*Int. J. Open Problems Compt. Math., Vol. 10, No. 2, June 2017*
*ISSN 1998-6262; Copyright © ICSRS Publication, 2017*
*www.i-csrs.org*

# Optimization Approach Based on Immigration Strategies for Symmetric Traveling Salesman Problem

**Chakir Tajani , Otman Abdoun**
Abdelmalek Essaadi University, Morocco

e-mail: chakir_tajani@hotmail.fr, abdoun@fpl.ma

**JaafarAbouchabaka,**
Ibn TofailUniversity, Morocco

e-mail: abouchabaka3@yahoo.fr

**Abstract**

***The Traveling Salesman Problem (TSP) is a combinatorial optimization problem of great importance which continues to interest several researchers in order to develop methods to achieve an optimal solution. Genetic algorithms (GAs) as meta-heuristic methods have been widely applied to this problem. Inspired by biological phenomena, we introduce two immigration operators, random immigration and structured memory immigration, forming two different algorithms. The performance of these algorithms is evaluated using benchmark datasets of symmetric TSP from TSPLIB library. The results of the proposed algorithms are compared with the standard genetic algorithm showing that the proposed algorithms improve the performance of GA in solving TSP problem effectively and specifically with the developed structured memory immigration.***

**Keywords**:*Combinatorial Problem; STSP; Genetic Algorithm; Immigration.*

# 1 Introduction

The Traveling Salesman Problem (TSP ) is one of the most famous problems in combinatorial optimization which consists of solving the problem of visiting all cities by a salesman with the condition that each city can only be visited once. Hence, the problem is to minimize the distance of the complete tour that the salesman may take. Several real world application of the TSP can behind, such as Vehicle Routing [1,2], Scheduling, Image Processing and Pattern Recognition [3].

The TSP problem is classified as an NP-complete problem such [4]. This explain the number of research and the methods developed and exploited to resolve it. There are some methods to find the approximate solutions [5, 6], but, these methods have exponential complexity, they take too much computing time or require too much memory. Then, meta-heuristics methods should be exploited.

Many exact and heuristic algorithms have been proposed to solve this problem in recent years such as branch-and-bound [7], neural network [8], [9] and artificial bee colony [10], an ant system with a cooperating agents strategy [11], a genetic algorithm [12],a scatter search with a particle filter strategy [13]; a particle swarm optimization [14,15,16], ant colony optimization [17,18]. The Bat algorithm [19]. This technique does not guarantee the best solution but it is to come as close as possible to the optimum value in a reasonable amount of time which is at most polynomial time.

The genetic algorithm is a one of the family of evolutionary algorithms [20]. The population of a genetic algorithm (GA) evolves by using genetic operators inspired by the evolutionary in biology. These algorithms were modeled on the natural evolution of species. We add to this evolution the concepts of observed properties of genetics (Selection, Crossover, Mutation, etc), from which the name Genetic Algorithm. They attracted the interest of many researchers, starting with Holland [21], who developed the basic principles of genetic algorithm, and Goldberg [22] that has used these principles to solve specific optimization problems. Other researchers have followed this path [23,24].

In a genetic algorithm, a population of individuals (possible solution) is randomly selected. These individuals are subject to several operations, called genetic operators (selection, crossover, mutation, insertion,...) to produce a new population containing in principle better individual. This population evolves more and more until a stopping criterion is satisfied and declaring obtaining optimal best solution. Thus; the performance of a genetic algorithm depends on the choice of operators who will intervene in the production of the new populations [25,26,30].

In the general structure of a genetic algorithm, we find various steps: Coding, method of selection, crossover and mutation operator and their probabilities,

insertion mechanism, and the stopping test. For each of these steps, there are several possibilities. The choice between these various possibilities allows creating several variants of genetic algorithms which can improve the GA. Several works are focused on the improvement of genetic operators, which has allowed the development of several adapted presentation, adapted crossover and mutation operators for TSP [27] and the comparison of their performance, and even the hybridization between two operators to benefit of their specificity and make the GA more efficient and accurate. But, it is well known that GAs gets stuck in local optima very often. One efficient way of avoiding this problem is maintaining the diversification in population. Then; an immigration operator which consists in randomly generating a finite number of individuals at regular intervals to replace a substantial percentage of the population [28], can be applied in addition to the usual genetic operator. In addition, knowing that the more different the immigrants are, the more progress and knowledge is brought.

