A New Genetic Algorithm For VRPTW

Kenny Qili Zhu
kzhu@comp.nus.edu.sg
National University of Singapore

April 13, 2000

Abstract

This paper presents a new genetic algorithm to solve Vehicle Routing Problem with Time Windows (VRPTW) to near optimal solutions. The objective of the problem is to serve a number of customers within predefined time windows at minimum cost, without violating the capacity and total trip time constraints for each vehicle. Combinatorial optimization problems of this kind are NP-hard and are best solved by heuristics[6]. Our GA implementation uses intuitive integer string representation and incorporates several new crossover operations and other techniques such as hybrid hill-climbing and adaptive mutation scheme using statistical measures, and achieve good results, especially in clustered datasets.

Keywords: *Genetic Algorithm, Vehicle Routing Problem, heuristics, search*

1 Introduction

VRPTW is a well-known NP-hard problem[12] which is encountered very frequently by the logistic team of a private company that makes decision about the distribution of goods and services. The problem involves servicing a number of customers, at different geographic locations, with various demands, within specific time windows. A fleet of vehicles set off from a depot to serve the customers and return to the depot eventually. The objective of the problem is to find routes for the vehicles to service all the customers at a minimal cost (in terms of travel distance, etc.) without violating the capacity and travel time constraints of the vehicles and the time window constraints set by the customers. To date, there is no consistent optimizing algorithm that solves the problem exactly by mathematical programming. Instead, many heuristic methods have been proposed to solve VRPTW to near optima.

Genetic algorithm, originally developed by Holland[7], is an adaptive heuristic that simulates the optimization process with the natural evolution of genes in a population of organics. The GA maintains a population of candidate members over many generations. The population members are string entities of artificial chromosomes. Chromosomes are usually fixed length binary or integer strings. A special selection mechanism picks up parent chromosomes to go though a crossover and mutation procedure and produce some children to replace themselves. An new generation is formed with all the parents replaced. The termination criterion of GA is convergence within a tolerable number of generations. Joe L. Blanton Jr, et. al.[1] first used GA to solve VRPTW in 1993. Thangiah, et. al.[15] made use of GA in his "cluster first, route second" GIDEON system in 1995. However generally, GA has not been very successful in the application of VRPTW. The author, on the other hand, successfully implemented GA with the help

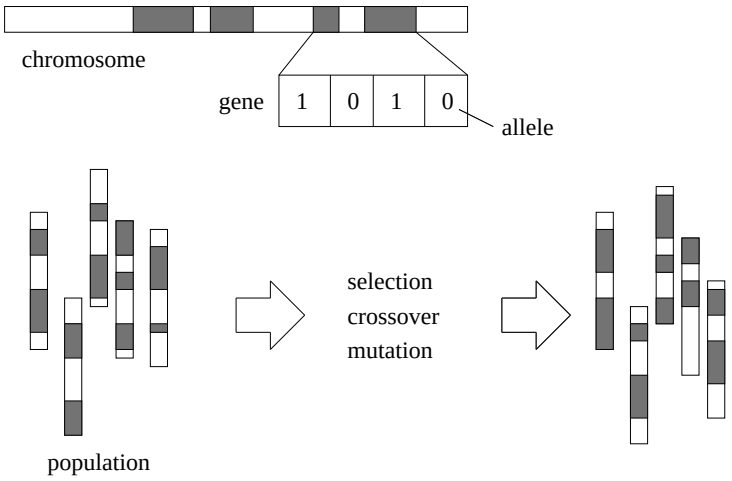

Figure 1: The basic concept of Genetic Algorithm

of new crossover operators and other local improvement techniques, and solved 56 Solomon benchmark problems[12] with better results.

