A GENETIC ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Lin Cheng

A Thesis Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Science

Department of Mathematics and Statistics

University of North Carolina Wilmington

2005

Approved by

Advisory Committee

Chair

Accepted by

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

The objective of the vehicle routing problem (VRP) is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at the same depot. A vehicle routing problem with time windows (VRPTW) requires the delivery be made within a specific time frame given by the customers. Prins (2004) recently proposed a simple and effective genetic algorithm (GA) for VRP. In terms of average solution cost, it outperforms most published tabu search results. We implement this hybrid GA to handle VRPTW. Both the implementation and computational results will be discussed.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to Dr. Yaw O. Chang for guidance on my thesis work. I also want to thank Dr. John Karlof and Dr. Matthew TenHuisen for serving on my thesis advisory committee and offering constructive ideas. Also, I want to thank all the faculty members in the Mathematics and Statistics Department who gave me valuable advice and help. I also want to thank my wife Angelina Zhou for her patience and help. This thesis is dedicated to my parents.

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Introduction to VRP

The Vehicle Routing Problem consists of a set of customers with known demands. For most cases, there is only one depot. Each vehicle can carry either a limited or unlimited goods and travel a limited distance. Each vehicle starts from a depot and delivers the goods required, then returns to the depot. Each customer is assigned to exactly one vehicle route. The total demand of any route must not exceed the vehicle capacity. The objective of the vehicle routing problem is to deliver to a set of customers with known demands with minimum-cost vehicle routes originating and terminating at a depot. Figure 1 shows a picture of the typical input for a vehicle routing problem. Figure 2 shows a possible output.

