Tsp Solving by Hybridization of GA and AS

Majid Joudaki, Mehdi Imani, and Niloofar Mazhari

***Abstract*—Traveling salesman problem (TSP) is one of the most famous combinational optimization problems. Today, many solutions have been offered by using different methods to solve this problem. Each one of these solutions has its own advantages and disadvantages and a comprehensive solution which proves itself as the most optimum one is not presented yet. But we are still waiting for better solutions which solve the problem in more optimum ways. In this paper we have proposed a hybridization solution using the two GA and AS for TSP solving and we have called it GA-AS. The results of this new solution which is presented in experimental results, shows that TSP is solved by using our proposed combinational solution (GA-AS) has better results than TSP solved only by using standard GA. Another new idea considered in this paper is to change the current GA generation in order to reach a better generation and therefore a better answer which is explained schematically.**

***Index Terms*—Genetic algorithm, ant system, traveling salesman problem**

I. Introduction

The Genetic Algorithm (GA) is an optimizing algorithm that models the processes of natural evolution[1]. The traditional GA usually consists of some operators such as crossover, selection and mutation which are simulated from biological and genetic processes. However, the traditional GA is inefficient for solving large optimization problems[2].

The GA is applied to the TSP which is a well known and important combinatorial optimization problem. In the TSP, each distance between two cities is given for a set of *n* cities. The goal is to find the shortest tour that visits each city exactly once and then returns to the starting city. Many books and papers introduce the procedure of how to find the shortest tour usually called as the optimum solution in TSP search algorithms[3].

A common application of the TSP is the movement of people, equipment and vehicles around tours of duty to minimize the total traveling cost. For example, in a school bus routing problem, it is required to schedule a school bus to pick up waiting students from the pre-specified locations. Post routing is another application of the TSP. The postman problem is modeled as traversing a given set of streets in a city, rather than visiting a set of specified locations. Moreover, the TSP plays an important role in general post problem, where the houses or streets are far away from each other[3].

A particularly successful metaheuristic is inspired by the behavior of real ants. Starting with Ant System, a number of algorithmic approaches based on the very same ideas were developed and applied with considerable success to a variety of combinatorial optimization problems from academic as well as from real-world applications [4].

The TSP is a typical example NP-hard combinatorial optimization problem which has pulls a very significant amount of research (Johnson & McGeoch, 1997; Lawler et al., 1985; Reinelt, 1994). The TSP has played a central role in ACO, because it was the application problem chosen when proposing the first ACO algorithm called Ant System (Dorigo, 1992; Dorigo, Maniezzo, & Colorni, 1991b, 1996) and it was used as a test problem for almost all ACO algorithms proposed later[4].

In this paper, after a brief explanation about the combination of evolutionary algorithms, we have classified the combinatorial algorithms. Then we have explained our proposed method and in the experimental results section we have proved our claim on GA-AS method being more optimum to resolve the TSP problem than standard GA method. Finally we have analyzed a new idea in GA algorithm schematically, in order to change the present generation to a better generation and therefore to find a better answer.

II. Combination of Evolutionary Algorithms

In recent years, much attention has been paid to the combination of algorithms to solve problems. Hybrid algorithms have proved their ability to find local optimal points. But still optimal use of hybrid algorithms is early in his way[5]. The combination of algorithms as experimental and change of the Commixtures of these algorithms has shown their efficiency, but there is still no clear and definite reason for amount of efficiency of a hybrid algorithm[6].

In order to justify the rational and scientific combination of algorithms first the search space structure must be investigated and efforts should be done to connect with the justifications and efficiency of algorithms. Then with regard to this knowledge, behavior of search algorithms will be investigated and some methods in designing hybrid algorithms should be represented[7].

