Iraqi Journal of Science, 2019, Vol. 60, No.5, pp: 1138-1153
DOI: 10.24996/ijs.2019.60.5.22

ISSN: 0067-2904

Exact and Local Search Methods for Solving Travelling Salesman Problem with Practical Application

Sajjad Majeed Jasim*, Faez Hassan Ali
Department of Mathematics, College of Science, Al-Mustansiryah University, Baghdad, Iraq

Abstract

This paper investigates some exact and local search methods to solve the traveling salesman problem. The Branch and Bound technique (BABT) is proposed, as an exact method, with two models. In addition, the classical Genetic Algorithm (GA) and Simulated Annealing (SA) are discussed and applied as local search methods. To improve the performance of GA we propose two kinds of improvements for GA; the first is called improved GA (IGA) and the second is Hybrid GA (HGA).

The IGA gives best results than GA and SA, while the HGA is the best local search method for all within a reasonable time for $5 \leq n \leq 2000$, where n is the number of visited cities. An effective method of reducing the size of the TSP matrix was proposed with the existence of successive rules. The problem of the total cost of Iraqi cities was also discussed and solved by some methods in addition to local search methods to obtain the optimal solution.

Keywords: Travelling Salesman Problem, Greedy Method, Improved Minimum Distance Method, Branch and Bound Technique, Genetic Algorithm and Simulated Annealing.

طرق الحل التام والبحث المحلية لحل مسالة البائع المتجول مع تطبيق عملي

سجاد مجيد جاسم* ، فائز حسن علي
قسم الرياضيات، كليه العلوم، الجامعة المستنصرية، بغداد، العراق.

خلاصة

في هذا البحث، سنقوم بتنفيذ بعض طرق الحل التام والبحث المحلية لحل مشكلة البائع المتجول. وقد تم اقتراح تقنية التفرع والتقيد (BABT)، كطريقة حل تام مع نموذجين. بالإضافة إلى ذلك، تم مناقشة وتنفيذ الخوارزمية الجينية التقليدية (GA) وخوارزمية محاكاة التلدين (SA) كطرق بحث محلية. ولتحسين أداء GA، تم اقتراح نوعين من التحسين لـ GA؛ النوع الأول سمي الخوارزمية الجينية المحسنة (Improved GA) والنوع الثاني سمي الخوارزمية الجينية المهجنة (Hybrid GA).

لقد اعطت الخوارزمية IGA نتائج أفضل من GA و SA ، في حين أن HGA هي ألافضل في طرق البحث المحلية المناقشة في هذا البحث وفي فترة زمنية مقبولة لـ $5 \leq n \leq 2000$، حيث n هو عدد المدن التي تمت زيارتها. ولقد تم اقتراح طريقة فعالة لتقليل حجم مصفوفة البائع المتجول في حالة وجود شروط اسبقية للمسالة. وكتطبيق لموضوع البحث، فقد تم حل مسالة تقليل الكلفة الإجمالية لزيارة جميع المدن العراقية باستخدام بعض طرق الحل التام بالإضافة إلى طرق البحث المحلية المطروحة في هذا البحث وقد تم الحصول على الحل الأمثل لهذه المسالة.

*Email: sajadm49@gmail.com

1. Introduction

The traveling salesman problem (TSP) is a classic combinatorial optimization problem (COP). The TSP searches for the shortest path among a set of cities with known distances between pairs of cities to find the route solution. The route solution can be articulated as a complete graph with a set of vertices, which is a set of edges weighted by the distance between two vertices (cities), to find the shortest route by visiting each city exactly once and returning to the original city [1].

Numerous approaches have been proposed and have obtained good solutions. However, they vary in terms of complexity and efficiency and in being able to solve the TSP at various levels of complexity and size (small, medium, and large). Earlier studies use linear programming, dynamic programming, and branch and bound, but their ability is limited to small problems. Later, artificial intelligent approaches were proved to have the ability to solve more complex problems; one of these approaches, the self-organized neural network, was later expanded as a metaheuristic. A metaheuristic can optimize a complex problem by searching through many candidate solutions with few or no assumptions about the problem being solved and without any guarantee of finding the optimal solution. Some metaheuristics use either a single solution-based approach (e.g. simulated annealing (SA)) or a population-based approach (e.g. genetic algorithm (GA)) [1].

Dorigo and Gambardella (1996) [2] they described an artificial ant colony capable of solving the TSP. Ants of the artificial colony are able to generate successively shorter feasible tours by using information accumulated in the form of a pheromone trail deposited on the edges of the TSP graph. Computer simulations demonstrate that the artificial ant colony is capable of generating good solutions to both symmetric and asymmetric instances of the TSP. The method is an example, like simulated annealing, neural networks, and evolutionary computation, of the successful use of a natural metaphor to design an optimization algorithm.

Basu and Ghosh (2008) [3] they review the Tabu Search literature on the TSP, point out trends in it, and bring out some interesting research gaps in their literature.

Hussain et al. (2017) [4] proposed a new crossover operator of genetic algorithm (GA) for TSP to minimize the total distance; this approach has been linked with path representation, which is the most natural way to represent a legal tour. Computational results are also reported with some traditional path representation methods like partially mapped and order crossovers along with new cycle crossover operator for some benchmark TSP instances and found improvements.

The main outlines of this paper are as follows: Section 2 discusses the background and formulation of TSP. Some heuristic methods to solve TSP are discussed in section 3. Branch and Bound Technique (BABT) which is introduced and used in section 4. Some local search methods to solve TSP are discussed in section 5. In section 6 we suggest some improvements for GA. In section 7 we proposed the hybrid GA. Reduce matrix using successive rules suggested in section 8. The optimal solution for Iraqi's cities problem discusses in section 9. Lastly, some conclusions are introduced in section 10.

2. TSP Background and Formulation

Since the 1950s, heuristic algorithms have been developed for finding an approximate solution for the TSP and other complex optimization problems in a reasonable time. Among the most widely used heuristic algorithms are evolutionary algorithms, swarm intelligence, and many others. These algorithms eventually have a strong impact on modern computer science research because they help researchers solve problems in a variety of domains for which solutions in their full generality cannot be found in a reasonable time, even with the world's fastest computers, being global search heuristics, and being easy to implement [5].

The mathematical formulation of TSP is as follows:

The cost (distance or time) between two cities i and j is marked with d_{ij}, then the total cost *C* is:

$$Min\, C = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \quad \text{s. t.}$$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n.$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1, \ldots, n.$$

x_{ij}=0 or 1, where n is the number of cities.[6]

3. Some Heuristic Methods to Solve TSP [7]

This section discusses two heuristic methods; Greedy method and Improved Minimum distance method.

The Greedy method (GRM) starts by sorting the edges by length, and always adding the shortest remaining available edge to the tour. The shortest edge is available if it is not yet added to the tour and if adding it would not create a 3-degree vertex or a cycle with edges less than n. This heuristics can be applied to run in $O(n^2\log(n))$ time.

The Minimizing Distance Method (MDM) is an efficient method for finding a good solution, but it has a weak point. This weak point has been manipulated by improved minimum distance method (IMDM) which is suggested in [7]. The IMDM has good achievement with high efficiency for solving TSP.

