PERGAMON

Computers & Operations Research 30 (2003) 787–800

computers & operations research

www.elsevier.com/locate/dsw

A genetic algorithm for the vehicle routing problem

Barrie M. Baker*, M.A. Ayechew

School of Mathematical and Information Sciences, Coventry University, Priory Street, Coventry CV1 5FB, UK

Abstract

This study considers the application of a genetic algorithm (GA) to the basic vehicle routing problem (VRP), in which customers of known demand are supplied from a single depot. Vehicles are subject to a weight limit and, in some cases, to a limit on the distance travelled. Only one vehicle is allowed to supply each customer.

The best known results for benchmark VRPs have been obtained using tabu search or simulated annealing. GAs have seen widespread application to various combinatorial optimisation problems, including certain types of vehicle routing problem, especially where time windows are included. However, they do not appear to have made a great impact so far on the VRP as described here. In this paper, computational results are given for the pure GA which is put forward. Further results are given using a hybrid of this GA with neighbourhood search methods, showing that this approach is competitive with tabu search and simulated annealing in terms of solution time and quality.

Scope and purpose

The basic vehicle routing problem (VRP) consists of a number of customers, each requiring a specified weight of goods to be delivered. Vehicles despatched from a single depot must deliver the goods required, then return to the depot. Each vehicle can carry a limited weight and may also be restricted in the total distance it can travel. Only one vehicle is allowed to visit each customer. The problem is to find a set of delivery routes satisfying these requirements and giving minimal total cost. In practice, this is often taken to be equivalent to minimising the total distance travelled, or to minimising the number of vehicles used and then minimising total distance for this number of vehicles.

Most published research for the VRP has focused on the development of heuristics. Although the development of modern heuristics has led to considerable progress, the quest for improved performance continues. Genetic algorithms (GAs) have been used to tackle many combinatorial problems, including certain types of vehicle routing problem. However, it appears that GAs have not yet made a great impact on the VRP as described here. This paper describes a GA that we have developed for the VRP, showing that this approach can be competitive with other modern heuristic techniques in terms of solution time and quality.

Keywords: Routing; Heuristics; Genetic algorithms

* Corresponding author.
E-mail address: b.baker@coventry.ac.uk (B.M. Baker).

0305-0548/02/$ - see front matter
PII: S0305-0548(02)00051-5

1. Introduction

The basic vehicle routing problem (VRP) consists of a large number of customers, each with a known demand level, which must be supplied from a single depot. Delivery routes for vehicles are required, starting and finishing at the depot, so that all customer demands are satisfied and each customer is visited by just one vehicle. Vehicle capacities are given and, frequently, there is a maximum distance that each vehicle can travel. In the latter case, a drop allowance may be associated with each customer, which is added to the total distance travelled by the vehicle to which the customer is assigned. Thus, a vehicle which visits many customers will not be able to travel as far as a vehicle that visits relatively few customers. Possible objectives may be to find a set of routes which minimises the total distance travelled, or which minimises the number of vehicles required and the total distance travelled with this number of vehicles. Various mathematical formulations of the VRP are given by, for example, Laporte [1].

Problems of realistic size are tackled using heuristics. There have been many contributions to the subject, including various extensions to the basic problem described above. Laporte [1] gives a survey, and an extensive bibliography has been compiled by Laporte and Osman [2].

The tabu search implementations of Taillard [3] and Rochat and Taillard [4] have obtained the best known results to benchmark VRPs. Various authors have reported similar results, obtained using tabu search [5–8], or simulated annealing [5,9]. However, it has been observed by Renaud et al. [10] that such heuristics require substantial computing times and several parameter settings.

Ant colony optimisation is another recent approach to difficult combinatorial problems with a number of successful applications reported, including the VRP [11,12]. With a 2-optimal heuristic incorporated to improve individual routes produced by artificial ants, this approach also has given results which are only slightly inferior to those from tabu search.

Genetic algorithms (GAs) have seen widespread use amongst modern metaheuristics, and several applications to VRPs incorporating time windows have been reported [13–20]. Applications of GAs have also been reported for the VRP with backhauls [21], for a multi-depot routing problem [22], and a school bus routing problem [23]. A hybrid approach to vehicle routing using neural networks and GAs has also been reported [24]. However, GAs do not appear to have made a great impact so far on the basic VRP. The aim of this study is to put forward a conceptually straightforward GA for the basic VRP, which is competitive with other modern heuristics in terms of computing time and solution quality. A hybrid heuristic which incorporates neighbourhood search into our GA is also considered. Computational results are given for benchmark problems, alongside some of the well-known results obtained using tabu search and simulated annealing.

2. Basis for a genetic algorithm

The principles of GAs are well known. A population of solutions is maintained and a reproductive process allows parent solutions to be selected from the population. Offspring solutions are produced which exhibit some of the characteristics of each parent. The fitness of each solution can be related to the objective function value, in this case the total distance travelled, and the level of any constraint violation. Analogous to biological processes, offspring with relatively good

fitness levels are more likely to survive and reproduce, with the expectation that fitness levels throughout the population will improve as it evolves. More details are given by Reeves [25], for example.