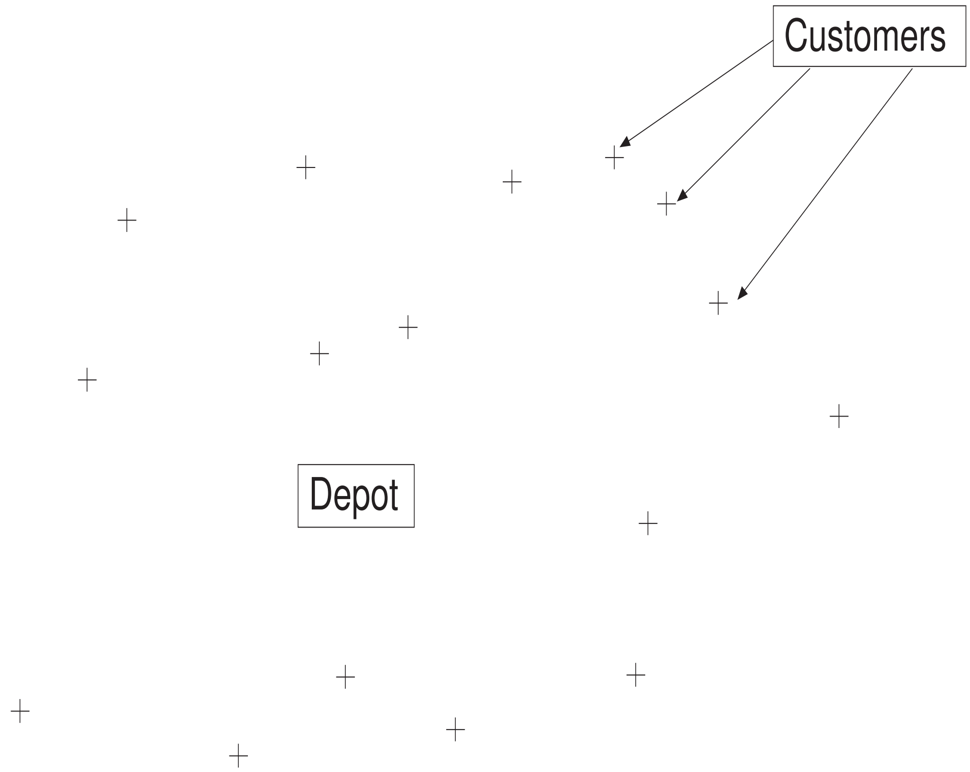

Figure 1: An input for a vehicle routing problem

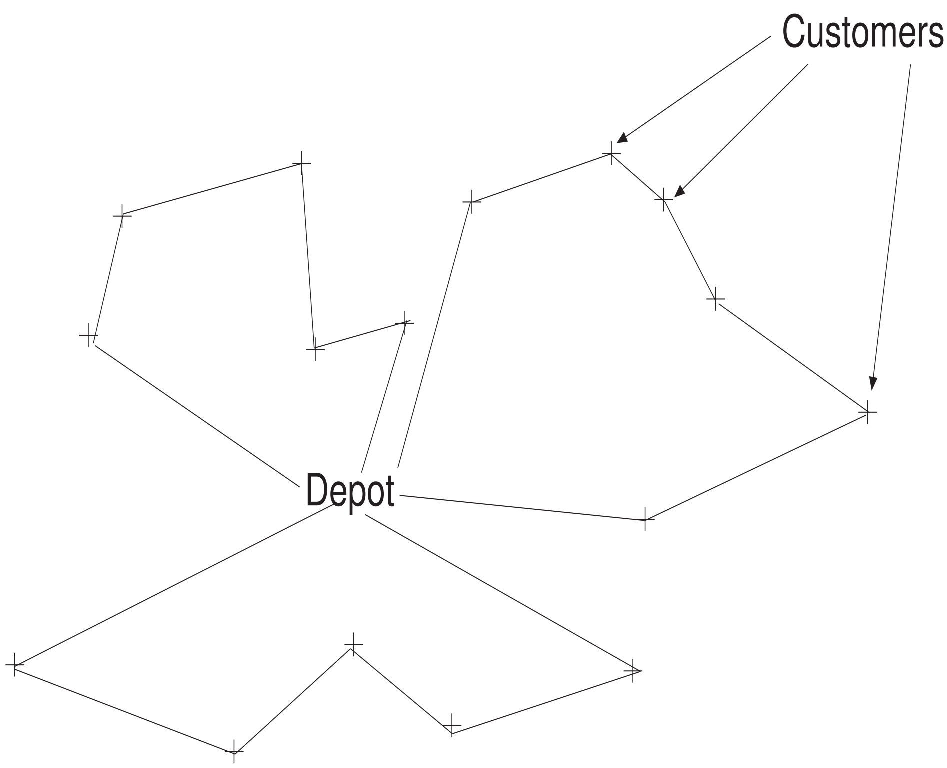

Figure 2: An output for the instance in Figure 1

The vehicle routing problem is a well-known integer programming problem which falls into the category of NP-Hard problems[11], meaning that the computational effort required to solve this problem increases exponentially with the problem size. For such problems it is often desirable to obtain approximated solutions, so they can be found quickly enough and are sufficiently accurate for the purpose. Usually this task is accomplished by using various heuristic methods, which rely on some insight into the nature of the problem.

The VRP arises naturally as a central problem in the fields of transportation, distribution, and logistics [5]. In some market sectors, transportation means a high percentage of the value is added to goods. Therefore, the utilization of computerized methods for transportation often results in significant savings ranging from 5% to 20% in the total costs, as reported in [10]. In some real world vehicle routing problems there are often side constraints due to other restrictions. Some of the well known models are:

- Every customer has to be supplied within a certain time window (vehicle routing problem with time windows - VRPTW).
- The vendor uses many depots to supply the customers (multiple depot vehicle routing problem - MDVRP).
- Customers may return some goods to the depot (vehicle routing problem with pick-up and delivering - VRPPD).
- The customers may be served by different vehicles (split delivery vehicle routing problem - SDVRP).
- The demands, service time and/or travel time are random (stochastic vehicle routing problem -SVRP).
- Satellite facilities are used to replenish vehicles during a route - (VRPSF).
- A VRP in which customers can demand or return some commodities (vehicle routing problem with backhauls -VRPB).

1.2 Time Window

The vehicle routing problem with time windows (VRPTW) is the same problem as the vehicle routing problem (VRP) with the additional time constrants. A time window $[e_i, l_i]$ is associated with each customer i, where the vehicle can not arrive earlier than time e_i and can not arrive later than time l_i. The VRP without time window can corresponds to the situation $e_i = 0$ and $l_i = \infty$ for all $1 \leq i \leq n$. In this paper, we will consider the case when $e_i > 0$ and $l_i = \infty$ for all i, that is, we only consider the case with restrictions on the earliest arrival time.

1.3 Genetic Algorithm

The principles of a genetic algorithm(GA) are well known. A population of solutions (chromosomes in the Genetic Algorithm) is maintained along with a reproductive process allowing parent solutions to be selected from the population. Offspring solutions are produced which exhibit some of the characteristics of each parent. The fitness of each solution can be related to the objective function value, in this case the total time travelled for all vehicles. Analogous to biological processes, offspring with relatively good fitness levels are more likely to survive and reproduce, with the expectation that fitness levels throughout the population will improve as it evolves. More details are given by Reeves [7].

A general formwork of GA can be described as follow: An initial population of chromosomes will be generated randomly and arrayed according to their corresponding objective function values. To ensure a better dispersal of solutions and to diminish the risk of premature convergence, we do not allow clone (identical) solutions in the population.

After the initial population is built up, two chromosomes are randomly selected from the population. A new chromosome C will be produced by either the standard crossover or the mutation procedure. (Both procedures will be discussed later.) If the new chromosome C is better than (in the optimization sense) any chromosome in the current population, C will be included and the worst one in the current population will be removed. This procedure is repeated until a stopping criteria is satisfied.

In this paper, we will review and implement a hybrid genetic algorithm proposed by Chreistian Prins [8] to solve the vehicle routing problem with earliest arrival time. Problem formulation and implementation of the GA will be discussed and computational results will be reported.

2 PROBLEM FORMULATION

The vehicle routing problem consists of a set of customers with known demands. Vehicles leave from a depot, deliver the goods to customers, and return to the depot. In general, there is only one depot. Each vehicle can carry Limited goods and travel a limited distance. Each customer is assigned to exactly one vehicle route and the total demand of any route must not exceed the vehicle capacity.