4. Branch and Bound Technique for Solving TSP

4.1 General Review of BABT

BAB technique (BABT) is most widely used in TSP by constructing a state space tree to find the optimal solution among all feasible solutions by taking the value of the objective function. Branch and bound was initially studies by Dantzig and a more description was provided by him in the applications of TSP. The BABT gives all feasible solutions by solving the problem, by trying the practical solution ad starting the value in the upper bound for finding the optimal solutions [8]. The general algorithm of the BABT is as follows:

Branch and Bound Technique (BABT) Algorithm [9]

Step 1: Choose a starting point.
Step 2: Choose one of the routes for that point.
Step 3: After choosing that route between the current point and unvisited point to add the distance. After doing that choose a new destination without choosing the same point.
Step 4: Keep doing this until we have gone through each point.
Step 5: Add up each distance of each subgroup.
Step 6: You will see the difference in routes and pick the smallest route.

4.2 Using BABT for Solving TSP with Two Models

This section discusses and the applying of BABT to solve TSP. It is very well known that BABT is one of the most important methods of the exact solution for COP. This method can act with different upper and lower bound to get very good results within a good time. The choosing process of upper bound (UB) and lower bound (LB) are figured as (UB-LB) this symbol of UB and LB we called it a model for BABT with notation BABT: UB-LB, we apply IMDM or GRM methods for finding UB.

The LB consists of two main parts such that LB equal sequenced nodes plus the unsequenced nodes, the sequence nodes: is the basic rout until the current node, while the unsequenced nodes: it's the subsequence obtained from all the cities after eliminating the sequence nodes which are obtained from applying IMDM method.

Now we will discuss two techniques for BAB, the first technique is the classical BABT with notation BABT1, The BABT1 algorithm is as follows:

BABT1 Algorithm

Step1: Input n, $D=[d_{ij}]$, i, j=1,...,n.
Step2: Calculate UB=Cost (N) using IMDM where N= {1,2,…,n}; k=0.
Step3: For each node in the search tree compute the LB= cost of sequencing nodes + cost of unsequenced nodes; where the cost of the unsequenced nodes is obtained by GRM, BABM or IMDM, k=k+1.
Step 4: Branch each node with LB $\leq$ UB for level k.
Step 5: If k < n then go to **step 3**.
Step 6: If the last level (k=n-2) of BAB algorithm, the optimal solution is obtained.
Step 7: Stop.

The second technique is similar for BABT1 but with modification. This modification includes finding a LB by branch form the least cost node and continue until getting the root node and calculate the LB, then we update the initial UB by the new LB, and then apply the same steps of BABT1. The BABT2 algorithm is as follows:

BABT2 Algorithm

Step 1: Input n, D=[d_{ij}], i, j=1,...,n.
Step 2: Calculate UB=Cost (N) using GRM where N= {1,2,…,n}, UB1=UB, k=0.
Step 3: Compute the New_LB= cost of sequencing nodes + cost of unsequenced nodes; where cost of unsequenced nodes is obtained by IMDM, if New_LB ≤ UB1 branch from this node and set UB1=New_LB, repeat until reaching the root node set UB=UB1, if all New_LB > UB1 then UB=UB1.
Step 4: For each node in the search, tree compute the LB is obtained by IMDM, k=k+1.
Step 5: Branch each node with LB ≤ UB for level k.
Step 6: If k < n then goto **step 4**.
Step 7: If the last level (k=n-2) of BAB algorithm, the optimal solution is obtained.
Step 8: Stop.

In the practical examples we will choose different n such that $5 \le n \le 2000$ with integer cost (distance) such that $d_{ij} \in [1,30]$ for $5 \le n \le 30$, $d_{ij} \in [1,100]$ for $30 < n \le 90$, $d_{ij} \in [1,150]$ for $90 < n \le 500$ and $d_{ij} \in [1,1000]$ for $500 < n \le 2000$. The most used important notations are:

- **C:** The cost of travel.
- **CT:** Complete time in seconds.
- **BT:** Is the time for the best solution that obtained by the Local Search Methods (LSM) in seconds.
- **R:** R∈[0,1].
- **E:** Difference between a set of methods with the first column in the table.
- **Iter:** Iteration range=[minimum Iteration, maximum Iteration].
- **Av-all:** Average for all examples.
- In comparison results showed in all the following tables the results are taken for an average of (3) examples.

Table-1 shows the comparison results between the complete enumeration method (CEM) from one side and with BABT1: IMDM-IMDM, BABT2: GRM-IMDM and IMDM from the other side for n=5,…,12, where the standard method is CEM.

Table 1-Comparison results between CEM with BABT1, BABT2 and IMDM for n=5,…,12.

n	CEM		BABT1			BABT2			IMDM		
	C	CT	C	E	CT	C	E	CT	C	E	CT
5	51	R	51	0	R	51	0	R	51.7	0.7	R
6	46.3	R	46.3	0	R	48	1.7	R	50.3	4	R
7	53	R	53	0	R	53	0	R	53	0	R
8	56.7	R	56.7	0	R	56.7	0	R	56.7	0	R
9	43.3	R	43.3	0	R	43.3	0	R	43.3	0	R
10	52.7	3.8	52.7	0	R	52.7	0	R	52.7	0	R
11	61.7	37.1	61.7	0	R	61.7	0	R	63.7	2	R
12	42.3	409.1	42.3	0	R	42.3	0	1.2	42.3	0	R
Av-all	50.9	56.7	50.9	0	R	51.1	0.2	R	51.7	0.8	R

From the results of Table-1 we notice that BABT1 is the best method from other methods so it is can be compared with other methods for n > 12.

Table-2 shows the comparison results between the BABT1: IMDM-IMDM from one side and with BABT2: GRM-IMDM and IMDM from the other side for n=13,…,20,25, where the standard method is BABT1.

Table 2-Comparison results between BABT1 with BABT2 and IMDM for n=13,…,20,25.

n	BABT1		BABT2			IMDM		
	C	CT	C	E	CT	C	E	CT
13	60.3	30.7	62	1.7	1.5	71	10.7	R
14	55	45.97	57	2	1.6	62	7	R
15	46.7	4.7	46.7	0	2.3	50.7	4	R
16	45	11.3	45	0	2.3	50.3	5.3	R
17	53.7	1.6	53.7	0	4.3	53.7	0	R
18	49.3	25.5	50	0.7	4.8	53	3.7	R
19	48.7	2.03	48.7	0	6.1	48.7	0	R
20	58.7	6.8	58.7	0	5.9	60	1.3	R
25	71.3	721	74.3	3	16.8	79.3	8	R
Av-all	54.3	94.4	55.1	0.8	5.1	58.7	4.4	R

Notice that from Table-2 that the BABT1 can be run for n ≤ 25 in reasonable time.

Table-3 shows the comparison results between the BABT2: GRM-IMDM from one side and with IMDM from the other side for n=30,…,80, where the standard method is BABT2.

Table 3-Comparison results between BABT2 with IMDM for n=30,…,80.

n	BABT2		IMDM		
	C	CT	C	E	CT
30	56	25	58	2	R
40	172.3	651.5	182	9.7	R
50	180.3	431.4	201.3	21	R
60	199.7	888.1	227	27.3	R
70	218.7	390.9	233.3	14.6	R
80	207.7	838.97	236.7	29	1.4
Av-all	172.5	537.7	189.7	17.3	R

5. Local Search Methods for Solving TSP

Metaheuristic (local search) algorithms are formally defined as algorithms inspired by nature and biological behaviors. They produce high quality solutions by applying a robust iterative generation process for exploring and exploiting the search space efficiently and effectively. Recently, metaheuristic algorithms seem to be hot and promising research areas. They can be applied to find near-optimal solutions in a reasonable time for different COP. Metaheuristic algorithms such as genetic algorithms (GA), particle swarm optimization (PSO), tabu search (TS), simulated annealing (SA), and ant colony optimizations (ACO) are widely used for solving the TSP [10].