The starting point for any GA is in the representation of each solution or population member. Typically, this will be in the form of a string or chromosome. Individual positions within each chromosome are referred to as genes. Although binary strings have been favoured by many GA researchers, some successful implementations use non-binary representations. For example, Chu and Beasley [26] use a non-binary representation in their approach to the generalised assignment problem (GAP), which requires jobs to be assigned to agents at minimum total cost, subject to resource limitations for each agent. In their representation, each chromosome consists of a string of decimal numbers, in index order of the jobs, specifying the agent number to which each job is assigned.

Structural similarities between the VRP and the GAP have been noted and successfully exploited in the past; for example, by Fisher and Jaikumar [27], and by Baker and Sheasby [28]. Our VRP representation is similar to Chu and Beasley's GAP representation and specifies, for each customer, the vehicle number to which it is assigned. Thus, given n customers and m vehicles, the chromosome for an individual solution has the form of a string of length n, with each gene value in the range $[1, m]$. Unlike some representations which have been used for the VRP, this does not specify explicitly the exact route which each vehicle should follow. However, with the assignment of customers to vehicles known, individual routes are implicitly specified, including the total distance travelled and the levels of any constraint violations. The solution of a travelling salesman problem (TSP) is required for each vehicle in order to make the transition from an implicit to an explicit solution, enabling a fitness value to be associated with each population member.

Two steps are taken to maintain as much structure as possible. Firstly, the customers are sorted and numbered so that consecutive customers are likely to be served by the same vehicle. For problems where the customers are randomly distributed around the depot, they are sorted according to increasing order of polar angle. When customers are located in clusters rather than randomly, they are sorted according to a nearest neighbour solution to the TSP, starting from the depot and visiting all customers. Secondly, we attempt to number the vehicles so that any vehicle, i, operates in approximately the same region for all population members. Then, we apply the usual reproductive processes for GAs to the population, generating new solutions that share certain route structures with their parents. These aspects are described in more detail in the following sections. Since a similar approach has proved successful with the GAP, we could reasonably expect such a GA to produce good solutions to the VRP.

3. Generating the initial population

Experiments were carried out with the initial population generated randomly, with an initial population of structured solutions, and with a mixed population containing both random and structured solutions. The expectation is that an initial population of reasonably structured solutions will evolve to high-quality solutions in a relatively small number of generations of the GA. However, a possible drawback is that such a population will lack the diversity needed to obtain near-optimal solutions, comparable in quality to those obtained using tabu search.

	Capacity constraints only	Capacity and distance constraints
Not tight:	(tightness < 0.95) $R_c \geq 0.9$	(tightness < 0.8) $R_d \geq 0.9$
Tight:	(tightness ≥ 0.95) $R_c \geq 0.75$	(tightness ≥ 0.8) $R_d \geq 0.75$

Fig. 1. Conditions for including last customer in sweep.

Two methods were used to generate a population of structured solutions. The first method is based on the sweep approach of Gillett and Miller [29]. For problems where the customers are randomly distributed around the depot, they are sorted according to increasing order of polar angle, as already described. Then, to generate each population member, a customer is chosen at random to begin the sweep process for one of the vehicles. Customers are allocated to the current vehicle in the order in which they have been sorted, until a constraint violation occurs. The last customer may or may not be removed from the vehicle before continuing the sweep with the next vehicle, according to the following rules. For problems with only capacity constraints, the remaining capacity of the vehicle before adding the last customer is expressed as a proportion of the demand of this customer, and is represented by R_c. For problems with both capacity and distance constraints, the additional distance which the vehicle could have travelled before adding the last customer is expressed as a proportion of the additional distance required to include the last customer (including any drop allowance) and is represented by R_d. Thus, a value of R_c or R_d close to 1 indicates a minor violation of the constraint, and a smaller value of R_c or R_d indicates a more significant violation. The decision rule used to determine whether the last customer is removed from the route takes into account the problem tightness, defined here as the total demand of the customers divided by the total capacity of the vehicles. The decision rule is then as shown in Fig. 1. For problems which are not tight, the last assignment is maintained only if the constraint violation is minor; for tight problems, a more significant constraint violation is allowed.

When customers are located in clusters rather than randomly, and sorted according to a nearest neighbour solution to the TSP, the sweep procedure as described above is applied without any further modification. This has proved to be an effective method of obtaining an initial population of reasonable quality solutions to problems where the customers appear in clusters.

The second method of generating structured solutions is based loosely on the generalised assignment approach of Fisher and Jaikumar [27]. These authors put forward a method of automatically generating a seed location for each vehicle. Using an approximation for distances travelled corresponding to all customer-seed allocations, a generalised assignment problem is solved to achieve an allocation of customers to vehicles which satisfies load constraints. For the GA, however, we are generating an initial population of solutions where the route structure can be fairly crude, and in which a fair degree of constraint violation can occur. Consequently, we have simplified slightly the method of generating the seed points and, instead of solving a generalised assignment problem, we use a random allocation for each customer to one of its two best seeds.

