Proceedings of the IASTED International Conference
Applied Simulation and Modelling (ASM 2012)
June 25 - 27, 2012 Napoli, Italy

SIMULATION TOOL BASED ON A MEMETIC ALGORITHM TO SOLVE A REAL INSTANCE OF A DYNAMIC TSP

Eneko Osaba, Roberto Carballedo, Fernando Díaz, Asier Perallos
University of Deusto, Deusto Institute of Technology
Av. Universidades, 24, Bilbao, Spain
{e.osaba, Roberto.carballedo, fernando.diaz, perallos}@deusto.es

ABSTRACT
Nowadays, public transportation has become an essential area which affects our quality of life. Therefore, the design of new roads, new vehicles or new stations is a complicated process that requires a preliminary study to analyze its impact. This paper shows the algorithm of a simulation tool that allows the definition of transport routes, in regular and on-demand transportation systems. The resulting application allows adjustment and modification of routes, depending on passenger demand. All this is achieved through the use of a memetic algorithm that combines a genetic algorithm and tabu search. The result of the work done is a simulation tool and a memetic algorithm used for solving a particular instance of the Dynamic TSP.

KEY WORDS
Simulation tool, Evolutionary computing, Dynamic TSP, Intelligent Transport System.

1. Introduction

The traveling salesman problem is one of the most famous and studied problem in combinatorial optimization. The TSP was first proposed around 1800 by WR Hamilton and the British mathematician Thomas Kirkman [1]. Its definition is as follows: Given a finite number of cities or nodes and a cost associated with the trip from city i to j, the objective is to find a permutation of cities that minimizes the cost of visiting every city only once.

The TSP is known for being NP-Hard [2]. This means that there is no known algorithm that guarantees optimal solutions on instances with a large number of nodes in a reasonable runtime. Therefore, the strategies in this kind of problem are focused on finding good solutions that approximate to the optimal, taking into account the basic criteria of the computational complexity: Needed time and resources employed.

Over the history there have been many studies on the TSP [13, 14]. In the same way, there have been many variations of TSP such as the TSP with Pickup and Delivery (TSPPD) [17] or m-TSP [18]. Another widely studied variant is the Dynamic TSP (DTSP). The interest in this type of problem has increased in recent years [15, 16], due to the enhanced of the technology and computational resources that have enabled the implementation of these systems into real scenarios.

This paper is the result of the work done in the resolution of a real Dynamic TSP instance. To solve that problem a hybrid evolutionary algorithm has been designed. Our new algorithm (a.k.a memetic algorithm) combines a genetic algorithm and a tabu search technique. The problem in hand is an instance of a DTSP. The new algorithm is the core of a web application that allows dynamic generation of bus routes. In this application the customer demand is dynamic, and so, the application can make changes in the routes during its execution.

The remainder of this document is organized as follows: section 2 presents a general description of the DTSP. Section 3 introduces our problem instance while the section 4 describes algorithm design details. Finally, section 5 concludes this paper and gives directions for further research.

2. Dynamic TSP – DTSP

In the basic version of the TSP, all information about the demand and routing is available from the beginning; before the construction of routes and also before the moment of the execution of these routes. In the DTSP, some of this relevant information is not available at the time of designing the route. This information may be unknown or even altered during the process of construction or execution of a route.

The first reference to a DTSP belongs to Wilson and Colvin [3], who proposed a dynamic variation of the ARP [4], where the client requests, consist of trips from an origin to a particular destination, could appear dynamically.

In this type of problems, vehicles must serve two types of requests: advanced and immediate requests. The firsts are those that customers have made before the start of the routing process, so they are known from the beginning. Moreover, the immediate requests, introduced by Psaraftis

DOI: 10.2316/P.2012.776-029

[5], are those that are received dynamically and which appear in real time while the path is being executed by the vehicle. The management of immediate demands is often very complex, and it requires a real-time planning process. According to the flexibility of the problem [6] the insertion of new requests can be more or less complex. For example, in problems with time windows, the adding becomes more complex than in environments without them.

In these kinds of problems, the most common source of dynamism is given by the arrival of requests during the execution of the route. Normally, these requests may require the supply of goods [7], or the request for a travel service [8]. Recently, the dynamic travel time, which is very common factor in the real world, has been taken into account in many studies [9]. Finally, some recent studies give dynamism to the demand of certain customers already known [10] and the availability of the vehicle [11], where the possible breakdown provides the dynamism to the system.