5.1 Simulated Annealing

Simulated Annealing (SA) is a trajectory-based optimization technique. It is basically an iterative improvement strategy with a criterion that accepts higher cost configurations sometimes. The first attempt to apply SA for solving the COP was in the 80s of the last century. An overview of SA, its theoretical development, and application domains is shown in. SA was inspired by the physical annealing process of solids in which a solid is first heated and then cooled down slowly to reach a lower state of energy. Metropolis acceptance criterion, which models how thermodynamic systems moves from one state to another state, is used to determine whether the current solution is accepted or rejected [10].

The original Metropolis acceptance criterion was that the initial state of a thermodynamic system was chosen at energy (Cost or C) and temperature (Temperature or t). Holding constant t, the initial configuration of the system is perturbed to produce new configuration and the change in energy ΔC is calculated. The new configuration is accepted unconditionally if ΔC is negative whereas it is accepted if ΔC is positive with a probability given by the Boltzmann factor shown in (1) to avoid trapping in the local optima:

$$e^{-\Delta\mathrm{Cost}/\mathrm{Temperature}} \qquad \ldots(1)$$

This process is then repeated until reaching a good sampling statistics for the current temperature, and then the temperature is decreased and the process is repeated until a frozen state (free energy state) is reached at t= 0 [10]. Algorithm of SA as follows:

Simulated Annealing Algorithm [10]

```
Step 1: Input: Temperature, FinalTemperature, cooling rate, ch;
Step 2: ch' = ch; Cost = Evaluate (ch');
Step 3: while (Temperature > FinalTemperature) do
            ch1 = Mutate (ch');
            NewCost = Evaluate (ch1);
            ΔCost = NewCost −Cost;
            if (ΔCost ≤ 0) OR ($e^{-\Delta Cost/Temperature}$ > Rand) then
                Cost = NewCost;
                ch' = ch1;
            end
            Temperature = cooling rate × Temperature
        end
Step 4: Output: the best ch'.
```

Where cooling rate is 0.95, Temperature is 10000, and final Temperature is 0, Rand as a uniform random and number of generation is 5000 iteration

5.2 Genetic Algorithm

Genetic algorithms (classical genetic algorithm) (GA) are derivative free stochastic approach based on biological evolutionary processes proposed by Holland. In nature, the most suitable individuals are likely to survive and mate; therefore, the next generation should be healthier and fitter than the previous one. A lot of work and applications have been done about GAs in a frequently cited book by Golberg. GAs work with a population of chromosomes that are represented by some underlying parameters set codes [4], where P is numbers of chromosomes and Pc is the probability of crossover.

Genetic Algorithm [4]

Step 1: Create an initial population of P chromosomes and evaluate the fitness for each one.
Step 2: Choose Pc*P parents from the current population via chosen selection.
Step 3: Select two parents to create offspring using crossover operator.
Step 4: Apply mutation operators for minor changes in the results.
Step 5: Repeat **Steps 4** and **5** until all parents chosen are selected and mated, and the remain (1-Pc)*P chromosomes are initialized randomly.
Step 6: Replace old population with new one, with elitism for best chromosome.
Step 7: Evaluate the fitness of each chromosome in the new population.
Step 8: Terminate if the number of generations meets some upper bound; otherwise go to **Step2**.

6. Improving Genetic Algorithm

In this section, we will attempt to improve the GA. We suggest to choose F number of chromosomes of the population of GA, where F= $\lfloor(P/4)\rfloor$, these F chromosomes are improved using three techniques, these technologies can be considered as a combination of the crossover and mutation operators.

6.1 GA Operators

These techniques are implemented on the origin chromosome as follows:

1- Simple Inversion Crossover [11]: Simple inversion selects two points along the length of the chromosome, which is in at these points, and the sub string between these points is reversed.

2- Swap mutation [12]: In pairwise swap mutation, the residues at the randomly chosen two positions swapped.

3-Displacement mutation [12]: Displacement mutation pulls the first selected gene out of the set of string and reinserts it into a different place then sliding the substring down to form a new set of string.

When applying the three techniques on one chromosome we obtained three new chromosomes and the 4th is the origin chromosome. This procedure is called mixing crossover mutation algorithm (MCMA), which act as follows:

MCM Algorithm

```
(ch$_1$, ch$_2$, ch$_3$, ch$_4$)=MCM (ch)
        ch$_1$=Simple inversion crossover (ch);
        ch$_2$=Swap (ch);
        ch$_3$=Displacement (ch);
        ch$_4$=ch;
end;
```

The proposed improved Genetic algorithm steps are as follows:

6.2 Improved Genetic Algorithm (IGA)

Step 1: Create an initial population of P chromosomes and evaluate the fitness for each one.

Step 2: Choose F=$\lfloor(P/4)\rfloor$ best chromosomes from the current population.

Step 3: Call MCMA (chromosome).

Step 4: Repeat **Steps 3** until all the F chromosomes are finished.

Step 5: Replace old population with new one.

Step 6: Evaluate the fitness of each chromosome in the new population.

Step 7: Terminate if the number of generations meets some upper bound; otherwise go to **Step2**.

6.3 Results of Applying IGA

Before describing the results of applying IGA we have to demonstrate the most GA and IGA parameters. These parameters are as follows: the population size (pop_size)=30, Probability of crossover (Pc)=0.8, Probability of mutation (Pm)=0.005 and number of generation (NG)=2000 for n=5,…,9, NG=4000 for n=10,…,14, and NG=5000 for n≥15.

Remark (1): For the initial population of GA, SA, and other improved algorithms, the GRM is suggested to be used to obtain one of the population chromosomes.

Table-4 shows the comparison results between GA and IGA, where the standard method is GA for different n.

Table 4-Comparison results between GA and IGA for different n.

n	GA				IGA				
	C	Iter	BT	CT	C	E	Iter	BT	CT
10	54	[0,3453]	0.7	2.4	53.7	-0.3	[0,29]	0.02	1.7
50	318.7	[0,1503]	0.3	3.3	273	-45.7	[236,1363]	0.4	2.4
100	677	[0,2968]	0.8	3.9	586.3	-90.7	[880,4401]	1.2	2.6
500	1262.7	[0,0]	0.02	9.6	1175.3	-87.4	[1420,4539]	3.5	5.5
1000	7589.3	[0,0]	0.04	19	7238.7	-350.6	[2183,4604]	7.5	10.2
1500	8162	[0,0]	0.1	29.3	7942.3	-219.7	[2337,4172]	9.8	16.6
2000	9424.7	[0,0]	0.1	40	9067.7	-357	[254,1174]	4.1	23.6

The [0,0] it mean no improvement in the initial solution.

Table-5 shows the comparison results between the BABT1: IMDM-IMDM (or CEM since they are identical) from one side and with IGA and SA from the other side for n=5,…,20,25, where the standard method is BABT1.

