Experiments in the use of Neighbourhood Search techniques for Vehicle Routing *

Tim Duncan
Artificial Intelligence Applications Institute,
University of Edinburgh, Edinburgh EH1 1HN

AIAI-TR-176
June 1995

Abstract

Distribution is a substantial cost in many companies and in some sectors it contributes a high percentage of the value added to goods. Therefore the potential for savings is large, and there is much interest in improving distribution efficiency. The Vehicle Routing Problem (VRP) is an abstraction of a class of problems that has been studied both as a theoretical problem and for its practical relevance to applications. In this paper we investigate Neighbourhood Search techniques for solving the VRP and describe some experiments with variations on the approach applied to a set of benchmark problems.

1 Introduction

Distribution is a major part of logistics and a substantial cost in many companies. In some sectors (*e.g.* soft drinks) it contributes a high percentage of the value added to goods. Therefore the potential for savings is large and there is much interest in improving distribution efficiency.

The Vehicle Routing Problem (VRP) is an abstraction of a class of problems that has been studied both as a theoretical problem and for its practical relevance to applications [Christofides *et al* 79, Osman 93]. There are a number of extensions to the VRP, however the basic problem consists of a set of customers to be served by a fleet of vehicles from a single depot (see Figure 1). Each customer places a demand for a certain quantity of goods and there are limits on vehicle capacity and the maximum distance that any vehicle can travel. The object is to find a set of routes which minimize the total cost of delivering goods to the customers.

*Presented at Expert Systems'95

This formulation covers a wide range of applications in the real world, not only delivery by vehicle but also collections, and other types of delivery (*e.g.* postal routes). A number of extensions to the basic VRP have been studied, including placing time constraints on deliveries, multiple depots, and the backhaul option (*i.e.* mixing delivery and collection), however these are beyond the scope of this paper and the reader is referred to recent surveys such as [Osman 93].

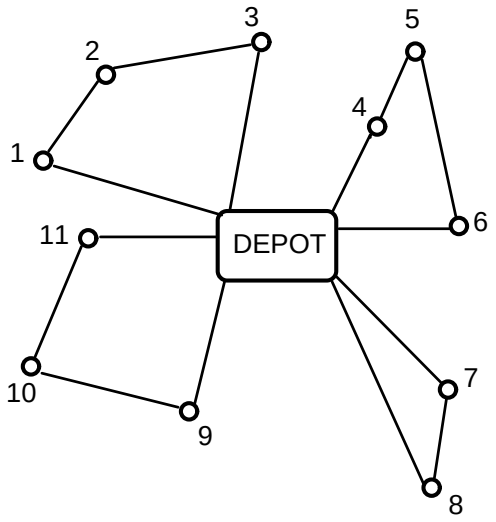

Figure 1: The Vehicle Routing Problem

As a generalisation of the Travelling Salesman Problem the VRP is NP-hard [Lenstra & Rinnooy Kan 81], *i.e.* there are no known methods for finding the optimal solution within polynomial time. Historically the focus in Operational Research (OR) has been on methods for generating a feasible solution which can then be improved through local search [Christofides *et al* 79]. Currently methods such as Genetic Algorithms and Tabu Search (see [Reeves 93] for overviews) are popular within Artificial Intelligence and OR circles. Constraint satisfaction (CSP) methods have not yet been very successful, *e.g.* [Christodoulou *et al* 94] had to extend their CSP approach with local search techniques.

Tabu Search and Genetic Algorithms belong to a class of methods known as Neighbourhood Search. These start with a solution (which need not be valid) and iteratively improve upon it by exploring the *neighbourhood* of states that can be reached by applying an operator to the current state. There are differences between the methods and variations exist within each technique, but all involve an interplay between the complementary pressures of *intensification* – selecting moves which improve the cost of the solution, and *diversification* – accepting non-improving moves in order to escape local optima.

In the next section we briefly outline these methods and show how they can be applied to the VRP. In the following section we report some experimental comparisons of systems embodying ideas from both Genetic Algorithms and Tabu Search, applied to a set of benchmark problems [Beasley 90].