We propose an immigration technique in which the immigrants are not random, but we adopt a technique based on structured memory immigration which consists in benefiting individuals exclude during the previous generations. Thus, a percentage of the most powerful individuals will immigrate after an interval of time instead of the same number of the lowest individuals in the last generation. The complexity of immigration is decreased by executing it only every several generations.

In this work, we are interested in the resolution of TSP by genetic algorithm. We will present each individual of population by the most adapted method of data representation, the path representation method, which is the most natural representation of a tour. The OX crossover operator and RSM mutation operator adapted to the TSP problem are used by introducing two immigration strategies in order to bring dynamism and then diversity to the current population to perform the algorithm and obtain a best optimal solution in a reduced number of iteration. This paper is organized as follows. The TSP model is presented in Section 2. The genetic algorithm with random immigration and structured memory strategies is prescribed. In Section 3, Computational experiments were performed through many standard instances of TSPLIB [29] showing that the immigration strategies perform the genetic algorithm, especially with the structured memory immigration.

# 2 The Problems description

Before describing the problem, the reason behind choosing the above mentioned TSP is that, they are very common and are derived from real world applications. Hence, the TSP provide a very good platform for testing the impact of an elite pool on the performance and generality (consistency and efficiency) of our proposed genetic algorithm.

The Traveling Salesman Problem (TSP) consist to finding the shortest closed route among n cities, having as input the complete distance matrix among all cities.

Let $d_{ij}$ be a non-negative integer that stands for the cost (distance) to travel from city i directly to city j.

If $d_{ij} = d_{ji}$ the problem is called symmetric traveling salesman problem (STSP).

The STSP subject of this paper can formally be stated as follows:

Given a set of cities and the cost of travel (or distance) between each possible pairs, the TSP, is to find the best possible way of visiting all the cities and returning to the starting point that minimize the travel cost (or travel distance).

The objective is to find a tour of the minimum total length, where the length is the sum of the costs of each arc in the tour.

The search space for the STSP is a set of permutations of *n* cities and the optimal solution is a permutation which yields the minimum cost of the tour.

In other words, an STSP of size *n*is defined by:

We consider a set of points $v = \{v_1, v_2, \dots., v_n\}$ which $v_i$ is a city.

The mathematical formulation of the ATSP is given by:

$$Min\,\{f(T), T = (T[1], T[2], \dots..., T[n], T[1])\} \quad (1),$$

Where;*f* is the evaluation function calculates the adaptation of each solution of the problem given by the following formula:

$$f = \sum_{i=1}^{i=n-1} d(T(i), T(i+1)) + d(T(n), T(1)) \quad (2),$$

With *d* define a symmetric distance matrix and *d(i, j)* the distance between the city $v_i$ and $v_j$ .

$T[i]$is a permutation on the set *{1, 2, . . . ,n}.T = (T[1], T[2], . . . . . , T[n], T[1])* is the scheduling form of a solution of the STSP.

In this study, we adopt the concept of calculating the distances between cities from [20], where the distance between city *i* and city *(i + 1)* is calculated as the Euclidean distance using the following equation:

$$d(T[i], T[i+]) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (3),$$

# 3 Proposed Immigration Approach

In this work, we consider the resolution of the TSP by Genetic Algorithms where we will present each individual by the most adapted and natural method of data representation, the path representation method (fig. 1), which is the most natural representation of a tour (a tour is encoded by an array of integers representing the successor and predecessor of each city).

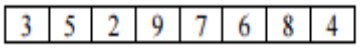


*Fig. 1.* Coding of a tour (3 5 2 9 7 6 8 4)

In this work, in the standard genetic algorithm, a population of individuals represented by "Path representation" is chosen at random and evaluated to select individuals who will undergo a set of genetic operators, namely, "*OX Crossover*" [25] and "*RSM Mutation*" [26]. Then, evaluation is maid for the new individuals (offspring's) to proceed to the insertion to give birth to a new population by serving the best individual of the old population by elitism, to give birth to a new population which will undergo the same procedure.

In the proposed approach, we introduce two strategies in the GA; the first is called "*Random Immigration*" after every generation and the second is called "*Structured Memory Immigration*" after a finite number of generations.

## 3.1 Crossover operator - OX -

The crossover in carried out according to the crossover probability $P_x$. In this study, we chose as crossover operator the *Ordered Crossover* method (OX) [25].

The Ordered Crossover method is presented by Goldberg [22], is used when the problem is of order based, for example in U-shaped assembly line balancing etc. Given two parent chromosomes, two random crossover points are selected partitioning them into a left, middle and right portion. The ordered two points crossover behaves in the following way: *child1* inherits its left and right section from *parent1*, and its middle section is determined.

