Chapter 6

A Genetic Algorithm

In this thesis the approach by Tan et al. (2001c) is followed whereby a Genetic Algorithm (GA) uses a path representation to code chromosomes (routes). For example, the chromosome string 4-5-2-3-1 represents a route that starts at node 4, followed by node 5, then 2, 3, and 1 before returning to node 4. Each element in the chromosome is referred to as an *allele*. For a problem with n customers, each chromosome will be an integer string with n elements. Although elementary crossover routines often destroy the validity of tours and routes, specific crossover routines have been developed to ensure that tours and routes remain valid, and keeps improving.

A slightly adapted version of the GA discussed in Algorithm 3.3 is provided in Algorithm 6.1. The GA requires a generation limit similar to the iteration limit for TS and SA. The population size determines the number of solutions in a single generation. The population subdivision parameters establishes the fraction of the population that will undergo specific genetic manipulation. To ensure the natural phenomena of *survival of the fittest*, the elitist parameter p_e ensures that the p_e fittest solutions in a given generation g is *exactly* copied to the next generation $g+1$. The mutation parameter p_m determines the number of chromosomes that will undergo random changes, or mutation. The crossover parameter, p_c, determines the number of solutions that will produce offspring by sharing elements of its chromosomes.

The algorithm is initialized with the generation of p solutions, each containing a single TSP string of nodes. Vas (1999) states that initial solutions can be generated either randomly or heuristically, while Tan et al. (2001c) suggest a combination of solutions: some generated using an efficient Push Forward Insertion Heuristic (PFIH), and the balance generated randomly.

Algorithm 6.1: Genetic Algorithm (GA) overview

```
    Input: Generation limit $g^{\max}$
    Input: Population size $p$
    Input: Population subdivisions $p_e$, $p_m$, and $p_c$ such that $p_e + p_m + p_c = p$
 1  $g \leftarrow 0$
 2  begin initialization
 3      generate feasible TSP solutions $x_1^0, \ldots, x_p^0$
 4  end
 5  repeat
 6      $g \leftarrow g + 1$
 7      cluster TSP solutions
 8      determine fitness of TSP solutions
 9      begin elite
10          Copy $p_e$ best solutions from generation $g$ to generation $g + 1$
11      end
12      begin mutation
13          Include $p_m$ mutated solutions in generation $g + 1$
14      end
15      begin crossover
16          Choose $\frac{p_c}{2}$ non-overlapping pairs of solutions from generation $g$
17          execute crossover perturbations
18          Include new solutions in generation $g + 1$
19      end
20  until $g = g^{\max}$
21  $x^\star \leftarrow \min_{i \in \{1,\ldots,p\}} \{x_i^g\}$
22  $\hat{x} \leftarrow$ locally optimized $x^\star$
```

The GA proceeds for $g^{\max}$ generations. During each generation, the single string solution, also referred to as a TSP solution, is clustered and assigned to vehicles. Each solution's fitness is calculated as the objective function of the specific solution. Based on the fitness, the algorithm reproduces the next generation through a combination of *cloning* the p_e fittest solutions exactly to the next generation, mutating p_m solutions through small changes referred to as perturbations, and creating p_c new offspring by performing crossover perturbations on a selection of generation g solutions.

The following sections discuss some of the elements of the GA in more detail.

6.1 Initialization

The simplest and computationally most efficient way of generating p initial solutions, each containing n customers, is to create p random permutations of integers between 1 and n. Each integer value represents a specific customer. To generate a population of 200 solutions (chromosomes), each with 200 nodes takes *MATLAB* on average 0.014 seconds (average obtained from 10,000 independent runs) on a standard *Intel Pentium Centrino* laptop computer with a 1.5GHz processor and 512MB of RAM.

As an alternative, initial solutions can be generated using the algorithm presented in Chapter 4, and adapted for the TS in Algorithm 5.2.

6.2 Clustering

Each chromosome represents a solution in the form of a single integer string, similar to the TSP strings proposed by Michalewicz (1992). The difficulty with having a single string to represent multiple tours and routes is that the chromosome needs to be *clustered*, and assigned to vehicles.

