Malaysian Journal of Science **21**: 131-141 (2002)

Application of genetic algorithms in vehicle routing problems

Noor Hasnah Moin

Institute of Mathematical Sciences, Faculty of Science, University of Malaya, 50603 Kuala Lumpur, Malaysia

ABSTRACT In this paper we propose four methods based on Genetic Algorithms for Vehicle Routing Problems. It is observed that algorithms that construct the routes sequentially produce superior results compared to parallel route building methods. In general, savings GA performs well for all the problems tested whilst vertex sequencing method produces superior solutions for small problems.

ABSTRAK Di dalam makalah ini kami mengutarakan empat kaedah yang berasaskan Algoritma Genetik bagi Masalah Perjalanan Kenderaan. Kami mendapati bahawa algoritma yang membina jalan secara jujukan menghasilkan keputusan yang lebih baik daripada kaedah pembentukan jalan secara selari. Secara amnya, kaedah penjimatan GA memberikan persembahan yang baik bagi semua masalah yang diuji sementara kaedah jujukan bucu memberikan keputusan yang lebih baik bagi masalah yang kecil.

(vehicle routing problems, genetic algorithms, heuristic methods)

INTRODUCTION

Vehicle Routing Problems (VRP), which was first described as *Truck Dispatching Problems* by Dantzig and Ramser (1959), lie at the heart of the distribution management. They involve designing optimal delivery routes, originating and terminating at one or several depots, for a fleet of vehicles that services a number of geographically scattered cities or customers subject to various side constraints. A common objective therefore is to find a set of routes for vehicles that satisfies a variety of constraints so as to minimise the total fleet operating cost.

The area of VRP has been intensively studied and a rich literature exists in the bibliographies contained in Laporte and Osman (1995). Applications of VRP cover a broad spectrum of problems such as school bus routing, design of dial-a-ride systems, collection of mail from mail boxes, distribution of gas industrial systems, snow ploughing, winter gritting, etc.

Since VRP has been shown to be *NP*-hard, the computational time for exact algorithms for relatively large problems is likely to be prohibitive. Furthermore, in most real world problems, the emphasis is no longer on getting the optimum value (which is often not known) but on achieving relatively *good* solutions in reasonable amount of times. Consequently research has been devoted to developing effective *heuristic methods* and *metaheuristics* that will allow large problems to be solved reasonably quickly.

Metaheuristic algorithms are iterative generation processes that guide a subordinate heuristic by combining, intelligently, different concepts for exploring and exploiting the solution search space. Metaheuristics include, among others, Tabu Search, Simulated Annealing and Genetic Algorithms.

Genetic Algorithm (GA) is a stochastic search technique that closely mimics the metaphor of natural biological evolution. GA explores the problem domain by maintaining a population of individuals, which represents a set of potential solutions in the search space. The survival of each individual into the next generation is determined by its fitness. The fitness of an individual is a performance measure based on an objective function that describes the problem. At each iteration, new individuals (*offspring*) are created by selecting old individuals according to their fitness and breeding them using genetic operators similar to natural genetics. The selection is carried out based on the principle of

the *survival-of-the-fittest* where stronger individuals are allowed to participate more in the reproduction of new individuals than the weaker ones, who may not even contribute at all. Using genetic operators, GA attempts to combine the good features found in each individual using a structured yet randomised information exchange in order to construct individuals which are better suited to their environment than the individuals from which they were created. Through the evolution of better individuals, it is hoped that the desired solution will be found.

In the last few years there has been a surge of interest in using GA, or techniques which are reminiscent of GA, in solving VRP and its related problems (Thangiah et al. (1993, 1995), Blanton and Wainwright (1993), Potvin and Bengio (1993), Potvin and Dube (1994), Stefanitsis et al (1995), Badeau et al (1996) and Chua et al. (1996)). In particular, the algorithm proposed by Badeau et al. (1996) which constructs an algorithm that follows some rules reminiscent of GA. They first decompose the initial solution into several subproblems, and each subproblem is optimised independently using a Tabu Search heuristic. Good solutions found in each subproblem are recorded after several decompositions and reconstructions. These solutions are concatenated to form a new solution. This stage is performed using a form of selection and recombination similar to GA.

Stefanitsis et al. (1995) and Chua et al. (1996) represented a VRP using the concept of *relaxed* TSP. Here, the solutions to VRP are represented as a TSP and the depots are encoded as multiple zero nodes having an infinite distance from each other. Stefanitsis et al., proposed an algorithm based on Constraint Logic Programming, which combines some aspects of Logic Programming from Artificial Intelligence and some constraint-satisfaction techniques, in order to produce good initial solutions before GA is applied.