To date, heuristic algorithms are required for dealing with real-life instances. Most of the proposed algorithms assume that the number of vehicles is unlimited, and the objective is to obtain a solution that either minimizes the number of vehicles and/or total travel cost. However, transport operations in the real world face the limit of time windows.

Before we formally give the mathematical formulation of the vehicle routing problem with time windows, we need to introduce some basic terminologies from graph theory.

Definition 1

An undirected graph G is an ordered pair G=(V, E) where

$$V=\{0, 1, 2, ..., n\},$$

V is the set of vertices or nodes; and

$$E=\{(i, j)\ i, j \in V\},$$

E is the set of unordered pairs of distinct vertices, called edges.

Definition 2

Let G=(V, E) be a graph, a path P is a sequence of vertices $i_1, i_2, ..., i_k$*, such that* $i_j \in V$ *and* (i_j, i_{j+1}) *is an edge in G, for all* $1 \leq j \leq k-1$*.*

Definition 3

Let G=(V, E) be a graph. G is called a connected graph if there is a path between every pair of vertices .

Now let G be an undirected graph corresponding to a vehicle routing problem where vertex 0 corresponds to the depot and vertex i corresponds to customer i, for $1 \leq i \leq n$. For convenience, let us define the following:

- c_{ij} : the cost associated with edge (i, j), the cost could be the travel distance, the travel time, or the travel cost from customer i to customer j. For simplicity, we assume $c_{ij} = c_{ji}$. If there is no edge between vertex i and j, $c_{ij} = \infty$.
- b_i : the vehicle arrival time at customer i.
- d_i : the service time for customer i.
- e_i : the earliest arrival time for customer i.
- L: maximal operation time for each vehicle.
- R_i: A route is a sequence $(i_1, i_2, ...i_k)$ where $i_j \in V$, such that a particular vehicle will serve customers $(i_1, i_2, ...i_k)$ according to the order of the sequence.
- W_{i_j}: the waiting time for vehicle i at customer j.

Consider a route $R_i = (i_1, i_2, ..., i_k)$, a vehicle arrives customer i_j at time

$$b_{i_{j-1}} + d_{j-1} + c_{i_{j-1},i_j}.$$

that is the arrival time at customer i_{j-1}, plus the service time at customer i_{j-1} and the travel time from customer i_{j-1} to customer i_j. To ensure the earliest arrival time constraint, we define

$$b_{i_j} = max[e_{i_j}, b_{i_{j-1}} + d_{i_{j-1}} + c_{i_{j-1},i_j}].$$

when the maximum is obtained at e_{i_j}, it implies the vehicle arrives earlier than the desired arrival time and it has to wait until the customer i_j is available. Let us define the waiting time for vehicle i at customer j as

$$W_{i_j} = max[0, e_{i_j} - b_{i_{j-1}} - d_{i_{j-1}} - c_{i_{j-1},j}].$$

An additional constraint

$$b_{i_k} + d_{i_k} + c_{i_k,0} \leq L,$$

is applied to ensure a vehicle will not operate over the operation time allowed, ie. R_i is a feasible route. Then the total travel time for route R_i is given by

$$C(R_i) = c_{0,i_1} + \sum_{j=2}^{k}(c_{i_{j-1},i_j}) + \sum_{j=1}^{k} W_{i_j} + \sum_{j=1}^{k} d_{i_j} + c_{i_k,0}.$$

A solution S is a collection of routes $R_1, R_2, ..., R_l$, such that each customer will be covered by exactly one route R_i. The total travel time for S is

$$C(S) = \sum_{i=1}^{l} C(R_i).$$

Our goal is to minimize $C(S)$ over all the possible feasible solutions S.

3 SPLITTING ALGORITHM

Before we review the original algorithm, we need to introduce the concept of chromosome:

Definition 4

A chromosome S is a permutation of n positive integers, such that each integer is corresponding to a customer.

A chromosome may be broken into several different routes. For example, a chromosome with 5 customers, $S = (1, 2, 3, 4, 5)$ may be broken into $R_1 = (1, 2)$ and $R_2 = (3, 4, 5)$, or $R_1 = (1, 2, 3)$ and $R_2 = (4, 5)$; etc.

Christian Prins [8] proposed a genetic algorithm dealing with the vehicle routing problem without time windows and focused on the total traval cost. We will review the main idea of the hybrid GA in Section 3.1 and discuss our implementation in Section 3.2.

3.1 Main idea of Hybrid Genetic Algorithm

Because each chromosome can be split into many different routes, Christian Prins proposed a splitting procedure which can find an optimal splitting so that the total cost is minimized for any given chromosome. The main idea can be described as follows. Without loss of generality, let $S = (1, 2, 3, ..., n)$ be a given chromosome. Consider an auxiliary graph $H = (\bar{V}, \bar{E})$ where $\bar{V} = \{0, 1, 2, ..., n\}$. An arc $(i, j) \in \bar{E}$ if

$$\bar{E_{ij}} = c_{0,i+1} + \sum_{k=i+1}^{j-1} (d_k + C_{k,k+1}) + d_j + c_{j,0} \leq L. \qquad (*)$$

Then $\bar{E_{ij}}$ is the total travel cost for the route $(i + 1, i + 2, ..., j)$. An optimal split for S corresponds to shortest path P from vertex 0 to vertex n in H.