Although Tan et al. (2001c) simply adds the first allele of the chromosome to the end of the current tour until vehicle capacity is met, the author of this thesis propose the clustering routine presented in Algorithm 6.2 to address multiple scheduling. The first allele of the chromosome is considered for insertion on each edge of each route of the current tour, and not only at the end of the route. If no position is found for the customer, a new route on the current tour is considered. If an additional route leads to infeasibilities, a new tour is initialized, and the customer is inserted. A customer is only orphaned if it can not be serviced by a dedicated tour.

Algorithm 6.2: GA clustering

```
    Input: population
1   foreach chromosome in population do
2       repeat
3           $found \leftarrow 0$
4           forall the routes of current tour do
5               forall the edges on current route do
6                   if feasible insertion then
7                       $found \leftarrow 1$
8                   endif
9               endfall
10          endfall
11          if found = 1 then
12              insert customer
13          else if multiple routes are feasible then
14              insert customer into new route
15          else
16              create new current tour
17              create new first route
18          endif
19      until all customers are routed, or vehicles are depleted
20      report orphans
21  endfch
```

6.3 Mutation

A proportion, p_m of all chromosomes in a given generation is *mutated* to ensure that the GA does not get stuck in a local optimum (Vas, 1999). The proportion is typically very low to ensure that good chromosomes remain intact. Michalewicz (1992) introduces a non-uniform mutation rate whereby the number of chromosomes mutated decreases to ensure that the solution space is searched widely during early generations, and only searched locally in later generations.

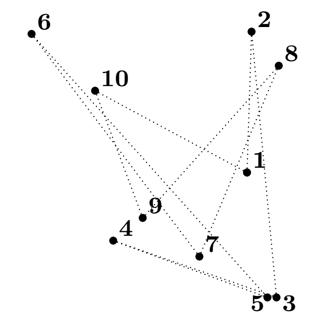
In the majority of applications binary representation is used and mutation involves changing a 0 value to 1, and vice versa. In this thesis the approach of Tan et al. (2001a) is followed whereby randomly selected customers are swapped in an integer string representation of a chromosome.

6.4 Crossover

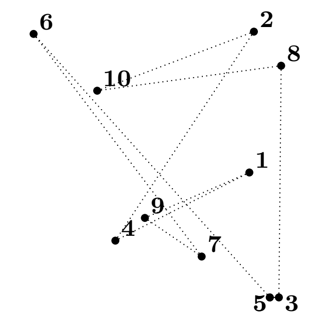
Crossover operators are concerned with producing offspring solutions for the next generation from two parent solutions from the current generation. Parents are selected using a biased roulette wheel. A number of the operators produce only a single offspring from the two parents, while others produce two offspring. To illustrate the various crossover operators, the first ten nodes of the $C2$-2-2 problem set is used.

6.4.1 Enhanced Edge Recombination (EER)

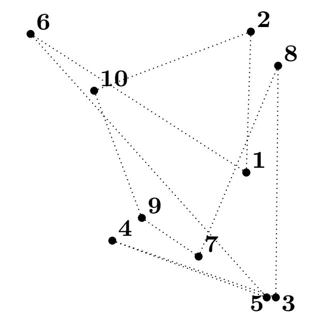
Whitley, Starkweather & Fuquay (as cited by Michalewicz (1992)) developed the Edge Recombination (ER) crossover technique which they claim transfer more than 95% of the edges from the parents to a single offspring. To illustrate the ER, consider two single string TSP solutions, A and B, illustrated in Figures 6.1(a) and 6.1(b) respectively. The edge table created in Table 6.1(a) lists for each node all the neighbouring nodes from both parent solutions. The single offspring, denoted by C, starts by selecting a starting element. Starkweather et al. (1991) state that the starting element can be either chosen randomly from the set of elements which has the fewest entries in the edge table, or a random choice between the starting element from either parent A or B. The latter option is used in this thesis. Of the elements that have links to the last element in C, choose the element which has the fewest number of unassigned links in the edge table entry, breaking ties randomly. The process is repeated until the new offspring chromosome is complete.