2 Chromosome Representation

Like in other GA applications, the members of a population in our GA for VRPTW are string entities of artificial chromosomes. The representation of a solution we use here is an integer string of length N, where N is the number of customers in question. Each gene in the string, or chromosome, is the integer node number assigned to that customer originally. And the sequence of the genes in the chromosome is the order of visiting these customers. If we have the following solution:

Route No. 1 is 0 -> 3 -> 2 -> 4 -> 5 -> 0

Route No. 2 is 0 -> 10 -> 6 -> 1 -> 12 -> 11 -> 0

Route No. 3 is 0 -> 9 -> 8 -> 7 -> 0

The chromosome string that represent the solution is now:

$$3 - 2 - 4 - 5 - 10 - 6 - 1 - 12 - 11 - 9 - 8 - 7 \tag{1}$$

Note we link the last customer visited in route i with the first customer visited in route i+1 to form one string of all the routes involved. However we do not put any bit in the string to indicate the end of a route, because such delimiters in a chromosome adversely restrain the validity of children produced by crossover operations later. To decode the chromosome into route configurations, we simply insert the gene values into new routes sequentially, similar to PFIH. There is a chance that we may not get back the original routes after decoding, but it is generally assumed that minimizing the number of routes helps in minimizing the total travel cost, therefore packing a route to its maximum capability implies a potential good solution as a result.

3 Creation of Initial Population

To create an initial population for GA, we start off with a *GA seed*. GA seed is a single good, feasible solution to the problem by some heuristic. Here we make use of Push-Forward Insertion Heuristic (PFIH), an effective algorithm first introduced by Solomon[12] and later intensively used by Thangiah[16] in his VRPTW research. PFIH guarantees a feasible solution if there is one. Under the assumption that the solution from PFIH is reasonably good and in the vicinity of the global best solution, we create an initial population in relation to this solution. The way to do that is by letting the PFIH solution S_0 and its random neighbors $\forall S \in N_\lambda(S_0)$ describe a portion of the starting population. The rest of the population is generated totally on random basis, or unrelated to S_0. The reason for having this mixed population is obvious: a population of members entirely from the same neighborhood can not go too far from there and hence give up the opportunity of explore other regions; and a completely random population may be largely infeasible hence take a long time to converge. The proportion of relevant chromosomes and random chromosomes are governed by a parameter RAND_RATIO. The higher this ratio, the more diverse the initial population becomes. This parameter also reflects the confidence level of the user to the PFIH solution. If there is a high chance that global optimum is located in $N_\lambda(S_0)$, then it is undoubtedly economical to have a small RAND_RATIO so that the population converges to the optimum sooner. The total population size is set at 100 and the number of generations ranges between 500 to 1000, a compromise between computation time and final result. Certainly the more generations the program runs, the more optimized the solution in most cases, unless the optimum has already been reached.

4 Selection

After we have a population of candidates, we need some mechanism to select parents for mating and reproduction. The tournament selection mechanism is used for this purpose. In tournament selection, two identical copies of the population of size N are maintained at every generation. In the beginning, both populations are arbitrarily ranked. For population P_1, each pair of adjacent chromosomes (with indices $2i$ and $2i+1$) in the population are compared, the one with smaller fitness value qualifies to be a potential parent and let us call it f_i. After comparing all the pairs in P_1, we have N "fathers", namely f_0, f_1, ..., f_{N-1}. Repeat this process for population P_2, and we get m_0, m_1, ..., m_{N-1}, a set of "mothers" as well. Subsequently f_i and m_i are mated, for all $i = 0, 1, ..., N-1$. The procedure is graphically illustrated in Figure 2. The implication in this selection scheme is that genetically superior chromosomes are given priority in mating but average entities have some chances of be selected, too, provided they happen to be compared with a "worse-off" chromosomes.

5 Reproduction

Reproduction is one of the most crucial functions in the chain of biological evolution. Similarly, reproduction in GA serves the important purpose of combining the useful traits from parent chromosomes and pass them on to the offsprings. It is believed that an efficient and smart reproduction mechanism is largely responsible for high GA performances. The reproduction of GA consists of two kinds of operations, crossover and mutation.