Table 5-Comparison results between BABT1 with IGA and SA for n=5,…,20,25.

n	BABT1		IGA					SA				
	C	CT	C	E	Iter	BT	CT	C	E	Iter	BT	CT
5	51	R	51	0	[0,0]	0.01	R	51	0	[10,19]	0.01	R
6	46.3	R	46.3	0	[0,2]	0.01	R	48.3	2	[6,59]	0.01	R
7	53	R	53	0	[0,5]	0.01	R	58	5	[0,43]	0.01	R
8	56.7	R	56.7	0	[9,13]	0.01	R	62.3	5.6	[0,73]	0.01	R
9	43.3	R	44.3	1	[4,20]	0.02	R	51.3	8	[23,83]	0.01	R
10	52.7	R	53.7	1	[0,29]	0.02	1.7	61.7	9	[0,0]	0.01	R
11	61.7	R	65	3.3	[21,56]	0.03	1.6	75.3	13.6	[0,16]	0.01	R
12	42.3	R	52.3	10	[10,246]	0.1	1.7	56	13.7	[0,41]	0.01	R
13	60.3	30.7	67	6.7	[25.45]	0.03	1.8	78.3	18	[3,90]	0.01	R
14	55	45.97	60.7	5.7	[18,41]	0.02	1.7	71	16	[53,107]	0.01	R
15	46.7	4.7	59	12.3	[0,124]	0.03	2.2	60.3	13.6	[0,12]	0.01	R
16	45	11.3	53.7	8.7	[11,71]	0.03	2.1	63.3	18.3	[0,16]	0.01	R
17	53.7	1.6	58.7	5	[39,70]	0.03	2.1	68.3	14.6	[0,0]	0.01	R
18	49.3	25.5	63	13.7	[0,80]	0.03	2.1	67.3	18	[0,0]	0.01	R
19	48.7	2.03	56.7	8	[0,232]	0.04	2.1	60.7	12	[0,54]	0.01	R
20	58.7	6.8	68	9.3	[0,85]	0.02	2.3	68.7	10	[0,0]	0.01	R
25	71.3	721	94.3	23	[176,541]	0.2	2.2	107.3	36	[0,1597]	0.02	R

Figure-1 shows the comparison results of Table-5.

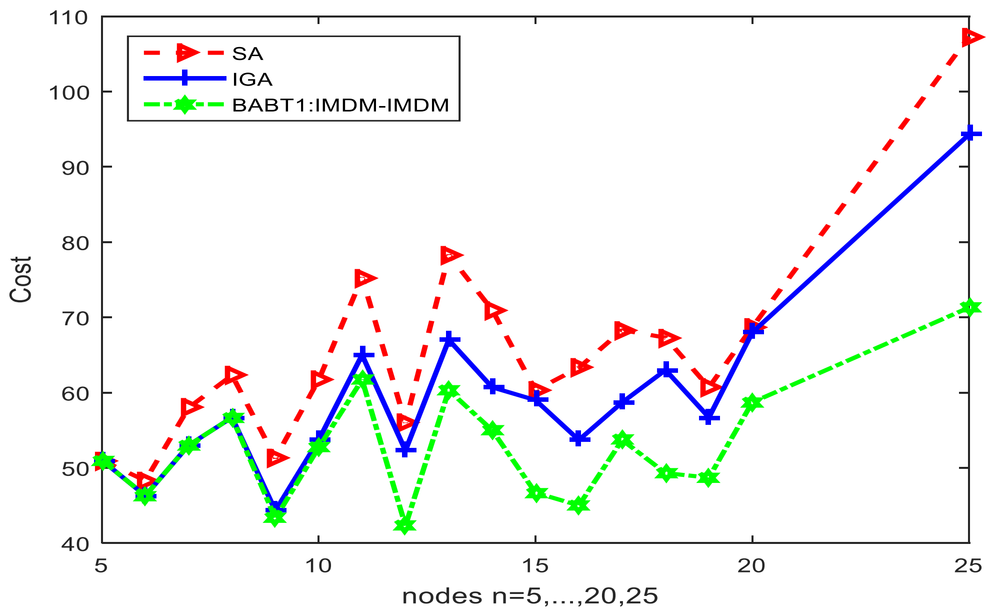

Figure 1-Comparison results between BABT1 with IGA and SA.

Table-6 shows the comparison results between the BABT2: GRM-IMDM from one side and with IGA and SA from the other side for n=30,…,80, where the standard method is BABT2.

Table 6-Comparison results between BABT2 with IGA and SA for n=30,…,80.

n	BABT2		IGA					SA				
	C	CT	C	E	Iter	BT	CT	C	E	Iter	BT	CT
30	56	25	86	30	[200,420]	0.2	2.3	90.7	34.7	[341,1864]	0.03	R
40	172.3	651.5	289	116.7	[535,1166]	0.4	2.3	295.7	123.4	[375,682]	0.02	R
50	180.3	431.4	273	92.7	[236,1363]	0.4	2.4	309	128.7	[0,507]	0.01	R
60	199.7	888.1	345	145.3	[2147,2392]	1.1	2.4	395.3	195.6	[992,3167]	0.1	R
70	218.7	390.9	366.7	148	[2200,4107]	1.7	2.5	407	188.3	[0,4818]	0.1	R
80	207.7	838.97	374.3	166.6	[2801,4983]	2.2	2.6	442.3	234.6	[2699,4163]	0.1	R

Figure-2 show the comparison results of Table-6.

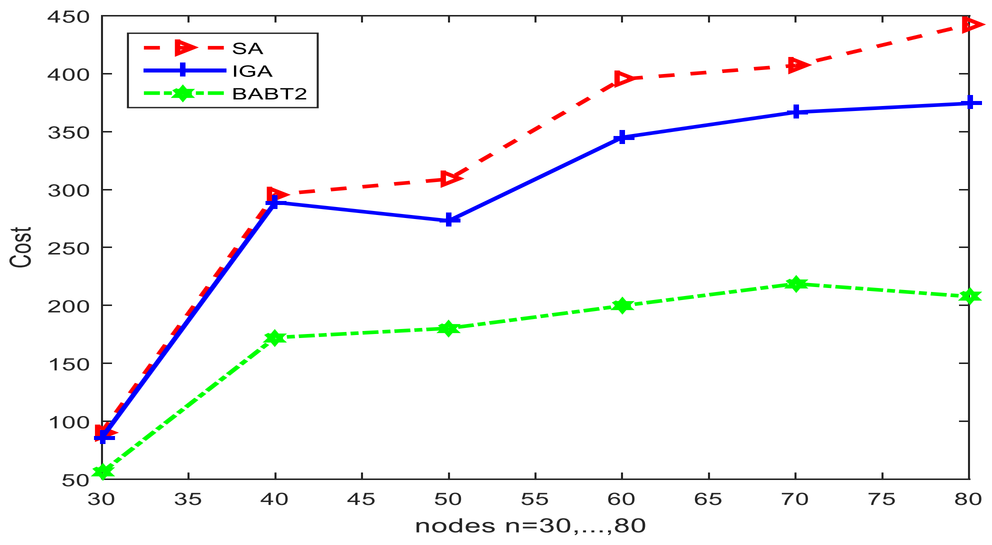

Figure 2-Comparison results between BABT2 with IGA and SA.

Table-7 shows the comparison results between the IMDM from one side and with IGA and SA from the other side for n=90,100,…,500, where the standard method is IMDM.

Table 7-Comparison results between IMDM with IGA and SA for n=90,100,…,500.

n	IMDM		IGA					SA				
	C	CT	C	E	Iter	BT	CT	C	E	Iter	BT	CT
90	247	1.9	401.3	154.3	[4334,3572]	2.1	2.6	456.3	209.3	[209,3412]	0.1	R
100	308.3	2.9	586.3	278	[880,4401]	1.2	2.6	601	292.7	[2883,3873]	0.1	R
200	387	18.4	795	408	[4444,4958]	3.1	3.3	845.7	458.7	[628,4238]	0.1	R
300	436.7	62.5	981.7	545	[3621,4256]	3.2	4.1	1055.3	618.6	[48,2515]	0.04	R
400	522.7	316.8	1091	568.3	[2387,4942]	3.7	4.9	1180	657.3	[1641,4141]	0.1	R
500	568.3	761.4	1175.3	607	[1420,4539]	3.5	5.5	1216.7	648.4	[3077,4301]	0.2	R

Figure-3show the comparison results of Table-7.

