A Diversity-controlling Adaptive Genetic Algorithm for the Vehicle Routing Problem with Time Windows

Kenny Q. Zhu
Department of Computer Science
National University of Singapore
Singapore 119260
kzhu@comp.nus.edu.sg

Abstract

This paper presents an adaptive genetic algorithm (GA) to solve the Vehicle Routing Problem with Time Windows (VRPTW) to near optimal solutions. The algorithm employs a unique decoding scheme with the integer strings. It also automatically adapts the crossover probability and the mutation rate to the changing population dynamics. The adaptive control maintains population diversity at user-defined levels, and therefore prevents premature convergence in search. Comparison between this algorithm and a normal fixed parameter GA clearly demonstrates the advantage of population diversity control. Our experiments with the 56 Solomon benchmark problems indicate that this algorithm is competitive and it paves way for future research on population-based adaptive genetic algorithm.[1]

1 Introduction

Vehicle Routing Problem with Time Windows is a well-known combinatorial optimization problem [13], which is encountered very frequently in decision-making about the distribution of goods and services. The objective of the problem is to find routes for vehicles to service all the customers at a minimal cost (in terms of number of routes and total travel distance), without violating the capacity and travel time constraints of the vehicles and the time windows imposed by the customers. Because finding solutions to VRPTW with fixed fleet size is an NP-complete problem [13], current studies of this problem are primarily limited to the application of heuristic methods such as tabu search, simulated annealing, and evolutionary computing.

Genetic algorithm, originally developed by Holland[6], is a stochastic heuristic that simulates the optimization process with the natural evolution of genes in a population of organisms. GA was first applied to solve VRPTW by Thangiah, *et al.*[14] in an algorithm called GIDEON. Subsequently it was studied in [8, 11, 7], etc. Most of the studies use hybrid methods which combine GA with other meta-heuristics such as tabu search. The focus has been on devising string representations and operator types particular to VRPTW, and trying to achieve better solutions to open benchmark problems such as the Solomon VRPTW problem sets[13]. A bit string representation was implemented by Thangiah, *et al.*[14], while an integer string representation for VRPTW can be found in [10].

Some studies have been devoted to adaptive GA and population diversity control. A good survey about the aspects of adaptive GA's can be found in [5]. Measures of population diversity were studied by Barker[1], Burke[3] and Morrison, *et al.* [9]. A Diversity-Control-Oriented Genetic Algorithm was presented in [12], in which the selection procedure is adaptive.

In this paper, we present a new adaptive GA for VRPTW based on population diversity. Here we measure the population diversity by pair-wise Hamming distance among genotypes in the population environment. We propose a novel function to control this diversity at a desirable level by adapting the crossover and mutation rates, in order for the algorithm to progress on the search landscape without premature convergence. We also introduce a new way of encoding VRPTW solutions into integer chromosomes(genotypes) and decoding the genotypes into phenotypes(fitness), despite the claim by other researchers that representing VRPTW solutions in GA is very difficult [11]. We benchmarked this adaptive algorithm against the standard Solomon problems, and obtained very competitive solutions to the published best in the literature. The main contributions of this paper are: (1) A simple integer representation of the chromosomes and a unique decoding scheme; (2) A novel population diversity control mechanism for VRPTW, which has achieved good results.

[1]Many thanks to Ziwei Liu for her kind assistance in this project.

2 Definition of VRPTW

The Vehicle Routing Problem with Time Windows is given by a set of identical vehicles *V*, a special node called the *depot*, a set of customers *C* to be visited, a directed network connecting the depot and the customers. Let us assume there are *R* vehicles, and K customers. For simplicity, we denote depot as customer 0. Each arc in the network corresponds to a connection between two customers. A *route* is defined as starting from the depot, through a number of customers and back at the depot. The number of routes in the traffic network is equal to the number of vehicles used, that is, at most R. Therefore, exactly *R* directed arcs leave the depot and *R* arcs return to the depot. A cost (distance) c_{ij} and a travel time t_{ij} are associated with each arc of the network. Every customer in the network must be visited only once by exactly one of the vehicles. Since each vehicle has a limited capacity q_r, and each customer has a varying demand m_i, q_r must be greater than or equal to the summation of all demands on any route. Any customer i must be serviced within a pre-defined *time window* $[e_i,\ l_i]$. Vehicles arriving later than the latest arrival time are penalized while those arriving earlier than the earliest arrival time incur waiting w_i. Vehicles should also complete their individual routes within a stipulated *trip time* r_k, which is also the time window of the depot. A vehicle cost c_r is associated with each route. c_r is large enough such as minimizing the total cost translates to minimizing the routes and then the total distance. c_r can be factored into the arc cost c_{0i}. Formally,