We have chosen the operator OX as a crossover operator in this study because it is considered one of the best genetic operators used in the resolution of the traveling salesman problem [25].

**Input: Parents** $x_1=[x_{1,1},x_{1,2},\ldots\ldots,x_{1,n}]$ and $x_2=[x_{2,1},x_{2,2},\ldots\ldots,x_{2,n}]$
**Output:Children** $y_1=[y_{1,1},y_{1,2},\ldots\ldots,y_{1,n}]$ and $y_2=[y_{2,1},y_{2,2},\ldots\ldots,y_{2,n}]$
---
**Initialize**

- Initialize y1 and y2 being a empty genotypes;
- Choose two crossover points a and b such that $1\leq a\leq b\leq n$;

$j_1 = j_2 = k = b+1$;
$i = 1$;
**Repeat**
**if** $x_{1,i} \notin \{x_{2,a}, \ldots, x_{2,b}\}$ **then**
{
$y_{1,j1} = x_{1,k}$ ;
$j_1$++;
}
**if** $x_{2,i} \notin \{x_{1,a}, \ldots, x_{1,b}\}$ **then**
{
$y_{2,j1} = x_{2,k}$ ;
$j_2$++;
}
k=k+1;
**Until**$i \leq n$
$y_1 = [y_{1,1} \ldots\ldots y_{1,a-1}\ x_{2,a}\ \ldots\ldots x_{2,b}\ y_{1,a}\ \ldots\ldots y_{1,n-a}]$;
$y_2 = [y_{2,1} \ldots\ldots y_{2.a-1}\ x_{1,a}\ \ldots\ldots x_{1,b}\ y_{2,a}\ \ldots\ldots y_{2.n-a}]$;

*Fig.2.* Algorithm of crossover operator OX

## 3.2 Mutation operator - RSM -

In the reverse sequence mutation operator (RSM) [26], we take a sequence *S* limited by two positions *i* and *j* randomly chosen, such that $i < j$. The gene order in this sequence will be reversed by the same way as what has been covered in the previous operation. The algorithm (Fig. 3) shows the implementation of this mutation operator with example in (Fig 4.).

***Input:*** **Parents** $x_1=[x_{1,1},x_{1,2},\ldots\ldots,x_{1,n}]$ and $x_2=[x_{2,1},x_{2,2},\ldots\ldots,x_{2,n}]$
***Output:*****Children** $y_1=[y_{1,1},y_{1,2},\ldots\ldots,y_{1,n}]$ and $y_2=[y_{2,1},y_{2,2},\ldots\ldots,y_{2,n}]$
---

Choose two crossover points a and b such that $1\leq a\leq b\leq n$;
**Repeat**
***Permute*** $(x_a, x_b)$;
a = a + 1;
b = b − 1;
**until a<b**

*Fig.3.* Algorithm of RSM operator

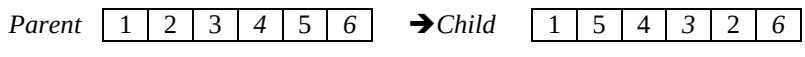


*Fig.4.* Mutation operator RSM

## 3.3 Immigration strategies

In addition to the standard operators of a genetic algorithm, namely the crossover and the mutation, we introduce a new operator called "*Immigration Operator*" which will introduce a diversity of the population and then more dynamism and exploration of different probable solutions of the problem, In order to obtain a better optimal solution compared to that obtained by the standard genetic algorithm and for the genetic algorithm based on the immigration operator.

Two strategies are proposed, the first one is called "*Random Immigration*" where the randomly created individuals are inserted into the population every generation by replacing the worst individuals or some individuals randomly selected individual (Fig.5.).

```
begin
Initialize population P randomly with constraints validation
evaluate population P
for (iitr=1; iitr<=iter; iitr++)
Sel:=Select For Reproduction(P)              // N individuals
CX := Crossover(Sel, Px)                     // Px is the crossover probability
Mut := Mutate(CX, Pm)                        // Pm is the mutation probability
Evaluate new individuals Mut                 // EvaluteMut and sort in descendant
   P' = Elitisme(Mut(1; N/2))                // Perform elitism
If  mod( iitr, ItInser) == 0 then            // Execution of RIGA
        Generate n random immigrants
        evaluate these random immigrants
        replace the worst individuals in P
endIf
endfor
end
```

*Fig.5.* Random immigration genetic algorithm - RIGA-

However, in order to benefit of the previous generations and of some individuals that not be able to be introduced in the population. After a defined interval of time (some generation), we give chance of the best individual to immigrate to the new population. This new operator is called "*Structured Memory Immigration Operator*" which is the second strategy (Fig. 6.).