The set of seed points is generated as follows. Customer cones are defined by drawing rays from the depot that bisect the angles between adjacent customers according to the order in which they are sorted, as described by Fisher and Jaikumar [27]. Then vehicle cones are formed by aggregating

customer cones and fractions of customer cones to obtain an equal demand distribution between vehicle cones. Each seed is located along a ray which bisects the angle of the corresponding vehicle cone. The distance of each seed from the depot is equal to the maximum of the customer distances for that vehicle cone. Assuming a symmetric distance matrix, d_{ij}, with 0 representing the depot, the cost of allocating customer j to seed i is approximated by

$$c_{ij} = d_{0j} + d_{ij} - d_{0i}.$$

Then, if a_j and b_j are the smallest and second smallest seed allocation costs for customer j, each customer is allocated to either its best seed or to its second best seed, with probabilities given by $b_j/(a_j + b_j)$ and $a_j/(a_j + b_j)$, respectively.

Individual vehicle routes are then found using the 2-optimal method, followed by a 3-optimal refinement. No attempt is made to satisfy load or distance constraints during the allocation process. Additional population members can be generated by repeating the random allocation process.

In both cases, individuals are not allowed to enter the population if they duplicate another member. A population of size 30 was used for the smallest test problems (50 customers) and populations of size 50 were used for larger sizes of problem. The sweep and generalised assignment approaches were each used to generate half the initial population. In preliminary experiments, the use of randomly generated individuals in the population only slowed the convergence of the GA with no improvement in the quality of the best solution obtained, and so randomly generated individuals were not included in the final versions of the GA.

4. The reproductive process

Two parent solutions are selected from the population by the binary tournament method. Thus, two individuals are chosen from the population at random. The one with the better fitness value is chosen as the first parent. The process is repeated to obtain a second parent.

Offspring are produced from the two parent solutions using a standard crossover procedure. Best results were obtained using the 2-point crossover, in which two points in the chromosome are chosen randomly. One offspring consists of the gene values from parent one which are to the left of the first point and to the right of the second point, along with the gene values from parent two which are between the two chosen points in the chromosome. A second offspring is produced by swapping round the parents and then using the same procedure. Offspring which duplicate existing members of the population are discarded.

As already stated, the aim is to number the vehicles so that, for all population members, vehicles with the same number operate in roughly the same region. Then, in a typical instance of the VRP, if the depot is roughly centrally located with customers uniformly distributed, the 2-point crossover is equivalent to dividing the plane into two sectors with two randomly drawn rays from the depot, using the vehicle allocations from one parent within one sector and the vehicle allocations from the other parent in the second sector. To achieve the vehicle numbering, the average of the x-coordinates and the average of the y-coordinates are calculated for the customers visited by each vehicle, giving points $(\bar{x}_i, \bar{y}_i)$, $i = 1, \ldots, m$. Then the polar angles of the points $(\bar{x}_i, \bar{y}_i)$ are used to order the vehicles, with vehicle 1 corresponding to the angle closest to zero (which may be an angle just less than 360°) and, thereafter, the remaining $m - 1$ vehicles being numbered in order of

increasing angle. Before mating the two chosen parents, if the angle for vehicle 1 in either parent is closer to the angle of vehicle 2 in the other parent than it is to the angle of vehicle 1, then the vehicles of one parent will be re-numbered to make the angles for the first vehicles match more closely.

Fig. 2 illustrates the application of 2-point crossover to a problem with 20 customers, in which a solution with four vehicles is sought. Customers and vehicles have been numbered according to increasing polar angle, as described in the previous paragraphs. Figs. 2(a) and (b) show two well-structured parents, whose representations are as follows:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P1:	1	1	1	1	2	2	2	3	3	2	3	3	3	3	4	4	4	4	4	1
P2:	1	1	2	2	1	2	2	2	2	3	3	3	3	3	4	4	4	4	1	4

Crossover points have been generated between customers 6 and 7, and between customers 15 and 16. Rays, shown as dashed lines, have been inserted in the corresponding locations in Fig. 2, dividing the region into two sectors. Application of 2-point crossover yields the following offspring solutions, each having the vehicle allocations from parent P1 within one sector, and from parent P2 within the other sector.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
O1:	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4	1
O2:	1	1	2	2	1	2	2	3	3	2	3	3	3	3	4	4	4	4	1	4

Fig. 2(c) shows the offspring O1, which has the best route structure amongst the four solutions.

The possibility of one vehicle route lying mostly or entirely within another is allowed for. If the distance from the depot of $(\bar{x}_{i+1}, \bar{y}_{i+1})$ is less than half the distance from the depot of $(\bar{x}_i, \bar{y}_i)$ for any i, and the difference between the vehicle angles is less than $180^\circ/m$, then the two vehicle numbers are swapped, so that the inner route always comes before the outer route.