Conventional single/double point crossover operations are relevant to string entities that are orderless,

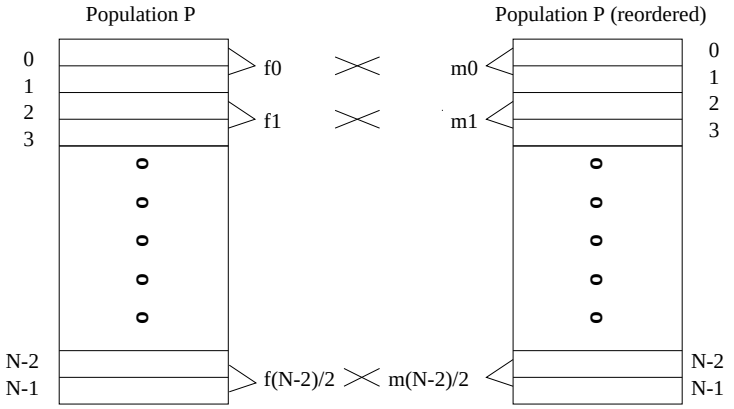

Figure 2: Tournament Selection

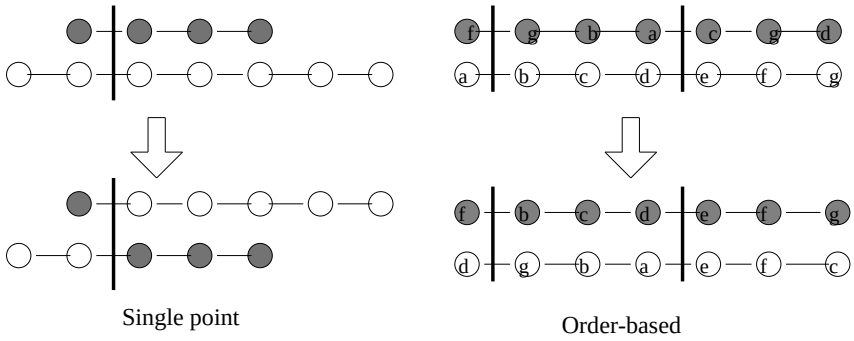

Figure 3: Simple Crossover vs. Ordered Crossover

or of different length. They put two integer/binary strings side by side and make a cut point (or two cut points) on both of them. A crossover is then completed by swapping the portions after the cut point (or between two cut points) in both strings (See Figure 3). In the context of VRPTW, where each integer gene appears only once in any chromosome, such simple procedure unavoidably produces invalid offsprings that have duplicated genes in one string. To prevent such invalid offsprings from being reproduced, we define a set of order-based crossover operators below.

The PMX Crossover[9] The PMX proceeds by choosing two cut points at random:

Parent 1: $h\ k\ c\ e\ f\ d \quad b\ l\ a \quad i\ g\ j$

Parent 2: $a\ b\ c\ d\ e\ f \quad g\ h\ i \quad j\ k\ l$

The cut-out section defines a series of swapping operations to be performed on the second parent. In the example case, we swap b with g, l with h and a with i, and end up with the following offspring:

Offspring: $i\ g\ c\ d\ e\ f \quad b\ l\ a \quad j\ k\ h$

Performing similar swapping on the first parent gives the other offspring:

Offspring: $l\ k\ c\ e\ f\ d \quad g\ h\ i \quad a\ b\ j$

Heuristic Crossover This operator is concerned with distances between nodes. In the following example, a random cut is made on two chromosomes and we look at the two genes b and g immediately after the cut points.

Parent 1: $h\ k\ c\ e\ f\ d \quad b\ l\ a\ i\ g\ j$

Parent 2: $a\ b\ c\ d\ e\ f \quad g\ h\ i\ j\ k\ l$

One of them say b is picked to be the first gene in the child, and other other gene g has to be swapped away with b in Parent 2 to avoid repetition subsequently. After swapping, we have this:

Parent 1: $h\ k\ c\ e\ f\ d \quad b\ l\ a\ i\ g\ j$

Parent 2: $a\ g\ c\ d\ e\ f \quad b\ h\ i\ j\ k\ l$

Now we compare the physical distance between node b and l, d_{bl}, and the distance between b and h, d_{bh}. If $d_{bl} > d_{bh}$, we then choose h to be the next node and swap l and h in the first parent or delete h in the first parent to avoid duplication later. This process is continued until a new chromosome of the same length and comprising all the 12 alphabets are formed. Note that the result varies depending on whether we swapping or deletion is undertaken. We named heuristic crossover with swapping *HeuristicCrossover 1* and *HeuristicCrossover 2* otherwise.