In this paper we propose four new methods based on GA for VRP. The paper is organised as follows: The notations and necessary terminologies are introduced in the following section. Then, the algorithms are described in the subsequent section. This is followed by results and discussions. The results for all the benchmark problems in the literature are presented in the subsequent section and finally, conclusion is given in the last section.

PROBLEM DEFINITION

Let $G=(V,A)$ be a graph where $V=\{0,1,2,\ldots,N\}$ is a set of vertices representing customers with the depot located at vertex 0, and A is the set of all arcs. We associate a non-negative distance (or sometimes referred to as cost) matrix $C=(c_{ij})$ with every arc (i,j), $i \neq j$. c_{ij} represents the distance travelled (or the cost of travelling, or the travel time), from city i to city j. Note that when C is symmetric, i.e. the distance from city i to city j is the same as the distance from city j to city i, it is often convenient to replace a set of arcs A by a set of undirected arcs or edges E. Note that the travel time from city i to city j is commonly assumed to be proportional to the distance between the two cities.

It is also assumed that there are m vehicles available which are based at the depot and m is bounded respectively by the maximum and minimum number of vehicles available, m_U and m_L, i.e. $m_L \leq m \leq m_U$. When $m_L = m_U$, then m is said to be *fixed*; and to be *free* when $m_L = 1$ and $m_U = N-1$ where N is the size of the problem. In addition, it is often desirable to associate a fixed cost f with the use of a vehicle when m is not fixed. However, in most of the problems tested, the cost is ignored unless otherwise stated. It is also assumed that all the vehicles have the same capacity (*homogeneous vehicles*), although in some variants of the basic VRP it is possible for vehicles to have different capacities (*heterogeneous vehicles*).

Therefore, the VRP consists of designing a set of least cost routes in such a way that the following conditions are satisfied:

(i) Each city in V\{0} is visited exactly once by exactly one vehicle
(ii) All vehicle routes start and end at the depot
(iii) Some side constraints are satisfied
(iv) The number of side constraints which can be considered in VRP is large. Bodin et al. (1983) and Golden and Assad (1988) give excellent reviews of the many variations of VRP.

APLICATION OF GA FOR VRP

We will now describe the four algorithms that we have developed for the VRP. We note that the main difference between these procedures is the

way in which problem-specific knowledge is embedded in the structure of the algorithm.

The first approach implements the improvement methods suggested by Ulder et al. (1991). The results reported by Ulder et al. (1991) using this approach on a set of Traveling Salesman Problem for up to 662 customers have been very encouraging. On the other hand, the second, third and fourth approaches employed the construction heuristic methods developed for VRP in the assignment of customers to clusters. These three methods differ in the criteria used in constructing the clusters and also the way in which the clusters are built, i.e. either sequentially or in parallel.

Sequential Methods

I. Vertex Sequencing

In this method, the VRP is represented as a Traveling Salesman Problem. Following the order of the customers in each chromosome, the customers are assigned to a vehicle until adding the customer to that particular vehicle violates either the capacity or the route length constraint of that vehicle. In such cases, a new vehicle is initiated. In this approach we assume that the number of vehicles to serve all customers is not bounded, thus ensuring that all solutions obtained are feasible. Subsequently, each chromosome in the population is locally optimised using the 2-*opt* method proposed by Lin (1965). We note that the chromosome is first partitioned using the *clusterbuilder* and the chromosomes are optimised locally with respect to the way they are partitioned. Each individual in the population thus represents a local optimum, and the idea is that combination of these local optima through the process of selection and recombination will hopefully lead to a better local optimum, and will eventually converge to a global optimum.

The clusterbuider partitions the customers based on a tour construction heuristic of the route-first, cluster-second approach (Beasely, 1983). Assuming that the sequence of customers in a chromosome is give by (v_1, v_2, ...,v_i, v_{i+1}, ..., v_n) and starting form v_0, where v_0 denotes a depot, the clusterbuilder assigns the customers to the vehicles (or clusters) as follows:

The first vehicle contains all customers starting from the first customer on the tour and up to, but excluding, the first customers v_i *whose inclusion in the route would cause a violation of either capacity or maximal length constraint. Then starting from* v_i, *the process is repeated until all customers have been allocated a vehicle.*

The objective value of each chromosome is evaluated as (Gendreau et al., 1994)

$$F_1(S) = \sum_r \sum_{(v_i, v_j) \in R_r} c_{ij}$$

$$F_2(S) = F_1(S) + \alpha \sum_r \left[\left(\sum_{v_i \in R_r} q_i \right) - Q \right]^+ + \beta \sum_r \left[\left(\sum_{(v_i, v_j) \in R_r} c_{ij} + \sum_{v_i \in R_r} \delta_i \right) - L \right]^+ \quad (1)$$

where $[x]^+ = \max(0, x)$; α and β are two positive parameters, R is the set of all the routes and Q and L are the maximum load of each vehicle and the maximum length constraints for each vehicle respectively. We note that the route length consists of the travel distance (in this case it is assumed to be proportional to the distance between the two cities) and the service time δ_i.

