Designing delivery districts for the vehicle routing problem with stochastic demands

Dag Haugland* Sin C. Ho* Gilbert Laporte†

October 19, 2005

Abstract

This paper considers the problem of designing districts for vehicle routing problems with stochastic demands. In particular, demands are assumed to be uncertain at the time when the districts are made, and these are revealed only after the districting decisions are determined. Tabu search and multistart heuristics for this stochastic districting problem are developed and compared. Computational results show that tabu search is superior over multistart.

Keywords: districting, vehicle routing, tabu search, stochastic programming

*Department of Informatics, University of Bergen, N-5020 Bergen, Norway. {dag,sin}@ii.uib.no

†Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7. gilbert@crt.umontreal.ca

1 Introduction

Districting involves partitioning customer locations into districts according to some criteria. In general, these criteria include contiguity and balance constraints. Each district is independently responsible for the operations performed within its borders. One possible operation is the determination of vehicle routes. From a planning perspective, there is a fundamental difference between districting and routing. While districting decisions are made at a strategic or tactical management level, routing decisions are operational and made on a regular basis. It is possible to achieve long-term savings when designing districts because of the impact of this decision on routing. Some typical applications of districting include the design of political constituencies (Bozkaya et al., 2003), of sales territories (Fleischmann & Paraschis, 1988; Skiera & Albers, 1998; Drexl & Haase, 1999), of emergency and health care zones (Pezzella et al., 1981; Blais et al., 2003), of school boards (Ferland & Guénette, 1990), of police districts (D'Amico et al., 2002), etc. In this paper we focus on the design of delivery districts for the vehicle routing problem with stochastic demands.

The *Vehicle Routing Problem* (VRP) is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{0, 1, \ldots, n\}$ is the set of vertices with $1, \ldots, n$ representing customers, and 0 denoting the depot, and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ is the arc set. With every vertex $i > 0$ is associated a nonnegative demand D_i. With each arc (i, j) is associated a least travel cost c_{ij} so that (c_{ij}) satisfies the triangle inequality. A fleet of m homogenous vehicles of capacity q is stationed at the depot. The classical VRP consists of designing m delivery routes such that (i) every route starts and ends at the depot; (ii) every customer is visited exactly once; and (iii) the total demand of any vehicle route does not exceed q. An overview of the VRP can be found at Laporte & Semet (2002) and Gendreau et al. (2002).

In this article we consider the *Stochastic Vehicle Routing Problem* (SVRP), in which demands D_i are random variables. (Other classes of stochastic VRPs, namely with random travel times or customer presence are described in Gendreau et al., 1996.) In the SVRP, vehicle routes are planned in the first stage, and demands are only revealed later. As a result, some vehicle routes may violate the capacity constraints, i.e. route failures may occur. There exist different recourse policies to handle these situations. One possible policy is to follow the planned route until the vehicle is emptied, return to the depot to replenish, and then continue with the deliveries at the customer on the planned route where the failure occurred. Other policies, producing lower second stage costs, consist of planning preventive breaks (returns to the depot) in anticipation of future higher demands or of reoptimizing the remaining portion of the planned route after each customer visit (see, e.g. Dror et al., 1989).

We consider the problem of partitioning a set of customers into at most m contiguous districts such that all customers within the same district are serviced by the same vehicle. The allocation of customers to districts is fixed, even if a different district solution could prove more cost effective if a different demand pattern occurred. The actual demand is observed only after the districts have

been defined, and a travel plan must be produced within each district. This approach makes sense from a managerial point of view since district boundaries tend to remain the same over time while customer demand varies from day to day. Such situations arise commonly in contexts where delivery districts are fixed and defined by grouping together smaller units such as postal code areas. The idea of having districts is usually motivated by the drivers' need to be acquainted with their area and customer base. Parcel and furniture delivery districts are prime examples of this type of arrangement.

Since we assume that routing decisions are not fixed until the demand is observed, the VRP to be solved in the route planning phase is deterministic. Demands are revealed before the vehicles leave the depot, and our recourse policy is to plan breaks so as to minimize the routing cost. The breaks must be positioned such that the total demand realized in each subtour does not exceed the vehicle capacity. In the first stage, however, demand must be considered as a stochastic variable, and the goal is to define the districts in such a way that the expected total travel cost is minimized. A formal description of the problem is provided in Section 2.