The fist graph of Fig. 3 shows a sequence S=(a, b, c, d, e). The number associates with each edge is the travel cost. Let us assume the service cost (delivery cost) is 0, then the auxiliary graph H is the second gragh in Fig. 3. The edge in H with weight 55 corresponds to the travel cost of the trip $(o, a, b, 0)$. The weight associated with the edges are similarly defined. The shortest path from 0 to e is (o, b, c, e) with minimal cost 205. The last graph gives the resulting splitting with three trips.

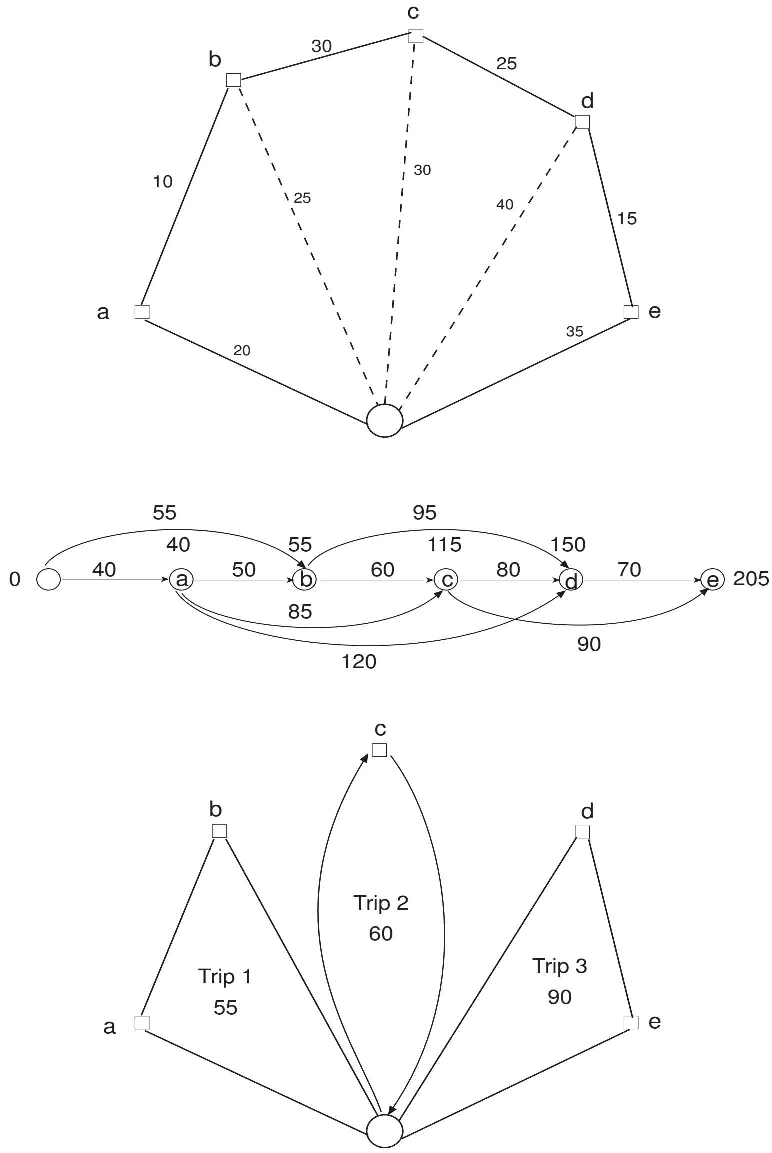

Figure 3: An example of chromosome evaluation

The auxiliary graph helps us understand the idea how to split a given chromosome

S into optimal trips. But in practice, we do not have to construct such graph H. Prince proposed the following procedure:

$V_0 := 0$

$for \quad i := 1 \quad to \quad n \quad do \quad V_i := +\infty \quad endfor$

$for \quad i := 1 \quad to \quad n \quad do$

$\quad cost := 0; \quad j := i$

$\quad repeat$

$\quad\quad if \quad i = j \quad then$

$\quad\quad\quad cost := c_{0,S_j}, +d_{S_j} + c_{S_{j,0}}$

$\quad\quad else$

$\quad\quad\quad cost := cost - c_{S_{j-1,0}} + c_{S_{j-1},S_j} + d_{S_j} + c_{S_{j,0}}$

$\quad\quad endif$

$\quad\quad if \quad (cost \leq L) \quad then$

$\quad\quad\quad if \quad V_{i-1} + cost < V_j \quad then$

$\quad\quad\quad\quad V_j := V_{i-1} + cost$

$\quad\quad\quad\quad P_j := i - 1$

$\quad\quad\quad endif$

$\quad\quad\quad j := j + 1$

$\quad\quad endif$

$\quad until \quad (j > n) \quad or \quad (cost > L)$

$endfor$

Let $S = (1, 2, ..., n)$ be a given chromosome. Two labels V_j and P_j for each vertex j in S are computed. V_j is the cost of the shortest path from node 0 to node j in H, and P_j is the predecessor of j on this path. The minimal cost is given at the end by V_n. For any given i, note that the incrementation of j stops when L is exceeded. We will modify this part to handle the time window. Once the split procedure is completed, the optimal routes can be determined by the shortest path.