$$Min \sum_{i=0}^{K} \sum_{j=0}^{K} \sum_{k=0}^{R-1} c_{ij} x_{ijk} \tag{1}$$

Subject to:

$$\sum_{k=0}^{R-1} \sum_{j=1}^{K} x_{ijk} \leq R \quad \text{for } i = 0 \tag{2}$$

$$\sum_{k=0}^{R-1} \sum_{j=0,\, j\neq i}^{K} x_{ijk} = 1 \quad \text{for } i = 1,\, 2,\, ...,\, K \tag{3}$$

$$\sum_{i=0,\, i\neq h}^{K} x_{ihk} - \sum_{j=0,\, j\neq h}^{K} x_{hjk} = 0, \quad \forall h \in [1,\, K];\ k \in [0,\, R-1] \tag{4}$$

$$u_i - u_j + K x_{ij} \leq K - 1 \quad \text{for } i = 1,\, 2,\, ...,\, K; \quad j = 1,\, 2,\, ...,\, K;\ i \neq j \tag{5}$$

$$\sum_{i=0}^{K} m_i \sum_{j=0,\, j\neq i}^{K} x_{ijk} \leq q_k \quad \forall k \in [0,\, R-1]; \tag{6}$$

$$\sum_{i=0}^{K} \sum_{j=0,\, j\neq i}^{K} x_{ijk}(t_{ij} + f_i + w_i) \leq r_k, \quad \forall k \in [0,\, R-1] \tag{7}$$

$$t_0 = w_0 = f_0 = 0 \tag{8}$$

$$\sum_{k=1}^{R} \sum_{i=0,\, i\neq j}^{K} x_{ijk}(t_i + t_{ij} + f_i + w_i) \leq t_j, \quad \forall j \in [1,\, K] \tag{9}$$

$$e_i \leq (t_i + w_i) \leq l_i \quad k \in [0,\, R-1] \tag{10}$$

3 Algorithm

When traditional GA's are used for multi-modal function optimizations, the solutions are often stuck at local optima. This is because there are two conflicting forces in the GA search dynamics: convergence from selection and recombination, and divergence from crossover and mutation. Fixed control parameters such as crossover and mutation rates are usually not able to balance the two forces, causing the population to either converge too quickly or drift off easily. Adaptive control to population diversity, which will be introduced in Section 4, provides a robust way of escaping from local optima and balancing the forces of divergence and convergence. In what follows, we first present the basic algorithm skeleton which is common to both fixed parameter and adaptive algorithm.

GA-1 Generate initial population of N chromosomes with a mix of random and good solutions.

GA-2 Decode chromosome to obtain fitness values in the population. Set fixed crossover rate p_c and mutation rate p_m, or calculate the Hamming-based diversity, and set p_c and p_m.

GA-3 Create a new population by repeating the following steps:
1. Select two parent chromosomes from a population by Tournament Selection;
2. With a probability p_c, crossover the parents to form two new offsprings, otherwise copy the parents to become offsprings;
3. With probability p_m, mutate the new offsprings;
4. Place the new offsprings in a new population;

GA-5 Replace the old population with the newly generated population;

GA-6 If the stop criterion is satisfied, stop, perform a 2-interchange (GB) local search[14] on the best chromosomes in the current population and return the global best solution; else go to GA-2.

3.1 Chromosome Representation

The individuals of a population in the adaptive GA for VRPTW are string entities of artificial chromosomes. The

representation of a solution is a string of K integers, where K is the number of customers in question. Each *gene* in the string, or the chromosome, is the *integer node number* designated to a customer. The chromosome is the concatenation of the routes in the solution. In other word, one chromosome is the encoding of one solution. For instance, the following solution:

```
Route 1: 0 -> 3 -> 2 -> 4 -> 5 -> 0
Route 2: 0 -> 10 -> 6 -> 1 -> 12 -> 11
         -> 0
Route 3: 0 -> 9 -> 8 -> 7 -> 0
```

can be encoded into:

$$3-2-4-5-9-8-7-10-6-1-12-11 \quad (11)$$

Two observations about the representation:

1. Depots are not coded in as delimiters, so that ordinary crossover operations can be used.

2. The routes have been ranked by the first customer number in each route before the encoding. In the creation of the initial population, duplicated solutions (differing by the order of the routes) can be detected and removed.