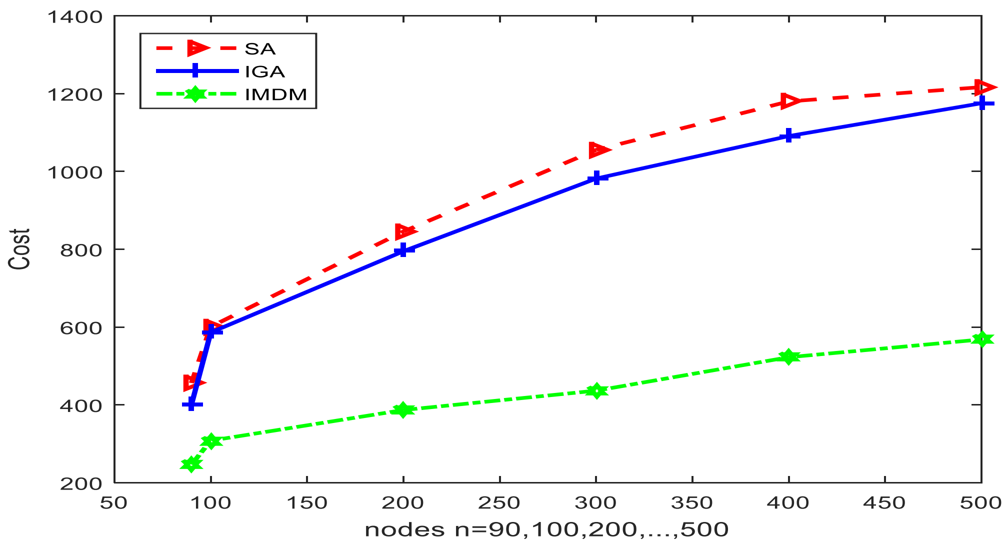

Figure 3-Comparison results between IMDM with IGA and SA.

7. Hybrid Genetic Algorithm

From classical GA and IGA, we see the good performance of IGA, but the IGA is still far from the results of some method like BABT or IMDM, especially for large n so that makes us suggest a hybrid between the IGA and SA to increase the performance of IGA. In this section we suggest to employ the SA in some important position of the IGA, so we suggest using the SA to improve chromosome which is the output of MCMA procedure. The suggested algorithm is called Hybrid GA (HGA) and the main steps as follows:

Hybrid Genetic Algorithm (HGA)

Step 1: Create an initial population of P chromosomes and evaluate the fitness for each one.
Step 2: Choose F=⌊(P/4)⌋ best chromosomes from the current population.
Step 3: Call MCMA(chromosome).
Step 4: Repeat **Steps 3** until all the F chromosomes are finished.
Step 5: For each chromosome of the population Call SA (chromosome).
Step 6: Replace old population with a new one.
Step 7: Evaluate the fitness of each chromosome in the new population.
Step 8: Terminate if the number of generations meets some upper bound; otherwise, go to **Step2**.

Table-8 shows the comparison results between the HGA from one side and with IGA and SA from the other side for different n, where the standard method is IGA.

Table 8-Comparison results between IGA and SA with HGA for different n.

n	IGA				SA				HGA				
	C	Iter	BT	CT	C	Iter	BT	CT	C	E	Iter	BT	CT
10	53.7	[0,29]	0.015	1.7	61.7	[0,0]	0.01	R	53.3	-0.4	[10,44]	0.3	32.3
50	273	[236,1363]	0.409	2.4	309	[0,575]	0.01	R	263.7	-9.3	[1508,2245]	16.7	45
100	586.3	[880,4401]	1.2	2.6	601	[2883,3873]	0.1	R	518.3	-68	[2379,3503]	30.3	49.9
500	1175.3	[1420,4539]	3.5	5.5	1216.7	[3077,4301]	0.2	R	1105.3	-70	[3152,4565]	75	99.8
1000	7238.7	[2183,4604]	7.5	10.2	7446	[997,1577]	0.1	R	6865.3	-373.4	[4387,4499]	152.4	172.5
1500	7942.3	[2337,4172]	9.8	16.6	8087.3	[0,4174]	0.2	R	7720.3	-222	[1411,4717]	190.5	263.4
2000	9067.7	[254,1174]	4.1	23.6	9277	[0,4294]	0.3	R	8606	-461.7	[1723,3224]	172.9	368.8

Figure-4 shows the comparison results of Table-8.

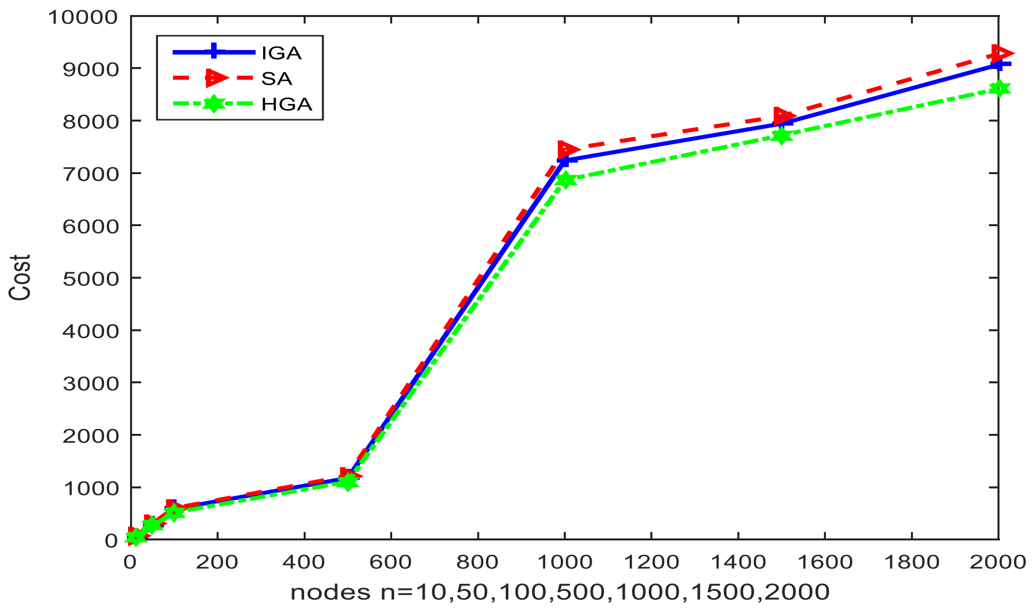

Figure 4-Comparison results between IGA with SA and HGA.

8. Reduce Matrix using Successive Rules

The successive rule (SR) plays a very important role in solving the combinatorial optimization problem (COP), especially TSP, these rules may be mandatory. The SR's will be helpful to reduce the numbers of cities that mean reduce the size of the problem, and this implies to reduce the required computation time to solve the problem. If the size of the TSP matrix is (n×n) and the number of SR's is m, then the size of the matrix after the reduction is (n-m)×(n-m). In order to use the SR, we suggest the following Matrix reduction algorithm:

Matrix Reduction Algorithm (MRA)

Step 1: Input n, D=[d_{ij}], i, j=1,...,n.