Merge Crossover Unlike heuristic crossover, which rearranges the parents according to distance to produce children, merge crossover operated on the basis of a predefined time precedence. This time precedence is often summarized from the time windows imposed by each node. The author created such precedence from the latest arrival time of each node. In the same example:

Parent 1: $h\ k\ c\ e\ f\ d \quad b\ l\ a\ i\ g\ j$

Parent 2: $a\ b\ c\ d\ e\ f \quad g\ h\ i\ j\ k\ l$

Similar to heuristic crossover, a random cut point is selected and a first gene b is chosen randomly from the first parents and swapping is done to second parent. The node which comes earlier in the time precedent becomes the next gene in the new chromosome. Here we define two kinds of merge crossover just like the heuristic crossover case, *MergeCrossover 1* and *MergeCrossover 2*, to differentiate the swapping and deletion schemes.

It is noted that both heuristic crossover and merge crossover produce only one offspring from a pair of parents. Inspired by the fact that both geographic locations and time sequences are important in vehicle routing, we decided to combine the two crossover operators so that two parents now produce two children, each of which comes from either the heuristic crossover or the merge crossover. Naturally we thus have four combined operators from the four combinations, namely *H1M1*, *H1M2*, *H2M1* and *H2M2*. Experimental results indicated that H1M2 outperforms the rest especially in clustered data sets.

Not every pair of parents ought to reproduce in every generation. How many parents are to crossover is governed by the *probability of crossover*, a fix real number between 0 and 1. In our GA, we set it at 0.77, a moderate value usually used in other GA implementation as well. When a couple of parents are determined not to crossover, they are copied verbatim to the next generation.

Mutation is a supplementary operation to crossover. The main purpose of mutation is to avoid overly homogeneous population by bring random, unrelated traits into the present population and increase the variance of the population. Several mutation operators have been proposed in the literature and they are shown in Figure 4. Because the chromosomes are of fixed length in our implementation, only swap node and swap sequence are used here.

There is another important parameter associated with the mutation operations. This is the probability of mutation, or $P_{mutation}$, taking on values from [0, 1]. Earlier research has shown that excessive $P_{mutation}$ values drives the GA into convergence sooner than necessary, often resulting in undesirable local optimal

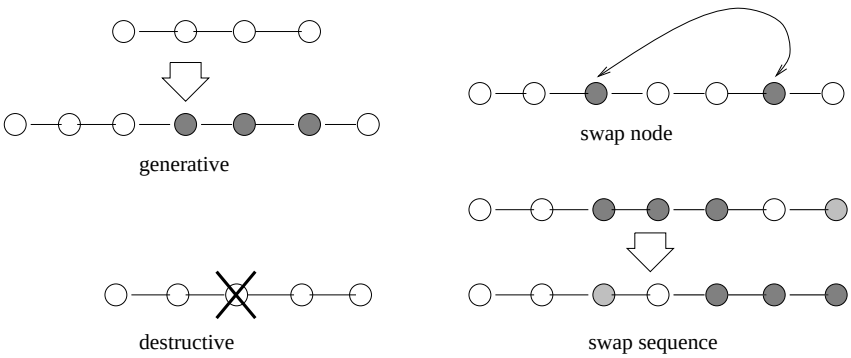

Figure 4: Several Common Mutation Operations

solutions; small $P_{mutation}$ produces the opposite result: intolerably slow convergence. In this paper, the author developed a unique *adaptive mutation probability scheme*. The scheme adapts $P_{mutation}$ to the standard deviation of the population:

$$S = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i - \overline{x})^2}{N-1}} \tag{2}$$

where N is the population size, x_i is the fitness value of individual i and $\overline{x}$ is the average fitness of the population. If S is greater or equal to a threshold value MINPOPDEV=5, a minimum $P_{mutation} = P_{min} = 0.06$ is used, otherwise $P_{mutation} = P_{min} + 0.1 \times (MINPOPDEV - S)$. Hence we can see a minimum mutation probability of 0.06 is always guaranteed.