2 Neighbourhood Search

2.1 Genetic Algorithms

Genetic Algorithms (GA) are based on an analogy with biological evolution. A population of chromosomes (*i.e.* strings representing possible solutions) is maintained. New chromosomes are produced by combining members of the population, and these replace existing chromosomes. The probability of a chromosome being involved in this process depends on its fitness (*i.e.* an evaluation function). GAs are performing a type of neighbourhood search by applying operators to chromosomes to produce new ones. The two operators most commonly used are *crossover* and *mutation*, however these frequently have to be adapted to take account of context (*i.e.* the structure implicit in the problem's representation as a chromosome). Other common variations in GAs include: incremental versus generational replacement, *i.e.* whether new chromosomes replace old ones *en bloc* or one at a time; elitism – the best chromosome always becomes part of the new generation; and different ways of relating fitness to the evaluation function, *e.g.* ranking.

2.2 Tabu Search

In contrast Tabu Search (TS) works on a single solution at a time, although it always keeps a record of the best solution found. In exploring the neighbourhood of a solution TS evaluates all the moves in a *candidate list*. The number of moves examined is one parameter of the search. The best move on the candidate list is generally accepted, unless the same move or its inverse has been made recently in which case it is *tabu*. A tabu move may still be accepted however, if the solution it produces is better than any other produced so far. This is known as *aspiration*. TS introduces diversification when there are no improving moves available. In such situations frequently tried moves are discouraged, while moves with big *influence* (*i.e.* capable of introducing large structural change to the solution) are encouraged.

2.3 Representation for the Vehicle Routing Problem

The two prerequisites for a neighbourhood search approach are a representation for the problem and a set of operators which can applied to that representation. The VRP as shown in Figure 1 can be transformed so that it can be represented using a single vector. The depot is replaced by a dummy customer between each route (see Figure 2). In this way the four routes can be represented as one large one. The end of the vector should be imagined as connected back to the first element.

The standard operators employed in GAs cannot be used since our representation has structure (*e.g.* each customer must appear once and once only) and crossover or mutation would produce badly formed strings. Instead we need an operator which will preserve the structure while generating permutations of the string in order to allow us to explore its neighbourhood. Three such operators are illustrated in Figures 3–5.

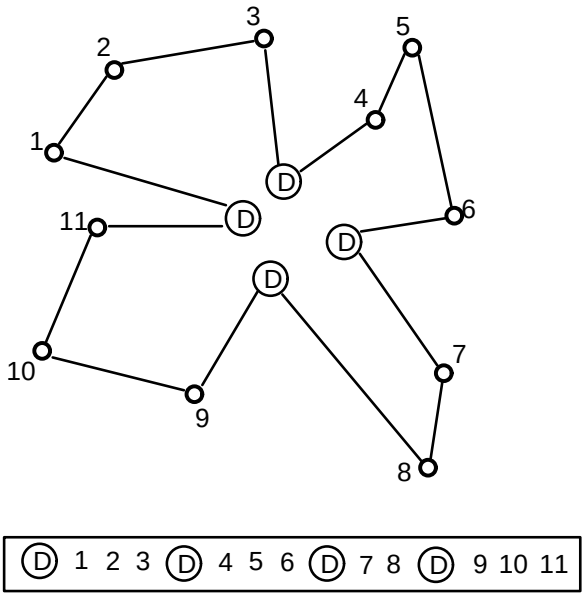

Figure 2: Representing the routes as a string

The "Swap" operator: identifies two elements of the vector (i and j in Figure 3) and simply swaps their positions.

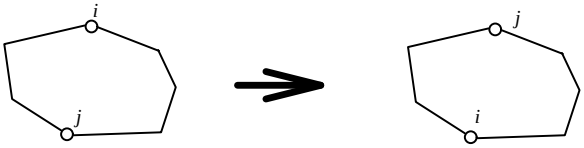

Figure 3: The "Swap" operator

The "Re-Insert" operator: removes one element (x in Figure 4) from the vector and replaces it between two adjacent nodes (i and $i+1$).

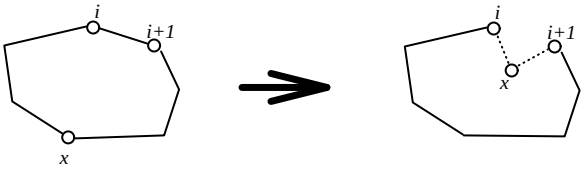

Figure 4: The "Re-Insert" operator

The "Relink" operator: cuts the links between two pairs of customers (i – $i+1$ and j – $j-1$ in Figure 5) and re-links them across the pairs.