To understand this dynamism, the following figure shows the execution of an instance of DTSP [12]:

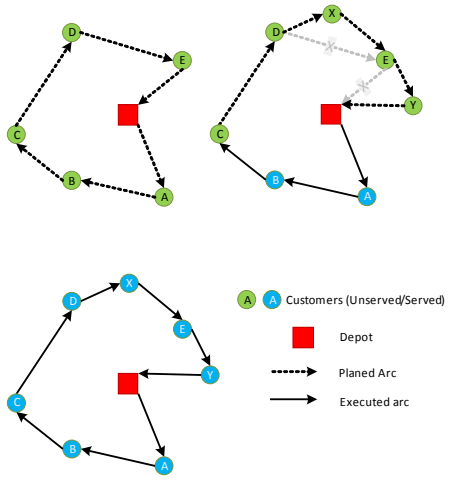

Figure 1: Example of DTSP

The picture shows that the initial route constructed with the known requests is {A, B, C, D, E}. While the vehicle is moving, two immediate requests come (X and Y), so, the initial route is modified to satisfy the demand of the new customers: {A, B, C, D, X, E, Y}.

Thanks to technological and computer advances, this kind of problem has gained popularity in recent years. Successful performance of this system requires special equipment in order to transmit information between the dispatch center and the vehicles. The introduction of the Global Positioning System (GPS), the development and expansion of smartphones, and the accuracy of the Geographic Information Systems (GIS) have made these systems to be applicable in real world scenarios.

The following figure shows the communication architecture of our simulation tool. This architecture enables bidirectional communication between vehicles and control center. This allows to know the position of the vehicles (obtained with a GPS system), and that the vehicles receive modifications of the routes in real time.

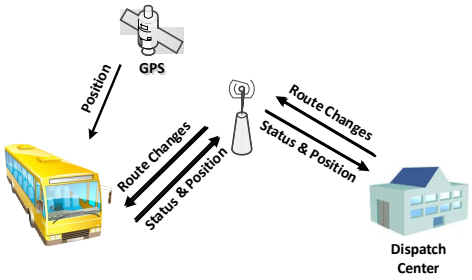

Figure 2: Communications Architecture

3. Description of our DTSP Problem

Nowadays there are many kinds of public transports, on demand or regular ones. This requires the public transport to be more demanding and sophisticated. But before performing variations to the regular service, by adding more transport units, new stops or creating new routes or lines, it is necessary to make preliminary studies to with the objective of improving the quality of service and the client satisfaction without making mistakes. To achieve this excellence different techniques and tools are used and applied before deploying the changes, in order to guarantee that the staff will be able to have an idea of the impact of the new variations.

In this work it has been developed a simulation tool, oriented to bus public transport, on demand or regular, in rural or urban environments. With this tool, different tests can be made to check the performance improvement when adding a new stop to a route or the creation of a new line.

With this tool the users will be able to create different environments composed of stations, that user may place wherever he desires. The application will calculate the optimized route to navigate through all the placed stations

using the artificial intelligence algorithm developed that will be further explained later. Once the route is created, the user may make requests, which the system will manage on an efficient way. The system will also be able to make a simulation of how the bus would be completing the route and managing the dynamic requests made in real time.

It's important to remark the difference between primary and secondary stations. The primary stations are the ones the bus must pass through, the secondary are the ones that will only be part of the route if it exists enough demanding from the users. This feature allows creating on demand transport lines, and helps to decide which stations will be secondary and the threshold that marks adding them to the route.

Figure 3, shows the application's main screen. In this screen several deployable tabs contain diverse information. The top ones for example, have information about the route of each of the buses and the history of the active bus. The bottom one contains a history of all the requests made, a panel to make transportation requests and some controls to manage the simulation of the bus tour.

Figure 3: application's main screen

Apart from this, a mobile application has been developed. Thanks to this, the users can view current bus route, in map or text format. Furthermore, it also offers the possibility of making requests in the same way as the web application. To finish, in Figure 4 a conceptual schema of the final system architecture is shown.

