016/j.amc.2007.03.019

Chen, S. M., Chien, C. Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques, *Expert Systems with Applications,* 38(12), 14439-14450. http://dx.doi.org/10.1016/j.eswa.2011.04.163

Cordeau, J. F., Dell'Amico, M., Iori, M. (2010). Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading, *Computers & Operations Research*, 37(5), 970-980. http://dx.doi.org/10.1016/j.cor.2009.08.003

Créput, J. C., Koukam, A. (2009). A memetic neural network for the Euclidean traveling salesman problem, Neurocomputing, 72(4-6), 1250-1264. http://dx.doi.org/10.1016/j.neucom.2008.01.023

Gutin, G., Punnen, A. (2002). *The Traveling Salesman Problem and its Variations*, Kluwer Academic Publishers, Dordrecht.

Hernández-Pérez, H., Rodríguez-Martín, I., Salazar-González, J. J. (2009). A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem, *Computers & Operations Research*, 36(5), 1639-1645. http://dx.doi.org/10.1016/j.cor.2008.03.008

Jaafar, A., Samsudin, A. (2013). An Improved Version of the Visual Digital Signature Scheme The International Arab Journal of Information Technology, *The International Arab Journal of Information Technology*, 10(6), 20-32.

Johnson, D.S., McGeoch, L.A. (2002). Experimental analysis of heuristics for the STSP, in G. Gutin and A.P. Punnen (eds.), *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, The Netherlands, 369–443.

López-Ibáñez, M., Blum, C. (2010). Beam-ACO for the traveling salesman problem with time windows, *Computers & Operations Research*, 37(9), 1570-1583. http://dx.doi.org/10.1016/j.cor.2009.11.015

Masehian, E., Akbaripour, H., Mohabbati-Kalejahi, N. (2014). Solving the n-Queens Problem using a Tuned Hybrid Imperialist Competitive Algorithm, The International Arab Journal of Information Technology, 11(6).

Masutti, T. A. S., Castro, L. N. D. (2009). A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, *Information Sciences,* 179(10), 1454-1468. http://dx.doi.org/10.1016/j.ins.2008.12.016

Renaud, J., Fayez, F., Boctor, F. (1998). An efficient composite heuristic for the symmetric generalized traveling salesman problem, *European Journal of Operational Research*, 108(3), 571-584. http://dx.doi.org/10.1016/S0377-2217(97)00142-2

Rochat, Y., Taillard, E.D. (1995). Probabilistic diversification and intensification in local search for vehicle routing, *Journal of Heuristics*, 1(1), 47–167. http://dx.doi.org/10.1007/bf02430370

Shi, X. H., Liang Y. C., Lee H. P., Lu C., Wang, Q.X. (2007). Particle swarm optimization-based algorithms for TSP and generalized TSP, *Information Processing Letters*, 103(5), 169-176. http://dx.doi.org/10.1016/j.ipl.2007.03.010

Tarantilis, C. D., Kiranoudis, C. T. (2002). BoneRoute: An adaptive memory-based method for effective fleet management. *Annals of operations Research*, 115(1-4), 227-241. http://dx.doi.org/10.1023/A:1021157406318

Tarantilis, C. D., Kiranoudis, C. T. (2007). A flexible adaptive memory-based algorithm for real-life transportation operations: two case studies from dairy and construction sector, *European Journal of Operational Research*, 179(3), 806-822. http://dx.doi.org/10.1016/j.ejor.2005.03.059

Thiago, A.S., Masutti, L., Castro, N. (2009). A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Information Sciences*, 179(10), 1454-1468. http://dx.doi.org/10.1016/j.ins.2008.12.016

Yadlapalli, S., Malik, W.A., Darbha, S., Pachter, M. (2009). A Lagrangian-based algorithm for a Multiple Depot, Multiple Traveling Salesmen Problem, *Nonlinear Analysis: Real World Applications*, 10(4), 1990-1999. http://dx.doi.org/10.1016/j.nonrwa.2008.03.014

Yousefikhoshbakht, M., Didehvar, F., Rahmati, F. (2013). Modification of the Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem, *Romanian Journal of Information Science and Technology*, 16(1), 65-80.

Yousefikhoshbakht, M., Sedighpour, M. (2013). New Imperialist Competitive Algorithm to solve the travelling salesman problem, *International Journal of Computer Mathematics*, 90(7), 1495-1505. http://dx.doi.org/10.1080/00207160.2012.758362

Yousefikhoshbakht, M., Sedighpour, M. (2012). A Combination of Sweep Algorithm and Elite Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem, *Proceedings of the Romanian academy A*, 13(4), 295-302.

Wang, C. (2010). A hybrid algorithm based on genetic algorithm and ant colony optimization for Traveling Salesman Problems. *The 2nd International Conference Information Science and Engineering*, 4257-4260. http://dx.doi.org/10.1109/ICISE.2010.5689028

Weber, B. (2006). *Distributed Hybrid Metaheuristics for Optimization*, Work paper, 1-12.

Wong, L. P., Low, M. Y. H., Chong, C. S. (2008). A bee colony optimization algorithm for traveling salesman problem. *Modeling & Simulation, 2008. AICMS 08. Second Asia International Conference*, 818–823. http://dx.doi.org/10.1109/AMS.2008.27

Zhong, W., Zhang J., Chen, W. (2007). A novel discrete particle swarm optimization to solve traveling salesman problem. *Evolutionary Computation*, 3283–3287.