3.2 Fitness of the Chromosomes

The fitness value corresponds to the quality of a solution. Since the vehicle cost c_r is not available in our benchmark Solomon problems, in this paper, we devise the fitness function using both the number of routes and the total distance traveled by all vehicles:

$$f_i = R_i + \frac{D_i}{D_{max}}, \quad (12)$$

where f_i is the fitness of chromosome i in the population, R_i is the number of routes in the solution represented by chromosome i, D_i is the total distance in that solution, and D_{max} is the absolute maximum total distance traveled in any solution:

$$D_{max} = \sum_{i=1}^{K} 2d_{0i}, \quad (13)$$

where d_{0i} is the distance between the depot and a customer i. From now on, d_{ij} replaces c_{ij} in Eqn. (1) as the cost of the arc from customer i to j. By Eqn. (12), solution S_1 is better than solution S_2 if S_1 has fewer routes. When both have the same number of routes, S_1 is superior if its total distance is smaller. In other words, *the better the solution, the smaller the fitness value*.

3.3 Decoding of Chromosome

To find out the fitness of a given chromosome (gene sequence) $(g_1, g_2, \ldots, g_K)$, we need to decode the chromosome back to a VRPTW solution. Since there is no delimiters in our chromosome encoding, there are exponential number of ways to break a string of numbers down to segments.

Lemma 1 *At least one feasible solution can be decoded from* $(g_1, g_2, \ldots, g_K)$.

Proof: One feasible but worst possible solution is breaking the chromosome up into K routes, with each route containing only one node. The capacity and trip time constraints are obviously satisfied. If we assume triangular inequality, i.e. $\forall(i,\ j),\ d_{0i} < d_{0j} + d_{ji}$, then the time window constraint is satisfied, too, except waiting time is likely to be incurred. This assumption is valid if Euclidean distances are used in the problem definition. □

There are finite feasible solutions, and one of them has the best fitness. This best solution can be obtained by a weighted dag, G, with $K+1$ vertices and M edges, and a shortest-path algorithm. The following procedure describes the construction of G.

We start G with $K+1$ vertices marked with unique integers from 0 through K. For every possible subsequence $(g_{i+1}, \ldots, g_j)$ in C, we add an edge between vertices i and j if sequence $(0, g_{i+1}, \ldots, g_j, 0)$ represents a feasible route r in a VRPTW solution. For example, in the chromosome in (11), the (0, 5, 0) is always a possible route, and since 5 is in the 4^{th} place in the chromosome, edge (3, 4) is added to the graph. Similarly, if (0, 5, 9, 8, 0) is another feasible route, edge (3, 6) is added to G. The weight of the edge is calculated as:

$$w_r = 1 + \frac{d_r}{D_{max}}, \quad (14)$$

where d_r is the trip distance of route r. The definition of the weight corresponds to the fitness of that route. It takes at most $K \times (K+1)/2$ steps to complete G.

We then use the Bellman-Ford algorithm to find the optimal path P from vertex 0 to K, which takes $O(K \times M)$ steps, where M is at most $K \times (K+1)/2$. Matching the path to the chromosome C gives the best possible solution in that setting. The fitness of the chromosome can be computed from the resulting solution, or $\sum_{r \in P} w_r$. The procedure costs $O(K^3)$ time, which is acceptable for practical size K, given the modern computing power.

3.4 Creation of Initial Population

The initial population of size N is created with a mix of "good" and "bad" chromosomes without duplication.

The good ones consist of a feasible solution from a *Push-Forward Insertion Heuristic*, *I1*[13], and some of its 2-opt neighbors. A *2-opt operation* is characterized by moving one or two customers from one route to another or within a route; or swapping one or two customers between two routes or within a route. A set of all 2-opt operations on a solution defines a *2-opt neighborhood*. Good building blocks in these elite solutions are expected to drive the search toward optimum. The "bad" ones are randomly generated chromosomes, which introduce variance to the population and prevent the evolution process from converging too quickly. The percentage of random chromosomes in the initial population is θ (random factor).