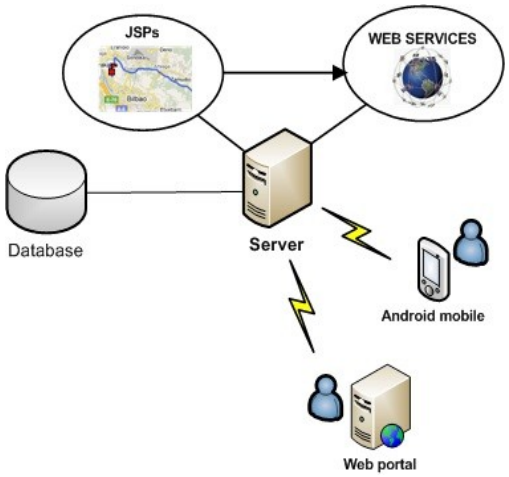

Figure 4: Structure of the application

4. Proposed memetic algorithm

Population-based algorithms are based on the laws of the evolution of species. These algorithms work with populations of chromosomes (solutions of the problem). Best chromosomes interact together to generate new chromosomes and these new individuals are added to the population. After a finite number of generations, or iterations, the algorithm ends its execution and returns the best chromosome in the population. This best chromosome represents the final solution to the problem in hand.

There are different types of population-based algorithms such as evolutionary or genetic algorithms. In the literature there are many hybrid approaches that combine different techniques. Two examples of this can be found in [19] and [20].

Our work has been focused in the design and the development of a new hybrid algorithm which combines a genetic algorithm with a tabu search strategy. The design of a new population-based algorithm involves primarily four major steps or phases: generation of the initial population, selection of the best chromosomes to be parents, crossover and mutation. Below the approach taken for the mentioned steps is described.

4.1 Generation of the initial population

As we explained before, each chromosome (or individual) represents a solution of the problem in hand. In particular, for a vehicle routing problem, the solution is a route that is defined by means of a string of numbers. One of the most important steps of a genetic algorithm is to create the individuals of the first or initial population.

In many cases, these initial chromosomes are generated randomly. But our intuition tells that the better the initial population, the better the following. For that reason, in the proposed algorithm we suggest a mix of randomness with the use of an initialization criterion. This technique generates some individuals using a well-known initialization technique, while other are generated randomly, until completion of the entire population. In this way the process begins with a population in which there are a number of acceptable solutions and a number of random solutions. We have implemented 4 initialization strategies:

Nearest neighbor criterion: The first location of the route is selected at random. Then, and repeatedly, the location closest to the last selected location is added to the route.

Insertion criterion: This strategy starts the process with a partial route composed of two cities. These cities are the two that are nearby one another, i.e. the closest locations. Below, a location closest to any of the locations that are already part of the partial path, is added to the partial route. Each selected location is inserted into the route in the position which supposes a smaller cost (or distance) increase.

Farthest insertion criterion: This is a modification of the previous strategy. After several studies, it has been demonstrated to be more efficient the selection of the location that is farthest from any location that is part of the partial route. Once a location is selected, it is inserted in the position involving a minor cost increase.

Solomon I1 insertion heuristic [21]: In 1987, Solomon proposed three heuristics for the VRPTW problem. Among all these, the so-called I1 is the one that offers the best results. This strategy begins its execution with the longer edge (locations that are farthest each other). Then, for each unrouted location, calculate the cost of inserting the location at any position on the current route. The location having the smaller increase in the cost is inserted in the path. The process continues until all locations are routed.

These 4 strategies were tested with different instances of TSP (obtained from TSPLIB [22] and [23]). The results of our tests are presented in Table 1. The tests showed up that the last two ones (farthest insertion and I1) offer the best results. This does not mean that either of the two other is not adequate to be used in our proposed algorithm, as demonstrated later.

Table 1
Initial solutions obtained by each criterion

TSP Instance	Oliver30	Eilon50	Eilon75
Optimal solution	420	421	535
Nearest neighbor *	509	566	620
Insertion	466	475	622
Farthest Insertion	**452**	485	**588**
Solomon I1	471	**464**	597

* These values are approximate because the result depends on the initial location (which is selected at random)

4.2 Parent selection

The crossover or crossing is the process in which the chromosomes in a population interact with each other to generate new individuals. Generated individuals are referred to as children, and as in natural laws, each child has to have pair of parents. As intuition suggests, the better the parents, the better the children. For that reason, parents selection process, is one of the most important steps of a genetic algorithm. There are many ways for the selection of the parents. In this case we have made use of an elitist method, which selects the best chromosomes in order to be parents.