This way, the introduced immigrants are more adapted to the current environment than the random immigrants. Then, the new operator introduces a diversity of the population and more dynamism and exploration of different probable solutions of the problem, in order to obtain a better optimal solution compared to that obtained by the standard genetic algorithm and for the genetic algorithm based on the immigration operator.

```
begin
Initialize population P randomly with constraints validation;
evaluate population P;
for(iitr=1; iitr<=iter; iitr++)
Sel:=Select For Reproduction (P)     // N individuals
CX  := Crossover (Sel, Px)           // Px is the crossover probability
Mut := Mutate (CX, Pm)               //Pm is the mutation probability
Evaluate new individuals Mut         // EvaluteMut and sort in descendant
    P' = Elitisme(Mut(1; N/2))       // Perform elitism
ImPop = (Mut(N/2; N))
ifiitr%ItInsert == 0 then            // Execution of SMIGA
        evaluate immigrants subpopulation ImPop
        replace the n worst individuals in P
endif
endfor
end
```

*Fig.6.* Structured memory immigration genetic algorithm -SMIGA-

# 4 Experimental results

## 4.1 Implementation and instances

The algorithms are coded in *C++* and run on a PC *HP Intel Corei31.7 GHz* and *4GB* of RAM.

In this article, the proposed genetic algorithm using the two strategies for the resolution of TSP is tested on two categories of STSP-instances: One category (C1) with a limited number of breakpoints which does not exceed a maximum of 42 jumps and the other (C2) with a wider length exceeding 100 breakpoints. Results are presented separately, starting with those obtained by small TSP instances: *BURMA14, RG17, ULYSSES22, FRI26, MATBAYS29 and DANTZIG42*. Then, the performance of the proposed approachis tested with large instances: *ATT48, ST70, PR76, KROA100* and *EIL101*. All instances used in numerical tests are standard instances of the TSP library [29]. The effectiveness of the Immigration Approach (*AIG*) "*Structured Memory Immigration Algorithm*", proposed in this paper, is demonstrated through comparisons with a *Variant Genetic Algorithm* (*VGA*) and Standard Immigration (*SIG*) called here "Random immigration genetic algorithm".

The genetic parameters of the *AIG, VGA* and *SIG* used to obtain the results presented are: *Npop = 10*, Select: *Roulette [25]*, *Niter = 100*, *Px = 0.6*, *Pm* $\in$ *[0.001, 0.2]* and Insert: *Elitism*. They were chosen after preliminary tests, in order to achieve a good compromise between calculation time and the quality of the solutions.

Indeed, the time to build a solution is not negligible; the total number of iterations must remain quite small. Diversification explores the space of solutions in order to search for a subset of individuals to obtain good results.

The results are given in table 1 and table 2. The first column indicates the TSP instances used. The other characteristics appearing in the tables present the results were given on average per subgroup of instances of equivalent size, for the proposed AIG method comparing it with other optimization methods: VGA and SIG. The two columns *Init* and *Bsol* expose respectively the initial solution and the optimal solution obtained after a well defined number of iterations.

## 4.2 Results for the first group of instances - C1-

Table 1. shows the numerical values found after execution of the new approach proposed in this paper and the similar values generated by standard immigration (SIG) and the VGA genetic variant on small (C1) TSP instances. The results show that the meta-heuristic immigration AIG improves those of the heuristic VGA and also SIG. The gain can even be quite significant, respectively *23.52%* and also *10.34%* on a small instance like *Burma14*.

Thus, the effectiveness of the AIG method is sufficiently influential to induce a difference of convergence towards the optimal solution (*24.36%*). More precisely for the instance *RG17*, by way of indication, the solution of the VGA method is of a total cost equal to *3407* against *2577* in the solution of the AIG approach. On the *MATBAYS29* instance, the AIG finds *15528* as the optimum solution compared to *16797* provided by SIG.

With the application of the three methods VGA, SIG and AIG to the resolution of the first category (C1) of the small size TSP instances: *BURMA14, RG17, ULYSSES22, FRI26, MATBAYS29 and DANTZIG42* is represented respectively in fig.6(a), fig.6(b), fig.6(c), fig.6(d), fig.6(e) and fig.6(f).