3.5 Selection

A modified *binary tournament selection* is adopted. We create the mating population by two identical passes, with each pass generating half of the mating population. A pass is described as:
(1) Start with empty mating population Q;
(2) Randomly shuffle the original population P;
(3) Starting from the first chromosome, compare two adjacent ones p_{2i} and p_{2i+1} and append the one with better fitness value to the mating population as q_i;
(4) Repeat step (3)

Subsequently q_{2i} and q_{2i+1} will mate for all $i = 0, 1, ..., N/2 - 1$. Note that although a chromosome might be selected in both passes, it will not be mating with itself because it will appear in different halves of the mating population.

3.6 Reproduction

Reproduction is one of the most crucial functions in the chain of genetic evolution. It serves the important purpose of combining the useful traits from parent chromosomes and passing them on to the offsprings. There are two types of operations, *crossover* and *mutation*. The rates of application of these operators, namely *crossover rate* p_c and *mutation rate* p_m, are the key to this algorithm, and in this paper, they are adaptive to the population diversity.

3.6.1 Crossover

Conventional single/double point crossover operators are only applicable to genetic representations where the permutation of the genes does not matter. In the context of VRPTW, where each integer element appears only once in any chromosome, we use the two order-based crossover operators: *Partially Matched Crossover (PMX)* and *Order Crossover (OX)* described in [4] to prevent invalid offsprings from being reproduced. Each of them has 50% chance of being applied at reproduction time.

3.6.2 Mutation

Because the chromosomes are of fixed length in our implementation, the following three types of mutation are used here with equal probability (33.3%).

One-step route reduction This operator aims at reducing the number of routes in the given chromosome. We decode the chromosome into a best possible solution S, attach a probability p_r to every route r in S, where $1/p_r$ is proportional to the number of customers in r and $\sum_{r \in S} p_r = 1$. Construct a roulette wheel according to the probabilities, and randomly select a route by the roulette wheel. We then attempt to remove every customer from that route and insert it into other routes, in hope of eliminating the route altogether.

One-step cost reduction This operator aims at reducing the total travel distance of the solution that is decoded from the chromosome. We randomly select two routes from the decoded solution and attempt to shift some nodes or exchange some nodes between the two routes. The operation ends as soon as the total distance is reduced.

Gene relocation This operator directly applies to the chromosome string. It simply shifts a gene (integer element) from its present location to some random location in the chromosome. This is the key operator to promote population diversity.

3.7 Recovery and Termination

Both crossover and mutation operations introduce forces of diversification. Two good parents may produce a bad offspring. To avoid adverse population drifts, an *elite recovery* process is carried out at the end of each generation, in which the best ϵN individuals in the previous populations are retained in this population without duplication, where $0 \leq \epsilon < 1$. As a result, the minimum fitness in the population improves monotonically over generations.

Our adaptive algorithm stops at n^{th} generation when the best solution has not been improved for the last $0.3n$ generations, subject to a minimum of g_{min} and a maximum of g_{max} generations. The one can easily show the following lemma.

Lemma 2 *If a global best is found, the stopping criterion guarantees termination of the program.*

4 Adaptive Control

4.1 Motivation

Before going into the specifics of adaptive control, we first justify the philosophy that *a diversified population helps prevent premature convergences and achieve better solution*. While the ways to measure diversity of a genetic

population suffice, we choose to use the sum of the *Hamming distances* among the chromosomes in the genotype space.

Definition 1 (Genotype) *A genotype is the genetic blueprint of a chromosome. In the context of GA, this is the physical sequencing of the genes in an individual.*

The Hamming distance between genotypes u and v is defined as:

$$Ham(u, v) = \sum_i |sgn(u[i] - v[i])|, \qquad (15)$$

where $u[i]$ and $v[i]$ are the i^{th} gene of u and v, respectively. And the Hamming based population diversity of population P is:

$$\mathcal{GD}(P) = \frac{1}{2} \sum_{i \neq j} Ham(P[i], P[j])), \qquad (16)$$

where $P[i]$ is the i^{th} genotype in P.