6 Further Improvements

Hill-climbing is a supplementary measures that takes advantage of local greedy search to improve the chromosomes produced from crossover and mutation procedures. In hill-climbing, a portion of the population are randomly selected and decoded into their respective solution form. These solutions then undergo a few iterations of removal and re-insertion operations and are eventually updated with the new, improved solutions. Note this is not a complete λ-interchange[16] procedure due to the moderately large time requirement by each λ-interchange. Furthermore, to reduce the complexity and to prevent from over-reliance on good solutions, the probability of a chromosome to be selected for hill-climbing is only set at 0.5.

However, even after hill-climbing, there is still possibility of degradation of the entire new generation. To restore some of the good chromosomes in the parent generation, the worst 4% chromosomes in the child generation is substituted with the best 4% in the parents. Note the percentage of recovery should be less than the mutation rate at any time, due to the consideration that it is desirable to have some mutated chromosomes escaping the recovery process, hence bring the population out of a premature convergence.

With all the strategies we have described in the last few sections, finally the genetic algorithm is stated as such:

Genetic Algorithm (GA) For VRPTW:

GA-1. *Generate initial population of N chromosomes (partly from PFIH and its mutation and partly from totally random selections). Calculate time precedence according the latest arrival time*

of each node;

GA-2. *Evaluate the fitness f(x) of each chromosome x in the population, calculate average fitness and standard deviation, thus set mutation probability;*

GA-3. *Create a new population by repeating following steps until the new population is complete;*
1.[Selection] Select two parent chromosomes from a population by Tournament selection;
2.[Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents;
3.[Mutation] With a mutation probability mutate new offspring at a random locus (position in chromosome);
4.[Accepting] Place new offspring in a new population;
5.[Hill-Climbing] Performing hill-climbing with 1-neighborhood first-best search;
6.[Recovery] Replace the worse 4% chromosomes in the new population with the best 4% in the parents population;

GA-4. *Update the old population with the newly generated population;*

GA-5. *If the certain number of generation is reached, stop, perform a 2-interchange (GB)[16] on the best solution in the current population and return the improved solution;*

GA-6. *Else go to step 2.*

7 Results and Conclusion

The author tested the algorithm with different crossover combinations on 56 Solomon VRPTW instances of 100 nodes. After 500 generations, we achieve solutions very competitive to those solved by Tabu Search and Simulated Annealing algorithms, and in 4 out of 56 cases, the results are better than or equal to the best in the publication. Table 1 summarizes solutions obtained from the 6 categories of problems in the Solomon problem set, by GA-PMX and GA-H1M2 only, as these two crossover operators out perform the rest.

In Figure 5 and Figure 6, we illustrate two sample solutions to Problem R104 in the Solomon problem set. While the total costs resulting from these two solutions are comparable, they follow quite different paths. Nodes in R1 category problems are homogeneously distributed like depicted in the figures. These problems are a lot harder to solve than C category problems. But one can still see that GA has roughly sectored the nodes into areas and nodes in one area is visited on one route. One can also see that because H1M2 crossover comes with concerns of time precedence and distance, the sectoring is even more apparent. Because of the tendency of sectoring in our algorithm, it is assumed that GA can do well in average VRPTW problems in which time windows are moderately wide.

The main contribution of this research is a new application of GA to VRPTW. Previously, Thangiah used cluster first route second method when applying GA to VRPTW. In his GIDEON system, the angular differences were coded in the chromosomes. GA was only used to sector the customers within clusters. Other heuristics like 2-opt and simulated annealing had to complement GA in routing the customers within one cluster. Strictly speaking, it is only a hybrid heuristic that constitutes some GA element. The author's current implementation, however, is a complete GA with more straightforward