III. Classifying Hybrid Algorithms

All evolutionary algorithms in fact simulate a natural process. The main character of this process is the population that process is working on, the other is convergence (i.e., the algorithm converging to a final goal or in other words all the candidate solution ways becoming one) is that it should be completely monotonous[9]. If this process began with a Random Population, after a certain period of time we will have a collection of members exactly similar to each other[6]. Monotony of members is related to fixing the amount of fitness capability average of population (another name of this fitness capability is fitness function). If we

Manuscript received October 13, 2012; revised November 28, 2012
Mehdi Imani is with Ktvc, Iran (e-mail:m.imani@gmail.com)

follow the changes of this amount during a certain time, we will clearly see that this stabilization happens very quickly. (See fig. 1)[6].

IV. Sequential Hybridization

With an experimental point of view, it will be determined that after a period of time, the population remains quite monotonous and its fitness amount will not increase much. Also the difference of fitness amounts related with each member of collection is very low. This shows that process has got stuck in a local optimal point and the possibility that it can escape from this local optimal point would be very low. These points lead us to identifying and solving the two aspects of problem[6]:

When the algorithm gets stuck in these points, it cannot distinguish between Global Optimum (point) and local optimal points. So we need to find a way to exit the local optimal points in order to find other optimal points.

Using the current optimal point in order to find any more effective and more impressive Global Optimum point in the near.

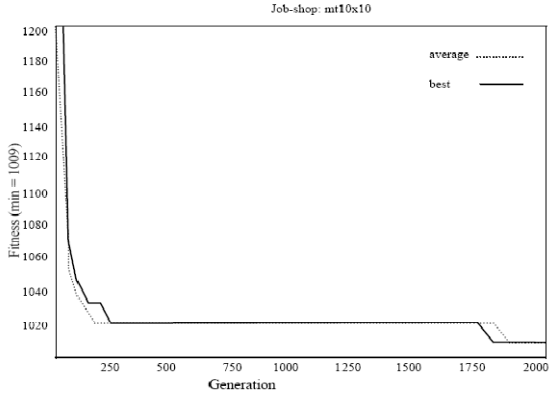

Fig. 1. Evolution during time of the mean fitness of the population of a genetic algorithm solving an instance of the Job-Shop Scheduling Problem (JSP, on instance MT10_10). We have represented the mean fitness of the population (plain line) as well as the fitness of the best solution found so far (dashed line). This evolution is typical, regardless of the problem that is solved, the representation of individuals, or the operators that are used. This evolution is a fundamental property of the process at work in Evolutionary algorithms[6].

First aspect may be solved by restarting EA with a new population in the hope that this algorithm will not get stuck in previous local optimal point. This purpose may be achieved by several different execution of the algorithm. about the second aspect, as experimental point of view, it is clear that other optimal points near to the found optimal point can be found by other algorithms (except EA) to be quite effective. Therefore using a local search algorithm, like Hill Climbing algorithm (HC) or Tabu Search (TS) or the combination of one of these two algorithms with EA can be effective and appropriate [9].

Such issues are suitable for the Sequential Hybridization (SH), i.e. using of a collection of algorithms that result of one algorithm is used as input to another algorithm. In this design, an Initial Population can be prepared for EA using a greedy algorithm. The considerable point in SH algorithm is the time that one algorithm stops and another algorithm starts. This starting and stopping time can lead to undesirable results if it takes a long time[9]. Experimentally, it should not be a long time. “Fig. 2,”

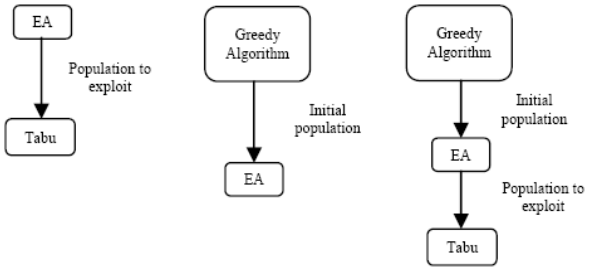

Fig. 2. Sequential hybridization three instances of hybridization scheme are represented. There may be more than three algorithms to be pipelined[6].