During the generation of, say, 100,000 offspring, there may be three or four occasions when the 2-point crossover produces a pair of offspring in which neither uses all the vehicles. When this occurs, an offspring is produced using uniform crossover, in which each gene value is chosen randomly from one of the two corresponding gene values of the parents.

The performance of our GA was found to be improved by applying a simple mutation to new offspring, in which two genes are selected at random and their values are exchanged. Thus, two randomly chosen customers are switched between vehicles, except in cases where the two customers happen to be on the same vehicle. Other types of mutation such as shifting one or more randomly chosen customers to a neighbouring vehicle were less effective.

5. Replacement scheme

The GA which we have developed uses the steady-state approach, in which eligible offspring enter the population as soon as they are produced, with inferior individuals being removed at the same time, so that the size of the population remains constant. The ranking replacement method is used, as described by Beasley and Chu [30]. For each individual, the total travel distance and any constraint violation are recorded as separate values. The travel distance is referred to as the fitness of

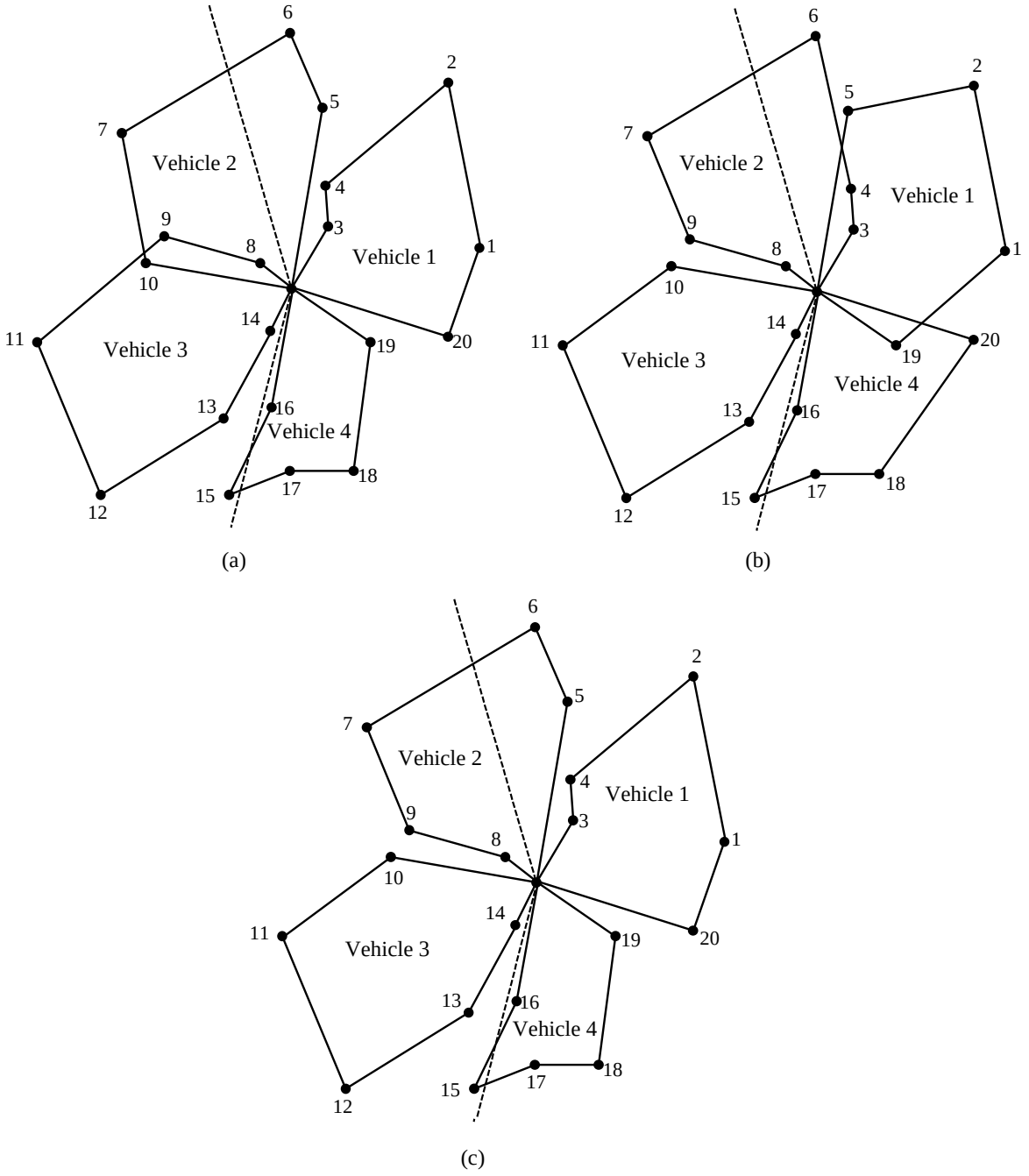

Fig. 2. (a) Parent 1. (b) Parent 2. (c) Offspring solution.

the individual. Any excess weight on a vehicle is expressed as a proportion of the vehicle capacity, and any excess distance is expressed as a proportion of the maximum vehicle distance. Then the total excess across all vehicles is referred to as the unfitness of this individual.

