CEJOR (2006) 14:193-207

CEJOR

Route stability in vehicle routing decisions: a bi-objective approach using metaheuristics

Kenneth Sörensen

University of Antwerp, Faculty of Applied Economics
Prinsstraat 13, B–2000 Antwerp, Belgium
e-mail: kenneth.sorensen@ua.ac.be

Abstract In this paper, we argue that vehicle routing solutions are often tactical decisions, that should not be changed too often or too much. For marketing or other reasons, vehicle routing solutions should be *stable*, i.e. a new solution (when e.g. new customers require service) should be as similar as possible to a solution already in use. Simultaneously however, this new solution should still have a good quality in the traditional sense (e.g. small total travel cost). In this paper, we develop a way to measure the difference between two vehicle routing solutions. We use this distance measure to create a metaheuristic approach that will find solutions that are "close" (in the solution space) to a given baseline solution and at the same time have a high quality in the sense that their total distance traveled is small. By using this approach, the dispatcher is offered a choice of Pareto-optimal solutions, allowing him to make a trade-off between changing his existing solution and allowing a longer travel distance. Some experiments are performed to show the effectiveness of the approach.

Key words: Vehicle routing, Route stability, Bi-objective optimization, Distance measure

1 Introduction

In the research literature, the vehicle routing problem (VRP) is generally treated as a purely operational decision. It is supposed that any change in the problem data will trigger re-optimization of the problem and that the new solution—of which the quality is guaranteed by the optimization procedure—will be implemented, regardless of the solution previously in use. For many reasons, these assumptions are unwarranted in a large number of practical situations.

First, when small data changes occur, a dispatcher will usually not start from scratch and calculate an entirely new solution, but instead will attempt to "repair" the existing solution using some simple procedure. In general, this will result in a new solution that is fairly similar to the current one. Even when a complete re-optimization of the vehicle routing problem is

practically feasible however, it might still be undesirable to generate a new solution that is completely different from the one that is implemented.

- Many companies nowadays attempt to build strategic relationships with their customers. This can imply that the same driver visits the same customers at approximately the same time each period (e.g. each day or each week).
- A frequent changing of the routes might create confusion among the drivers who have to execute them. This could lead to more errors and consequently higher costs.
- Dispatchers that are used to driving approximately the same routes each period, are more familiar with the specific characteristics of each route. This enables them e.g. to make last-minute changes to the routes in a more intelligent way.

As this discussion points out, a certain *stability* or robustness *in the solution space* is desirable for many practical routing situations. A dispatcher should be able to find solutions that are "close" to a given baseline solution and simultaneously have a high solution quality (e.g. a small total travel cost). The problem of solution stability (also called solution robustness [21]) is relatively new. Some of the problems mentioned above have been pointed out by Bertsimas and Simchi-Levi [2]. Ribeiro and Lorenço [20] find that in real-life problems, "marketing objectives" like the relationship between drivers and the customers they serve, are often more important than cost considerations. Thangiah et al. [25] find that in the routing of school buses, there is a preference for routes to remain the same throughout an entire semester.

The concept of route stability, which can be considered as robustness in the solution space, is illustrated in figure 1. An extra customer is added to an existing *baseline solution* in figure 1 and the problem is re-optimized. The solution in figure 1 is similar to the baseline solution. The solution in figure 1 is much less similar. Visually, it is easily verifiable that the difference (or distance) between solution 1 and 1 is much larger than that between solution 1 and 1. In many cases, there will be a preference of the dispatcher to use solution 1, even though its quality (e.g. total distance) might be lower than that of solution 1. However, this preference is usually not absolute in the sense that solution 1 will be preferred when its quality is *much* better than that of 1.

Finding vehicle routing solutions that are "close" to a given baseline solution requires a *distance measure* to calculate the difference (or similarity) between two given solutions. Such a distance measure, based on the *edit distance*, is developed in section 3. We should stress at this point already that the distance between two solutions, measured in order to control route stability, needs to be measured in the solution space, and not in the objective function space. In other words, the solution chosen by the decision maker should *itself* be close to a baseline solution, not its quality.