V. Sequential Hybridization of EA Algorithms

With regard to the problem size (TSP), parallel implementation of EA algorithms can be useful.

Therefore different EA implementation should be used.

A. Parallel Synchronous Hybridization (PSH)

The idea of this method is to replace EA operators with search methods. In this situation instead of using a blindly operator, related to fitness and applying it to main instances, a search algorithm is used. It investigates main instances, and replaces the old instances with new obtained ones. This search algorithm can be a SA algorithm or TS algorithm or any kind of them. Since different algorithms act exactly synchronous, this kind of algorithm is called PSH (See Fig. 3).

B. Parallel Asynchronous Hybridization (PAH)

In this kind of combination, several different algorithms are used in the search Space which cooperates with each other to find the optimal points. Finally PAH acts much better than algorithms which cooperate in hybridization.

There are two different type of PAH:

- Homogeneous**:** all colleague algorithms are similar.
- Heterogeneous**:** a state that different algorithms are being used.

Another point of view, we can investigate three types of cooperation:

- Global**:** in this position, all algorithms search similar search space. The goal here is to expand search space as much as possible. As an example, one Global PAH, consisting of EA and TS[10], has been proposed to solve the problem of designing network. The population used by EA is being updated by TS asynchronous algorithm. The best answer has been achieved by using several TS algorithms in generating an Elite Population. A similar strategy to expand the search space is using several different search algorithms, which any of them has its own view of the problem. Then the results of different searches are being compared with each other.
- Partial**:** in this case the main problem is divided to some sub-problems which any of them has its own search space. Then each algorithm is allocated to one of these search spaces.

Then each algorithm is allocated to one of these search spaces. In general, the combination of these sub-problems in order to find an optimal point is overshadowed by the limitations of algorithms which have participated in hybridization. Therefore algorithms cooperate with regard to

these limitations and find a single solution for the problem[6].

- Functional**:** in this case, algorithms solve different problems. For example, to solve Quadratic Assignment Problem (QAP) parallel TS are used, while a genetic algorithm (GA) has to formulate the problem to reach an optimal answer. A frequency memory saves information related to all of the observed solutions while executing TS. Referencing to this memory, GA generates solutions which are in unexpanded areas (see Fig. 4).
- The main point in all this kind of hybridization is the connection facility that allows different algorithms to exchange information (see Fig. 5). This aspect may be propounded as following:
- What elements are exchanged?
- How?
- How these elements are used by other algorithms?

The classification of various types of algorithms hybridizations has been shown in general schematic in Fig. 6.

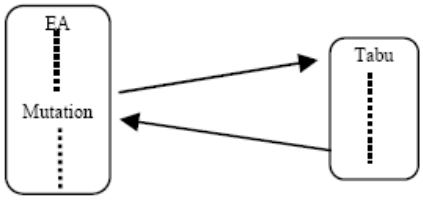

Fig. 3. Parallel synchronous hybridization. for instance, a tabu search is used as a mutation operator in an EA[6].

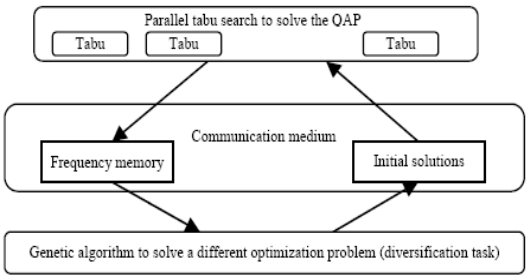

Fig. 4. Parallel asynchronous heterogeneous functional hybridization. several search algorithms solve different problems

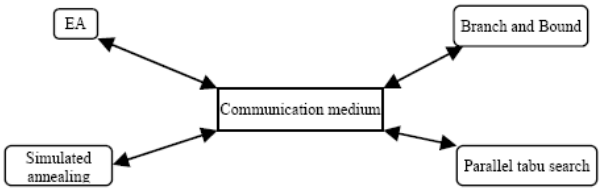

Fig. 5. Parallel asynchronous heterogeneous hybridization several search algorithms cooperate, co-adopt, and co-evolve a solution.