3.2 Implementation for the Splitting Procedure

Recall E_{ij} in (*) represents the total travel time for route $(i+1, i+2, ..., j)$. To incorporate the earliest arrival time e_i, the total travel time E_{ij} for the same route can be calculated by

$$E_{ij} = c_{0,i+1} + \sum_{k=i+1}^{j-1}(W_i + d_k + c_{k,k+1}) + W_j + d_j + c_{j,0}.$$

If $E_{ij} \leq L$, then the arc (i, j) exists in the auxiliary graph H, and the shortest path from 0 to n in H corresponding to an optimal split for S.

To accommodate this modification, the splitting procedure can be modified as follows:

$V_0 := 0$

$for \quad i := 1 \quad to \quad n \quad do \quad V_i := +\infty \quad endfor$

$for \quad i := 1 \quad to \quad n \quad do$

$\quad cost := 0; \quad j := i$

$\quad repeat$

$\quad\quad if \quad i = j \quad then$

$\quad\quad\quad cost := Max[c_{0,S_j}, e_{S_j}] + d_{S_j} + c_{S_j,0}$

$\quad\quad else$

$\quad\quad\quad cost := Max[cost - c_{S_{j-1},0} + c_{S_{j-1},S_j}, e_{S_j}] + d_{S_j} + c_{S_j,0}$

$\quad\quad endif$

$\quad\quad if \quad (cost \leq L) \quad then$

$\quad\quad\quad if \quad V_{i-1} + cost < V_j \quad then$

$\quad\quad\quad\quad V_j := V_{i-1} + cost$

$\quad\quad\quad\quad P_j := i - 1$

$\quad\quad\quad endif$

$\quad\quad\quad j := j + 1$

$\quad\quad endif$

$\quad until \quad (j > n) \quad or \quad (cost > L)$

$endfor$

Here c_{0,S_j} is the regular arriving time at customer S_j, and e_{S_j} is the earliest arrival time at customer S_j. If $e_{S_j} < c_{0,S_j}$, we will pick e_{S_j}. Otherwise, we will pick the regular arrival time c_{0,S_j}.

Let us consider an example with 10 customers. The vehicle service time limit L is 540 minutes and the service time d_i is 10 minutes for all customers. Figure 4 shows a sequence $S = (A, B, C, D, E, F, G, H, I, J)$ with the earliest arrival time in parentheses. For example, if the vehicle leaves the depot at 8 : 00AM, the 150 minutes corresponding to customer A means the earliest arrival time for customer A is 10:30AM. The number associated with each edge is the travel time.

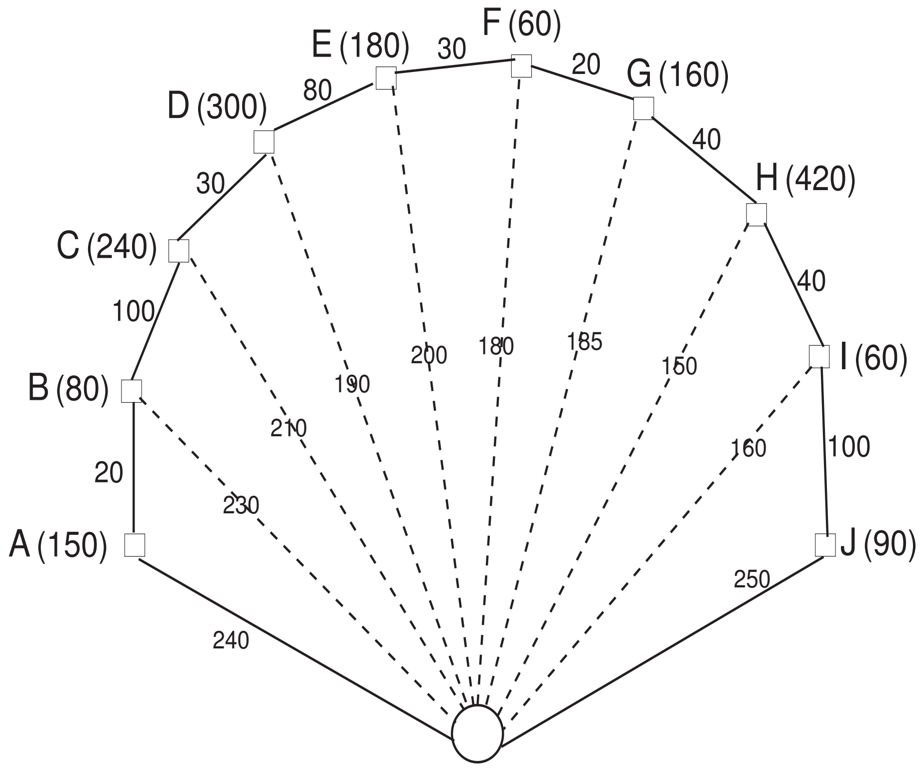

Figure 4: An example of the split procedure

Figure. 5 shows the optimal routes for this particular sequence. It contains 4 trips with the total travel time of 1840 minutes. The travel time for Trip 1 is 510 minutes, the travel time for Trip 2 is 410 minutes, the travel time for Trip 3 is 470 minutes, and the travel time for Trip 4 is 430 minutes.