(a) $A = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10]$



(b) $B = [5\ 6\ 9\ 1\ 4\ 2\ 10\ 8\ 3\ 7]$



(c) $C' = [1\ 2\ 10\ 9\ 7\ 8\ 3\ 4\ 5\ 6]$

Figure 6.1: Two parent solutions illustrating the ER crossover

Table 6.1: Edge lists

(a) ER edge list

City	**Links**
1	2, 4, 9, 10
2	1, 3, 4, 10
3	2, 4, 5, 8
4	1, 2, 3, 5
5	3, 4, 6
6	5, 7
7	6, 8, 9
8	3, 7, 9, 10
9	1, 7, 8, 10
10	1, 2, 8, 9

(b) EER edge list

City	**Links**
1	2, 4, 9, 10
2	1, 3, 4, 10
3	2, 4, 5, 8
4	1, 2, 3, 5
5	3, 4, -6
6	-5, -7
7	-6, 8, 9
8	3, 7, 9, 10
9	1, 7, 8, 10
10	1, 2, 8, 9

Suppose element 1 is selected from A as starting element in C. Since 1 has been assigned to C, all occurrences of 1 is removed from the edge list. Element 1 has links to 2, 4, 9 and 10, each having 3 remaining links in the edge table. Element 2 is randomly chosen as next element in C and all element 2's occurrences are removed from the edge table. Element 2 has links with 3, 4 and 10, of which 4 and 10 have only 2 remaining links in the edge table. Element 10 is chosen randomly as the next element in C, having links to elements 8 and 9. Element 9 has the least (2) number of remaining links in the edge list, and chosen as the next element in C. The process continues until $C = [1\ 2\ 10\ 9\ 7\ 8\ 3\ 4\ 5\ 6]$.

To enhance the random breaking of ties when selecting among elements, Starkweather et al. (1991) changed the edge list to indicate common edges. This is achieved by flagging a common edge by inverting, for example, 3 to -3 if an element has a common edge to element 3 in both parents. Table 6.1(b) indicates the edge list with flagged common edges. When a tie exist between elements, preference is given to the element with the highest number of remaining flagged elements. If a tie still exists, it may be broken randomly. Following the same procedure as for the ER example above, a slightly different offspring $C' = [1\ 2\ 10\ 9\ 7\ 6\ 5\ 3\ 4\ 8]$ is obtained. The offspring chromosome is illustrated in Figure 6.1(c). The only *new* edge in the offspring is the edge connecting elements 6 and 1. Hence, 90% of the edges are transferred from the parents to the offspring solution.

6.4.2 Merged Crossover (MX)

The MX was first introduced by Blanton and Wainwright (1993) and is based on the notion of a global precedence among genes of any chromosome, rather than defining a precedence among genes specific to parents in a local crossover such as the EER. A number of precedence vectors have been established in literature.

Latest allowed arrival time

Chen et al. (1998) state that there is a natural precedence relationship among all customers based on the upper limit of their time windows. The precedence list, denoted by P, for the example problem is $P = [2\ 8\ 3\ 5\ 7\ 1\ 10\ 6\ 9\ 4]$, based on the time window details provided in Table 6.2.

To illustrate the crossover, we consider parents A and B from Figure 6.1. The first elements from both parents are considered: element 5 from B appears before element 1 from A in the precedence list P, and is selected as first element in offspring C. To maintain

Table 6.2: Time window details for customers from the $C2$-2-2 problem set

Customer, i	**Earliest allowed arrival,** e_i	**Latest allowed arrival,** l_i
1	2808	2968
2	668	828
3	1021	1181
4	0	3481
5	1922	2082
6	0	3451
7	2597	2757
8	906	1066
9	0	3475
10	0	3445

validity, elements 1 and 5 are swapped in parent A.

$A = [\mathbf{5}\ 2\ 3\ 4\ \mathbf{1}\ 6\ 7\ 8\ 9\ 10]$

$B = [5\ 6\ 9\ 1\ 4\ 2\ 10\ 8\ 3\ 7]$

$C = [5 \star \star \star \star \star \star \star \star \star]$

Next, the second elements of each parent is considered. As element 2 from A appears before element 6 from B in the precedence list, element 2 is placed in the offspring, and elements 2 and 6 are swapped in parent B.