4.3 Crossover process

In the process of crossing, as explained above, two chromosomes interact to form new chromosomes. There are plenty of crossover functions or operators. Below, we detail the crossover operators that we have implemented, although only one of them has been used for the final algorithm.

4.3.1 Order crossover (OX1)

The Order crossover was proposed by Davis [24]. This operator constructs the children by choosing small sub-routes of one's parent and maintaining the order of the locations of the remaining parent. For example, assuming these two individuals:

P = (1 2 3 4 5 6 7 8)
M = (2 4 6 8 7 5 3 1)

Now two cut-off points are randomly selected. These cut points are similar for both parents. Assuming that these cut-off points are between the position two and three, and the position five and six:

P = (1 2 | **3 4 5** | 6 7 8)
M = (2 4 | **6 8 7** | 5 3 1)

After that, two children (new solutions of the problem) are created. First, the segments between the cut-off points are preserved:

H1 = (* * | **3 4 5** | * * *)
H2 = (* * | **6 8 7** | * * *)

Finally, starting with the second cut-off point, the remaining locations are inserted in the same order they appear in the other relative, bearing in mind that the already routed locations are omitted. When the end of the route is reached, the process continues at the beginning of corresponding parent individual. The resulting children of this example are these ones:

H1 = (8 7 | **3 4 5** | 1 2 6)
H2 = (4 5 | **6 8 7** | 1 2 3)

4.3.2 Modified Order crossover (MOX)

This crossover was proposed by Shubhra [25]. The process starts by selecting a cut-off point that divides the parents into two sub-routes. Assuming the following parents:

P = (**1 2 3 4** | 6 9 8 5 7)
M = (**2 1 9 8** | 5 6 3 7 4)

The locations that are on the left of the cut-off point maintain the position they had in the original individual. In our example, the locations 1, 2, 3 and 4 of the first parent are in their original positions of the second child; the same happens for the locations 2, 1, 9 and 8 of the second parent:

H1 = (**1 2** * * * **9 8** * *)
H2 = (**2 1** * * * * **3** * **4**)

The remaining locations are inserted into the children in the order appearing in the opposite parent. Then, the resulting children would be the following:

H1 = (**1 2** 5 6 3 **9 8** 7 4)
H2 = (**2 1** 6 9 8 5 **3** 7 **4**)

4.3.3 Very Greedy crossover (VGX)

This operator presented by Bryant [26] is more "directed" than the previous two. In generating children, it that takes into account the distances between locations. First selects a location at random. Assuming that these two chromosomes are the parents:

P = (1 2 3 4 5 6 7 8)
M = (2 4 6 8 7 5 3 1)

Having selected location 2 as the initial portion of a new chromosome, the new child is:

H = (2 * * * * * * *)

Now, the unrouted locations that are adjacent (in both parents) to the last routed location are selected as potential locations. In this case, those locations are: 1, 3 and 4. From all possible locations, the one that is closed to the last routed location, added to the child. Assuming that location 4 is the closest to location 2, now the new child would be:

H = (2 4 * * * * * *)

This process is repeated until all locations have been routed. For example, in the next step, the possible nodes would be 3, 5 and 1. Between those three locations, the one that is closest to the location 4 is selected.

After testing these three operators, it was concluded that the VGX crossover is the most effective, so that is the one used in our genetic algorithm.

4.4 Mutation process

This process is executed after making all crossings, and it is performed on the resulting children. During the execution of the algorithm, the population is likely to be directed toward a local minimum. For this reason, once created new individuals, they undergo a mutation process with a certain probability. As in the nature, the aim of the mutation process is to evolve the chromosomes and ensure the diversity of the new population.

There are many ways to perform the mutation. The most common ones are based on the exchange of locations at random. In this case, we have implemented a mutation process that modifies each chromosome with a probability of 20 per cent. If the mutation is realized, a number of locations of the chromosome are modified. The number of changed locations is defined by a configuration parameter of the algorithm is called “MutationFactor”.

For example, assuming we have the following chromosome, composed by 10 locations:

C = (1 2 3 4 5 6 7 8 9 0)

Mutation factor is 0.2; i.e. 20% of the locations of the route will be modified. As the number of locations of the route is 10, 2 locations will be modified. The mutation process is performed as follows: first two locations are selected at random (e.g. 3 and 7), and their positions are exchanged. Then the process is repeated (locations 2 and 9 are exchanged):

C' = (1 2 **7** 4 5 6 **3** 8 9 0)
C'' = (1 **9** 7 4 5 6 3 8 **2** 0)

And the resulting chromosome will be as follows:

CM = (1 **9** **7** 4 5 6 **3** 8 **2** 0)

In order to preserve the quality of the population, both chromosomes (the original one and the mutated), are saved.