An upper bound on the number of vehicles, $\overline{m}$, for each problem instance is determined according to some rules. In our implementation the algorithm is run once without imposing any restriction on the number of vehicles and the maximum number of vehicles required for the solution to be feasible is selected. The values of the parameters have been chosen arbitrarily.

Although the initial populations, in most of the GA-based algorithms, consist of randomly generated individuals, in difficult problems such as VRP it would be advantageous to start with some structured solutions and this has been shown to produce better solutions in the context of the TSP (Ulder et al., 1991). To accomplish this, simple heuristics based on some of the construction heuristics for VRP can be used to generate reasonably good initial structures for the first population. It is observed that an algorithm starting from a randomly generated solution, on average, requires a larger population size in order to converge to the desired solution. In our implementation a restricted 2-*opt* heuristic is used to initialise the starting population. A restricted 2-*opt* ensures that the computational time taken in initialising the population is not exhaustive. Also, this creates an initial population with some structures whilst maintaining some degree of diversity within the

population. Consequently, this helps prevent undesirable premature convergence to some sub-optimal values.

II. Savings GA

In this procedure we investigate the possibility of designing a clusterbuilder that assigns customers to a vehicle in a deterministic way instead of following the order of their appearance in the chromosome. This is achieved by using the existing constructive heuristics such as the savings method of Clark and Wright (1964).

In the savings method, initially all the customers are assumed to be served by one vehicle and the saving obtained if two customers are merged on a route are calculated according to the equation (Paessen, 1988)

$$S_{ij} = c_{0j} + c_{0j} - \theta_1 c_{ij} + \theta_2 \left(c_{0i} + c_{0j}\right) \qquad (2)$$

for $i, j = 1, 2, \ldots, N$ and $\theta_1 \in [1, 3]$ and $\theta_2 \in [0, 1]$. This ensures that the radial distance, the distance between the customers and the depot, is also taken into account. Preliminary experiments have indicated that the quality of the final solutions is sensitive to the choice of θ_1 and θ_2. Generally, taking θ_1 around 1.5 and θ_2 around 0.5 produces satisfactory results.

In the savings heuristic, the savings obtained for each pair of customers are arranged in descending order and the customers are merged into a cluster starting from the highest to the lowest savings as described in the savings method proposed by Clark and Wright (1964). This procedure cannot be implemented directly in GA since it not only requires a lot of computational time, but also produces a single solution regardless of the sequence in the chromosome. Therefore the clusterbuilder has to be modified accordingly to suit GA. We will now describe the new clusterbuilder implemented in savings GA.

Starting with the first customer on the sequence as the current customer, the savings obtained when it is merged with other customers on a single route are calculated. The customer with the most savings is selected. If the selected customer has already been routed, then the customer with the next best savings is chosen. Then, taking this new customer as the current customer, the procedure is repeated until either the capacity or the route length of the vehicle is violated. In such circumstances, the firs customer on the sequence that has not been routed is assigned to the next vehicle. The algorithm iterates until all the customers have been allocated to a vehicle or the number of vehicles used is $\overline{m} - 1$ *in which case the remaining customers that are not yet routed are assigned to vehicle* $\overline{m}$ *.*

After all the customers have been routed, the objective value of each chromosome is evaluated as in equation (1).

Parallel Methods

Most sequential-based heuristics are concerned with building good clusters at the beginning of the process. As a result, they tend to leave to the end customers that may be far apart from each other geographically. Inevitably, this produces a last route which is of poor quality. This has been recognised as the intrinsic weakness of sequential-based methods (Potvin and Rousseau (1993), Altinkemer and Gavish (1991)). Algorithms that construct several routes simultaneously, often referred to as parallel route building methods, have been proposed to alleviate this problem. In this paper we examine two parallel route building methods or parallel methods based on GA.

1. Set Partitioning Approach

One of the methods we have studied is based on algorithms proposed by Jones and Beltramo (1991) designed for partitioning problems. This idea was first investigated by Blanton and Wainwright (1993) for VRP with time windows. In this procedure, as with any other parallel clusterbuilding methods, the number of vehicles required must be predetermined. In our implementation this is done by taking the number of vehicles found in the sequential approach as a starting point. The algorithm is then run several times to determine the appropriate number of vehicles. In cases where no feasible solutions are obtained in any of the runs, the number of vehicles is then increased by one and a similar process is carried out until the probability of getting feasible solutions in each run is relatively high. In GA selecting the correct number of vehicles is vital since, unlike other parallel-based heuristics, the number may not be reduced throughout the runs. Our implementation adapts the approach proposed by Blanton and Wainwright (1993).