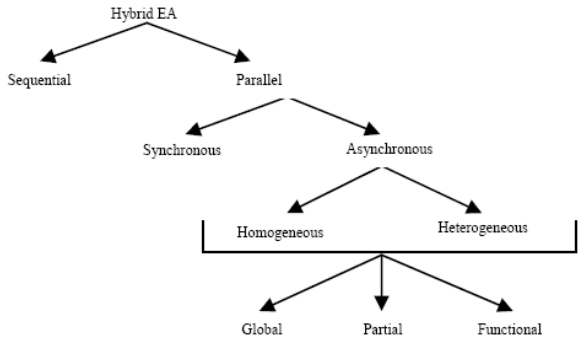

Fig. 6. A classification of hybrid EAs[6].

VI. GA-AS

In this paper, we have proposed a compound solution using the combination of Genetic Algorithm (GA) and Ant System (AS)[4] to solve the Traveling Salesman Problem (TSP). The results of this new solution shows that TSP solving by using this compound algorithm has better results than TSP solving only by using GA. In here, we have used hybrid 2optGA algorithm[9] for implementing GA algorithm.

The manner of using 2optGA is as follows:

- Mutation by using 2opt: 2opt method is one of the most famous methods in local search for TSP algorithms. This method improves the length of the tour by moving the edges of a Tour (by tour we mean a sub-Graf having cycle) and reversing the sub-tours (see Fig. 7). For example consider a tour like this[11]:

At first, $\overline{ab}$ and $\overline{cd}$ edges will be eliminated. Then direction of tour from b to c will be reversed, and $\overline{ac}$ and $\overline{bd}$ edges will be added to the tour. This action is useful while $\overline{ab} + \overline{cd} > \overline{ac} + \overline{bd}$ is correct. This action will be applied to all the pair of edges which have this condition and the length of the tour will be improved. This action will be repeated until there is no more improvement to be done[12].

- Greedy Crossover: While a 2opt method is applied to a solution, it is possible that the solution get stuck in the local optimal point. In this case applying 2opt has no influence on making the condition better[12]. For example, suppose that there are two tours, that each one is an optimal local solution for the main problem; the better solution will be achieved by combination of these two possible solutions (see Fig.8 and Fig.9).