Step 2: Read m numbers of SR's (m<n), k=0.

Step 3: We have the SR $c_i \rightarrow c_j$, k=k+1.

Step 4: Reduce matrix D by Remove (cancel) two row's and two columns c_i and c_j to obtained the reduced matrix D′.

Step 5: Let row C'_r=row{c_j}/{$d_{c_j c_i}$}, column C'_c={col{c_i}/{$d_{c_j c_i}$}}+$d_{c_i c_j}$, add row C'_r and column C'_c in the last of the matrix D′.

Step 6: If k<m goto **step 3**.

Step 7: Print the reduced matrix D′ with dim (n-m×n-m).

Step 8: Stop.

The follows example shows matrix reduction.

Example 1-Let's have the following TSP:

	A	B	C	D	E	F	G
A	–	6	9	1	7	9	8
B	10	–	2	9	2	7	8
C	2	10	–	10	8	4	2
D	10	2	10	–	1	10	5
E	7	10	8	8	–	1	5
F	1	10	10	8	1	–	7
G	3	5	7	4	1	4	–

The optimal route (obtained from some exact method) for the above TSP table is: **A→D→B→C→G→E→F→A**, with C=10.

Now suppose we have the following SR: **(B→C)** for the same example, we treat the two cities **B** and **C** as a single city say **H**, so we obtain the following reduced table:

	A	D	E	F	G	$H=C_i \to B \to C$
A	−	1	7	9	8	6+2=8
D	10	−	1	10	5	2+2=4
E	7	8	−	1	5	10+2=12
F	1	8	1	−	7	10+2=12
G	3	4	1	4	−	5+2=7
$H=C \to C_i$	2	10	8	4	2	−

Where C_i =A, D, E, F and G according to i.

The optimal route for the above TSP table is: **A→D→H→G→E→F→A**, with C=10.

Now suppose we have the following SR's: **(B→C** and **E→F)** for the same example, we treat the two cities **E** and **F** as a single city say **I**, so we obtain the following reduced table:

	A	D	G	H	I
A	−	1	8	8	8
D	10	−	5	4	2
G	3	4	−	7	2
H	2	10	2	−	9
I	1	8	7	12	−

The optimal route for the above TSP table is: **A→D→H→G→I→A**, with C=10.

Remark (2): The symmetric and asymmetric matrices when applying MRA are transformed into an asymmetric matrix.

9. Optimal Solution for Iraqi's Cities Problem using Some Methods and LCM

In this section we exploit the TSP to evaluate the minimum total cost (distance or time) for Iraqi cities. So some methods are investigated to solve this problem; these methods are; Branch and Bound Technique (BABT), HGA, IGA, GA and SA.

9.1 Iraqi's Cities Problem (ICP) Definition

The Iraq's cities problem is asymmetric TSP. Iraq consists of 18 governorates, the traveling cost between each two governorates centers is known. We wish to find the minimum total cost of these cities starting from the capital city; Baghdad, then returns to it without repeat the path between any two cities. The symbol of each city is as in the following table:

City	**Baghdad**	**Baqubah**	**Diwaniyah**	**Hillah**	**Ramadi**	**Karbala**	**Najaf**	**Kut**	**Tikrit**
Symbol	Bg	Bq	Dw	Hl	Rm	Kb	Nj	Ku	Tk
City	Kirkuk	Samawah	Sulaymaniyah	Nasiriyah	Erbil	Amarah	Mosul	Duhok	Basrah
Symbol	Kr	Sm	Sl	Ns	Er	Am	Ms	Dh	Bs

In this subsection, we will use the TSP as an application to compute the minimum total cost for n=18 Iraqi's cites. First, we demonstrate the distance in km in a Table-9 for the 18-cites which represents the governorates centers [13].

Table 9-The distance (Km) between the Iraqi's cities

	Bg	**Bq**	**Kb**	**Hl**	**Rm**	**Dw**	**Nj**	**Ku**	**Tk**	**Kr**	**Sm**	**Sl**	**Ns**	**Er**	**Am**	**Ms**	**Dh**	**Bs**
Bg	−	**69**	**107**	**115**	**120**	**162**	**172**	**182**	**242**	**267**	**272**	**335**	**345**	**364**	**373**	**404**	**477**	**548**
Bq	**69**	−	**180**	**187**	**181**	**234**	**244**	**226**	**172**	**213**	**345**	**272**	**417**	**310**	**467**	**362**	**435**	**618**
Kb	**107**	**180**	−	**47**	**148**	**132**	**77**	**211**	**293**	**373**	**234**	**442**	**315**	**470**	**369**	**505**	**578**	**502**
Hl	**115**	**187**	**47**	−	**194**	**86**	**58**	**165**	**306**	**387**	**171**	**455**	**269**	**484**	**323**	**519**	**592**	**457**
Rm	**120**	**181**	**148**	**194**	−	**279**	**223**	**301**	**185**	**288**	**371**	**400**	**443**	**384**	**491**	**398**	**471**	**631**
Dw	**162**	**234**	**132**	**86**	**279**	−	**83**	**130**	**346**	**463**	**100**	**531**	**192**	**560**	**246**	**595**	**668**	**379**

Nj	**172**	**244**	**77**	**58**	**223**	**83**	**–**	**225**	**370**	**447**	**164**	**515**	**257**	**543**	**311**	**578**	**651**	**450**
Ku	**182**	**226**	**211**	**165**	**301**	**130**	**225**	**–**	**368**	**433**	**242**	**432**	**182**	**530**	**192**	**580**	**653**	**373**
Tk	**242**	**172**	**293**	**306**	**185**	**346**	**370**	**368**	**–**	**121**	**457**	**234**	**529**	**218**	**554**	**231**	**304**	**717**
Kr	**267**	**213**	**373**	**387**	**288**	**463**	**447**	**433**	**121**	**–**	**540**	**112**	**612**	**98**	**616**	**174**	**246**	**799**
Sm	**272**	**345**	**234**	**171**	**371**	**100**	**164**	**242**	**457**	**540**	**–**	**640**	**107**	**668**	**255**	**703**	**776**	**295**
Sl	**335**	**272**	**442**	**455**	**400**	**531**	**515**	**432**	**234**	**112**	**640**	**–**	**613**	**180**	**624**	**284**	**336**	**799**
Ns	**345**	**417**	**315**	**269**	**443**	**192**	**257**	**182**	**529**	**612**	**107**	**613**	**–**	**713**	**146**	**753**	**826**	**200**
Er	**364**	**310**	**470**	**484**	**384**	**560**	**543**	**530**	**218**	**98**	**668**	**180**	**713**	**–**	**722**	**85**	**164**	**900**
Am	**373**	**467**	**369**	**323**	**491**	**246**	**311**	**192**	**554**	**616**	**255**	**624**	**146**	**722**	**–**	**760**	**833**	**179**
Ms	**404**	**362**	**505**	**519**	**398**	**595**	**578**	**580**	**231**	**174**	**703**	**284**	**753**	**85**	**760**	**–**	**75**	**946**
Dh	**477**	**435**	**578**	**592**	**471**	**668**	**651**	**653**	**304**	**246**	**776**	**336**	**826**	**164**	**833**	**75**	**–**	**1016**
Bs	**548**	**618**	**502**	**457**	**631**	**379**	**450**	**373**	**717**	**799**	**295**	**799**	**200**	**900**	**179**	**946**	**1016**	**–**

In the same time, we have to estimate another cost which is represented by time factor by using distance cost mentioned in the Table-9 In order to estimate the time cost we have to use the following transformation:

$$T = M / V \qquad \ldots(2)$$

Where T is the time, M is the distance and V is the velocity factors respectively.