The algorithm uses a *greedy heuristic*, which constructs the routes by taking the first m customers (where m is the number of vehicles available) in the permutation to initialise each vehicle. The remaining customers are added to the vehicles in the order they appear in the chromosome, always adding to the vehicle that yields the best objective value. This strategy resembles the *nearest neighbour* rule whereby each customer assigned to a vehicle is chosen to be closest to the customer last assigned to that vehicle. Since our sole objective is to minimise the total route length, this is achieved by taking $v^* = \arg\{\min_{k=1,\square,m} c_{il_k}\},\ i = m+1,\square,N$ where i is the current customer to be inserted and l_k is the last customer that appears on route k. Customer i is added to vehicle v^* only if inserting this customer does not violate either the capacity or the route length constraints of vehicle v^*, else the customer is considered to be unserviced. An objective value for each chromosome is thus defined as follows:

$$F(S) = \begin{cases} u & \text{if } u > 0 \\ \sum_{r=1}^{m} \sum_{(v_i,v_j)\in R_r} c_{ij} - M & \text{otherwise} \end{cases} \qquad (3)$$

where u is the number of customers that are unserviced and M is a very large positive constant. Note that if the solution is feasible, a large negative constant, - M, is added to the total route length to ensure that the solutions are ordered in the correct manner. The value of M is chosen such that it always exceeds the total route length. In our implementation, as in Blanton and Wainwright (1993), a constant of 100000 is chosen since in all cases the total route length obtained never exceeded this value.

II. Parallel Savings GA

The final method we developed is similar to the set partitioning procedure except that a different measure is used to allocate the subsequent customers to the vehicles. This idea is similar to the parallel insertion based on savings method described in Altinkemer and Gavish (1991). Here, starting from the first customer in the chromosome, for each customer i that is not yet routed, the saving obtained when i is connected to the last customer on each vehicle is calculated, and the customer is then assigned to the vehicle giving the largest saving. A vehicle v^* is therefore selected according to the criterion $v^* = \arg\{\max_{k=1,\square,m} s_{il_k}\},\ i = m+1,\square,N$ where s_{il_k} is the saving obtained when customer i is connected to the last customer on route k. The savings are calculated as in equation (2).

We note that if adding customer i to a particular vehicle results in an infeasible solution due to violation of one of the constraints, then the next vehicle, in descending order of savings, is selected. If adding customer i to each of the vehicles in turns violates one or all of the constraints, then this customer is considered unserviced and a penalty is incurred on the total amount of unserviced demand. Hence, the new objective function can be written as

$$F(S) = \sum_{r} \sum_{(v_i,v_j)\in R_r} c_{ij} + \alpha \sum_{l=1}^{u} q_l \qquad (4)$$

where u is the total number of unserviced customers.

RESULTS AND DISCUSSIONS

We evaluated the performance of these algorithms on three sets of problems that exhibit different characteristics. The 50-customer and the 150-customers problems are randomly generated whilst the locations of the customers in the 100-customers problem are arranged in clusters.

Each algorithm was simulated for five times with different random number generators and the variables such as population size (N), crossover rate, mutation rate, selective pressure and insertion rate were fixed at 50, 0.8, 10/N, 1.5 and 0.8, respectively, for both algorithms. We used stochastic universal sampling method to assign for each individual the expected number of offspring to be produced in the next generation. The fitness of each individual was assigned using the non-linear ranking method (Chipperfield et al., 1993) and the reproduction strategy ensures that the least fit individuals are replaced by the offspring.

For the vertex sequencing method we have adapted the modified enhanced edge recombination operator whereby the edge that connects the first and the last customer and those connecting customers on two separate vehicles are omitted in the construction of the edge list. The order-based crossover was employed in the savings GA. Uniform order-based crossover is

applied in both the set partitioning and parallel savings GA. Scramble sub-list mutation was applied in all the algorithms and all algorithms were terminated after 150 iterations.

Tables 1 -- 3 tabulate the best, the worst and the average solutions found over the five runs, the standard deviation and the average CPU times for each algorithm. Figures 1 – 3 display the average performance of these algorithms for each of the problem tested.