Initially the vector should contain a dummy depot node for each available vehicle. Where a solution can be generated with less than the maximum number of vehicles, adjacent dummy nodes signify an empty route. Since the aim is to

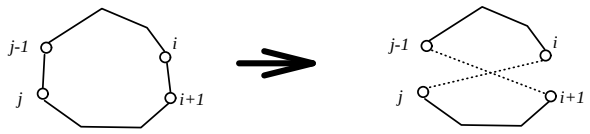

Figure 5: The "Relink" operator

minimize the total length of the routes, the lengths of each route are summed to give the cost of a particular solution. Where an individual route breaks the maximum weight or maximum distance constraint an appropriate penalty factor is added. Naturally the size of the penalty should be proportional to the degree of constraint violation.

3 Experiments

In this section we describe some experiments applying these ideas to a set of problems from the OR library [Beasley 90]. A number of variations were incorporated into the scheduling system and each was tested with the datasets VRPNC1, VRPNC2, VRPNC3, VRPNC6, and VRPNC14 (originally drawn from [Christofides *et al* 79]).

The first three of these contain data for 50, 75, and 100 customers respectively, with capacity restrictions on vehicles but none on distance. The VRPNC6 dataset is VRPNC1 with the additional restriction that no route can exceed 200 units of distance. The VRPNC14 dataset contains 100 customers with capacity and distance restrictions. An additional feature of VRPNC14 is that the customers are geographically clustered.

The evaluation function used in all the experiments reported here was based on the total distance travelled, with penalties for schedules which broke either the weight or distance constraints. In both cases the penalty was the amount overweight or over-distance, times 1000.

Random: The first version of the system was loosely based on GA principles. A population of 100 chromosomes were generated in a pseudo-random manner. The dummy depot nodes for the 20 available vehicles were first distributed evenly over the string, and then the customers were assigned to the remaining positions randomly. Twenty vehicles was felt to be a suitably large number to cover any of the datasets. All three of the operators were employed under a scheme of generational replacement with elitism. That is at each cycle the best chromosome was copied directly into the new population, and then the rest of the population were generated by applying one of the operators (chosen randomly and with random parameters) to one of the top 10% of chromosomes from the previous generation. Fitness was based simply on the evaluation function.

The results for VRPNC1 are depicted graphically[1] in Figure 6. This graph also shows the result of using the Relink operator only, and the Swap and Re-insert

[1]The first two points on the graph have been omitted. The values of these two initial solutions

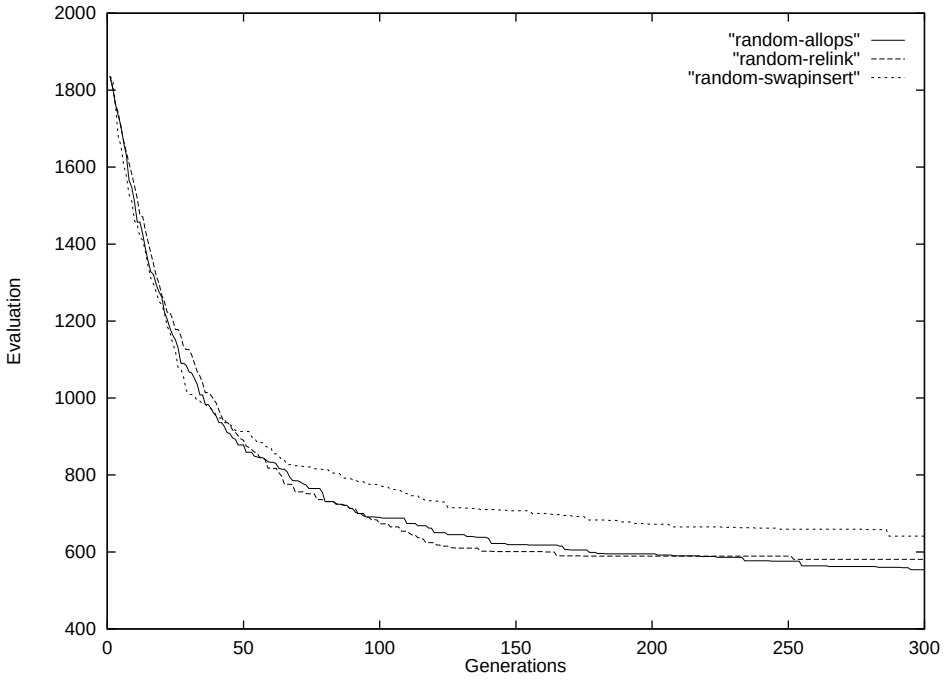

Figure 6: Comparing operators (random initial population)

operators together. There appears to be very little difference between the operators, Swap and Re-insert are marginally worse and in a couple of the datasets Relink was better than Allops, but the differences are not large enough to be significant.