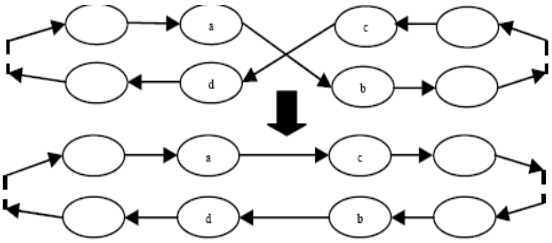

Fig.7. The 2opt method

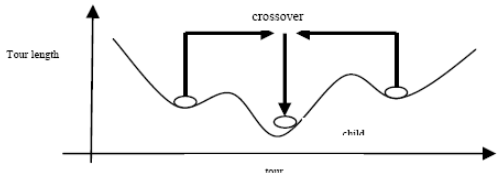

Fig. 8. Pop-up from local minima

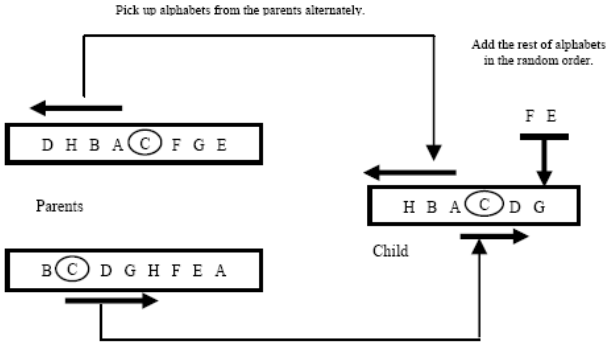

Fig. 9. Greedy subtour crossover

The combination of these two tours and creating new ones

is possible by using Greedy Crossover algorithm. The way this algorithm works is as follows[7], [9],

```
Algorithm: Greedy Subtour Crossover
Inputs: Chromosomes g_a = (a_0, a_1, ..., a_{n-1}) and g_b =
(b_0, b_1, ..., b_{n-1})
Outputs: The offspring chromosome g.
  Procedure crossover(g_a, g_b) {
   f_a ←true
   f_b ←true
   Choose town t randomly
   Choose x, where a_x = t
   Choose b, where b_y = t
   g ← t
   do{
        x ← x − 1 (mod n),
        y ← y + 1 (mod n),
        if f_a =true then {
           if a_x ∉ g then {
              g ← a_x . g,
            } else {
               f_a = false.
              }
         }
        if f_b = true then {
           if b_y ∉ g then {
              g ← g.b_y
            } else {
               f_b = false.
              }
         }
    } while f_a = true or f_b = true
   if |g| < |g_a| then {
     add the rest of towns to g in the
     random order
   }
   return g
}
```

As it is clear, GA starts with an initial random population and after operating Crossover and Mutation and producing the next generation, continues its work. After producing next generations as many as enough, the algorithm will be stopped and the best chromosome in last generation (a chromosome that has a higher fitness than the other ones) will be elected as a solution.

Other algorithm is Ant System (AS) which is called by GA in the process of producing initial random population. The initial generation in GA includes ants that have enough knowledge about the problem (which have better quantity of fitness function) and have been the best of their own generation. The hybridization of these solutions (ants) and creating new solutions by using GA can lead to better and more useful results.

VII. Experimental Results

We have compared our proposed method that is combinatorial GA-AS algorithm, with standard GA algorithm. In this comparison we have used random produced models to evaluate the effectiveness of these two algorithms (see Fig. 10). In presented chart the result of this comparison is shown. In this chart, the horizontal axis shows number of cities and vertical axis represents fitness value related to each of the algorithms. In both algorithms, this amount is obtained by reversing the Euclidean Distance, which is achieved by the algorithms (I.e. fitness value = 1/Euclidean Distance). According to the chart for all the tested instances, the results of our proposed algorithm are better than the standard GA results. The results are obtained by executing both of the algorithms on a similar system with the memory of 1 GB and core2due processor with the frequency of 2 GHz. Worth mentioning point in these observations is that in instances with less number of cities, executing time of every two algorithm (GA-AS, GA) is almost equal but with increasing the number of cities (problems with great size) execution time of the GA-AS method is considerably longer than the standard GA execution time and this is one of the weakness points of our proposal method (GA-AS). Certainly taking a long time in GA-AS method is due to AS which causes a considerable delay in execution of this algorithm. Another point which can be observed in the chart is that the number of anomalies in GA-AS chart is much less than the number of anomalies in the chart of GA and GA-AS chart has more predictable treatment.

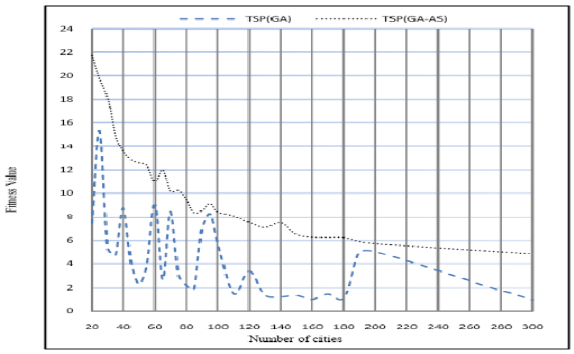

Fig. 10. Fitness changes (1/distance) against the increase in the number of cities. Comparing with the standard GA. our proposed algorithm has rather predictable results.