In the ranking replacement method, the population is partitioned into four subsets with respect to the child. Representing the population by P and denoting the fitness and unfitness values of a population member $p \in P$ by $f(p)$ and $u(p)$, respectively, and the child by c, the four subsets are defined as

$$S_1 = \{p \in P:\ f(p) \geqslant f(c),\ u(p) \geqslant u(c)\},$$

$$S_2 = \{p \in P:\ f(p) < f(c),\ u(p) \geqslant u(c)\},$$

$$S_3 = \{p \in P:\ f(p) \geqslant f(c),\ u(p) < u(c)\},$$

$$S_4 = \{p \in P:\ f(p) < f(c),\ u(p) < u(c)\}.$$

Then the population member to be replaced by the child is chosen from the first non-empty set in the order S_1, S_2, S_3. In our implementation, if all three of these sets are empty, then the child does not enter the population. In any subset, the member selected for replacement is that with the worst unfitness, with ties broken by worst fitness. In this way, the evolution of the population should achieve feasible solutions, but without being so strongly biased towards achieving feasibility that good quality solutions are never achieved, or vice versa. Our experiments confirmed that the GA performs better with ranking replacement than with either worst unfitness or worst fitness replacement.

Thus, the steps of the pure GA can be summarised as follows.

```
Generate an initial population of structured solutions
Evaluate fitness and unfitness of each individual in the population
Repeat
    Select two parents from the population using two binary tournaments
    Produce two offspring from the parents
    Mutate offspring
    Evaluate fitness and unfitness of offspring
    Choose favoured offspring
    If entry criteria are satisfied by chosen offspring
        Choose population member to be replaced
        Offspring enters population and the chosen member is removed
    End if
Until stopping criterion is satisfied
```

In choosing the favoured offspring, feasible solutions are favoured over infeasible solutions. If both child solutions are infeasible, the one with smaller unfitness value is favoured.

The stopping criterion can be defined in terms of elapsed computer time, number of generations, or number of generations with no improvement in the best solution found. Computational results for the pure GA are given later, alongside those for our hybrid algorithm and well-known published results.

6. Accelerating the convergence of the GA

Convergence of the GA described above has been accelerated by incorporating strategies for moving to neighbouring solutions. The first is a simple reassignment procedure to improve the

unfitness value of each population member as it is generated, either for the initial population or by the reproductive process. Whenever a solution is generated with non-zero unfitness value, for each vehicle with a constraint violation a customer is selected which, when re-assigned to its best position in the next numbered vehicle, causes the least increase in distance. If the total unfitness (without re-optimisation of either route) is reduced, then the re-assignment is maintained. When all vehicle routes with a constraint violation have been considered, the process is repeated, considering re-assignment to the previous numbered vehicle in each case. The process continues to alternate in this way until the unfitness value of the individual is zero, or no reduction in unfitness value occurs in a complete cycle. At this stage the routes are re-optimised using the 2-optimal procedure.

Secondly, two types of systematic neighbourhood search are performed on each individual of the initial population and then every 10,000 generations. The first is a 2-optimal search, in which all the vehicle routes are represented as a single TSP solution, with the depot replicated between each route. The fitness and unfitness values for each potential move are calculated and the best move is made at each iteration. The second type of systematic search is based on the λ-interchange used by Osman [5], with $\lambda = 1$. In this case, the moves that are considered consist of either removing one customer from a route and inserting it into its best position in an adjacent route, or swapping two customers between adjacent routes by removing one customer from each route and then inserting each customer into its best position in the other route. Again, a best-improve strategy is employed and, when no further improvement can be obtained with this type of move, the individual routes are re-optimised using the 2-optimal method.

7. Computational results

The heuristics described in the previous two sections were coded in C and, using a Pentium 266 MHz, they were applied to the 14 vehicle routing problems which can be downloaded from the OR-library (see Beasley [31]), and which have been widely used as benchmarks. The first 10 problems have customers that are randomly distributed with the depot in an approximately central location. In the last four problems, the customers are grouped into clusters. Initial testing focused on the pure GA applied to these 14 problems, to establish the effects of various strategies. Overall, these experiments reveal relatively small differences, and the GA remained effective with each of the variants that were tried. These experiments are described briefly as follows.

Variations on the method of population generation were tried first, in combination with the 2-point crossover, tournament selection, ranking replacement and mutation as previously described. The main findings from this experiment were that convergence from a population of random solutions was faster initially than for structured solutions which had been generated as described in Section 3. At 40,000 generations, the best population member from the random start had a total distance which was 14.5% above the best member from the structured start. Thereafter, there was little difference in convergence rates, and the best population member from a random start was still 8.1% above the best from a structured start after 100,000 generations. When an initial population of random solutions was combined with structured solutions, the gap at 100,000 iterations, averaged across the 14 problems, was reduced to 1.6%. Thus, no overall benefit from using random solutions in the initial population could be demonstrated.