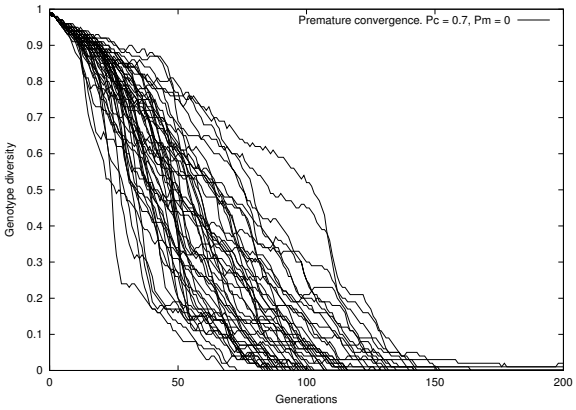

Figure 1. Rapid decline of genotype diversity in fixed settings

In a fixed-parameter GA, the genotype diversity declines very rapidly. Figure 1 illustrates 50 runs of our base algorithm on Solomon problem R101, with $p_c = 0.7$ and $p_m = 0$. The initial population is completely random, that is, $\theta = 1$, and there is no recovery of elite chromosomes, so $\epsilon = 0$. The diversity drops to zero in most runs by 150 generations, when the algorithm has converged a local optimum. If we rank the average diversity $\mathcal{GD}_{avg}$ over 200 generations for every run, and also the mean fitness of the population at the 200^{th} generation for each run, we arrive at a scatter plot in Figure 2. Every point in the plot stands for one of the 50 runs. An observable trend is a north-west to south-east diagonal band of points, which indicates that a run with larger average diversity results in a better quality

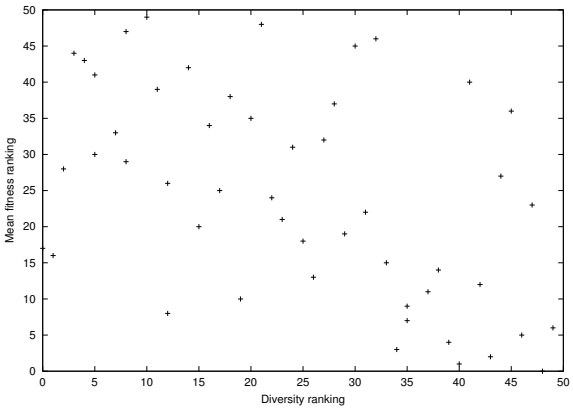

Figure 2. Correlation between diversity and solution quality

population (smaller mean fitness). This subtle correlation suggests that higher population diversity give rise to better solutions.

4.2 Controlling Diversity by Crossover and Mutation Rates

There are a number of ways to control population diversity. In this paper, we propose an adaptive control that is based on varying crossover and mutation rates. To demonstrate the effects of crossover and mutation operators on the genotype diversity, we ran the basic, non-adaptive GA with a range of $0.3 \leq p_c \leq 0.9$ and $0.1 \leq p_m \leq 0.7$ on problem R101, and the results were summarized in Figure 3 and 4. From these figures, increasing crossover and mutation rates promotes diversity and delays the convergence of the algorithm, though such effect is not proportional to the value of p_c and p_m. The most effective region of control for R101 appears to be for $0.6 \leq p_c \leq 0.8$ and $0.1 \leq p_m \leq 0.2$. The main idea now is to calibrate p_c and p_m so that the population diversity $\mathcal{GD}$ could be sustained at a healthy level, a level at which search can progress but does not converge easily.

We apply the following adaptive function on p_c and p_m to maintain diversity at a target level throughout the run.

$$p' = p \left[1 + \frac{\xi(\mathcal{GD}_t - \mathcal{GD})}{\mathcal{GD}} \right], \qquad (17)$$

where p is the current value of crossover or mutation rate, p' is the value in the next generation, subject to the absolute min/max value of p_{min} and p_{max} (in this paper 0 and 1 respectively), $\mathcal{GD}$ is the current generation genotype diversity, $\mathcal{GD}_t$ is the target diversity, and ξ is a sensitivity constant. The crossover and mutation rates are updated every