4.5 Application of the tabu algorithm

This step is the one that distinguishes this algorithm from a traditional genetic algorithm. After crossover and mutation process, with the aim of optimizing the new chromosomes, an execution of a tabu algorithm is applied. As in the mutation process, this process applies only to a small part of the population, in order not to increase excessively the execution time. This process is applied to each individual with 20% probability.

The tabu algorithm implemented has the following characteristics:

- **Movement criterion**: Vertex Insertion. In this function a node of the route is selected, this node is removed from the route and it is inserted into another position to generate a new solution [27].
- **Tabu criteria:** Node I. This is the strictest criterion. The ID of the last node used in the process of successor generation is inserted into tabu list. This node cannot be moved of its position until he leaves the tabu list [28].
- **Tabu list size and duration of tabu status:** The list will have a size of N / 4 where N is the number of nodes in the environment. With strict criteria, the size of the list has to be small. Several tests were performed in which it was concluded that N / 4 was the right size. The list has a queued process in which each movement is considered prohibited for a number of runs equal to the size of the list.
- **Aspiration criteria:** The aspiration criterion is an extended concept in the implementation of tabu algorithms. When this criterion is satisfied, allows the algorithm to execute prohibited or tabu movements. In the implemented algorithm, for example, there is an aspiration criterion, which allows tabu moves to be made if the execution of this movement can improve the current solution, and not return to the immediately preceding solution.

As can be seen, the algorithm has the same characteristics as an autonomous tabu algorithm, with the difference that the initial solution is not generated by the process itself, instead it is provided by the memetic algorithm.

In order to not increase excessively the execution time of the algorithm, the difference with an autonomous algorithm occurs in the number of iterations that have to be executed without the improvement of the best solution found so far. In the simple version, this value was close to the neighborhood. In this case, this value is equal to a fifth of the neighborhood, so, the execution time is reduced drastically.

4.6 Selection of survivors

The population in a memetic algorithm has a finite size. Therefore, it is necessary to reduce the number of chromosomes after crossover and mutation and discard those that are less interesting. This is accomplished by the selection of survivors function. In this case is used a function called ElitistRandomSurvaivals. This function selects the surviving population in two parts, one part will consist of the best chromosomes, while the other part will be selected at random. For example, if we have a population of 50 chromosomes, and we have to reduce it to 30 individuals, this function will select the 15 best chromosomes according to their fitness, and the remaining 15 will select at random from the 35 remaining.

The reason to select half of the population at random is to maintain some diversity, in order to avoid local optimums.

4.7 Results of the algorithm

These are the results obtained by the implemented algorithm by applying them to different known instances. Tests have been performed on a laptop Intel Core i5 - 2410, with 2.30 GHz and a 4 GB RAM.

There have been performed 50 iterations for each instance.

Table 2
Final results of the algorithm:

	Hits*	Media	Media %	Ex. Time
Oliver30	50/50	420	0%	0.3 seg
Eilon50	48/50	425.04	0.009%	11 seg.
Eilon75	40/50	535.3	0.05%	55 seg.
Eil101	50/50	629	0%	10 min.

*A hit is an execution in which the algorithm has found the optimal solution.

5. Conclusion and future work

The work presented is the result of a research project funded by the Basque government. The project focuses on the design of a software tool that assists in the creation of routes and schedules of passenger transport systems. For this, we have developed an application that is based on evolutionary computing techniques to simulate passenger demand and adjust the routes and frequency of the services to meet those demands. The result of work done is a software tool, and a metaheuristic algorithm that can be used for solving optimization problems.

We are currently working on adapting this algorithm to the problem DVRP. Apart from this, we plan to perform various studies to determine the influence of the mutation process in the algorithms and the influence of population survival functions.

References

[1] E.L. Lawler, J.K. Lenstra, K. Rinnooy and D.B. Shmoys. The Traveling Salesman Problem: A guided tour of combinatorial optimization. *Wiley - Interscience Publication*, 1985

[2] M.R. Garey, and D.S. Johnson. Computers and Intractability; a Guide to the Theory of Np-Completeness. *W. H. Freeman & Co*. 1990.