Figure 6. illustrates the evolution of the cost function with respect to the number of iterations. The curves of the graph synthesize the average performance of the proposed method to solve the instances of type C1. Besides the fact that this figure highlights the need for AIG genetic immigration to obtain good quality solutions, according to the requirements retained, the AIG method can provide the optimal solution in a very reduced number of iterations.

*Table 1.* The obtained solution for small size TSP instances after 100 iterations

| | **VGA** | | **SIG** | | **AIG** | |
|---|---|---|---|---|---|---|
| | **Init** | **Bsol** | **Init** | **Bsol** | **Init** | **Bsol** |
| BURMA14 | 47 | 34 | 50 | 29 | 48 | 26 |
| RG17 | 3941 | 3407 | 4629 | 2822 | 3975 | 2577 |
| ULYSSES22 | 150 | 104 | 144 | 95 | 144 | 86 |
| FRI26 | 2390 | 1871 | 2308 | 1776 | 2399 | 1695 |
| MATBAYS29 | 22688 | 19100 | 22569 | 16797 | 24412 | 15528 |
| DANTZIG42 | 2722 | 2135 | 2605 | 2028 | 2729 | 1926 |

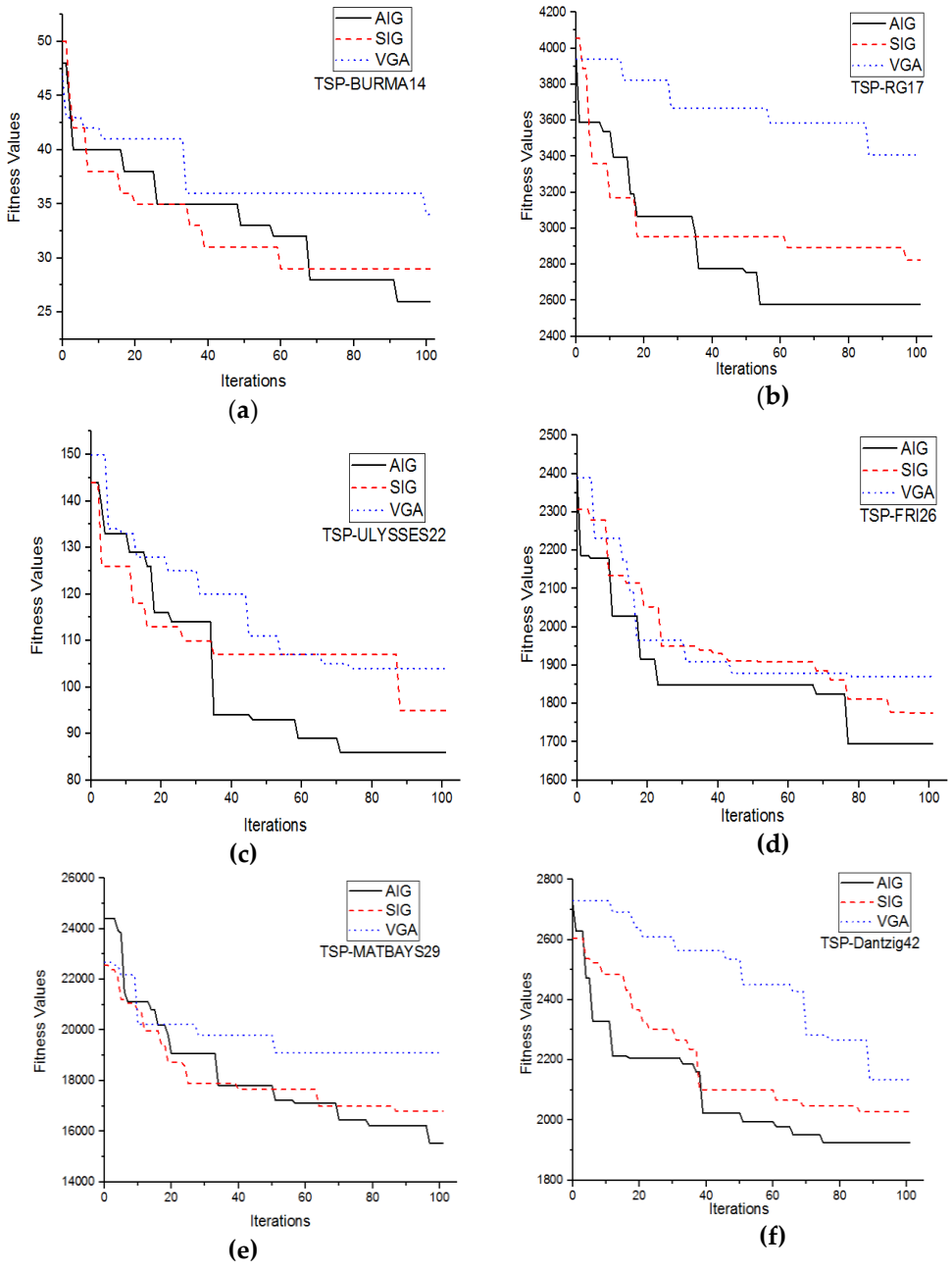


*Fig.6.* Convergence to the global optimum for small TSP instances:(a) TSP-BURMA14;(b) TSP-RG17; (c) TSP-ULYSSES22; (d) TSP-FRI26; (e) TSP-MATBAYS29; (f)TSP-DANTZIG42.

## 4.3 Results for the second group of instances - C2 -

To test the function of the proposed approach in the resolution of ATSP instances of different sizes, we have varied the deployment of the proposed approach with large (C2) instances, comparing its generated numerical result with the other VGA approaches and SIG.

Table 2. shows that AIG exceeds SIG and VGA performance in terms of cost. Once again, if on average the number of breakpoints in a TSP business traveler problem increases, can reach more than *100* jumps, a more careful observation shows that an increase in the size of the TSP instance does not necessarily influence the convergence of the proposed AIG approach to optimize the total cost of the solution.