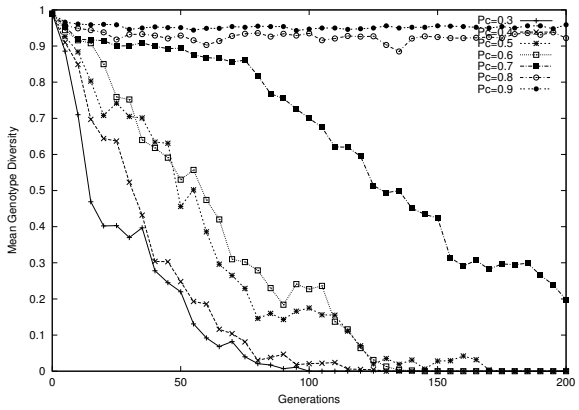

Figure 3. Effects of p_c on diversity

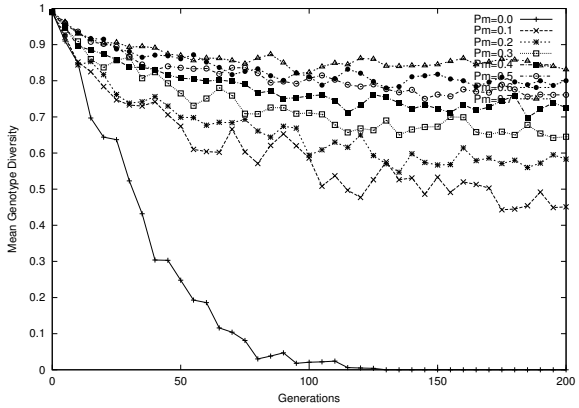

Figure 4. Effects of p_m on diversity

generation according to (17). Note that this equation partly depends on the initial values of p_c and p_m, which are usually problem specific.

5 Experiments and Discussion

The benchmark problems were all taken from the well-known 56 Solomon VRPTW instances (100 nodes) [13]. There are several objectives in our experiments: (1) to determine relationship between target diversity $\mathcal{GD}_t$ and search quality, and hence suggest a effective target genotype diversity for VRPTW; (2) to compare and contrast our adaptive algorithm against the fixed parameter counterpart; (3) to benchmark our best solutions against the published best solutions and solutions from other evolutionary heuristics for the Solomon instances. We use the follow default control parameters unless otherwise stated.

$$
\begin{aligned}
N &= 50 \text{ (Population size)} \\
\theta &= 0.9 \text{ (Random factor)} \\
p_{c,0} &= 0.8 \text{ (Initial probability of crossover)} \\
p_{m,0} &= 0.1 \text{ (Initial mutation rate)} \\
\epsilon &= 0.02 \text{ (Elite recovery rate)}
\end{aligned}
$$

A larger sensitivity ξ corresponds to a more pro-active response to a diversity drift, which means the control changes the crossover/mutation rate more rapidly. A smaller ξ, on the other hand, means more gradual change in the rate, and that translates into *slower* fluctuation of the diversity, a phenomenon we call *ripple effect* (Figure 5). We postulate that bigger ripples allow for both *global exploration* and *local exploitation*. Hence, we set $\xi = 0.01$ for crossover and $\xi = 0.02$ for mutation control.

5.1 Target Diversities

$\mathcal{GD}_t$	0.1	0.3	0.5	0.7	0.9
R101	20.344	20.342	**19.864**	20.342	20.345
	20.607	20.787	22.930	24.300	28.312
R201	4.287	4.288	**4.284**	4.291	4.316
	4.447	5.165	6.657	9.050	14.706
C104	10.150	10.150	**10.147**	10.152	10.168
	10.254	10.376	13.302	12.565	12.764
C204	4.134	4.135	**4.129**	4.135	4.140
	4.255	4.625	4.999	5.593	8.893
RC101	17.260	17.260	**17.259**	17.260	17.261
	17.361	17.624	17.948	20.164	22.989
RC201	5.256	5.248	**5.246**	5.257	5.268
	5.604	5.496	5.882	5.566	12.862

Table 1. Search quality under different $\mathcal{GD}_t$

With an identical, mixed population, we compared the effects of $\mathcal{GD}_t$ from 0.1 to 0.9 at step of 0.2 to the population's best and mean fitnesses on several selected problems from the problem set. All the problems tested here are considered "hard" problems from each category for which optimal solutions are not likely to be found within 500 generations. Table 1 summarizes the average results from 5 runs in each problem. The first number in each cell is the average minimum(best) fitness and the second number is the average of mean fitness of the population at 500^{th} generation. The numbers in boldface of each row are the best solutions for the problem of that row. Apparently, in all problems, setting $\mathcal{GD}_t = 0.5$ gives better search quality. Hence for the remaining experiments, we fix $\mathcal{GD}_t = 0.5$.