Different crossover methods were compared using the structured initial population and the same selection, replacement and mutation methods as in the previous experiment. This experiment was run for 20,000 generations with each of the 14 problems using 1-point, 2-point and uniform crossovers and the total distance for the best population member was recorded for each problem. Averaged over the 14 problems, the best distances for 1-point and uniform crossovers were, respectively, 0.5% and 0.6% higher than those for 2-point crossover.

In the next experiment, the ranking replacement method was replaced by a worst fitness/unfitness replacement method, in which the population member with the worst unfitness was chosen for replacement, unless all population members had zero unfitness, in which case the member with worst fitness was chosen for replacement. The child solution replaced the chosen member even if it was worse, as this caused a much larger number of child solutions to enter the population and gave rise to a better performance of the GA compared with only allowing the child to enter if it was better than the population member chosen for replacement. The experiment was run for 20,000 generations for all 14 problems. Averaged over the 14 problems, the best distances for worst fitness/unfitness replacement were 1.3% higher than those for ranking replacement.

Two types of mutation were tried in conjunction with 2-point crossover, tournament selection, ranking replacement and the structured initial population. The first method, as previously described, simply exchanges the values for two randomly chosen genes in each new offspring. The second method shifts a randomly chosen customer to a neighbouring vehicle (i.e. neighbouring with respect to the numbering system for the vehicles described in Section 2). Each method was applied to each of the 14 benchmark problems for 20,000 generations. There was no significant difference between the effects of the two mutation methods on the performance of the GA. However, when the GA was run with no mutation, there was some evidence that the algorithm was less successful at finding solutions with zero unfitness value. For the final tests, mutation which exchanged randomly chosen gene values was chosen.

Table 1 shows well-known published results obtained using simulated annealing and tabu search. The distances obtained by Taillard [3] using tabu search remain the best known for 12 of the 14 problems. Rochat and Taillard [4] subsequently published solutions to the other two problems which improved upon those published by Taillard [3], as shown in Table 1. The results shown alongside for the pure GA were obtained from 100,000 generations with the smallest problems (with 50 customers) and 200,000 generations for the remainder. The best solution value in the initial population for the pure GA is included in Table 1, for each problem, to confirm that the GA itself has a significant effect. For four of the problems, it is seen that the initial population did not include a feasible solution.

The hybrid GA, incorporating the neighbourhood search techniques described in Section 6, was terminated whenever there was no improvement in 20,000 successive generations, thereby enabling a slight reduction in average CPU, despite the additional computations of the neighbourhood search. Table 2 shows the computation times that have been reported, alongside the corresponding values from Table 1 for the GA and the hybrid GA.

To give an indication of the variability of the GA's performance from one run to another, a further five runs were carried out for each problem, starting with different seed random numbers. Table 3 shows the best, worst and average values over the six runs for each problem.

Overall, the pure GA compares favourably with the simulated annealing algorithm of Osman [5], but, on average, the distances shown in Table 1 are 2.49% greater than the best known results

Table 1
Comparison of GA with published results

Cities	Capacity	Drop allowance /distance limit	Vehicles	Osman[a] (SA)	Osman[b] (TS)	Rochat and Taillard[c] (TS)	Gendreau et al.[d] (TABUROUTE)	Pure GA[e] (Initial)	Pure GA[f] (Final)	% Gap	Hybrid[g] GA	% Gap
50	160	0/∞	5	528	524/524	524.61	524.61	534.12	524.81	0.04	524.61	0.00
75	140	0/∞	10	838	844/844	835.26	835.32	885.41(0.27)	849.77	1.74	838.89	0.43
100	200	0/∞	8	829	838/835	826.14	826.14	869.22	840.72	1.76	829.47	0.40
150	200	0/∞	12	1058	1044/1052	1028.42	1031.07	1090.51	1055.85	2.67	1034.80	0.62
199	200	0/∞	17	1376	1334/1354	1298.79/1291.45	1311.35	1420.71	1378.73	6.76	1327.78	2.23
50	160	10/200	6	555	555/555	555.43	555.43	561.87	560.29	0.87	555.43	0.00
75	140	10/160	11	909	911/913	909.68	909.68	942.93(0.19)	914.13	0.49	909.68	0.00
100	200	10/230	9	866	878/866	865.94	865.94	917.73	872.82	0.79	867.68	0.20
150	200	10/200	14	1164	1184/1188	1162.55	1162.89	1229.48(0.02)	1193.05	2.62	1166.27	0.32
199	200	10/200	18	1418	1441/1422	1397.94/1395.85	1404.75	1502.18(0.13)	1483.06	6.25	1425.27	1.96
120	200	0/∞	7	1176	1043/1042	1042.11	1042.11	1158.31	1060.24	1.74	1046.92	0.46
100	200	0/∞	10	826	819/819	819.56	819.56	925.75	877.80	7.11	819.56	0.00
120	200	50/720	11	1545	1545/1547	1541.14	1545.93	1627.42	1562.25	1.37	1546.31	0.34
100	200	90/1040	11	890	866/866	866.37	866.37	923.35	872.34	0.69	867.13	0.09
									Average	2.49		0.50

[a]Simulated annealing by Osman [5].
[b]Tabu search + first-best-admissible/best-admissible by Osman [5].
[c]Tabu search by Taillard [3]/Tabu search by Rochat and Taillard [4].
[d]Tabu search by Gendreau et al. [6].
[e]Best solution in initial population for pure genetic algorithm (unfitness shown in brackets when greater than zero).
[f]Best solution obtained by pure genetic algorithm and % above best published.
[g]Best solution obtained by genetic algorithm hybridised with neighbourhood search and % above best published.