Table-10 describes the time cost in minutes depending on the distance cost mentioned in the Table-9 using constant velocity, V= 70 km/hour by car. (taking in consideration the Iraqi streets circumferences, where the specified constant velocity (V) is the minimum velocity)

Table 10-The Time (minute) between the Iraqi's cities

	Bg	**Bq**	**Kb**	**Hl**	**Rm**	**Dw**	**Nj**	**Ku**	**Tk**	**Kr**	**Sm**	**Sl**	**Ns**	**Er**	**Am**	**Ms**	**Dh**	**Bs**
Bg	**–**	**59**	**92**	**99**	**103**	**139**	**147**	**156**	**207**	**229**	**233**	**287**	**296**	**312**	**320**	**346**	**409**	**470**
Bq	**59**	**–**	**154**	**160**	**155**	**201**	**209**	**194**	**147**	**183**	**296**	**233**	**357**	**266**	**400**	**310**	**373**	**530**
Kb	**92**	**154**	**–**	**40**	**127**	**113**	**66**	**181**	**251**	**320**	**201**	**379**	**270**	**403**	**316**	**433**	**495**	**430**
Hl	**99**	**160**	**40**	**–**	**166**	**74**	**50**	**141**	**262**	**332**	**147**	**390**	**231**	**415**	**277**	**445**	**507**	**392**
Rm	**103**	**155**	**127**	**166**	**–**	**239**	**191**	**258**	**159**	**247**	**318**	**343**	**380**	**329**	**421**	**341**	**404**	**541**
Dw	**139**	**201**	**113**	**74**	**239**	**–**	**71**	**111**	**297**	**397**	**86**	**455**	**165**	**480**	**211**	**510**	**573**	**325**
Nj	**147**	**209**	**66**	**50**	**191**	**71**	**–**	**193**	**317**	**383**	**141**	**441**	**220**	**465**	**267**	**495**	**558**	**386**
Ku	**156**	**194**	**181**	**141**	**258**	**111**	**193**	**–**	**315**	**371**	**207**	**370**	**156**	**454**	**165**	**497**	**560**	**320**
Tk	**207**	**147**	**251**	**262**	**159**	**297**	**317**	**315**	**–**	**104**	**392**	**201**	**453**	**187**	**475**	**198**	**261**	**615**
Kr	**229**	**183**	**320**	**332**	**247**	**397**	**383**	**371**	**104**	**–**	**463**	**96**	**525**	**84**	**528**	**149**	**211**	**685**
Sm	**233**	**296**	**201**	**147**	**318**	**86**	**141**	**207**	**392**	**463**	**–**	**549**	**92**	**573**	**219**	**603**	**665**	**253**

Sl	**287**	**233**	**379**	**390**	**343**	**455**	**441**	**370**	**201**	**96**	**549**	**–**	**525**	**154**	**535**	**243**	**288**	**685**
Ns	**296**	**357**	**270**	**231**	**380**	**165**	**220**	**156**	**453**	**525**	**92**	**525**	**–**	**611**	**125**	**645**	**708**	**171**
Er	**312**	**266**	**403**	**415**	**329**	**480**	**465**	**454**	**187**	**84**	**573**	**154**	**611**	**–**	**619**	**73**	**141**	**771**
Am	**320**	**400**	**316**	**277**	**421**	**211**	**267**	**165**	**475**	**528**	**219**	**535**	**125**	**619**	**–**	**651**	**714**	**153**
Ms	**346**	**310**	**433**	**445**	**341**	**510**	**495**	**497**	**198**	**149**	**603**	**243**	**645**	**73**	**651**	**–**	**64**	**811**
Dh	**409**	**373**	**495**	**507**	**404**	**573**	**558**	**560**	**261**	**211**	**665**	**288**	**708**	**141**	**714**	**64**	**–**	**871**
Bs	**470**	**530**	**430**	**392**	**541**	**325**	**386**	**320**	**615**	**685**	**253**	**685**	**171**	**771**	**153**	**811**	**871**	**–**

Now we suggest to apply more than one method like BABT1 (IMDM-IMDM), BABT2: (GRM-IMDM), IGA, HGA, GA and SA for distance cost in the Table-9 and time cost in the Table-10 for n=18. Before we describe the results we suggest using SR to be certain that what we obtain is an optimal path. These SR's are as follows:

1. Since the subpath **MDE**=**Ms→Dh→Er** is the only and the minimum cost of available path this mean that we have n=16 (Table-11)).

Table 11-The distance between 16-Iraqi's cities after applying SR (MDE) for Table-10

	Bg	**Bq**	**Kb**	**Hl**	**Rm**	**Dw**	**Nj**	**Ku**	**Tk**	**Kr**	**Sm**	**Sl**	**Ns**	**Am**	**Bs**	**MDE**
Bg	**–**	**69**	**107**	**115**	**120**	**162**	**172**	**182**	**242**	**267**	**272**	**335**	**345**	**373**	**548**	**643**
Bq	**69**	**–**	**180**	**187**	**181**	**234**	**244**	**226**	**172**	**213**	**345**	**272**	**417**	**467**	**618**	**601**
Kb	**107**	**180**	**–**	**47**	**148**	**132**	**77**	**211**	**293**	**373**	**234**	**442**	**315**	**369**	**502**	**744**
Hl	**115**	**187**	**47**	**–**	**194**	**86**	**58**	**165**	**306**	**387**	**171**	**455**	**269**	**323**	**457**	**758**
Rm	**120**	**181**	**148**	**194**	**–**	**279**	**223**	**301**	**185**	**288**	**371**	**400**	**443**	**491**	**631**	**637**
Dw	**162**	**234**	**132**	**86**	**279**	**–**	**83**	**130**	**346**	**463**	**100**	**531**	**192**	**246**	**379**	**834**
Nj	**172**	**244**	**77**	**58**	**223**	**83**	**–**	**225**	**370**	**447**	**164**	**515**	**257**	**311**	**450**	**817**
Ku	**182**	**226**	**211**	**165**	**301**	**130**	**225**	**–**	**368**	**433**	**242**	**432**	**182**	**192**	**373**	**819**
Tk	**242**	**172**	**293**	**306**	**185**	**346**	**370**	**368**	**–**	**121**	**457**	**234**	**529**	**554**	**717**	**470**
Kr	**267**	**213**	**373**	**387**	**288**	**463**	**447**	**433**	**121**	**–**	**540**	**112**	**612**	**616**	**799**	**413**
Sm	**272**	**345**	**234**	**171**	**371**	**100**	**164**	**242**	**457**	**540**	**–**	**640**	**107**	**255**	**295**	**942**
Sl	**335**	**272**	**442**	**455**	**400**	**531**	**515**	**432**	**234**	**112**	**640**	**–**	**613**	**624**	**799**	**523**
Ns	**345**	**417**	**315**	**269**	**443**	**192**	**257**	**182**	**529**	**612**	**107**	**613**	**–**	**146**	**200**	**992**
Am	**373**	**467**	**369**	**323**	**491**	**246**	**311**	**192**	**554**	**616**	**255**	**624**	**146**	**–**	**179**	**999**
Bs	**548**	**618**	**502**	**457**	**631**	**379**	**450**	**373**	**717**	**799**	**295**	**799**	**200**	**179**	**–**	**1185**
MDE	**364**	**310**	**470**	**484**	**384**	**560**	**543**	**530**	**218**	**98**	**668**	**180**	**713**	**722**	**900**	**–**

With the same idea can get a time matrix with SR (MDE).