To illustrate the impact of adaptive control on the search progress, we plot a sample run of problem R101 with different $\mathcal{GD}_t$'s in Fig. 5. A complete random initial population was used here, and there is no elite recovery. Population converges monotonically with small $\mathcal{GD}_t$ and drifts off unboundedly with large $\mathcal{GD}_t$.

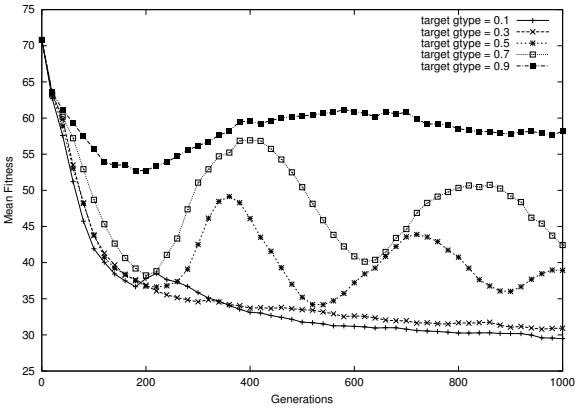

Figure 5. Sample search progress under target diversities 0.1 - 0.9 (Problem R101, $\epsilon = 0$)

5.2 Adaptive vs. Non-adaptive

To compare the adaptive algorithm with the non-adaptive one, we first run the adaptive GA 5 times with $g_{min} = 500$ and $g_{max} = 5000$. We compute the average p_c and p_m in runs of the adaptive algorithm for all 56 problems, and use these settings in the fixed-parameter GA for the respective problems. Table 2 lists the average numbers of routes and total distances in the best solutions for all problems in the 5 runs, presented in categories. Running time of these problems are also included. The run-times are in seconds on a Pentium 4 2.4 GHz/512MB RAM. One can see that the adaptive algorithm leads in the solution quality consistently in all problem sets. The results clearly show that adaptive control plays a role in improving solution quality and possibly directing search to unknown regions to avoid being trapped in a locality.

5.3 Our Best Solutions Against Other Evolutionary Results

We now compare the results we have obtained from the previous subsection with several evolutionary methods in the literature and the best ever solutions by all heuristic methods to our knowledge. The results are presented in Table 3. The average solutions come in a pair of average number of routes and average cost.

According to Table 3, our adaptive algorithm, dubbed DAGA, has managed to find all optimal solutions in problem set C1 and C2, and fared quite competitively in R2. In general, the total distance is better than GIDEON and GENEROUS, although it is somewhat behind HG1/2. The number of routes are not very good in R1, RC1 and RC2. This is probably because the other heuristics (except for I1) use more sophisticated crossover and mutation operators to improve route numbers, e.g. the route-based crossover, two-level exchange mutation and local search mutation in [11]. Our crossover operators are completely generic to any sequence based problems. The results suggest that using VRPTW specific operators or hybrid methods may be essential in providing GA solutions to VRPTW.

Another observation is that compared with other methods, it takes our algorithm more CPU cycles to reach the current solutions, especially in C2, R2 and RC2. As we know these three sets of problems involves long routes, therefore the dag constructed for such solutions has more edges and it takes longer to construct and to find the shortest path. In addition, when the population diversity happens to swing well below the target level, most of the individuals are close to a local optimum. At this time, it takes much longer for the mutation operators to reduce a route or some distance. It is also possible that the search may be trapped in some loops for some time. In this respect, tabu search can be helpful in preventing the crossover or mutation to produce the same offsprings repeatedly.

Nevertheless, the results in Table 3 has not weakened our claim that population diversity control and adaptation of crossover and mutation operator is a very useful alternative to improving search quality, because we have shown that VRPTW can be solved from a different angle even with such simple representation and straightforward operators.

In conclusion, we have presented a new genetic algorithm for VRPTW based on adaptive control of the crossover and mutation rates. The algorithm adapts p_c and p_m to the population dynamics, in order to maintain a stable population diversity at a desirable level. The new algorithm features an intuitive integer encoding of the VRPTW, and uses a unique method to decode chromosomes into VRPTW solutions and hence the fitness values. Our experiments indicate that the adaptive population diversity control is a very useful means to prevent premature convergence and to improve search results.