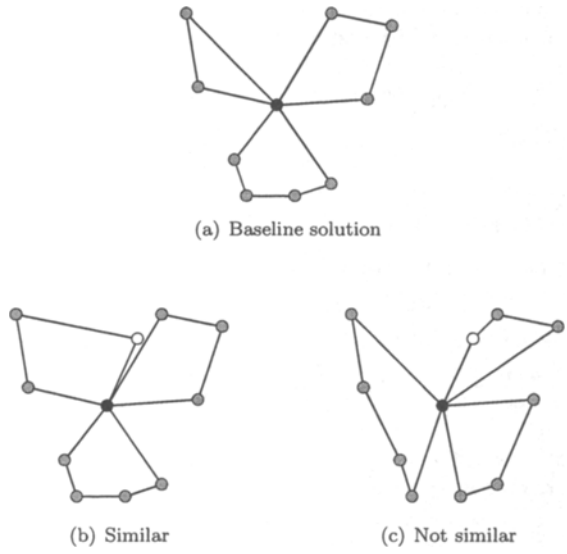

Figure 1: Solution robust vehicle routing example

The approach for route stability developed in this paper uses a metaheuristic to generate a set of good solutions. For each solution, the distance to a given baseline solution is measured and a set of efficient solutions is generated. From these solutions, a decision maker can choose which solution best satisfies his or her preferences.

The organization of this paper is as follows. In the next section, we briefly introduce the VRP and survey the literature, especially with respect to metaheuristic solution approaches for this problem. In section 3, a distance measure is developed to accurately determine the distance between two solutions. In section 4, an algorithm is developed that uses this distance measure to determine solutions that have a small travel cost and a small distance to a baseline solution simultaneously. Finally, section 5 discusses an experiment to show the effectiveness of the approach.

2 Problem description and literature review

In this paper, we focus on the *capacitated, distance-constrained* vehicle routing problem (CDVRP or simply VRP). The VRP is defined on a graph $G = (V, E)$ with $V = v_0 \cup \{v_1, \ldots, v_n\}$. The set $\{v_1, \ldots, v_n\}$ represents a set of customers and v_0 represents a depot. With each edge, a travel cost between customers is associated. Each of the n customer has a non-negative known demand q_i $(i = 1, \ldots, n)$. This demand must be serviced by a homogeneous set of vehicles, all having capacity Q. Travel costs c_{ij}

between customers i and j are known and constant. Sometimes an extra cost (called drop cost) is incurred for each customer visited.

The objective of the VRP is to determine a set of minimum-cost routes that satisfy the following constraints.

1. Each route begins and ends at the depot.
2. The total demand serviced in a single route does not exceed the capacity Q of the vehicles.
3. The total cost in a single route (sum of travel costs and drop costs) does not exceed a given maximum cost C.

The $\mathcal{NP}$-complete [13] vehicle routing problem is very difficult in practice and even some moderately-sized problems have not been solved to optimality [27]. Because of this fact as well as its practical importance, many attempts have been made to design efficient heuristics for the vehicle routing problem, starting with Clark and Wright [6]. Also a rather large number of optimal algorithms have been developed, e.g. the branch and bound algorithm of Fisher [9] or the branch and cut algorithm of Lysgaard et al. [15]. The most powerful optimal methods can now find optimal solutions for problems with up to 100 customers with some reliability, but often require large clusters of parallel computers, see e.g. [17].

While optimal algorithms fail to find solutions for large problem instances, simple heuristics quickly find solutions, but these are often of low quality. *Metaheuristics* provide a means to find better solutions than simple heuristics in a reasonable amount of time. Although many more applications of metaheuristics to the vehicle routing problem with time windows exist, several metaheuristic approaches have been proposed for the VRP. These approaches include tabu search [10, 18, 24, 30], granular tabu search [26], memetic algorithms [1, 16], ant algorithms [19], and others. For a detailed recent survey, including the details of these methods, we refer to Cordeau et al. [7].