$A = [5\ 2\ 3\ 4\ 1\ 6\ 7\ 8\ 9\ 10]$

$B = [5\ \mathbf{2}\ 9\ 1\ 4\ \mathbf{6}\ 10\ 8\ 3\ 7]$

$C = [5\ 2 \star \star \star \star \star \star \star \star]$

The process is repeated until the offspring chromosome is completed with $C = [5\ 2\ 3\ 1\ 4\ 6\ 7\ 8\ 9\ 10]$. The MX approach is denoted by MX_{l_i}.

Earliest allowed arrival time

Louis et al. (1999) suggest using the earliest allowed arrival time, given by e_i in Table 6.2, to establish the precedence list, denoted by MX_{e_i}. Executing their recommendation results in a precedence list $P = [4\ 6\ 9\ 10\ 2\ 8\ 3\ 5\ 7\ 1]$. When using the precedence list on parents A and B from Figure 6.1, an offspring chromosome $C = [5\ 6\ 9\ 4\ 1\ 2\ 10\ 8\ 3\ 7]$ results.

Time window compatibility

Two novel ways of establishing a precedence list are suggested in this thesis. In the first novel approach denoted by MX_{twc}, the total compatibility for each customer is calculated using either (4.8) or (4.9), sorted in ascending order to create the precedence list. Ties are broken arbitrarily. The resulting precedence list sees incompatible nodes placed earlier in the chromosome. More compatible nodes are subsequently inserted to fill routes and tours.

Angles

The second novel way to establish the precedence list is to reconsider the fundamental way in which the the crossover operator is used. The simplicity, yet success of the sweep algorithm proposed by Gillett and Miller (1974) is incorporated in this MX approach denoted by $MX_{\angle}$. The angle for each customer is calculated, and the angles are sorted in ascending order to determine the precedence list. The resulting crossover ensures that customers that are located close to one another are assigned to the same route, time windows permitting.

With the depot's location indicated by an open circle in Figure 6.2, the precedence list

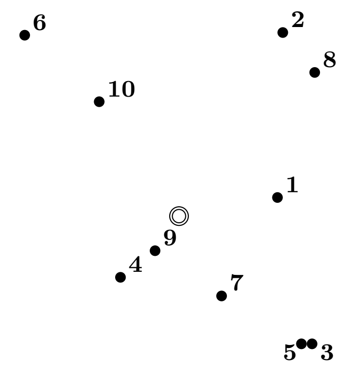


Figure 6.2: Depot and first 10 customers from the $C2$-2-2 problem set

$P = [1\ 8\ 2\ 10\ 6\ 9\ 4\ 7\ 5\ 3]$ is established.

6.4.3 Partially Matched Crossover (PMX)

PMX is a genetic operator often used with TSP problems using integer string representation (Goldberg and Lingle, 1985). The operator selects two parent chromosomes using the biassed roulette wheel, and produces two offspring chromosomes, as opposed to the previous operators producing only a single offspring. Consider again the two parent chromosomes A and B given in Figure 6.1. Two crossing positions a and b are randomly selected such that

$1 \leq a < b \leq \|A\| + 1$, where $\|A\|$ denotes the number of elements (alleles) in chromosome A. For illustrative purposes let $a = 3$ and $b = 7$. To create offspring 1, denoted by C_1, the strings between the crossing positions from parent 2 is copied to C_1.