References

[1] Allen L. Barker and Worthy N. Martin. Dynamics of a distance-based population diversity measure. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 1002–1009, Piscataway, NJ, 2000. IEEE Service Center.

[2] Olli Bräysy and Wout Dullaert. A fast evolutionary metaheuristic for the vehicle routing problem with time windows. *International Journal of Artificial Intelligence Tools*, 12(2), 2002.

[3] Edmund Burke, Steven Gustafson, and Graham Kendall. A survey and analysis of diversity mea-

Category	p_c/p_m	Fixed	Time	Category	Adaptive	Time
C1	0.88/0.49	10/835.6	72	C1	10/828.9	80
C2	0.89/0.36	3/610.9	804	C2	3/589.9	735
R1	0.85/0.56	13.3/1263.4	131	R1	12.8/1242.7	305
R2	0.84/0.48	3.2/1021.4	1288	R2	3/1016.4	1308
RC1	0.85/0.57	13.1/1437.2	177	RC1	13/1412.0	239
RC2	0.84/0.49	3.9/1249.7	765	RC2	3.7/1201.2	883

Table 2. Fixed parameter vs. adaptive algorithm

Prob.	Best ever [1]	I1 [2]	GIDEON [3]	GEN [4]	HG1/2 [5]	DAGA [6]
C1	10/828.4	10/951.9	10/892.1	10/838	10/828.4	10/828.9
C2	3/589.9	3.1/692.7	3.0/749.1	3/590	3/589.9	3/589.9
R1	11.9/1228.1	13.6/1436.7	12.8/1300.3	12.6/1296.8	11.9/1228	12.8/1242.7
R2	2.7/961.3	3.3/1402.4	3.2/1124.7	3/1117.7	2.7/970.0	3/1016.4
RC1	11.5/1395.1	13.5/1596.5	12.5/1474.1	12.1/1446.2	11.6/1392.6	13/1412.0
RC2	3.3/1139.4	3.9/1682.1	3.4/1411.1	3.4/1360.6	3.3/1144.4	3.7/1201.2

Table 3. Average of the best results in each category against the best known

sures in genetic programming. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 716–723. Morgan Kaufmann Publishers, 2002.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Pub. Co., 1989.

[5] Francisco Herrera and Manuel Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing*, pages 95–125, 1996.

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

[7] Jörg Homberger and Hermann Gehring. Two evolutionary meta-heuristics for the vehicle routing problem with time windows. *INFORMS Jorunal on Computing*, 37(3):297–318, 1999.

[8] J.L. Blanton Jr. and R.L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[9] Ronald W. Morrison and Kenneth A. De Jong. Measurement of population diversity. In *5th International Conference, Evolution Artificielle, EA 2001, Le Creusot, France, Selected Papers*, pages 31–41. Springer, 2001.

[10] F. B. Pereira, J. Tavares, P. Machado, and E. Costa. GVR: a new genetic representation for the vehicle routing problem. In *Proceedings of ICS 2002 13th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 95–102, 2002.

[11] Jean-Yves Potvin and Samy Bengio. The vehicle routing problem with time windows - part ii: Genetic search. *INFORMS Journal on Computing*, 8:165–172, 1996.

[12] Hisashi Shimodaira. DCGA: A diversity control oriented genetic algorithm. In *9th International Conference on Tools with Artificial Intelligence (ICTAI '97), the Proceedings*, pages 367–374. IEEE Computer Society, 1997.

[13] Marius M. Solomon. Algorithms for vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 1987.

[14] S. R. Thangiah, K. E. Nygard, and P. L. Juell. Gideon: A genetic algorithm system for vehicle routing with time windows. In *Proceedings of the Seventh Conference on Artificial Intelligence Applications*, pages 322–325, 1991.

[1] Best published: From [2]

[2] I1: 24-63 secs on DEC-10, 1 run.[13]

[3] GIDEON: 2 mins on Solbourne 5/802, runs unknown.[14]

[4] GENEROUS: 4-41 mins on SPARC 10, runs unknown.[11]

[5] HG1/HG2: 13-19 mins on Pentium 200MHz, 10 runs.[7]

[6] DAGA: Our diversity-controlling adaptive GA, 1-21 mins on Pentium 4 2.4GHz, 5 runs.