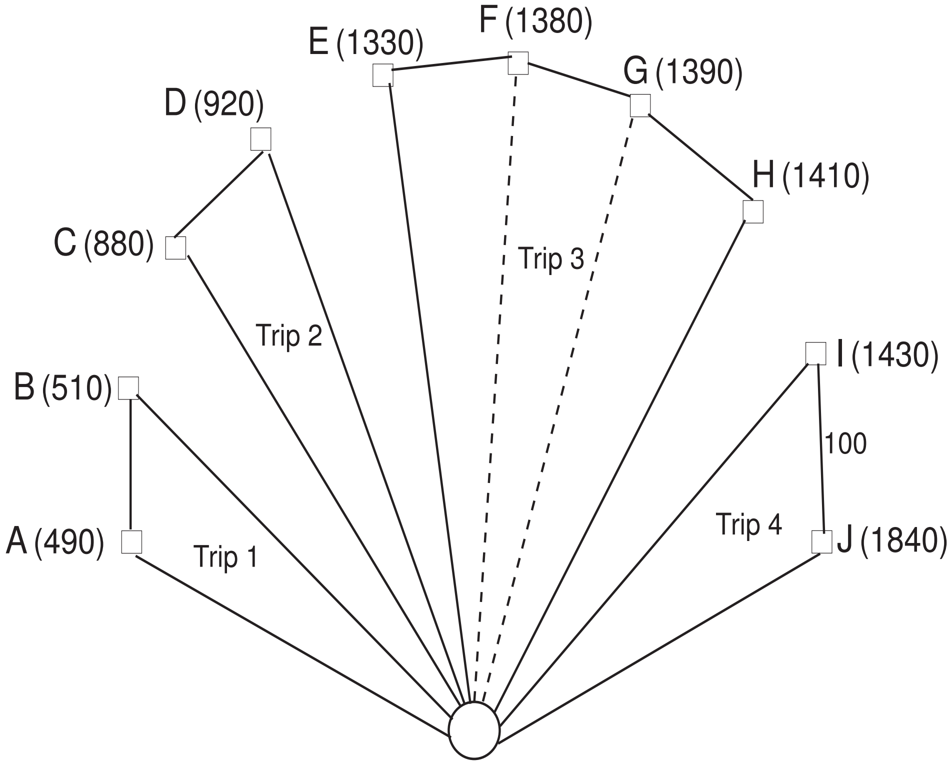

Figure 5: A feasible solution of the split procedure

4 CROSSOVER

As mentioned earlier, the reproductive process in GA can be done either by crossover or by mutation procedures. A crossover (or recombination) operation is performed upon the selected chromosomes. We will use the *order crossover* which will be explained in the following example.

Figure. 5 shows how crossover constructs a child C. First, two chromosomes are randomly selected from the initial population and the least-cost one becomes the first parent P_1. Two cutting sites i and j are randomly selected in P_1, here i=4 and j=6. Then, the substring $P_1(i)...P_1(j)$ is copied into $C(i)...C(j)$. Finally, P_2 is swept circularly from $j+1$ onward to complete C with the missing nodes. C is also filled circularly from $j+1$. The other child maybe obtained by exchanging the roles of P_1 and P_2.

Rank	1	2	3	4	5	6	7	8	9	10
				i=4 ↓		j=6 ↓				
P1	9	8	7	5	10	3	6	2	1	4
P2	9	8	7	6	5	4	3	2	10	1
C	7	6	4	5	10	3	2	1	9	8

Table 1: Example of crossover

Figure 5 demonstrates the process. Let $i = 4$ and $j = 6$, so $C(4) = P_1(4)$, $C(5) = P_1(5)$, and $C(6) = P_1(6)$. Now $C(7)$ should equal to $P_2(7)$. Because $C(6) = 3$, we shift to $P_2(8)$, so $C(7) = P_2(8) = 2$. Now $C(8)$ should equal to $P_2(9)$. Again $C(5) = 10 = P_2(9)$, we will skip $P_2(9)$ and let $C(8) = P_2(10) = 1$. Repeat this process until C is constructed.

5 LOCAL SEARCH AS MUTATION OPERATOR

The classical genetic algorithm framework must be hybridized with some kind of mutation procedure. For the VRPTW, we quickly obtained much better results by replacing simple mutation operators (like moving or swapping some nodes) by a local search procedure.