Nearest Neighbour: It has sometimes been suggested that *seeding* a GA with a good initial solution leads to better results [Reeves 93]. There are many algorithms for constructing feasible solutions (*e.g.* the well known "savings" algorithm of [Clarke & Wright 64]). We decided to try a very simple one: first a customer is picked at random, then the customer's five nearest neighbours are considered. The route is extended by adding the nearest unassigned customer which does not violate the weight or distance constraints. When the route cannot be extended a new one is started. If all the vehicles are used the remaining customers are added to the end of the vector. This would mean that the vector is not a valid solution, but nevertheless it provides a better starting point than random.

The results of this approach, again for VRPNC1 comparing operators, are shown in Figure 7, and the random and nearest neighbour strategies are compared in Figure 8.

The "nearest neighbour" strategy naturally produces better initial solutions, although the two strategies rapidly converge in most of the datasets. There are however still over 280 points difference after 300 cycles in datasets VRPNC3 and VRPNC14.

were so large that they had the effect of reducing the scale on the Y axis and obscuring the shape of the graph.

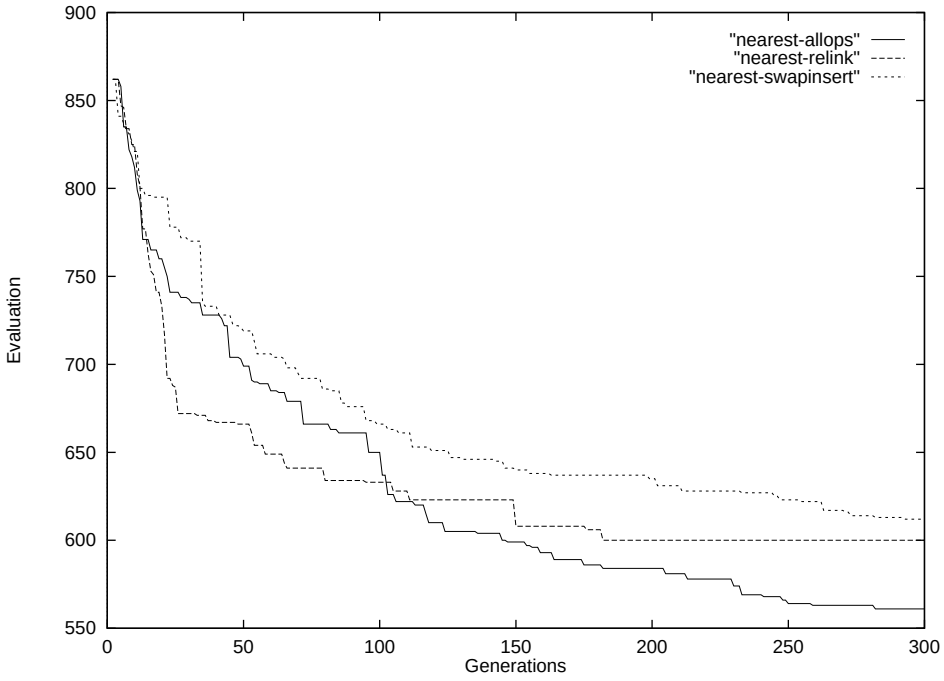

Figure 7: Initial population generated using "nearest neighbour"