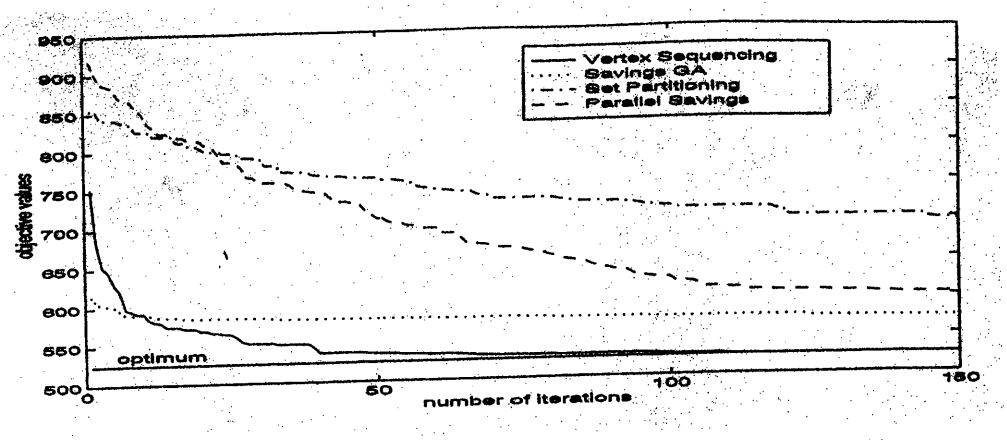

Figure 1. Average performance of various methods for the 50-customers problem

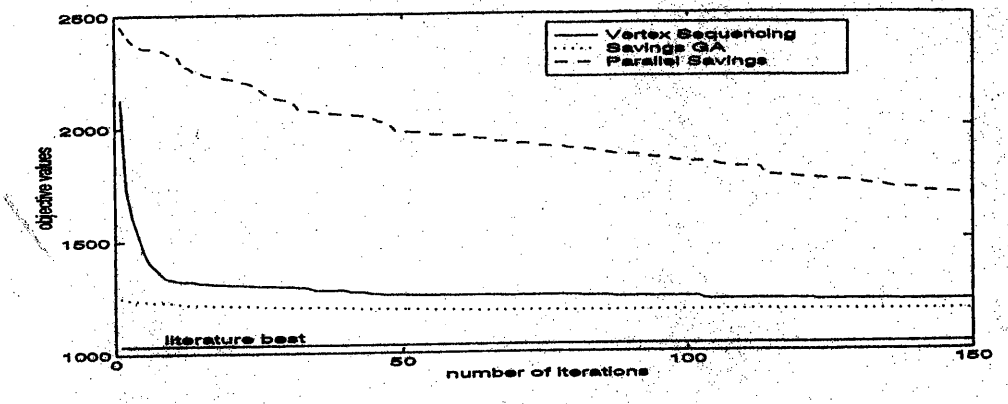

Figure 2. Average performance of various methods for the 150-customers problem

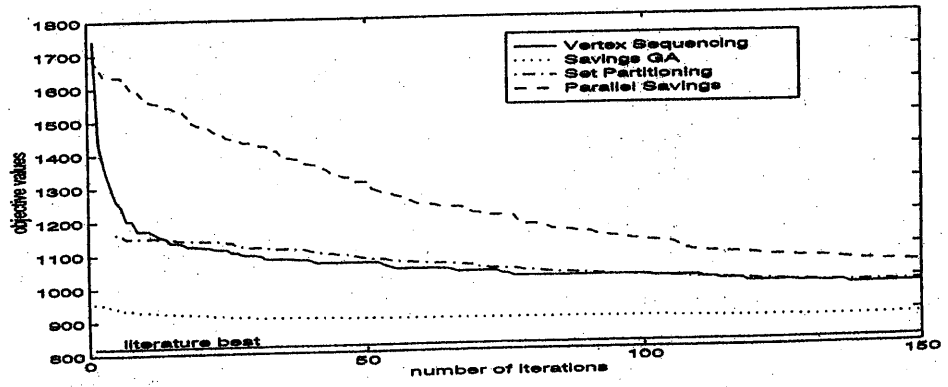

Figure 3. Average performance of various methods for the 100-customers problem

Table 1. Results for 50-Customer Problem

Methods	Best	Worst	Average	Standard Deviation	Average CPU (secs.)
Vertex Seq.	524.61	524.93	524.67	0.14	1842.34
Savings GA	565.31	585.71	571.82	8.70	1023.34
Set Partitioning	665.00	737.00	695.80	26.94	802.52
Parallel Savings	583.55	605.35	602.24	16.71	1472.58
Best Published Results: **524.61**					

Table 2. Results for 150-Customer Problem

Methods	Best	Worst	Average	Standard Deviation	Average CPU (secs.)
Vertex Seq.	1201.70	1254.10	1222.86	21.03	8886.88
Savings GA	1163.90	1185.80	1177.27	8.23	4202.90
Set Partitioning	1792.40	*infeasible*	--	--	--
Parallel Savings	1546.50	1775.60	1611.32	97.03	15347.8
Best Published Result: **1034.9**					