A child C produced by crossover could be improved by local search with a fixed mutation rate p_m which stands for the rate or the probability of mutation and is a fundamental parameter in genetics and evolution. Here p_m is the measure of the impact of mutations on the child C. The mutation procedure can be described as follows. A chromosome C is converted into a VRPTW solution. Then for all possible pairs of distinct vertexes (u, v), such that i is ahead of v, the following simple moves are tested. Let x and y be the successors of u and v in their respective trips. The following rules are used to create new chromosomes.

- M1. Remove u from C and insert it after v, ie, the new sequence is $(...v, u, ...)$.
- M2. If x is a client, remove (u, x) from C then insert (u, x) after v, ie, the new sequence is $(...v, u, x, ...)$.
- M3. If x is a client, remove (u, x) from C then insert (x, u) after v, for example, the new sequence is $(...v, x, u, ...)$.
- M4. Swap u and v, for example, the new sequence is $(...v, ..., u, ...)$.
- M5. If x is a client, swap (u, x) and v, ie, the new sequence is $(...v, ...u, x, ...)$.
- M6. If x and y are clients, swap (u, x) and (v, y), ie, the new sequence is $(...v, y, ...u, x, ...)$.
- M7. If u and v are in the same trip, replace (u, x) and (v, y) by (u, v) and (x, y), ie, the new sequence is $(...u, v, ...x, y, ...)$.

- M8. If u and v are not in the same trip, replace (u, x) and (v, y) by (u, v) and (x, y), ie, the new sequence is $(...u, v, ...x, y, ...)$.

- M9. If u and v are not in the same trip, replace (u, x) and (v, y) by (u, y) and (x, v), ie, the new sequence is $(...u, y, ...x, v, ...)$.

When a new chromosome is produced, we will find the optimal routes using the splitting procedure. Let $\bar{C}$ be the new chromosome produced by the mutation procedure. If $\bar{C}$ is better than the worst in the current population at the end of mutation process, $\bar{C}$ will be added into the current population, and the worst one will be moved. If $\bar{C}$ is better than the current best solution, we consider the current mutation process as a productive mutation.

The population is implemented as an array Π of σ (σ is the population size) chromosomes, always sorted in increasing order of cost to ease the basic genetic algorithm iteration. Thus, the best solution is Π_1.

Clones (identical solutions) are forbidden in Π to ensure a better dispersal of solutions. This also allows a higher mutation rate p_m by local search LS, giving a more aggressive genetic algorithm. To avoid comparing chromosomes in details and to speed-up clone detection, we impose a stricter condition: the costs of any two solutions generated by crossover or mutation must be spaced at least by a constant $\triangle > 0$.

A population satisfying the following condition will be said to be well-spaced. The simplest form with $\triangle = 1$ (solutions with distinct integer costs) was already used in an efficient Genetic Algorithm. That is,

$$\forall P_1, P_2 \in \Pi : P_1 \neq P_2 \Rightarrow |F(P_1) - F(P_2)| \geq \triangle$$

where P_1 and P_2 are the parents we select from the initial population, and $F(P_1)$ and $F(P_2)$ are the costs of P_1 and P_2

The stopping criterion for GA is either after reaching a maximum number of general iterations, α_{max} (number of crossovers that do not yield a clone) or a maximal number of iterations without improving the best solution, β_{max}. Let σ be the population size, p_m be the mutation rate, and space $\Delta = 5$. The GA can be described as follows:

Step 1: (Initial population)

Randomly generate σ chromosomes.

Step 2: (Sort Initial population)

Sort the initial population according to the total travel time of each chromosome.

Step 3: (setup counter)

Set α, β=0

Step 4: (Crossover)

Let $\alpha = \alpha + 1$. Select two parents (P_1) and (P_2) by binary tournament. Apply Crossover to (P_1, P_2) and select one child C at random.

Step 5: (Mutation)

Compare r (randomly generated) and p_m. If $r > p_m$ then go to Step 6. Otherwise, mutation is applied to the child and C' is generated. The split procedure is then used to get the solution of C'.

Step 6: (Productive iteration)

Since Π_1 is the best solution of the population, if $(F(C') - F(\Pi_k)) \geq \Delta, 1 \leq k \leq \sigma$, then let $\beta = 0$ and shift C' to re-sort population. Otherwise, $\beta = \beta + 1$.

Step 7: (Check stopping criterion)

If $\alpha = \alpha_{max}$ or $\beta = \beta_{max}$, stop. Otherwise, go to Step 4.

7 COMPUTATIONAL RESULTS

The Genetic Algorithm was implemented in the Visual Basic script, on a Pentium-4 PC clocked at 2.7 GHz under the operating system Windows XP. We ran two different randomly generated data sets with n=10. For each data set we ran 3 different population sizes with 3 different mutation rates. For all the cases, we set α_{max}=1000 and β_{max}=100. The best solution was calculated by complete enumeration for both date sets.

In the following tables, the first column is the population size, the second column is the mutation rate, and the third column is the best solution in the initial population with the number of trips in the parentheses. Columns 4 and 5 give the final values of α and β, respectively. Column 6 gives the final solution produced by the GA with number of trips in the parentheses. Column 7 gives the runtime in seconds. Column 8 is the best solution with the number of trips in the bracket. The last column is the relative error.