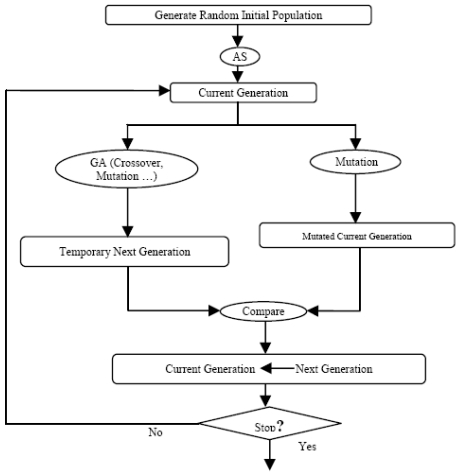

Fig. 11. Flowchart of the proposed algorithm and the manner of combining GA and AS.

VIII. Produce A Better Generation

Another new work that has been done in this paper is to change the present generation of GA in order to achieve a better generation and eventually to find a better answer. The manner of doing this work is that the present generation will be mutated to a new generation with a less possibility, and then the present generation's elements will be compared to the next generation's elements and the best elements of this collection will be elected for the next future generation and this action will be repeated for the other next generations (see Fig. 11).

Schematically, the way we do this work and reach a

solution to the case will be as following:

1) producing the initial population as random
2) Applying AS algorithm on the initial Population and improving proposed solutions
3) a) Applying GA algorithm and creating the next temporary generation
   b) Applying a mutation on the present generation and creating a better generation if possible
4) Comparing the result population of stage 3.1 with the result population of stage 3.2 and selecting the best chromosomes of both generations and creating the next generation
5) Replacing the current population with the new one
6) Considering the criteria of the problem and decide whether to continue solving the problem or to finish it.

IX. Conclusion

Two new ideas have been put forward in this paper. The first idea is to resolve the TSP problem in more optimum ways that we have called it GA-AS and we have proved our claim in experimental results section.

The second idea is to change the present generation of GA and to reach a generation with better chromosomes and eventually to find a better answer, which has been explained schematically. Research in Genetic Algorithm and Ant System is being continued and we are still waiting for more optimum solutions for TSP solving to be presented.

References

[1] S. B. Liu, K. M. Ng, and H. L. Ong, "A New Heuristic Algorithm for the Classical Symmetric Traveling Salesman Problem," in *Proceedings of world academy of science, engineering and technology*, vol. 21, 2007, pp. 267-271.

[2] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European journal of Operation Research*, vol. 137, 2002, pp.50-71.

[3] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, 2004.

[4] Marco Dorigo, Christian Blum, "Ant colony optimization theory: A survey," *Theoretical Computer Science*, vol. 344, 2005, pp. 243 – 278.

[5] C. Grosan, A. Abraham, and H Ishibuchi, "Hybrid Evolutionary Algorithms," *Computational Intelligence*, 2007, vol. 75.

[6] P. Preux and E. G. Talbi, "Towards hybrid evolutionary algorithms," *International Transactions in Operational Research*, vol. 6, pp. 557–570, 1999.

[7] C. M. White and G. G. Yen. "A Hybrid Evolutionary Algorithm for TSP," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1473–1478.

[8] P. Gang, I. Iimura, and S. Nakayama, "An Evolutionary Multiple Heuristic with Genetic Local Search for Solving TSP," *International Journal of Information Technology*, vol. 14, 2008, no. 2.

[9] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. "An Analysis of Several Heuristics for the Traveling Salesman Problem," *SIAM Journal on Computing*, vol. 6, 1977, pp. 563–581.

[10] T. Kaji. "Approach by Ant Tabu Agents for Traveling Salesman Problem," in *Proceedings of the IEEE International Conference on Systems*, Man, and Cybernetics, vol. 5, 2001, pp. 3429–3434.

[11] L. J. Schmitt and M. M. Amini, "Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study," *European journal of Operation Research*, vol. 108, 1998, pp. 551-570.

[12] S. Chatterjee, C. Carrera, and L. A. Lynch, "Genetic algorithms and traveling salesman problems," *European journal of Operation Research*, vol. 93, pp.490-510, 1996.