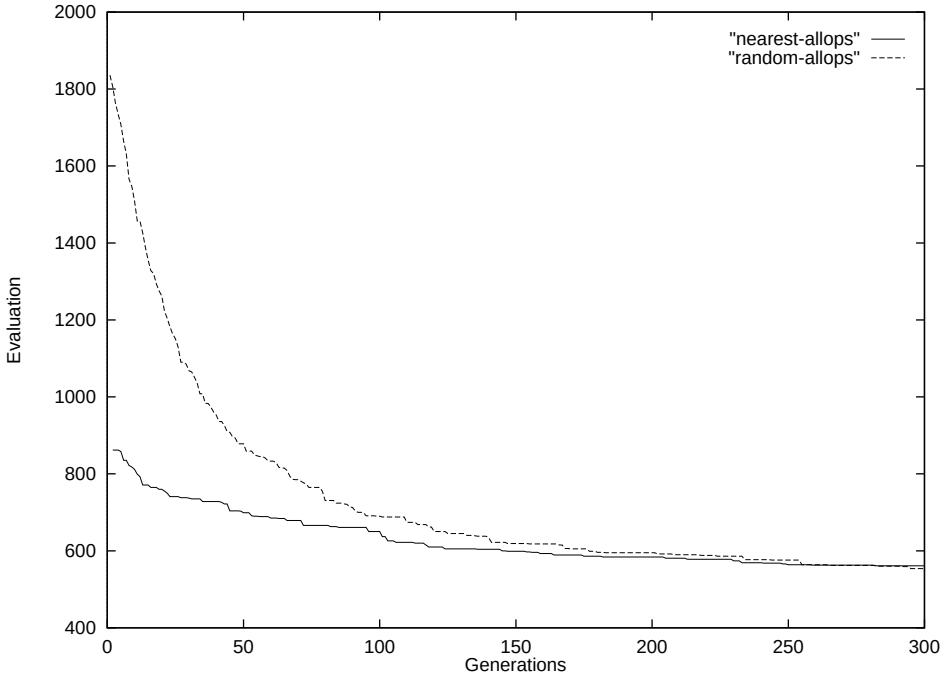

Figure 8: Comparing strategies for initial population generation

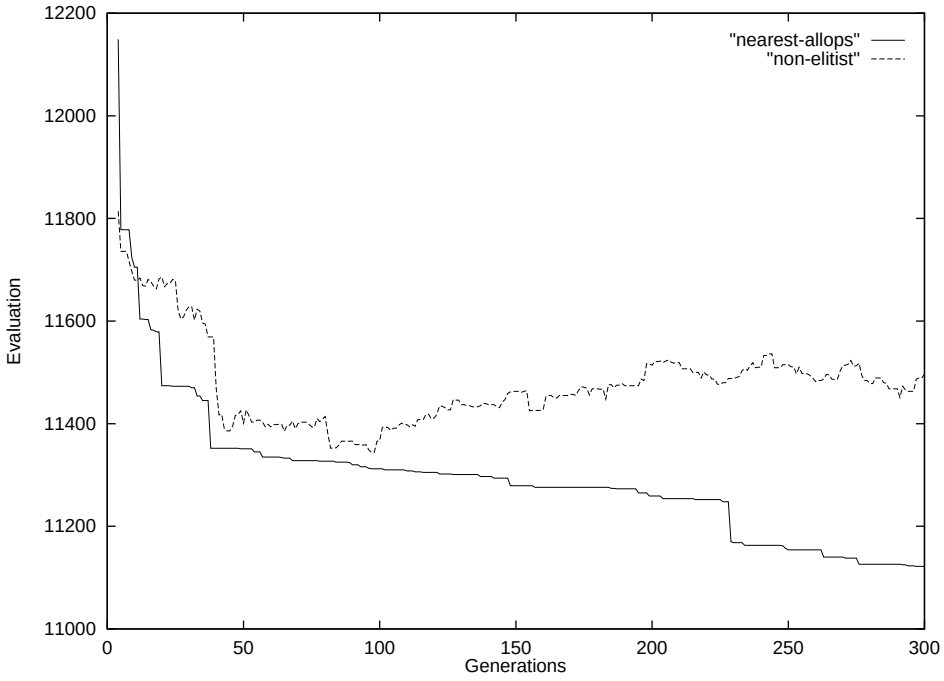

Figure 9: The effect of elitism

Elitism: In order to assess the effect that elitism was having on the results we ran the system without it. The effect was clear, being pronounced in datasets VRPNC2, VRPNC3, and VRPNC14. Without elitism the graphs are no longer monotonic, usually this is exhibited as small variations around a main trend of improvement, however in VRPNC14 this trend is lost (see Figure 9). It is clear that elitism is worthwhile.

Incremental vs Generational Replacement: In order to compare incremental (or *steady-state*) and generational (*i.e. en bloc*) replacement we modified the cycle so that each chromosome generated was used to replace the worst chromosome in the population. In order to make possible comparison with other results reported here, n chromosomes were generated and replaced each cycle, where n is the population size minus one (because of elitism). The results are inconclusive, occasionally showing incremental generation to be better, but not consistently across datasets. Figure 10 illustrates the results for VRPNC1.

Tabu: Converting the scheduler to use a Tabu Search approach involved a few more changes. The first difference is that the system works on a single, current solution rather than a population of solutions. The initial solution is constructed using the "nearest neighbours" method as before. In place of the application of random operators to the population, the TS version of the system employs a more systematic exploration of the neighbourhood. That is it uses nested loops to iterate over the possible parameters of the available operators, comparing the results to find the best move to make. An explicit candidate list is not required.