Table 2
Comparison of CPU times with published results

Cities	Vehicles	Osman[a] (SA)	Osman[b] (TS)	Gendreau et al.[c] (TABUROUTE)	Pure[d] GA	Hybrid[e] GA
50	5	167.4	114.0/67.2	360	213	23
75	10	6434.3	178.7/70.8	3228	765	617
100	8	9334.0	1543.0/675.0	1104	1148	717
150	12	5012.3	3560.0/3075.0	3528	2475	1961
199	17	2318.1	3246.0/1972.7	5454	3999	5261
50	6	3410.2	173.0/140.2	810	217	429
75	11	626.5	1056.7/203.0	3276	786	449
100	9	957.2	2998.0/1200.0	1536	1134	1904
150	14	84301.2	4755.8/2443.6	4260	2258	2242
199	18	5708.0	4561.0/3310.1	5988	3687	6433
120	7	315.8	1445.4/1398.4	1332	1633	1483
100	10	632.0	892.2/407.5	960	1160	1285
120	11	7622.5	2834.0/1343.0	3552	1694	1063
100	11	305.2	1175.9/5579.0	3942	1197	585
Average		9081.8	2038.1/1563.3	2809.3	1597.6	1746.6

[a]Simulated annealing by Osman, seconds using VAX 8600 (listed in [33] as 0.48 Mflop/s).
[b]Tabu search + first-best-admissible/best-admissible by Osman, seconds using VAX 8600 (listed in [32] as 0.48 Mflop/s).
[c]Tabu search by Gendreau et al., seconds using Silicon Graphics work station (5.7 Mflop/s).
[d]Pure genetic algorithm, seconds using Pentium 266 MHz (83.7 Mflop/s).
[e]Genetic algorithm hybridised with neighbourhood search, seconds using Pentium 266 MHz (83.7 Mflop/s).

Table 3
Performance of pure GA over six independent runs

Cities	Vehicles	Best	Worst	Average
50	5	524.81	524.81	524.81
75	10	849.77	860.36	857.61
100	8	839.73	846.03	842.55
150	12	1055.85	1059.92	1057.43
199	17	1374.73	1390.92	1380.99
50	6	559.04	560.29	560.08
75	11	909.94	924.31	916.18
100	9	872.82	882.97	879.61
150	14	1188.22	1217.11	1198.67
199	18	1451.63	1483.06	1463.19
120	7	1060.24	1087.47	1077.07
100	10	863.73	878.37	871.38
120	11	1560.79	1579.76	1568.49
100	11	872.34	891.15	887.70

obtained using tabu search. The best results shown in Table 3 give a slight improvement, at the expense of much greater computation times. The hybrid GA does better, with distances averaging only 0.5% above the best known values.

8. Conclusion

The straightforward GA that has been described here performs well, although it does not equal tabu search in terms of solution quality. Incorporating simple types of neighbourhood search into the GA produces a significant improvement, giving distances that are only 0.5% above best known results on average, with solution times that are not excessive.

Thus, it has been demonstrated that GAs are an effective approach to solving the basic VRP. Although the pure GA that we have developed works reasonably well, it appears to be better to view the GA more as a means of diversifying our exploration of the solution space, alongside neighbourhood search. In this sense, it is a strong competitor with other modern heuristics for the VRP.