With regard to calculation times, no method requires more than one second with less than *40* jumps. On the other hand, for the biggest instances, if VGA obtains results in less than 10 s, the immigration approach can sometimes require several seconds more. However, as in previous instances, the gain on cost is interesting and the calculation times are reasonable for strategic decisions.

The fig.7emphasize the effectiveness of the proposed AIG approach in solving all variants of the ATSP problem with even large instances, fig.7(e) and fig.7(f), in a reasonableiteration number.

The numerical results emphasize the importance of the proposed approach, consisting of a good selection of genetic operators enhanced by the integration of the improved immigration technique, which results in a good compromise between the total cost and the number of iteration in order to find the optimal solution.

The goal is not necessarily to try to reduce the execution CPUtime, but to find a good compromise between the number of iterations executed by the genetic approach and the values of the fitness function, so as to find a combination minimizing the total cost of the optimal solution.

The best quality of the solutions obtained by the AIG is however to the detriment of the average computing times which go from *6 seconds* for VGA to *17 seconds*. It should be noted, however, that the decisions taken are of a strategic type, in the long term, and therefore that the gains made deserve a few extra seconds.

*Table 2.* Values of optimum TSP instances of large size after 100 iterations

| | **VGA** | | **SIG** | | **AIG** | |
|---|---|---|---|---|---|---|
| | **Init** | **Bsol** | **Init** | **Bsol** | **Init** | **Bsol** |
| ATT48 | 151923 | 109174 | 139863 | 94492 | 140537 | 91776 |
| ST70 | 3477 | 2948 | 3411 | 2496 | 3498 | 2378 |
| PR76 | 546786 | 471677 | 538220 | 393494 | 539447 | 379843 |
| KROA100 | 163133 | 130308 | 158079 | 119650 | 158725 | 113962 |
| EIL101 | 3158 | 2595 | 3187 | 2471 | 3269 | 2383 |