A possible way of finding solutions that are close to a given baseline solution and hence increase solution robustness, has been proposed in Jaillet [11] and Bertsimas et al. [4] for the traveling salesman problem and by Bertsimas [3] for the VRP (see also Bertsimas and Simchi-Levi [2]). In this approach—coined *a priori optimization*—a single route is designed that includes all customers. When the demand becomes known, two different procedures are proposed: (1) all customers are visited, but only the customers that have non-zero demand are serviced or (2) only the customers with non-zero demand are visited. A vehicle on the route is then forced to return to the depot when its capacity has been reached. A natural objective for this problem is to minimize the total expected travel distance. A priori routing tackles the problem of route stability by never changing the route. This technique has some drawbacks however, the most important one being the fact that a solution can only be split into pieces by a very simple heuristic which may result in a very low-quality solution.

As mentioned in the introduction, measuring solution stability requires the existence of a *distance measure* between solutions. The distance measure used in this paper is based on the edit distance. The edit distance was first introduced by Levenshtein [14] in the context of error-correcting codes and later extended by Wagner and Fischer [29].

The algorithm developed in this paper is a memetic algorithm with population management (MA|PM). This algorithm uses distances between two vehicle routing solutions to actively control the diversity of a small population of individuals. MA|PM (formerly called GA|PM or *genetic* algorithms with population management) have been shown to produce highly competitive results on several benchmark problems, e.g. scheduling problems [22, 23], and current research is being undertaken to test and improve their performance.

3 Distance between two vehicle routing solutions

A VRP solution can be represented as a set of permutations, one for each trip. Each trip is determined by the order in which the customers appear in it. In this section, we develop a distance measure for vehicle routing solutions, based on the *edit distance*, also called *Levenshtein distance*. For a more elaborate discussion of some issues related to distance measures, including some other distance measures for permutation problems, we refer to Sörensen [21]. Another commonly used distance distance measure is the number of common edges between two solutions, but a comparison between these distance measures is beyond the scope of this paper.

3.1 Edit distance

The edit distance is used to calculate the distance between strings, composed of characters from a finite alphabet Σ. Λ is the *null-character*, signifying the absence of a character.

An *elementary edit operation* $(x, y) \neq (\Lambda, \Lambda)$ is an ordered pair of characters from the set $\Sigma \cup \Lambda$. An elementary edit operation is called a *substitution* iff $x \neq \Lambda$ and $y \neq \Lambda$. It is called an *insertion* iff $x = \Lambda$ and a *deletion* iff $y = \Lambda$.

An ordered set of edit operations that transforms a string s into a string t is called an *edit transformation* of s into t.

Elementary edit operations can be weighted by a weight function γ, that assigns a non-negative real number $\gamma(x, y)$ to each elementary edit operation (x, y). Using the function γ, a non-negative real value can be assigned to any edit transformation $E = [(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)]$. The weight of

edit transformation E is equal to

$$\gamma(E) = \sum_{i=1}^{m} \gamma(x_i, y_i).$$

The *edit distance* $\delta(s,t)$ between strings s and t is defined as

$$\delta(s,t) = \min\{\gamma(E)|E \text{ is an edit transformation of } s \text{ into } t\}$$

If $\gamma(x,y) = 1$ for all x, y then $\delta(s,t)$ is equal to the number of elementary edit operations required to transform string s into t.