Table 3. Results for 100-Customer Problem (Clustered)

Methods	Best	Worst	Average	Standard Deviation	Average CPU (secs.)
Vertex Seq.	956.74	10004.10	981.95	16.86	4224.26
Savings GA	880.86	895.76	886.34	6.29	2679.30
Set Partitioning	928.00	1016.00	986.00	35.74	2609.6
Parallel Savings	1003.20	1121.90	1043.96	54.45	4525.36
Best Published Result: **819.6**					

The graphs show that the vertex sequencing approach converged to better solutions for problems with fewer customers. In contrast, the savings GA method performs well for larger problems, with the set partitioning approach converging to a better solution than the parallel savings method for the clustered problems. Also, it is observed that savings GA starts with superior solutions due to the effectiveness of the savings method, but for small problems, it is quickly overtaken by the vertex sequencing approach. The improvement in the savings GA is very slow compared to the other methods. The poor performance of the vertex sequencing approach in larger problems may be attributed to the effectiveness of the underlying local optimiser. It is also noted that the set partitioning approach starts with superior solutions than either the vertex sequencing or parallel savings method for the 150-cluster problem. This may be attributed to the fact that the clusterbuilder places great emphasis on the distance between the customers.

In general, the tables show that the vertex sequencing method requires more computational time than the savings method because of the underlying 2-opt heuristic. The set partitioning approach requires the least computational time with the parallel savings GA takes the longest time. The 2-opt heuristic is known to be less effective and the very large computing time involved prevented us from experimenting with more effective local search methods. It is also observed that the set partitioning and the parallel savings methods give large standard deviations compared to the other two methods, which is undesirable.

One of the interesting characteristics displayed by vertex sequencing, savings GA and parallel savings GA is that the solutions iterate between feasibility and infeasibility, since these methods allow the algorithm to accept infeasible solutions if the reduction in the objective value is significant enough. However, in the case of set partitioning, the algorithm is more concerned with finding a feasible search space and once this is found, the algorithm constrains the search to be within this area only. This approach is

particularly effective for VRP with time windows (for which the original algorithm was developed) where a feasible solution is hard to obtain because of the added constraint imposed for each customer.

As mentioned earlier, one possible cause of the poor performance of the parallel methods is that the first K customers that were used to initialise K vehicles may consist of customers that are close to each other. Indeed this is undesirable since more vehicles will be used to serve these customers, which may in fact require only one vehicle to serve them.

We note that all the programmes were written in MATLAB using the genetic algorithm TOOL BOX (Chipperfield et al., 1993). All programmes were run on Silicon Graphic computer.

BENCHMARK PROBLEMS

Since vertex sequencing and savings GA perform relatively better than the other two algorithms, we evaluated the performance of these two algorithms on the 14 benchmark problems given in the literature. We note that Problems 1 to 10 are uniformly randomly generated problems and problems 6 to 10 are the same as Problems 1 to 5 with the additional of route length constraints. The locations of the cities in Problems 11 to 14 are arranged in clusters with problems 13 and 14 incorporating the route length constraints.

The algorithms were simulated for several runs and Table 5 tabulates the best published solutions from Gendreau et al. (1997), solutions from Kelly and Xu (1999), solutions obtained from our algorithms, the number of vehicles required and the percentage deviation from optimum solutions (or best published result) for all the benchmark problems. The value of the parameters for the results shown in Table 5 is given in Table 4.

A smaller population were selected for vertex sequencing method because of the time consuming 2-*opt* heuristic. We note that all the programmes were terminated after 250 iterations and other parameters such as the selective pressure (the bias towards the best individual), crossover rate and insertion rate were fixed at 1.5,0.8, and 0.8 respectively.

It should be noted that the penalty constant is fixed at 1.0. In situations where no feasible solutions can be found, the penalty constant is increased to 100 and if the algorithm still fails to find a feasible solution, then the parameter is increased to 1000. The maximum number of vehicles is always chosen to be one more than the number of vehicles required in the best published solution. The main idea here is to reduce the effect of the penalty constant as much as possible, since the choice of a suitable penalty value is not a trivial task in GA. With the addition of the route length constraint, the choice of appropriate penalty constants becomes a crucial factor. Limited experiments have shown that imposing a large penalty value often results in inferior solutions. The results again show that vertex sequencing performs relatively well for smaller problems, with the savings GA yielding better results for larger problems. The performance of vertex sequencing becomes worse with the additional route length constraints. In addition, it fails to find any feasible solution within reasonable computational time for Problem 10 even though the number of vehicles is increased. Note that the percentage deviation from the best published results for vertex sequencing varies from 0 to 25.3. On the other hand, savings GA produces a better percentage deviation on the whole, varying from 2.9 to 12.7. It is interesting to note that, on average, 80 % of the improvements in the best solution are obtained in less than 100 iterations and the improvement obtained after this is only marginal.