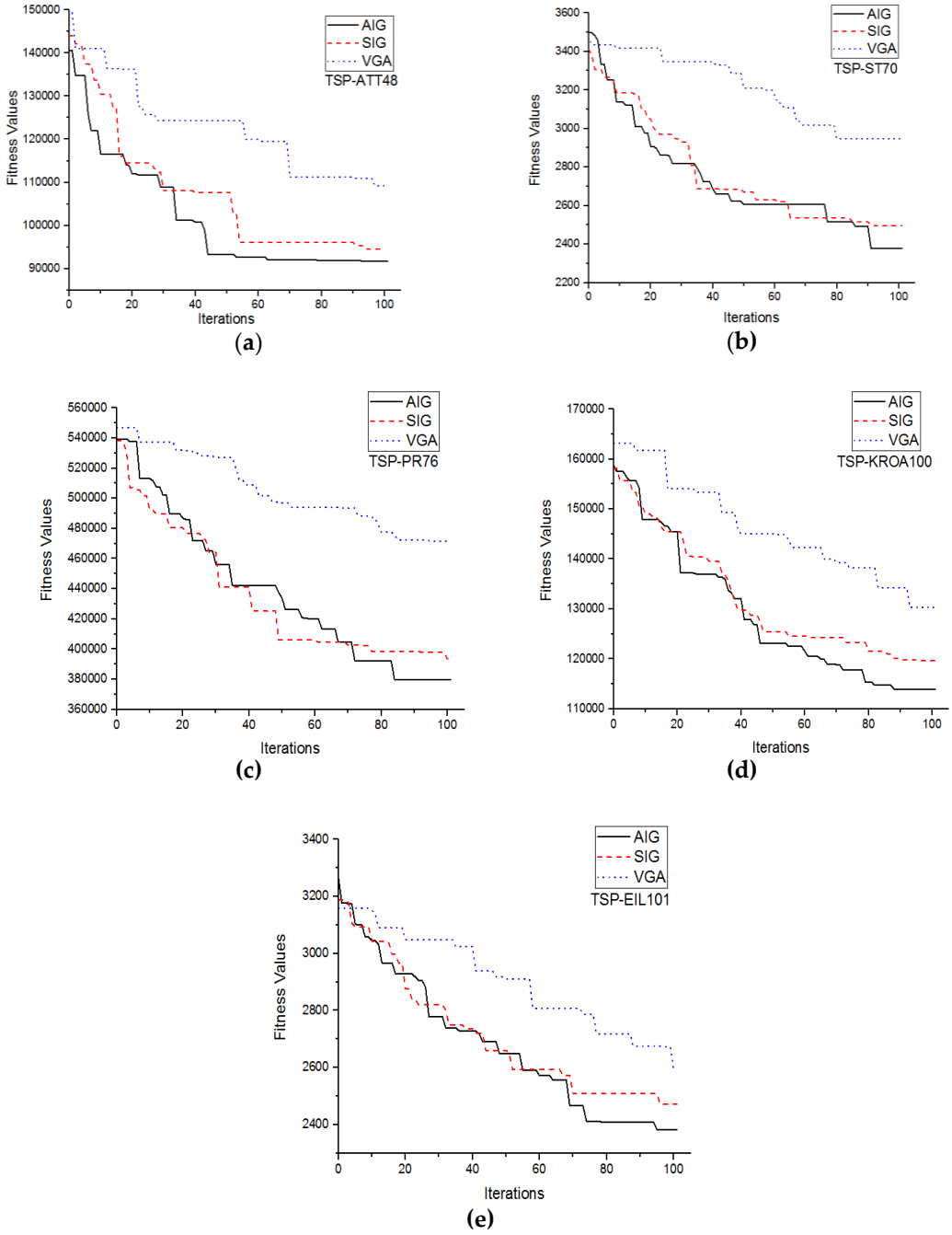


*Fig.7.* Convergence to optimum for large TSP instances:(a)TSP-ATT48; (b) TSP-ST70; (c) TSP-PR76; (d) TSP-KROA100; (e)TSP-EIL101.

# 5 Conclusion

In this work, in order to improve the performance of genetic algorithm to solve the symmetric traveling salesman problem as a typical problem of great importance in the real world. In addition to the standard operations of a GA, two immigration strategies are introduced "standard immigration" and "structured memory immigration". The results obtained with the standard instances of the TSPLIB show the operators effectiveness in bringing diversity to the genetic population and dynamism. Thus, thereafter, the performance of the solution even after a finite number of iterations and especially with the proposed operator "structured memory migration" which allows us to benefit, after a number of iterations, better old individuals who could not be inserted into previous populations.

# 6 Open Problem

Inspired by natural phenomena, we have introduced the immigration strategy that has improved the performance of the genetic algorithm. In future work, we want to introduce this strategy to other extensions of the TSP problem and especially to the DSTP. On the other hand, the adaptation of genetic algorithms to other transport problems can be envisaged with new representations of some transport problems or the coding of solutions to a great influence on the performance of AGs.

# References

[1] Y. Marinakis, M. Marinaki, and G. Dounias, "A hybrid particle swarm optimization algorithm for the vehicle routing problem", Engineering Applications of Artificial Intelligence, 2010, 23(4), p. 463-472.

[2] K Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the dubins vehicle," Automatic Control, IEEE Transactions, 2008, 53(6), p. 1378-1391,

[3] D. Banaszak, G. A. Dale, A. N. Watkins, and J. D. Jordan, "An optical technique for detecting fatigue cracks in aerospace structures", Proc. 18th ICIASF, 1999, p. 27/1-27/7.

[4] M. Garey, and D. Johnson, “Computers and Intractability”, W.H. Freeman, San Francisco, 1979.

[5] M. Dorigo, and L. M. Gambardella, "Ant colonies for the traveling salesman problem", BioSystems, 1997, 43, p. 73–81.