The edit distance has several interesting properties. It is a metric, i.e. $\forall s,t,u$: $\delta(s,t) \geq 0$ (non-negativity), $\delta(s,t) = 0 \Rightarrow s = t$ (separation), $\delta(s,t) = \delta(t,s)$ (symmetry) and $\delta(s,t) + \delta(t,u) \geq \delta(s,u)$ (triangle inequality). The edit distance is also able to calculate the distance between strings of different lengths, in which not all characters of the alphabet appear, and in which characters may be repeated. A simple implementation in java can be tried on line at *http://www.merriampark.com/ld.htm.*

3.2 Calculation of the edit distance

For two strings s and t of length $|s|$ respectively $|t|$, the time-complexity of the dynamic programming algorithm proposed by Wagner and Fischer [29] is $O(|s| \times |t|)$, i.e. $O(n^2)$ if the lengths of both strings is about n. The space complexity can be reduced to $O(n)$ if only the value of the edit distance is needed (and not the edit sequence that this distance corresponds to). A more efficient algorithm is given by Ukkonen [28]. The worst-case time complexity is $O(n \times d)$, the average complexity is $O(n + 2 \times d)$ where n is the length of the strings and d is their edit distance. This is fast for similar strings where d is small, i.e. if $d \ll n$. Other, even more efficient algorithms have been proposed, but these are outside the scope of this paper.

3.3 Adaptation of the edit distance to vehicle routing problems

For a given VRP with n customers an alphabet of cardinality $n + 1$ is defined, having a character for each customer and one extra character for the depot. A solution to the VRP can then be represented as a string of characters of this alphabet. A distance measure between such strings s_1 and s_2 has the following characteristics.

1. The distance $d(s_1, s_2)$ is equal to the minimal number of edits needed to transform s_1 into s_2, taking into account that
2. trips are independent and therefore can be taken in any order, and
3. trips can be executed in any direction (forward or backward).

To calculate a distance measure $d(s_1, s_2)$ between two VRP solutions s_1 and s_2, the following procedure can be followed.

1. Let $T_1 = \{\tau_1^1, \tau_2^1, \ldots, \tau_k^1\}$ and $T_2 = \{\tau_1^2, \tau_2^2, \ldots, \tau_l^2\}$ be the sets of trips in solutions s_1 and s_2 respectively. Let $|T_1| = k$ be the number of trips in T_1 and $|T_2| = l$ the number of trips in T_2. Empty trips are added to the solution with the fewest trips, so that both solutions have the same number of trips.

2. Create a square matrix $D = d_{ij}$ of size $\max(k, l)$.

3. Let $d_{ij} = \min(\delta(\tau_i^1, \tau_j^2), \delta(\tau_i^1, \tilde{\tau}_j^2))$. $\tilde{\tau}_j^2$ is the reverse tour of τ_j^2 going from the last customer in τ_j^2 to the first.

4. Calculate $d(s_1, s_2)$ where

$$d(s_1, s_2) = \min \sum_{ij} d_{ij} x_{ij}, \tag{1}$$

s.t.

$$x_{ij} \geq 0 \qquad \forall i, j, \tag{2}$$

$$\sum_j x_{ij} = 1 \qquad \forall i, \tag{3}$$

$$\sum_i x_{ij} = 1 \qquad \forall j. \tag{4}$$

$d(s_1, s_2)$ is equal to the total cost of the minimum cost assignment having the matrix D as linear cost matrix.

The algorithm finds the matching between trips in the first solution and trips in the second solution that minimizes the total edit distance between the two solutions. We calculate the assignment problem using the Jonker–Volgenant algorithm [12]. According to Dell'Amico and Toth [8], this algorithm consistently performs well. It should be noted that in some rich routing models (e.g. models with time windows), tours cannot be reversed. In this case, the calculation of the distance is slightly simplified as step 3 can be changed to "Let $d_{ij} = \delta(\tau_i^1, \tau_j^2)$".

Although the computational time required to calculate this distance measure is relatively high, we believe it to be a very accurate measure of the difference between two vehicle routing solutions. Verbally, this distance measure calculates the *minimal number of changes required to transform the first solution into the second one.* It is possible to assign weights to each individual edit operation or to groups of edit operations, adding to the flexibility of this distance measure.

3.4 Example

For a vehicle routing problem with 10 customers, labeled a to j, 2 solutions are given: s_1 = abc|defg|hij and s_2 = bcdef|gjiha.