Table 4. Value of the parameters involved

Problem number	Population size		Mutation rate		Penalty constants	
	Vertex seq.	Savings GA	Vertex seq.	Savings GA	α	β
1	30	50	1/*N*	10/*N*	1.0	--
2	75	100	1/*N*	10/*N*	1.0	--
3	100	150	10/*N*	20/*N*	1.0	--
4	100	200	10/*N*	20/*N*	1.0	--
5	100	200	10/*N*	20/*N*	1.0	--
6	50	100	1/*N*	10/*N*	1.0	1.0
7	75	100	1/*N*	10/*N*	1.0	100
8	100	150	10/*N*	20/*N*	1.0	1000
9	100	200	10/*N*	20/*N*	1.0	1000
10	100	200	10/*N*	20/*N*	1.0	1000
11	100	200	10/*N*	20/*N*	1.0	--
12	75	150	10/*N*	20/*N*	1.0	--
13	100	200	10/*N*	20/*N*	1.0	100.0
14	100	150	10/*N*	20/*N*	1.0	100.0

Table 5. Results for 14 benchmark problems

Prob. Num.	Number of cities	Vertex Sequencing	Savings GA	XK[a]	Best Published Results[b]	% Deviation	
						Vertex Seq.	Vertex Seq.
1	50	524.61 (5)[c]	565.31 (6)	524.61 (5)	524.61 (5)	0	7.7
2	75	850.92 (10)	894.43 (10)	835.26 (10)	835.26 (10)	1.9	7.0
3	100	900.12 (8)	906.45 (8)	826.14 (8)	826.14 (8)	9.0	9.7
4	150	1200.1 (12)	1159.1 (12)	1028.42 (12)	1028.42 (12)	16.7	12.7
5	199	1552.6 (17)	1422.3 (17)	1310.97 (17)	1298.79 (16)	19.5	9.5
6	50	561.45 (6)	601.32 (6)	555.43 (6)	555.43 (6)	1.2	8.3
7	75	1023.1 (12)	957.17 (12)	909.68 (11)	909.68 (11)	12.5	5.2
8	100	951.02 (9)	939.83 (9)	865.94 (9)	865.94 (9)	9.8	8.5
9	150	1456.7 (16)	1300.5 (15)	1171.33 (14)	1162.55 (14)	25.3	11.9
10	199	--	1575.4 (19)	1425.97 (18)	1397.94 (18)	--	12.7
11	120	1184.3 (7)	1110.3 (7)	1042.11 (7)	1042.11 (7)	13.7	6.5
12	100	955.54 (10)	876.42 (10)	819.56 (10)	819.56 (10)	16.9	6.9
13	120	1801.8 (12)	1586.6 (11)	1581.98 (11)	1541.14 (11)	16.9	3.0
14	100	888.91 (11)	907.90 (11)	866.37 (11)	866.37 (11)	2.6	4.8

[a] Kelly and Xu (1999)
[b] Gendreau et al (1997)
[c] Number of vehicles

CONCLUSION

We have investigated several ways of modelling VRP using GA. Our results have shown that the sequential algorithms, i.e. vertex sequencing and savings GA, produce solutions that are superior to the solutions obtained using the parallel approach. The vertex sequencing method, which has performed well for problems up to 50 customers, has demonstrated that GA is able to produce very competitive results using a relatively small population if the local search heuristic employed in the improvement stage is effective for the problems in hand. The savings GA method, which exhibit smaller fitness variance (Moin, 2002), has produced relatively good solutions in larger problems. Although in small problems, such as the 50-customers problem, the vertex sequencing method converges to better solutions, the power of savings GA manifests itself in larger problems where the local optimiser used in the vertex sequencing method ceases to be effective.