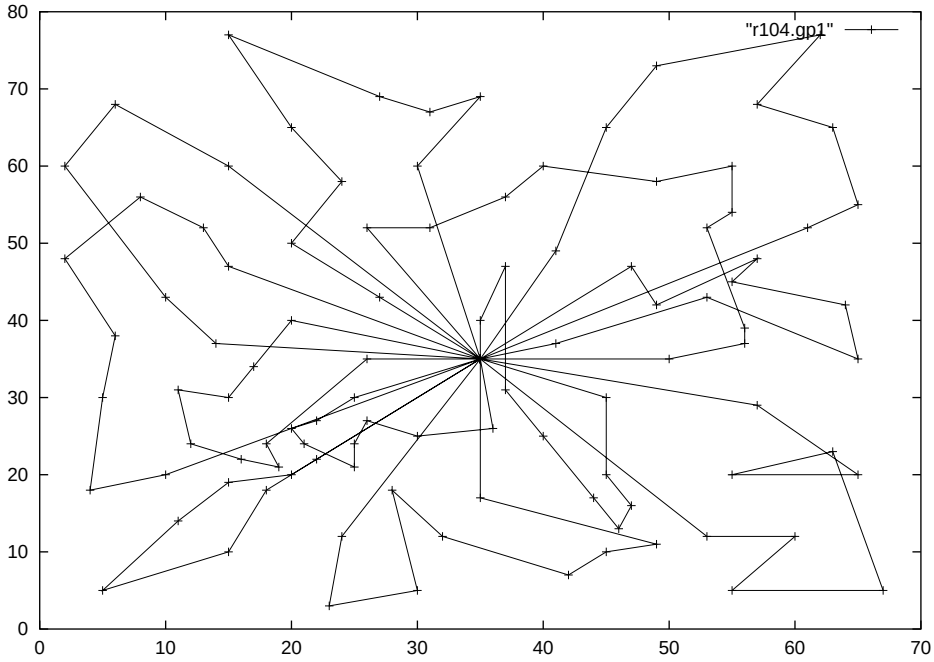

Figure 5: Solution Plot for GA-PMX on R104

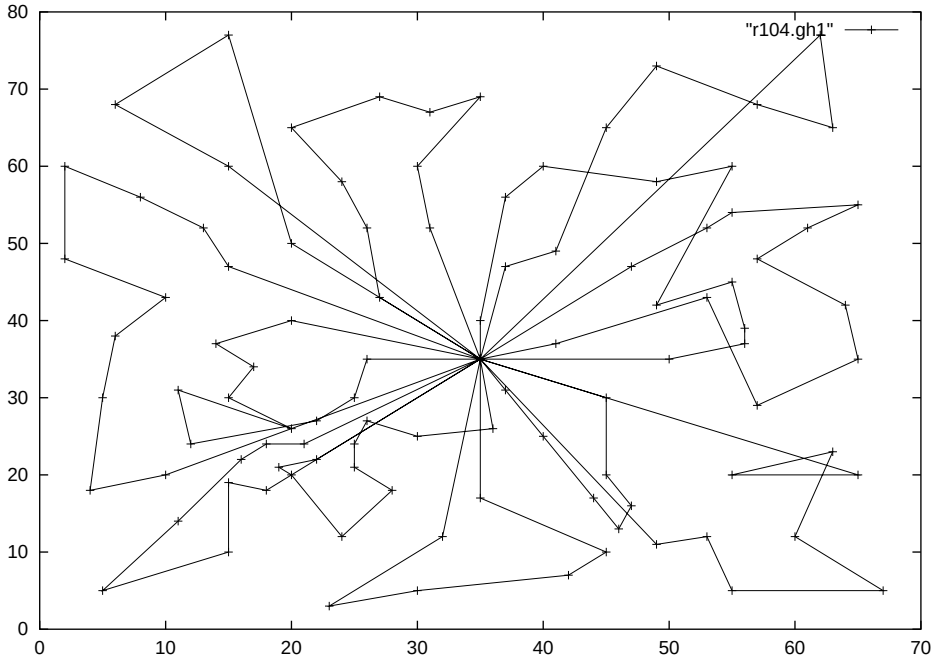

Figure 6: Solution Plot for GA-H1M2 on R104

Prob.	Pub. Best	GA-PMX	GA-H1M2	Best Found
C1	827.49	870.15	867.36	C105, C109
C2	589.86	657.25	625.40	C201, C205
R1	1174.66	1352.77	1369.97	None
R2	941.18	1159.62	1193.6	None
RC1	1354.06	1543.77	1577.64	None
RC2	1081.29	1360.53	1377.86	None

Table 1: Relative Average Cost for Our Heuristics Against the Best Known
Note: This table compares the average minimal costs between the published best solutions and our solutions. The last column shows the problems for which we obtain best solutions.

representation. The results from this GA system prove better than that from the GIDEON system in all 56 Solomon instances.