References

[1] Laporte G. The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 1992;59:345–58.
[2] Laporte G, Osman IH. Routing problems: a bibliography. Annals of Operations Research 1995;61:227–62.
[3] Taillard É. Parallel iterative search methods for vehicle routing problems. Networks 1993;23:661–73.
[4] Rochat Y, Taillard É. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1995;1:147–67.
[5] Osman IH. Metastrategy simulated annealing and Tabu search algorithms for the vehicle routing problem. Operations Research 1993;41:421–51.
[6] Gendreau M, Hertz A, Laporte G. A Tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[7] Rego C, Roucairol C. A parallel Tabu search algorithm using ejection chains for the vehicle routing problem. In: Osman I, Kelly J, editors. Meta-heuristics: theory and applications. Boston: Kluwer, 1996.
[8] Barbarosoglu G, Ozgur D. A Tabu search algorithm for the vehicle routing problem. Computers & Operations Research 1999;26:255–70.
[9] Hiquebran DT, Alfa AS, Shapiro A, Gittoes DH. A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem. Engineering Optimization 1994;22:77–107.
[10] Renaud J, Boctor FF, Laporte G. An improved petal heuristic for the vehicle routing problem. Journal of the Operational Research Society 1996;47:329–36.
[11] Bullnheimer B, Hartl RF, Strauss C. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research 1999;89:319–28.
[12] Gambardella LM, Taillard É, Agazzi G. A multiple ant colony system for vehicle routing problems with time windows. In: Corne D. Dorigo M, Glover F, editors. New ideas in optimization. New York: McGraw-Hill, 1999. p. 63–76 [Chapter 5].
[13] Thangiah SR, Nygard PL, Juell PL. GIDEON: a genetic algorithm system for vehicle routing with time windows. In: Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications, Miami, FL, 1991. p. 322–8.
[14] Blanton JL, Wainwright RL. Multiple vehicle routing with time and capacity constraints using genetic algorithms. Proceedings of the Fifth International Conference on Genetic Algorithms 1993:452–459. Los Altas, CA.
[15] Thangiah SR, Gubbi AV. Effect of genetic sectoring on vehicle routing problems with time windows. In: Proceedings of the IEEE International Conference on Managing Intelligent System Projects, Washington, DC, 1993. p. 146–53.

[16] Thangiah SR, Vinayagamoorthy R, Gubbi AV. Vehicle routing with time deadlines using genetic and local algorithms. In: Proceedings of the Fifth International Conference on Genetic Algorithms. Los Altas, CA: Morgan Kaufmann, 1993. p. 506–15.
[17] Schmitt LJ. An empirical computational study of genetic algorithms to solve order-based problems: an emphasis on the TSP and the VRP with time constraints. Ph.D. dissertation, Fogelman College of Business and Economics, University of Memphis, 1994.
[18] Thangiah SR. Vehicle routing with time windows using genetic algorithms. In: Chambers L, editor. Application handbook of genetic algorithms: new frontiers, vol. II. Boca Raton, FL: CRC Press, 1995. p. 253–77.
[19] Thangiah SR. An adaptive clustering method using a geometric shape for vehicle routing problems with time windows. In: Proceedings of the Sixth International Conference on Genetic Algorithms. Los Altas, CA: Morgan Kaufmann, 1995. p. 536–43.
[20] Potvin JY, Bengio S. The vehicle routing problem with time windows—Part II: genetic search. INFORMS. Journal on Computing 1996;8:165–72.
[21] Potvin JY, Duhamel C, Guertin F. A genetic algorithm for vehicle routing with backhauling. Applied Intelligence 1996;6:345–55.
[22] Salhi S, Thangiah SR, Rahman F. A genetic clustering method for the multi-depot vehicle routing problem. In: Smith GD, Steel NC, Albrecht RF, editors. ICANNGA'97, Vienna. New York: Springer, 1998. p. 234–7.
[23] Thangiah SR, Nygard PL. School bus routing using genetic algorithms. In: Proceedings of the SPIE Conference on Applications of Artificial Intelligence X: Knowledge Based Systems, Orlando, FL, 1992. p. 387–97.
[24] Potvin JY, Dubé D, Robillard C. A hybrid approach to vehicle routing using neural networks and genetic algorithms. Applied Intelligence 1996;6:241–52.
[25] Reeves CR, editor. Modern heuristic techniques for combinatorial problems. Oxford: Blackwell Scientific Press, 1993.
[26] Chu PC, Beasley JE. A genetic algorithm for the generalised assignment problem. Computers & Operations Research 1997;24:17–23.
[27] Fisher ML, Jaikumar R. A generalized assignment heuristic for vehicle routing. Networks 1981;11:109–24.
[28] Baker BM, Sheasby JE. Extensions to the generalised assignment heuristic for vehicle routing. European Journal of Operational Research 1999;119:147–57.
[29] Gillett B E, Miller LR. A heuristic algorithm for the vehicle dispatch problem. Operations Research 1974;22:340–9.
[30] Beasley JE, Chu PC. Constraint handling in genetic algorithms: the set partitioning problem. Journal of Heuristics 1998;4:323–57.
[31] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990;41:1069–72.
[32] Ayechew MA. Genetic algorithms and Lagrangean based heuristics for vehicle routing. Ph.D. thesis, Coventry University, UK, 2000.
[33] Dongarra JJ. Performance of various computers using standard linear equations software. Working Paper, Computer Science Department, University of Tennessee, USA, 2001, http://www.netlib.org/benchmark/performance.ps.

Barrie M. Baker holds a B.Sc. in Mathematics and an M.Sc. and Ph.D. in Operational Research, all from London University. He is a member of the academic staff at Coventry University, and his main interests are in network and distribution modelling and spreadsheet modelling.

Mossie Ayechew holds a B.A. in Accounting from Addis Ababa University, an M.Sc. in Computing from London University, and a Ph.D. from Coventry University for his research into genetic algorithms and Lagrangean-based heuristics for vehicle routing.