$$C_1 = [\star\ \star\ 9\ 1\ 4\ 2\ \star\ \star\ \star\ \star]$$

For each element in A between a and b, starting from position a, look for elements in A that have not been copied to C_1. In the example element 3 is identified. Element 3's position in A is occupied by element 9 in C_1, and hence element 9 in A's position is filled in C_1 with 3 such that

$$C_1 = [\star\ \star\ 9\ 1\ 4\ 2\ \star\ \star\ 3\ \star].$$

Next, element 5 in A is identified, as element 4 has already been copied to offspring C_1. Element 5's position in A is occupied by 4 in C_1, but since element 4 in A's position is already occupied in C_1 by element 1, element 1's position in A is identified for element 5 in C_1 such that

$$C_1 = [5\ \star\ 9\ 1\ 4\ 2\ \star\ \star\ 3\ \star].$$

Element 6 in A is identified next. Element 6's position in C_1 is occupied by element 2, which in turn, is located in position 2 in A. Hence, element 6 is placed in position 2 in C_1 such that

$$C_1 = [5\ 6\ 9\ 1\ 4\ 2\ \star\ \star\ 3\ \star].$$

As all elements in A between positions a and b have been considered, C_1 is completed by duplicating the remaining elements from A such that

$$C_1 = [5\ 6\ 9\ 1\ 4\ 2\ 7\ 8\ 3\ 10].$$

The second offspring, denoted by C_2, is created in a similar fashion with the resulting offspring being $C_2 = [1\ 2\ 3\ 4\ 5\ 6\ 10\ 8\ 9\ 7]$.

6.5 Evaluating crossover operators

The proposed GA algorithm is executed to identify appropriate crossover operators for the varying problem sets. Due to computational time complexity, a single problem is randomly selected from each problem set. Each crossover operator is then tested using 4 independent iterations. The fitness is calculated using an objective function which considers total travel time, number of vehicles used, and total lateness at customers. The GA is executed for a maximum of 200 generations, each having 100 chromosomes. Of every new generation, 80% of chromosomes were generated through crossover operators. Initially 10% of a newly created generation is established through mutation to ensure that the solution space is widely searched. A non-uniform mutation rate introduced by Michalewicz (1992) reduces the number of mutated chromosomes as the number of generations increases. Hence the solution space is only locally searched towards the end of the algorithm. The balance of a new generation is created by cloning (copying exactly) the best chromosomes from the previous generation.

Figure 6.3 illustrates the performance of the various crossover mechanisms for each problem set. The performances are expressed and calibrated according to the best crossover operators for the specific problem set. Actual results are provided in Tables 6.3a and 6.3b, providing the best fitness (objective function value) obtained over the four independent runs, as well as the average time required (in seconds) to find a solution.

Contrary to Blanton and Wainwright (1993) claiming that most of their MX operators outperform the PMX operator, the results in this thesis indicates six instances in which the PMX is either significantly better in terms of fitness, or significantly faster than any of the other crossover operators. In only two instances, $c2\_2\_3$ and $rc2\_3\_8$, did MX prove significantly faster than the other crossover operators, of which one instance is the newly proposed MX_{twc}.

Using a standard statistical t-test, the EER crossover operators proved to be consistently worse and slower than other mechanisms, and is consequently omitted from further analysis. The remaining operators are again subjected to a t-test, resulting in some operators to be identified as significant, hence labels (e) and (f) in Tables 6.3a and 6.3b.

Self regulation can be achieved through a biased selection of operators based on past performance. Initially each operator (except EER) is assigned equal probability of being selected. A parameter λ indicates the frequency (in terms of generations) of testing *all* operators, and the probabilities are consequently adjusted based on the relative performance of each operator, similar to the self organizing mechanism proposed for exchange operators of the TS algorithm in Chapter 5.