[6] A. Misevicius, "Using iterated Tabu search for the traveling salesman problem", Information Technology and Control, 2004, 3(32), p. 29–40.

[7] G. Finke, A. Claus, and E. Gunn, "A two-commodity network flow approach to the traveling salesman problem", CongressusNumerantium,1984, 41(1), p. 167-178.

[8] R. Pasti, and L. N. De Castro, "A neuro-immune network for solving the traveling salesman problem", in Neural Networks, 2006. IJCNN'06. International Joint Conference on. IEEE, 2006, p. 3760-3766.

[9] T. A. Masutti, and L. N. de Castro, "A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem", Information Sciences, 2009, 179(10), p. 1454-1468.

[10] D. Karaboga, and B. Gorkemli, "A combinatorial artificial bee colony algorithm for traveling salesman problem", in: 2011 international symposium on innovations in intelligent systems and applications (INISTA), 2011, p. 50–53.

[11] M. Dorigo, and V. Maniezzo, A. Colorni, "The ant system: optimization by a colony of cooperating agents", in: IEEE transactions on systems, man, and cybernetics– part B, 1996, 26, p. 29–42.

[12] J.H. Yang, C.G. Wu, H.P. Lee, Y.C. Liang, Solving traveling salesman problemsusing generalized chromosome genetic algorithm, Prog. Nat. Sci. 2008, 18, p. 887–892.

[13] J.J. Pantrigo, A. Duarte, Á. Sánchez, and R. Cabido, "Scatter search particle filter to solve the dynamic travelling salesman problem", in: G.R. Raidl, J. Gottlieb (Eds.), EvoCOP 2005, LNCS 3448, 2005, p. 177–189.

[14] Y. Marinakis, and M. Marinaki, "A hybrid genetic – particle swarm optimization algorithm for the vehicle routing problem", Expert Syst. Appl., 2010, 37 p. 1446–1455.

[15] K.P. Wang, L. Huang, C. G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem", in Machine Learning and Cybernetics, 2003 International Conference on. IEEE, 2003, 3, p. 1583-1585.

[16] Z. Yuan, L. Yang, Y. Wu, L. Liao, and G. Li, "Chaotic particle swarm optimization algorithm for traveling salesman problem", in Automation and Logistics, 2007 IEEE International Conference on. IEEE, 2007, p. 1121-1124.

[17] P.C. Chen, G. Kendall, and G.V. Berghe, "An ant based hyper-heuristic for the travelling tournament problem", in: Proceedings of the 2007 IEEE symposium oncomputational intelligence in scheduling (CI-Sched 2007), 2007.

[18] A. Puris, R. Bello, Y. Martinez, and A. Nowe, "Two-stage ant colony optimization forsolving the traveling salesman problem", in: J. Mira, J.R. Alvarez (Eds.), IWINAC2007, Part II, LNCS 4528, 2007, p. 307–316.

[19] X. S. Yang, "A new metaheuristic bat-inspired algorithm", in Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, 2010, p. 65-74.

[20] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic", European Journal of Operational Research, 2000, 126 (1), p. 106–130.

[21] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem", In Proc. of the second international conference on genetic algorithms (ICGA'87) Cambridge, MA: Massachusetts Institute of Technology; 1987.

[22] D. Goldberg, "Genetic Algorithm in Search, Optimization, and Machine Learning", Addison Wesley, 1989.

[23] L. Davis, D. Orvosh, A. Cox, and Y. Qiu, "A Genetic Algorithm for Survvivable", Network Design, ICGA, 1993, p. 408-415.

[24] Z. Michalewicz, "Genetic algorithms + data structures = evolution programs". Berlin: Springer, 1992.

[25] O. Abdoun, J. Abouchabaka, "A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem", International Journal of Computer Applications, 2011, 31(11), p. 49-57.

[26] O. Abdoun, C. Tajani, J. Abouchabaka, "Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem Analyzing", International Journal of Emerging Sciences, 2012, 2(1), p. 61-77.

[27] M. Albayrak, and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms", Expert Syst. Appl. 2008, 38(3) , p. 450–461.

[28] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, Cambridge, MA, 1992.

[29] G. Reinelt, The TSPLIB symmetric traveling salesman problem instances, 1995, Available in: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html.

[30] O. Abdoun, C. Tajani, J. Abouchabaka, and H. El Khatir, "Improved Genetic Algorithm to Solve Asymmetric Traveling Salesman Problem", Int. J. Open Problems Compt. Math., 2016, 9(4), p. 42-55.