Matrix D contains in each cell the minimum of (1) the edit distance between the string in the row and the string in the column and (2) the edit distance between the string in the row and the reverse of the string in the column (or between the reverse of the string in the column and the string in the row which can be shown to be the same). An asterisk label (*) indicates that one of the strings needs to be reversed to find the minimum distance. An empty trip (indicated by Λ) is added to the second solution so that it has the same number of trips as the first one.

As an example, we calculate the contents of element d_{21}, representing the distance between trip 2 of solution 1 and trip 1 of solution 2. The edit distance between defg and bcdef is 3 (see eq. 5 for a possible transformation using 3 edits). The edit distance between defg and fedcb (the reverse of bcdef) is 4 (see eq. 6 for a possible transformation). The value of d_{21} is therefore equal to $\min(3, 4) = 3$.

$$\texttt{defg} \xrightarrow{\text{remove g}} \texttt{def} \xrightarrow{\text{add b}} \texttt{bdef} \xrightarrow{\text{add c}} \texttt{bcdef} \tag{5}$$

$$\texttt{defg} \xrightarrow{\text{d}\to\text{f}} \texttt{fefg} \xrightarrow{\text{add d}} \texttt{fedfg} \xrightarrow{\text{f}\to\text{c}} \texttt{fedcg} \xrightarrow{\text{g}\to\text{b}} \texttt{fedcb} \tag{6}$$

Table 1: Matrix D with edit distances for the VRP

	bcdef	gjiha	Λ
abc	4	4*	**3**
defg	**3**	4*	4
hij	5	**2***	3

Solving an assignment problem with the matrix in table 1 as cost matrix assigns abc $\leftrightarrow \Lambda$, defg $\leftrightarrow$ bcdef and hij $\leftrightarrow$ ahijg (indicated in bold in the table), yielding a total cost of 8.

4 An algorithm for solution stability and solution quality

We have argued that, when a certain baseline solution x_0 is implemented and a re-optimization of the problem is performed, there will often be a preference for the new solution x to be as close as possible to x_0. Nevertheless, x should also have a travel cost that is as low as possible. Both objectives should be achieved simultaneously. Preferably, the decision maker should be presented with a diverse set of solutions, as uniformly spread along the Pareto frontier as possible.

In this section, an algorithm is developed that is able to find solutions that have both a small travel cost and a small distance to a given baseline

solution. We first develop a generic algorithm and afterward specify how both objectives can be achieved. It should be noted that the memetic algorithm developed in the next section extensively uses the distance measure developed in section 3 for an altogether different purpose: to maintain a small but diverse population.

4.1 A MA|PM for the VRP

Memetic algorithms with population management, developed in Sörensen [21] are structured like a simple memetic algorithm, but add some features to improve performance. First, a very small population (of e.g. 10 or 20 solutions) is used. Second, *population management* ensures the diversity of the population. Algorithm 1 is a schematic representation of a MA|PM.

Algorithm 1 MA|PM

```
1: initialize population P, population diversity parameter Δ
2: repeat
3:    select: p1 and p2 from P
4:    crossover: p1 ⊗ p2 → c1, c2
5:    tabu search: on c1 and c2
6:    for each child c do
7:       while dP(c) < Δ do
8:          mutate c
9:       end while
10:      remove solution: P ← P\b
11:      add solution: P ← P ∪ c
12:   end for
13:   update diversity parameter Δ
14: until stopping criterion satisfied
```

Like in Prins [16], VRP solutions are encoded as strings of customers without trip delimiters. A binary tournament selection scheme is used, i.e. two solutions are chosen randomly and the best one is used for crossover.

The crossover operator used is LOX (linear ordered crossover), originally defined for the VRP. Parent 1 is cut at two locations i and j and the corresponding string is placed in positions i to j of offspring 1. Offspring 1 is then completed by searching parent 2 circularly from position $j + 1$. A second offspring is created by reversing the roles of the two parents.