Table 6.3a: Analysis of random problems for each data set

Problem			**EER**	$\mathbf{MX}_{l_i}$	$\mathbf{MX}_{e_i}$	$\mathbf{MX}_{\mathbf{twc}}$	$\mathbf{MX}_{\angle}$	**PMX**
*c*1_2_8	**Fitness**	Value	90026	88895	88510	85272	85043	84607
		Relative	1.064	1.051	1.046	1.008	1.005	1.000
		t-Value	-3.096[a]	-1.916	-1.514	1.865	2.104	2.559
	Time	Value	26478	25708	25747	25776	25560	25361
		Relative	1.044	1.014	1.015	1.016	1.008	1.000
		t-Value	-4.569[b]	0.412	0.160	-0.028	1.369	2.656[c]
*c*2_2_3	**Fitness**	Value	213070	212736	212572	212318	212261	212078
		Relative	1.005	1.003	1.002	1.001	1.001	1.000
		t-Value	-3.821[b]	-1.559[e]	-0.448	1.272	1.658	2.898[c]
	Time	Value	24498	23783	23695	23416	23595	23657
		Relative	1.046	1.016	1.012	1.000	1.008	1.010
		t-Value	-4.725[b]	-0.059	0.516	2.336[f]	1.168	0.764
*r*1_2_1	**Fitness**	Value	37147	36779	36358	36144	35610	34350
		Relative	1.081	1.071	1.058	1.052	1.037	1.000
		t-Value	-2.672[a]	-1.764	-0.724	-0.196	1.123	4.234[d]
	Time	Value	20348	19746	19736	19697	19658	19441
		Relative	1.047	1.016	1.015	1.013	1.011	1.000
		t-Value	-4.650[b]	0.201	0.282	0.596	0.911	2.659[c]

[a] Rejected with 97.5% certainty

[b] Rejected with 99.0% certainty

[c] Accepted with 97.5% certainty

[d] Accepted with 99.0% certainty

[e] Rejected with 97.5% certainty, **EER** omitted

[f] Accepted with 97.5% certainty, **EER** omitted

Table 6.3b: Analysis of random problems for each data set

Problem			**EER**	**MX**$_{l_i}$	**MX**$_{e_i}$	**MX**$_{\mathbf{twc}}$	**MX**$_{\angle}$	**PMX**
$r2\_2\_5$	**Fitness**	Value	51472	50715	50392	50147	49975	50277
		Relative	1.030	1.015	1.008	1.003	1.000	1.006
		t-Value	-4.434[b]	-0.994[e]	0.474	1.588	2.369	0.997
	Time	Value	17685	17160	17279	17374	17612	17003
		Relative	1.040	1.009	1.016	1.022	1.036	1.000
		t-Value	-3.113[a]	1.797	0.684	-0.204	-2.430[e]	3.266[c]
$rc1\_2\_8$	**Fitness**	Value	42833	42310	41693	41327	41257	40772
		Relative	1.051	1.038	1.023	1.014	1.012	1.000
		t-Value	-3.679[b]	-1.983[e]	0.018	1.205	1.432	3.005[c]
	Time	Value	61481	60192	60236	60007	59988	60215
		Relative	1.025	1.003	1.004	1.000	1.000	1.004
		t-Value	-4.908[b]	0.701	0.510	1.507	1.589	0.601
$rc2\_2\_8$	**Fitness**	Value	37348	34763	34433	34294	34271	34045
		Relative	1.097	1.021	1.011	1.007	1.007	1.000
		t-Value	-4.909[b]	0.189[e]	0.840	1.114	1.160	1.605
	Time	Value	46954	46193	46561	46686	46605	47711
		Relative	1.016	1.000	1.008	1.011	1.009	1.033
		t-Value	-0.803	2.813[c]	1.064	0.470	0.855	-4.400[b]