Other authors, like Blanton Jr., et. al., applied GA with similar representation to VRPs, but almost all their works were concentrated on Traveling Salesman Problem (TSP), which involves only one route and has been proved NP-compelete. And these authors failed to present result data so no comparison can be made. The author has reason to believe that his research is among the first to solve complex VRPTW by GA and to obtain comprehensive results to all 56 100-node benchmark problems.

References

[1] Blanton, J. L. Jr. and R. L. Wainwright, *Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms (1993).

[2] Bodin, L., B. Golden, A. Assad and M. Ball, *Routing and Scheduling of Vehicles and Crews - the State of the Art*. Pergamon Press (1983).

[3] Desrochers, M., J. Desrosier and M. Solomon, *A New Optimization Algorithm for Vehicle Routing Problems with Time Windows*. Operations Research 40 (1992).

[4] Desrosier, J., Y. Dumas, M. Solomon and F. Soumis, *Time Constraint Routing and Scheduling*. Handbooks on Operations Research and management Science, Volume Network, North Holland, Amsterdam (1995).

[5] Fisher, M. L. , *Vehicle Routing*. Handbooks in Operations Research & Management Science, Vol 8 (1995).

[6] Golden, B. and A. Assad, *Vehicle Routings: Methods and Studies*. North Holland, Amsterdam (1988).

[7] Holland, J. H., *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor(1975).

[8] Homaifar, A., S. Guan and G. E. Liepins, *A New Approach to the Traveling Salesman Problem by Genetic Algorithms*. Proceedings of the Fifth International Conference on Genetic Algorithms (1993).

[9] Oliver, I. M. , D. J. Smith and J. R. C. Holland, *A Study of Permutation Crossover Operations on the Traveling Salesman Problem.* Proceedings of the Fourth International Conference on Genetic Algorithms (1991).

[10] Prinetto, P., M. Rebaudengo and M. Sonza Reorda, *Hybrid Genetic Algorithms for the Traveling Salesman Problem.* Proceedings of the Fifth International Conference on Genetic Algorithms (1993).

[11] Savelsbergh, M. W. P., *Local Search for Routing Problems with Time Windows.* Annuals of Operating Research (1985).

[12] Solomon, M. M., *Algorithms for Vehicle Routing and Scheduling Problems with Time Window Constraints.* Operations Research 35 (2) (1987).

[13] Thangiah, Sam R., Ibrahim Osman, Rajini Vinayagamoorthy and Tong Sun (1994). *Algorithms for Vehicle Routing with Time Deadlines.* American Journal of Mathematical and Management Science's special issue: Vehicle Routing 2000: Advances in Time Windows, Optimality , Fast Bounds and Multi-Depot Routing, 323-355.

[14] Thangiah, S. R. , *An Adaptive Clustering Method using a Geometric Shape for Vehicle Routing Problems with Time Windows.* Proceedings of the Sixth International Conference on Genetic Algorithms (1995)

[15] Thangiah, Sam R. (1995). *Vehicle Routing with Time Windows using Genetic Algorithms.* Application Handbook of Genetic Algorithms: New Frontiers, Volume II. Lance Chambers (Ed.), CRC Press,253-277 (1995).

[16] Thangiah, S. R., *Genetic Algorithms, Tabu Search and Simulated Annealing Methods for Vehicle Routing Problems with Time Windows.* In Practical Handbook of Genetic Algorithms: Complex Structures, L. Chambers (Ed.), CRC Press (1998).

[17] Whitney, D., T. Starkweather and D. Fuquay, *Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator.* Proceedings of the Fourth International Conference on Genetic Algorithms (1991).