Population size n	Mutation Rate r	Original Solution	α	β	Final Solution	Runtime t(seconds)	Best Solution	Relative Error
30	0.1	1380(3)	928	100	520(1)	90	490(1)	6.12%
	0.2	1550(2)	990	100	510(1)	91	490(1)	4.08%
	0.3	1280(3)	551	100	490(1)	98	490(1)	0%
50	0.1	1050(2)	1000	87	490(2)	90	490(1)	0%
	0.2	990(2)	1000	98	510(1)	90	490(1)	4.08%
	0.3	1360(1)	996	100	500(1)	98	490(1)	2.04%
100	0.1	1270(3)	1000	98	520(2)	90	490(1)	6.12%
	0.2	1823(4)	864	100	500(1)	93	490(1)	4.08%
	0.3	1900(4)	980	100	490(1)	106	490(1)	0%

Table 2: Computational results for date set 1

Table 2, sorted in increasing order of population size, shows very encouraging results. When the population size equals 30 and 100, the mutation rate equals to 0.3, and our solutions reach the best solution with β reaching the maximal number of iterations without improving the best solution, which means that the mutation procedure was applied to the child C at least 100 times.

In general, when the mutation rate increases, the relative errors are decreasing, and the number of trips is decreasing. Only two relative errors are greater than 5%.

One exception that we notice is that when the population size equals 50 and the mutation rate equals 0.1, our solution also reaches the best solution that has two trips instead of one.

Population size n	Mutation Rate r	Original Solution	α	β	Final Solution	Runtime t(seconds)	Best Solution	Relative Error
30	0.1	1787(4)	506	100	1022(2)	85	962(2)	6.3%
	0.2	2629(5)	1000	75	997(2)	88	962(2)	3.6%
	0.3	1860(4)	579	100	987(2)	96	962(2)	2.6%
50	0.1	1640(4)	583	100	987(2)	90	962(2)	2.6%
	0.2	1744(4)	573	100	987(2)	89	962(2)	2.6%
	0.3	1702(4)	516	100	962(2)	97	962(2)	0%
100	0.1	1787(4)	506	100	1022(2)	89	962(2)	6.3%
	0.2	1441(3)	1000	83	970(2)	93	962(2)	0.83%
	0.3	1518(3)	1000	71	962(2)	97	962(2)	0%

Table 3: Computational results for date set 2

Table 3 is based on the second data set, sorted in increasing order of population size. It shows results similar to those in table 1. When population size equals 50 and 100, mutation rate equals to 0.3. Our solutions also reach the best solution with β reaching the maximal number of iterations without improving the best solution. Also, as population size and mutation rate increase, the solutions are getting better, the relative errors are decreasing, and the number of trips is decreasing. Only two relative errors are greater than 5%.

8 CONCLUSION

We implement a hybrid GA Vehicle Routing Problem to solve The Vehicle Routing Problem with Time Windows. The final results are encouraging when the customer size is small. We will have more comparable experiments with large number of customers in the future.

For simplicity, we only consider the earliest arrival time and assume the capacity of the vehicle is unlimited. We will re-examine the module to handle the capacited cases in the future. We would also like to implement the current algorithm to handle a real time window.

REFERENCES

[1] J.F. Bard, L. Huang, M. Dror, and P. Jaillet, "A Branch and Cut Algorithm for the VRP with Satellite Facilities", *IIE Transactions 30, pp 821-834*

[2] W. Burrows.1988. "The Vehicle Routing Problem with Loadsplitting: A Heuristic Approach". In 24th Annual Conference of the Operational. *Research Society of New Zealand, pages 33-38,*

[3] Moses Charikar, Samir Khuller, Balaji Raghavachari, 2001. Algorithms for capacitated vehicle routing. *Industrial and Applied Mathematics*

[4] Hoong Chuin Lau, Melvyn Sim, Kwong Meng Teo, 2003. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research 559C569*

[5] G. B. Dantzig and R.H. Ramser, 1959. "The Truck Dispatching Problem". *Management Science 6, 8091.*

[6] M. Dror, G. Laporte, P. Trudeau, 1994. Vehicle routing with split deliveries, *Discrete Appl.Math. 50, 239-254*

[7] Reeves CR, editor. Modern heuristic techniques for combinatorial problems. Oxford: Blackwell Scientific Press, 1993.

[8] Chreistian Prins, 2003. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computer and Operations Research*

[9] M. M. Solomon, 1995. "Algorithms for the Vehicle Routing Problem with Time Windows". *Transportation Science, 29(2), pp. 156-166.*

[10] P. Toth, D. Vigo, 2001. The Vehicle Routing Problem. *Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia*

[11] A problem is NP-hard (Non-deterministic Polynomial-time Hard) if solving it in polynomial time would make it possible to solve all problems in class NP in polynomial time . That is, a problem is NP-hard if an algorithm for solving it can be translated into one for solving any other NP problem (nondeterministic polynomial time) problem. NP-hard therefore means "at least as hard as any NP problem".