After crossover, solutions are split into trips optimally (using the procedure outlined by Prins), so that they can be subjected to tabu search and later population management. The splitting procedure works by first building an auxiliary graph containing all feasible tours that have their customers in the same order as the encoded solution. It then finds the optimal splits by solving a shortest-path problem in this graph. For a more detailed explanation, we refer to Prins [16].

The tabu search procedure is able to quickly improve the quality of solutions it is set to work upon. A simple local search scheme is used, in which each possible swap of two customers is considered and the best-improving move is made. A tabu list is used to prevent the solution from getting stuck in a local optimum. The tabu search continues until the best solution found during the tabu search phase was not improved for a fixed number of moves. As the solutions returned by the crossover operator are generally of rather low quality, such a hybridization procedure is absolutely necessary to ensure that the algorithm finds good solutions. The two candidate solutions improved by the tabu search operator are then subjected to population management.

Population management works by calculating the distance of the candidate solution c to all other solutions in the population. The *distance to the population* $d_P(c)$ is equal to the smallest distance to any solution in the population. The candidate solution is added to the population if its distance to the population is larger than or equal to the *population diversity parameter* Δ. By using this criterion, solutions are only added to the population if they add sufficient new genetic material. The calculation of the distance to the population involves the use of the distance measure developed in section 3. The new solution replaces a solution chosen by binary tournament selection (i.e. the worst of two randomly selected solutions is removed from the population), thus maintaining the population size. If the distance to the population of the candidate solution is insufficient (i.e. smaller than the population diversity parameter Δ), it is mutated. The mutation operator randomly swaps two customers. This process is repeated until the candidate solution satisfies the diversity criterion, after which it is added to the population.

The value of the diversity parameter determines to a large extent the diversity of the population. A large value of Δ will allow only very different solutions in the population and will thus increase the diversity. A small value of Δ will likewise decrease the diversity. It follows that the value of Δ can be used to *control* the diversity of the population using *population management strategies*. See e.g. Sörensen and Sevaux [22] for more details.

4.2 Objectives

As mentioned, a solution is sought that simultaneously minimizes two objectives: the total distance traveled by all vehicles (which we will refer to as *travel cost* to avoid ambiguity) and the distance of the solution to the baseline solution, as measured by the distance developed in section 3. As explained previously, the algorithm uses a tabu search local optimization procedure inside the structure of a memetic algorithm with population management. The direction of the search is determined in two ways: (1) by the selection operator of the MA|PM and (2) by the move selection of the tabu search procedure. In principle, both can be used to guide the search in the direction of either of the objectives.

Experiments show however that it is both intractable and undesirable to direct the tabu search in the direction of minimization of distance to the baseline solution. First, the tabu search procedure requires too many evaluations of the solution quality and determining the distance to the baseline solution is too time-consuming. When using travel cost to determine as the criterion, the objective function evaluation can be easily short-circuited. It is easily verifiable that when customer k is inserted between customers i and j, the total cost of the route changes by $c_{ik} + c_{kj} - c_{ij}$ (provided that the route remains feasible). Similar types of reasoning apply for removing a customer from a route or swapping two customers in a route. Although individual distance measure calculations are negligibly small, using the distance measure to direct the tabu search increases total computing time dramatically as these constant-time shortcut calculations are replaced by the far more demanding distance calculations outlined in section 3. A second reason to not direct the tabu search towards the minimization of the distance to the baseline is that the tabu search procedure tends to be too effective in finding solutions close to the baseline solution. As a result, the quality of the solution in terms of travel cost tends to be too low. For these two reasons, we use the tabu search to direct the search in the direction of minimal travel cost only. This guarantees that all solutions generated by the algorithm are of relatively high quality.

The search is directed in the direction of minimal distance by using this criterion to select solutions from the population. The selection operator of the MA|PM therefore pulls the solution in the direction of minimal distance to the baseline, while the tabu search procedure is used to ensure the quality of the solution in terms of travel cost.