[a] Rejected with 97.5% certainty

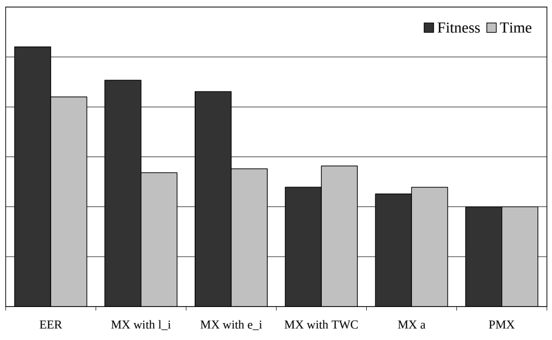
[b] Rejected with 99.0% certainty

[c] Accepted with 97.5% certainty

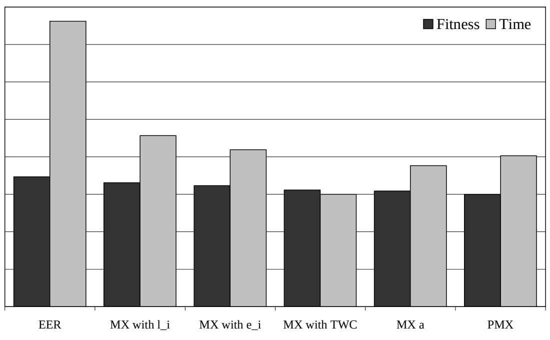
[d] Accepted with 99.0% certainty

[e] Rejected with 97.5% certainty, **EER** omitted

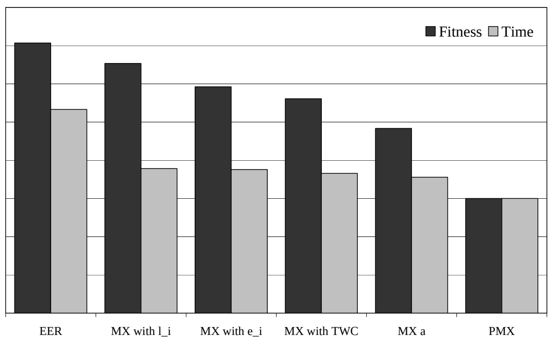
[f] Accepted with 97.5% certainty, **EER** omitted



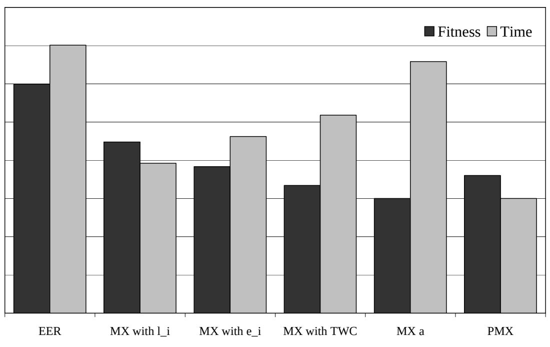
(a) $c1\_2\_8$



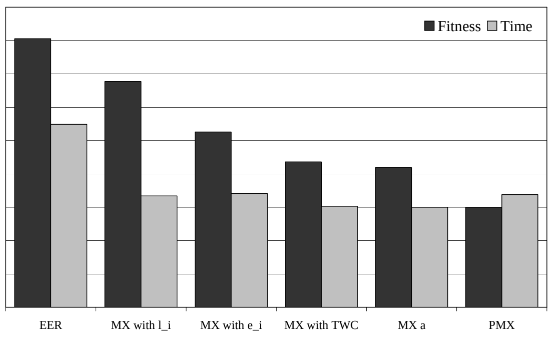
(b) $c2\_2\_3$



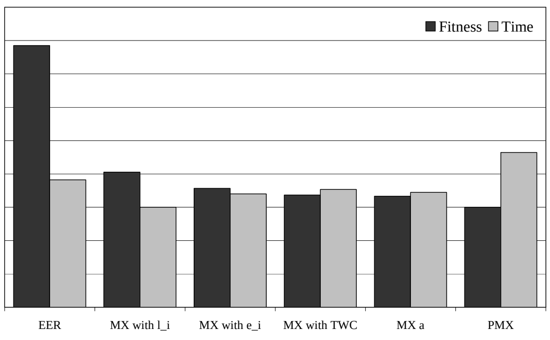
(c) $r1\_2\_1$



(d) $r2\_2\_5$



(e) $rc1\_2\_8$



(f) $rc2\_2\_8$

Figure 6.3: Results for a random problem from each set, expressed relative to the best crossover mechanism for each set.

6.6 Conclusion

In this chapter a GA with integer string representation is developed to test a variant of the VRP that uses time-dependent travel time and that accommodates time windows, a heterogenous fleet, and multiple scheduling. Six crossover mechanisms are tested, two of which are newly proposed in this thesis.

The results suggest that although there are performance differences among the crossover operators, few prove to be significant. Therefor, it is suggested that when integrating the multiple optimization algorithms, namely GA and TS, into the intelligent routing agent, internal learning or self regulation should be considered.