2. Let's add another certain subpath to the previous one, **ABN**=**Am→Bs→Ns**, this mean we have n=14 (Table-12)).

Table 12-The distance between 14-Iraqi's cities after applying SR (**MDE** and **ABN**) for Table-10

	Bg	**Bq**	**Kb**	**Hl**	**Rm**	**Dw**	**Nj**	**Ku**	**Tk**	**Kr**	**Sm**	**Sl**	**MDE**	**ABN**
Bg	−	69	107	115	120	162	172	182	242	267	272	335	643	752
Bq	69	−	180	187	181	234	244	226	172	213	345	272	601	846
Kb	107	180	−	47	148	132	77	211	293	373	234	442	744	748
Hl	115	187	47	−	194	86	58	165	306	387	171	455	758	702
Rm	120	181	148	194	−	279	223	301	185	288	371	400	637	870
Dw	162	234	132	86	279	−	83	130	346	463	100	531	834	625
Nj	172	244	77	58	223	83	−	225	370	447	164	515	817	690
Ku	182	226	211	165	301	130	225	−	368	433	242	432	819	571
Tk	242	172	293	306	185	346	370	368	−	121	457	234	470	933
Kr	267	213	373	387	288	463	447	433	121	−	540	112	413	995
Sm	272	345	234	171	371	100	164	242	457	540	−	640	942	634
Sl	335	272	442	455	400	531	515	432	234	112	640	−	523	1003
MDE	364	310	470	484	384	560	543	530	218	98	668	180	−	1101
ABN	345	417	315	269	443	192	257	182	529	612	107	613	992	−

With the same idea can get a time matrix with SR (**MDE** and **ABN**).

Notice that the symmetric matrix in Table-10 converted to asymmetric matrices in Tables-(11, 12)

9.2 Comparison Results of BABT and LSM to Solve ICP

Table-13 shows the results of applying BABT1 and BABT2 methods for n=18 (WOSR) and for n=16 and 14 (WSR).

Table 13-The results of applying BABT1 and BABT2 methods for different n WSR and WOSR

Method	n								
	18			16			14		
	Distance	Time	CT	Distance	Time	CT	Distance	Time	CT
BABT1	2502	2145	8	2502	2145	2	2502	2145	1.6
BABT2	2502	2145	3	2502	2145	2	2502	2145	1.6

Table-14 shows the results of the HGA, IGA, GA and SA methods for n=18 (without SR).

Table 14-The results of the HGA, IGA, GA and SA methods for n=18 without SR.

LSM	n=18				
	Distance	Time	Iter	BT	CT
HGA	2502	2145	[14,15]	0.1	45
IGA	2502	2145	[67,76]	0.1	2
GA	2627	2262	[3878,3700]	2.3	4
SA	2736	2345	[297,295]	0.01	0.1

For the above optimal costs we have the following unique symmetric path:
"**Bg→Bq→Sl→Kr→Er→Dh→Ms→Tk→Rm→Kb→Hl→Nj→Dw→Sm→Ns→Bs→Am→ Ku→Bg**".

To verify our optimal result Dynamic Programming (DP) software obtained from [14] implemented to solve Iraqi cities problem and obtained the same results BABT1 and BABT2 in compotation time 26007s.

10. Conclusions

1. The IMDM serves a good method to solve TSP so it is used as an UB and LB for BABT for different n.

2. The results of the practical examples of TSP, proof that the BABT1 is better than BABT2 in cost for n ≤ 25, while BABT2 is better in time for different n.
3. IGA is better than GA and SA in terms of results for different n, and faster almost double than GA.
4. For the time criterion, we notice that SA is the best.
5. The HGA is better than the IGA in terms of results for different n.
6. In general among all the proposed algorithms in this paper, for n≤25 we conclude that BABT1 is best algorithm (Figure-1), while for 25<n≤80 the BABT2 is the best (Figure (2)), for 80<n≤500 we see that IMDM is the best (Figure-3), lastly, for 500<n≤ 2000 we obtained that HGA is the best (Figure-4)).
7. From Table-13, we can notice the time effect when applying the SR for the ICP, as the number of SR increased the time complexity of the problem will decrease.
8. As future work, we recommended using other local search methods (Tabu Search, Particle Swarm Optimization, Bees Algorithm,…, etc.) to solve TSP.

Reference

1. Srour A., Othman Z. A. and Hamdan A. R. **2014.** "A Water Flow-Like Algorithm for the Travelling Salesman Problem", Hindawi Publishing Corporation, *Advances in Computer Engineering*, Volume 2014, Article ID 436312, http://dx.doi.org/10.1155/2014/436312.
2. Dorigo M. and Franklin A. **1996.** "Ant colonies for the traveling salesman problem", Université Libre de Bruxelles Belgium.
3. Basu S. and Ghosh D. **2008.** "A Review of the Tabu Search Literature on Traveling Salesman Problems", Indian Institute of Management Ahmedabad – 380015, India.
4. Hussain A., Muhammad Y. S., Sajid M. N., Hussain I., Shoukry A. M. and Gani. S. **2017.** "Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator", Hindawi Computational Intelligence and Neuroscience, Article ID 7430125, https://doi.org /10.1155/ 2017/7430125.
5. Tsai C., Tseng S., Chiang M., Yang C. and Hong T. **2014.** "A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case", Hindawi Publishing Corporation, *The Scientific World Journal*, Volume 2014, Article ID 178621, http://dx.doi.org /10.1155/ 2014/178621.
6. Akandwanaho S. M., Adewumi A. O. and Adebiyi A. A. **2014.** "Solving Dynamic Traveling Salesman Problem Using Dynamic Gaussian Process Regression", Hindawi Publishing Corporation, *Journal of Applied Mathematics*, Volume 2014, Article ID 818529, http://dx.doi.org/ 10.1155/2014/818529.
7. Jassim S. M. and Ali F.H. **2018.** "New Improved Heuristic Method for Solving Travelling Salesman Problem", *Iraqi Journal of Science*, **59**(4C): 2289-2300.
8. Ranjana P. **2018.** "Travelling salesman problem using reduced algorithmic Branch and bound approach", *International Journal of Pure and Applied Mathematics*, 118(20): 419-424, url: http://www.ijpam.eu, Special Issue, 2018.
9. Pineda P. **2017.** "Implementation and solutions of the Traveling Salesman Problem (TSP) in R", In Partial Fulfillment of Stat 4395-Senior Project Department of Mathematics and Statistics Spring. 2017.
10. Mohsen A. M. **2016.** "Annealing Ant Colony Optimization with Mutation Operator for Solving TSP", Hindawi Publishing Co2rporation, *Computational Intelligence and Neuroscience*, Volume 2016, Article ID 8932896, http://dx.doi.org/10.1155/2016/8932896.
11. Hameed W. M. **2005.** "The Role of Crossover in Genetic Algorithms (GAs)", M.Sc. thesis Submitted to the College of Science, AL-Mustansiriyah University.
12. Hung C. H. **2016.** "A Genetic Simplified Swarm Algorithm For Optimizing N- Cities Open Loop Travelling Salesman Problem", M.Sc. Faculty Of Computer Science And Information Technology University Tun Hussein Onn Malaysia.
13. https://www.google.com/maps, 2018.
14. https://www.mathworks.com/matlabcentral/fileexchange/31454-dynamic-programming-solution-to-the-tsp, 2011.