5 Experiments and results

We assume that a company uses a given baseline solution x_0 on a regular (e.g. daily) basis. A set of customers is added to the list of customers that require service. In general, adding the customers to the baseline solution in any simple way (e.g. by inserting them at their nearest insertion point) will render the baseline solution infeasible. Because of this, the MA|PM is used to find a new solution. The company prefers this new solution to be as close as possible to the baseline solution. This situation may occur in practice when a company has a set of "loyal" customers and other customers may call in to demand a visit.

In the experiment, the MA|PM is allowed 100 generations to find a baseline solution using the original list of customers from the Christofides et al. [5] instances. Then, the size of the customer list is increased by 10% (rounded upwards). New customers are generated randomly on the same grid as the original customers (i.e. x and y coordinates between 0 and 70). Demand of each customer is randomly generated between 0 and 30, comparable with the demand of the original customers.

Data file	n	Added	Crit.	f	d	CPU (s)
vrpnc01	50	5	f	556.13	14	137.80
			d	561.22	8	
vrpnc02	75	8	f	928.12	26	379.08
			d	973.67	25	
vrpnc03	100	10	f	907.95	51	744.94
			d	1012.29	36	
vrpnc04	150	15	f	1166.66	79	2070.34
			d	1287.65	66	
vrpnc05	199	20	f	1487.45	144	5110.79
			d	1585.95	109	
vrpnc06	50	5	f	606.41	20	167.77
			d	608.58	10	
vrpnc07	75	8	f	1007.89	36	515.94
			d	1048.10	29	
vrpnc08	100	10	f	931.41	25	678.53
			d	974.34	19	
vrpnc09	150	15	f	1352.90	94	2027.00
			d	1445.27	57	
vrpnc10	199	20	f	1633.28	116	4229.15
			d	1705.26	96	
vrpnc11	120	12	f	1291.59	89	1234.86
			d	1343.68	57	
vrpnc12	100	10	f	926.52	31	683.75
			d	956.76	18	
vrpnc13	120	12	f	1737.76	65	1602.42
			d	1952.10	52	
vrpnc14	100	10	f	1085.39	48	734.76
			d	1182.77	21	

Table 2: Results of the experiment with 10% customers added

The MA|PM is then allowed 200 generations to search for a solution that is both good and close to the baseline solution. For each of the solutions generated this way, travel cost $f(x)$ and distance to the baseline solution $d(x, x_0)$ are recorded. In table 2, the results are summarized. This table shows for each data set the objective function values of two solutions: the solution with minimal travel cost (first row) and the solution with minimal distance to the baseline (second row). All programming was done in MS Visual Basic, all experiments were performed on an AMD Athlon 1100 PC running MS Windows.

As can be seen from this table, the results show that a trade-off needs to be made between the travel cost and distance to the baseline solution, as decreasing one objective will generally lead to an increase of the other. In a more elaborate multi-criterion decision process, the decision maker can plot the values of both objectives against one another and choose any solution on the Pareto frontier. Figure 2 shows both the efficient and the dominated solutions found by the MA|PM for the data file vrpnc05. We should note of course that there is no guarantee that these solutions are on the real Pareto frontier as they may be dominated by solutions not encountered during the search.