[3] N. Wilson and N. Colvin. Computer control of the Rochester dial-a-ride system. Technical Report R77-31, *Dept. of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts*. 1977.

[4] H.A. Eiselt, M. Gendreau and G. Laporte. Arc Routing Problems, Part 1: The Chinese Postman Problem. *Operations Research*, 43, 1995, 231-242.

[5] H. Psaraftis. A dynamic-programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14, 1980, 130-154.

[6] A. Attanasio, J.F. Cordeau, G. Ghiani and G. Laporte. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. *Parallel Computing*, 30, 2004 377-387.

[7] A. Beaundry, G. Laporte, T. Melo and S. Nickel. Dynamic transportation of patients in hospitals. *OR Spectrum*, 32, 2010, 77-107.

[8] H. –K. Chen, C. -F Hsueh and M.-S. Chang. The real-time time-dependent vehicle routing problem. *Transportation research Part E. Logistics and Transportation Review*, 42, 2006, 383-408.

[9] C. Novoa and R. Storer. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196, 2009, 509-515.

[10] Q. Mu, Z. Fu, J. Lysgaard and R. Eglese. Disruption management of the vehicle routing problem with vehicle breakdown. *Journal of the Operational Research Society*, 62, 2011, 742-749.

[11] V. Pillac, M. Gendreau, C. Guéret and A.L. Medaglia. A review of Dynamic Vehicle Routing Problem. *Industrial Engineering*, 2011, 0-28.

[12] A. Larsen, O.B.G. Madsen and M.M. Solomon. Recent Developments in Dynamic Vehicle Routing System. Golden, B., Raghavan, S. and Wasil, E. *The Vehicle Routing Problem: Latest Advances and New Challenges, Springer*, 2008, 199-218.

[13] S. Basu and D. Ghosh. A review of the tabu search literature on traveling salesman problem. *IIMA Working Papers 2008. Indian Institute of Management Ahmedabad*, 2008, 1-16.

[14] P. Larranaga, C.M.H. Kuijpers, R.H. Murga, I. Innza and S. Dizdarevic. Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review*,13, 1999, 129-170.

[15] W. Li. A Parallel Multi-start Search Algorithm for Dynamic Traveling Salesman Problem. *Lecture Notes in Computer Science, 6630/2011*, 2011, 65-75.

[16] Y. Song, Y. Qin and X. Chen. Dynamic TSP Optimization Base on Elastic Adjustment. *ICNC Fifth International Conference on Natural Computation*, 5, 2009, 205-210.

[17] I. Dumitrescu, S. Ropke, J. F. Cordeau and G. Laporte. The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming*, 121, 2010, 269-305.

[18] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *The International Journal of Management Science*, 34, 2006, 209-219.

[19] P.C. Pop, S. Iordache. A hybrid heuristic approach for solving the generalized traveling salesman problem. *Genetic and evolutionary computation conference, GECCO 2011*, 2011, 481-488.

[20] G. Gutin and D. Karapetyan. A memetic algorithm for the generalized traveling salesman problem. *International Journal of Natural Computing Research*, 9, 2010, 47-60.

[21] M.M Solomon. Algorithms for the vehicle routing and scheduling problems with time windows. *IFORMS Operations Research*, 35, 1987, 254-265.

[22] TSPLIB,comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[23] D. Whitley, T. Starkweather and D. Fuquay. Scheduling Problems and Traveling Salesmen The Genetic Edge Recombination Operator. *International Conference on Genetic Algorithms*, 3, 1989, 133-140.

[24] L. David. Applying Adaptive Algorithms to Epistatic Domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, 1985, 162-164.

[25] S.S. Ray, S. Bandyopadhyay and S.K. Pal. New Operators of Genetic Algorithm for Traveling Salesman Problem. *Proceedings of the 17th International Conference on Pattern Recognition*, 2, 2004, 497-500.

[26] B. A. Julstrom. Very Greedy Crossover in a Genetic Algorithm for the TSP. *Proceedings of the 1995 ACM symposium on Applied computing*, 1995, 324-328.

[27] F.A.T Montane and R.D Galvao. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operation Research*, 33, 2006, 595-619.

[28] M. Malek, M. Guruswamy, M. Pandya and H. Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21, 1989, 59-84.