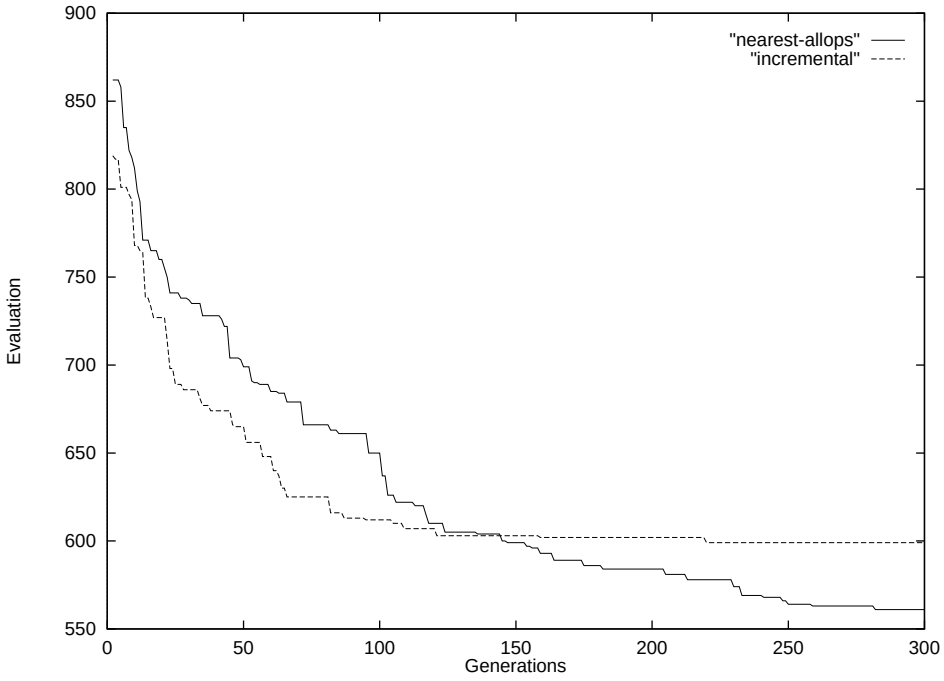

Figure 10: Incremental vs generational replacement

In fact since the method described above would require 4950 evaluations per operator each cycle for a 100 customer dataset, the system reduces the load by stepping through each loop by a factor of four and considers only the "relink" operator. This reduces the number of checks to 325, however his would also leave unexplored gaps in the neighbourhood, so we vary the starting point of the iteration relative to the cycle.

A *tabu list* of the five most recent moves is maintained and testing it checks both the move and its inverse since for swap and relink either will restore the previous state. Each move is also recorded in a frequency table so that this information can be used to promote diversification when no improving move is available. Apart from the current solution the system also keeps a record of the best value found so far. The results for VRPNC1 are shown graphically in Figure 11.

The results show that the TS approach reached acceptable values much quicker for all of the datasets apart from VRPNC3. However there seems to be a tendancy for the TS approach to reach a plateau. Since the number of candidates considered is obviously an important factor we ran the scheduler again with the step factors of the nested loops set at one and two respectively (which requires 2500 evaluations per cycle for the 100 customer dataset). The results are illustrated for VRPNC6 in Figure 12. There is a clear improvement in performance, although at the expense of longer processing time.