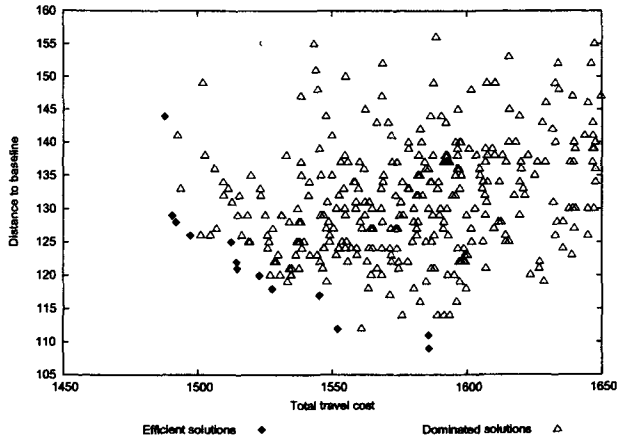

Figure 2: Efficient and dominated solutions for a run of the MA|PM

6 Conclusions

In this paper, we have argued that route stability can be an important criterion in vehicle routing. Marketing and other objectives often dictate a need for vehicle routing solutions to remain as similar as possible over time. We have shown how route stability can be measured by developing a distance measure between vehicle routing solutions. We have then used a memetic algorithm with population management or MA|PM to generate a non-dominated set of solutions that have both a small total travel cost and a small distance to a baseline solution. From this set, decision makers can choose a solution that best satisfies their preferences.

References

[1] J. Berger and M. Barkaoui. A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 54:1254–â1262, 2004.

[2] D. J. Bertsimas and D. Simchi-Levi. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Operations Research*, 44:286–304, 1996.

[3] D.J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40:574–585, 1992.

[4] D.J. Bertsimas, P. Jaillet, and A.R. Odoni. A priori optimization. *Operations Research*, 38:1019–1033, 1990.

[5] N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In N.Christofides, A.Mingozzi, P.Toth, and C.Sandi, editors, *Combinatorial optimization*, pages 315–338, Chichester, 1979. John Wiley.

[6] G. Clark and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[7] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. In A. Langevin and D. Riopel, editors, *Logistics Systems: Design and Optimization*, pages 270–297. Springer, 2005.

[8] M. Dell'Amico and P. Toth. Algorithms and codes for dense assignment problems: the state of the art. *Discrete Applied Mathematics*, 100:17–48, 2000.

[9] M.L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[10] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[11] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of customers is visited. *Operations Research*, 36:929–936, 1988.

[12] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[13] J.K. Lenstra and A.H. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–228, 1981.

[14] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10:707–710, 1966.

[15] J. Lysgaard, A.N. Letchford, and R.W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[16] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31(12):1985–2002, 2004.

[17] T.K. Ralphs. Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29:607–629, 2003.

[18] C. Rego and C. Roucairol. A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In I.H. Osman and J.P. Kelly, editors, *Meta-Heuristics: Theory and Applications*, pages 661–675, Kluwer, Boston, 1996.

[19] M. Reimann, K. Doerner, and R.F. Hartl. D-ants: Savings based ants divide and conquer the vehicle routing problem. *Computers & Operations Research*, 31:563–591, 2004.

[20] R. Ribeiro and H.R. Lorenço. A multi-objective model for a multi-period distribution management problem. In J. Pinho de Sausa, editor, *Book of Extended Abstracs of the Fifth Metaheuristics International Conference (MIC'2001)*, pages 97–102, Porto, 2001.

[21] K. Sörensen. *A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design.* PhD thesis, University of Antwerp, 2003.

[22] K. Sörensen and M. Sevaux. GA|PM: Genetic algorithms with population management for permutation problems. In *Proceedings of MIC 2003*, pages 38(1)–38(6), Kyoto, Japan, 2003.

[23] K. Sörensen and M. Sevaux. MA|PM: Memetic algorithms with population management. *Computers and Operations Research*, 33:1214–1225, 2006.

[24] É. D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23:661–673, 1993.

[25] S. R. Thangiah, B. Wilson, A. Pitluga, and W. Mennell. School bus routing in rural school districts. In *9th International Conference on Computer-Aided Scheduling of Public Transport (CASPT)*, San Diego, California, 2004.

[26] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle routing problem. *INFORMS Journal on Computing*, 15: 333–â346, 2003.

[27] P. Toth and D. Vigo. Exact algorithms for vehicle routing. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 1–31. Kluwer Academic Publishers, Boston, 1998.

[28] E. Ukkonen. Finding approximate patterns in strings. *Journal of Algorithms*, 6:132–137, 1985.

[29] R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

[30] J. Xu and J.P. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.