REFERENCES

1. Altinkemer, K. and Gavish, B. (1991). *Parallel Savings Based Heuristics for the Delivery Problem*, Operations Research **39**: 456-469.
2. Badeau, P., Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, É.D. (1996). *A New Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows*, Symposium on Combinatorial Optimization, London, pp 128.
3. Beasley, J.E. (1983). *Route First-Cluster Second methods for Vehicle Rauting*, OMEGA, **11**: 403-408.
4. Blanton, J.L., Jr., and Wainwright, R.L. (1993). *Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms*, in S. Forrest, (ed.), Proceedings of the Fifth International Conference on Genetic Algorithm, Morgan Kaufmann Publisher, San Mateo, California, pp 452-459.
5. Chipperfiled, A., Fleming, P., Pohlheim, H. and Fonseca, C., (1993). Genetic Algorithm TOOLBOX for Use with MATLAB, Department of Automatic Control and System Engineering, University of Sheffield, U.K.
6. Clarke, G. and Wright, J.W. (1964). *Scheduling of Vehicles from* a *Central Depot to a Number of Delivery Points*, Operations Research **12**: 568-581.
7. Chua, C., Daniel, R., Gallently, J., Ikonomou, A. and Tan, M. (1996). *The Vehicle Routing Problem: A Comparison of a Genetic Algorithm and a Tabu Search Implementation*, Symposium on Combinatorial Optimization, London, pp 127.
8. Dantzig, G.B. and Ramser, G.H. (1959). *The Truck Dispatching Problem*, Management Science **6**: 80-91.
9. Gendreau, M., Hertz, A., and Laporte, G., (1994). A Tabu Search Heuristic for the Vehicle Routing Problem, Management Science **40**: 1276-1284.
10. Gendreau, M., Laporte, G., Potvin, J.-Y. (1997). Local Search Algorithms for the Vehicle Routing Problem, Workshop on Heuristic Methods: Developments and Applications, Lancaster, U.K.
11. Gendreau, M., Laporte, G., Potvin, J.-Y. (1997). Local Search Algorithms for the Vehicle Routing Problem, Workshop on Heuristic Methods: Developments and Applications, Lancaster, U.K.
12. Grefenstette, J.J. and Baker, J.D. (1989). *How GAs work: A Critical Look at Implicit Parallelism*, in J. Schaffer, (ed.), Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, Los Altos, California, pp 20-27.
13. Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan, Ann Arbor.
14. Kelly, J.P., and Xu, J. (1999). *A Set-Partitioning Based Heuristic for the Vehicle Routing Problem*, INFORMS Journal on Computing **11**: 161-172.
15. Laporte, G., and Osman, I.H. (1995). *Routing Problems: A Bibliography*, Annals of Operations Research **61**: 227-262.
16. Lin, S. (1965). *Computer Solutions of the Traveling Salesman Problem*, Bell System Technical Journal **44**: 2245-2269.
17. Mitchell, M. (1996). An Introduction to Genetic Algorithms, MIT Press, Cambridge, Mass., USA.
18. Moin, N.H. (2002). Performance Analysis of Genetic Based Algorithms for Vehicle Routing problems Using the Fitness

Variance of Formae, Tamsui Oxford Journal of Management Sciences **18**.

19. Potvin, J.-Y. and Dube, D. (1994). {\em Improving a Vehicle Routing Heuristic Through Genetic Search}, Proceedings of the IEEE, pp 194-198.
20. Potvin, J.-Y. and Bengio, S. (1993) {\em A Genetic Approach to the Vehicle Routing Problem with Timw Windows}, Working Paper, CRT-953, Centre de Recherche sur les Transports, Universite de Montreal, Canada.
21. Stefanitsis, E., Christodoulou, N. and Psarras, J. (1995). *Combination of Genetic Algorithms and CLP in the Vehicle-Fleet Scheduling Problem*, in D.W. Pearson, N.C. Steele and R.F. Albrecht, Thangiah, S.R., 1995, Vehicle Routing with Time Windows using Genetic Algorithms, in L. Chambers, (ed), Practical Handbook of Genetic Algorithms: New Frontiers, **II**: 253-277.
22. Thangiah, S.R., Nygard, K.E., and Juell, P.L., (1991). GIDEON: *A Genetic Algorithm System for Vehicle Routing with Time Windows*, Proceedings Seventh IEEE Conference on Artificial Intelligence Applications, IEEE Computer Society press, Los Alamitos, California, pp 332-325.
23. Thangiah, S.R., Vinayagamoorthy, R., and Gubbi, A.V. (1993). *Vehicle Routing with Time Deadlines using Genetic and Local Algorithms*, in S. Forrest, (ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, California, pp 506-513.
24. Thangiah, S.R. (1995). *Vehicle Routing with Time Windows using Genetic Algorithms*, in L. Chambers, (ed), Practical Handbook of Genetic Algorithms: New Frontiers **11**: 253-277.
25. Ulder, N.L.J., Aarts, E.H.L., Bandelt, H.,-J, van Laarhoven, P.J.M. and Pesh, E. (1991). *Genetic Local Search Algorithms for the Traveling Salesman Problem*, in H.P Schwefel and R. Männer, (eds.), proceedings of the First International Conference on Parallel Problem Solving from Nature (PPSN), Lecture Notes in Computer Science, **496**: 109-116.