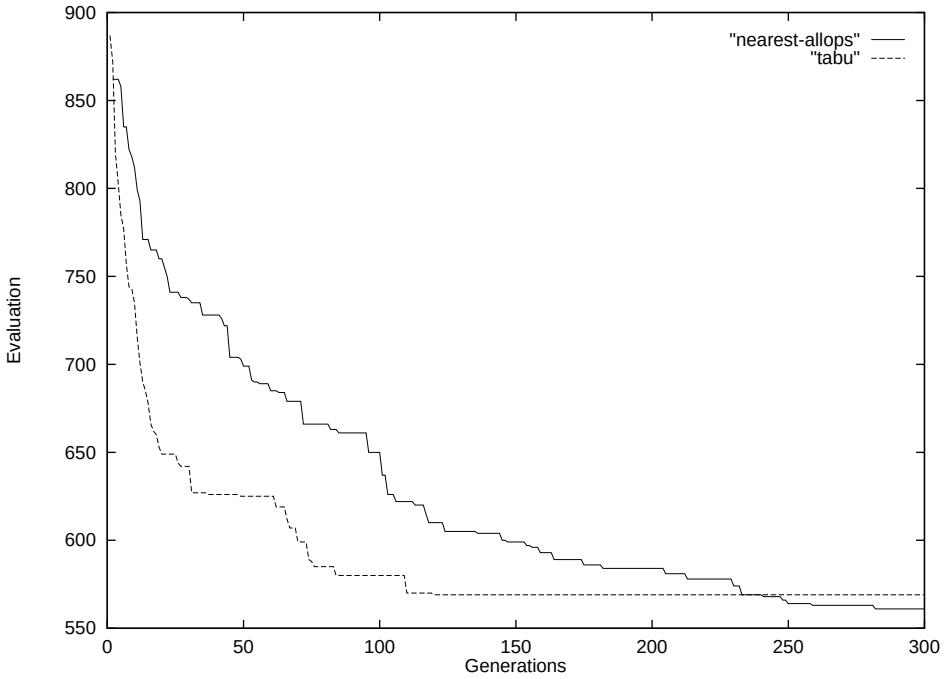

Figure 11: Tabu search

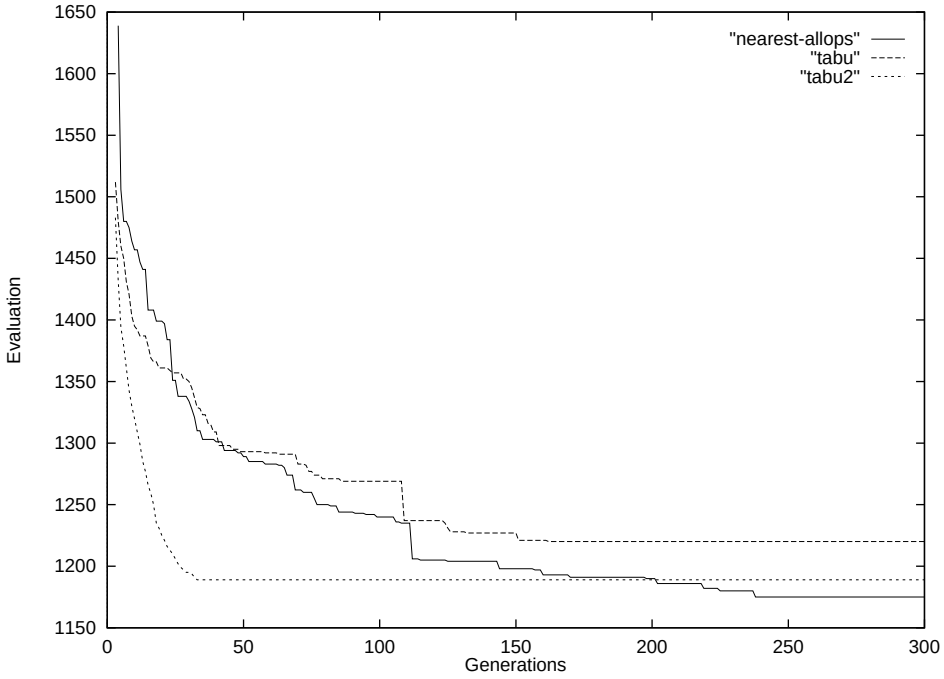

Figure 12: Tabu search: larger candidate list

Comparing Versions: Table 1 compares the results obtained by the various versions of the system for the five datasets.

DATA	RANDOM	NEAREST NEIGHBOUR			NON-	INCRE-	
	ALLOPS	ALLOPS	RELINK	SWAP/INSERT	ELITIST	MENTAL	TABU
VRPNC1	554	561	600	612	596	599	569
VRPNC2	985	921	909	945	1070	984	918
VRPNC3	1213	930	917	1042	1174	901	913
VRPNC6	1182	1175	1161	1199	1223	1185	1220
VRPNC14	11504	11122	11052	11202	11496	11049	11198

Table 1: Comparing best solution obtained by each variant

4 Discussion

Although we have presented Genetic Algorithms and Tabu Search as two very different approaches there are in fact similarities. Both for example maintain the best solution found (elitism in GAs); the same operators were used for both approaches; and whereas TS explores the neighbourhood explicitly through the candidate list, the GA explores it through randomization and by maintaining a large set of solutions under consideration.

Obviously the size of the TS's candidate list is an important factor and one would intuitively feel that larger population sizes for GAs would also be better. Although we have not examined this here, [Alander 92] suggests that between n and $2n$ is the ideal population size. This would make our choice of 100 about right for the datasets we were looking at. For GAs the proportion of chromosomes involved in producing the next generation is also an important factor since if this is too small there may not be enough variance to generate interesting solutions.

The strategy for constructing the initial solutions was shown to be important, and improvements were noticeable even for a relatively naïve algorithm like "nearest neighbours". Examining some of the more sophisticated approaches from the OR literature may be worthwhile. However, there is also the possibility that *seeding* might lead to premature convergence [Reeves 93]. It is also said that "fitness as evaluation" may lead to premature convergence too. Judging whether this has occurred or not is difficult, however we note that most of the graphs are beginning to level off by 300 cycles and that the results are not as good as those reported in [Christofides *et al* 79].

The operators used here seem to have been reasonably effective in generating new solutions. Swapping on its own does not appear as effective as relinking. This may be because the latter produces larger changes. On the other hand relinking sometimes reaches a plateau and swapping or inserting may help to get the system

out of it. It would also have been possible to use other operators such as *uniform order-based crossover* [Davis 91] in the GA. This is a modified crossover which maintains the structure of sequence-based chromosomes.

The possibility of premature convergence raises the question of whether diversification was effective in the TS implemented here, since diversification should encourage the system to explore new areas when no improving move can be found. The two attributes frequency and influence are used to promote diversification. Moves with large influence should be encouraged and frequently tried moves should be penalized. In the present system moves are recorded in terms of the operator and the two positions involved, however this may not give useful information since, for example, swaping X and Y at one point in the proceedings may have no relation to swapping the same positions later in the day. Recording the contents rather than the position may be more useful. The same point also applies to the tabu list.

It is difficult to know whether the maximum number of vehicles was well chosen or not. On the one hand this should be enough to allow feasible initial solutions to be constructed, but on the other hand an excess of dummy depot nodes may make it harder for the operators to produce interesting schedules. Even with 20 vehicles it was difficult to produce initial solutions for datasets VRPNC6 and VRPNC14. This was presumably due to the additional restrictions in these two datasets. Some solution construction algorithms work better with clustered data, although this was not noticeable here.

There are many ways in which the work presented here could be extended, perhaps the most important from the perspective of making the techniques applicable to real world problems would be to deal with time restrictions on deliveries.

5 Conclusions

We have outlined a representation and operators that can be used for applying Neighbourhood Search techniques (including ideas from Genetic Algorithms and Tabu Search) to the Vehicle Routing Problem. Table 1 compares the results obtained by the various versions of the system for the five datasets. In investigating these variants we have come to the following conclusions:

- Swapping on its own is not as effective as relinking.
- Maintaining the best value found (elitism) is a good idea.
- Systematic exploration of the neighbourhood is more effective, particularly during intensification.
- Seeding the algorithms with a feasible initial solution(s) may result in better solutions as well as being faster.
- There is no advantage in incremental vs generational replacement.

The techniques described provide a good starting point for tackling the VRP but there are many other factors which remain to be studied.

References

[Alander 92] J. Alander. On Optimal Population Size of Genetic Algorithms. In *Proceedings of CompEuro'92*, pages 65–70. IEEE Computer Society Press, 1992.

[Beasley 90] J. E. Beasley. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990. (See `http://mscmga.ms.ic.ac.uk/info.html`).

[Christodoulou *et al* 94] N. Christodoulou, E. Stefanitsis, E. Kaltsas, and V. Assimakopoulos. A Constraint Logic Programming Approach to the Vehicle-Fleet Scheduling Problem. In *Proceedings of Practical Applications of Prolog (PAP'94)*, pages 137–149, London, 1994.

[Christofides *et al* 79] N. Christofides, A. Mingozzi, and P. Toth. The Vehicle Routing Problem. In N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, chapter 11, pages 315–338. John Wiley and Sons, 1979.

[Clarke & Wright 64] G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12:568–581, 1964.

[Davis 91] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[Lenstra & Rinnooy Kan 81] J. Lenstra and A. Rinnooy Kan. Complexity of Vehicle Routing and Scheduling Problems. *Networks*, 11:221–228, 1981.

[Osman 93] I. H. Osman. Vehicle Routing and Scheduling: Applications, Algorithms and Developments. In *Proceedings of the International Conference on Industrial Logistics*, Rennes, France, July 5-8 1993.

[Reeves 93] C. R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Optimisation.* Blackwell Scientific Publications, Oxford, 1993.