Solutions fulfilling this goal typically consist of districts leaving a fair amount of flexibility in the routing decisions, that is an ability to adjust to the actual demand. The most flexible partition of the customers is obviously obtained by defining one all-inclusive district and leave the others empty. This would however lead to extreme expected travel cost for one district, while there would be no cost associated with the others. In order to balance the districts more evenly, we impose the constraint that the districts have to be constructed in such a way that the actual travel cost within each district never exceeds a given upper bound. Without such a constraint the planned solution would always consist of a single district, which does not make much practical sense.

The remainder of the paper is organized as follows. Section 2 provides a formulation of the model. The approach of approximating the expected cost is described in Section 3. A tabu search heuristic and a multistart heuristic are presented in Sections 4 and 5, respectively. This is followed by computational results in Section 6, and by the conclusion in Section 7.

2 Model formulation

The problem just described can be modeled as a *two-stage stochastic program with recourse*. In the first stage, the districting decisions are made, while in the second stage one VRP is solved for each district. The constraint relative to the composition of a district is that the travel cost is no larger than a given number.

Assure that the demands $D_1, \ldots, D_n$ are independent stochastic variables. For $S \subseteq \mathcal{V} \setminus \{0\}$, let $f(S, D)$ denote the stochastic routing cost function associated with the demand vector $D = (D_1, \ldots, D_n)$, let Ω be the set of all realizations of D, and for $d \in \Omega$ let $f(S, d)$ denote the routing cost given the realization d. Also let $E[X]$ be the expected value of a random variable X, and $P(\mathcal{E})$ the probability of an event $\mathcal{E}$. Given an upper bound t on the travel cost

in each district, we seek a partition $\{S_1, \ldots, S_m\}$ of $\mathcal{V} \setminus \{0\}$ into districts (some of which may be empty) of least expected routing cost, i.e., we solve

$$\min_{\{S_1,\ldots,S_m\}} \quad \sum_{j=1}^{m} E[f(S_j, D)] \tag{1}$$

$$\text{s.t.} \qquad f(S_j, d) \leq t \qquad j = 1, \ldots, m, \ d \in \Omega. \tag{2}$$

Note that the removal of (2) makes $\{S_1, \ldots, S_m\} = \{\mathcal{V} \setminus \{0\}, \emptyset, \ldots, \emptyset\}$ an optimal solution (see Section 1). It can be argued that (2) could be relaxed to either a probabilistic constraint stating that $P\left(f(S_j, D) \leq t\right)$ is to be above some lower bound, or to the constraint $E[f(S_j, D)] \leq t$. However, complying with the conventional format of two-stage recourse problems, we introduce neither probabilistic constraints nor constraints involving expectations.

A two-stage stochastic program is said to have *complete recourse* if the second stage problem is feasible regardless of the first stage decisions and the outcome of the random variables (Kall & Wallace, 1994). If this is true under the condition that the first stage decisions are feasible, we have *relatively complete recourse*. Even in the weaker of these cases, feasibility of the recourse can be disregarded when solving the first stage problem. For our model, problem (1)-(2) is said to have (k, ℓ)-complete recourse if for all $S \subseteq \mathcal{V} \setminus \{0\}$ the condition $\sum_{v \in S} E[D_v] \leq kq$ implies that there exists a set of $r \leq \ell$ feasible vehicle routes over S. Recourse completeness is formally defined as follows.

Definition 1. *Given k and ℓ, problem (1)-(2) is said to have* (k, ℓ)-complete recourse *when $\forall S \subseteq \mathcal{V} \setminus \{0\}$ the condition*

$$\textstyle\sum_{v \in S} E[D_v] \leq kq \tag{3}$$

implies that there exists a permutation $\pi_S = (v_1, \ldots, v_{|S|})$ of S, an integer $r \in \{1, \ldots, \ell\}$, and integers $0 = i_0 < i_1 < \cdots < i_r = |S|$ such that the following inequalities hold:

$$\textstyle\sum_{j=i_{h-1}+1}^{i_h} d_{v_j} \leq q \qquad \forall h = 1, \ldots, r, \ \forall d \in \Omega \tag{4}$$

$$g(\pi_S, i_1, \ldots, i_{r-1}) = \sum_{h=1}^{r} \left(c_{0 v_{i_{h-1}+1}} + c_{0 v_{i_h}} + \sum_{j=i_{h-1}+1}^{i_h - 1} c_{v_j v_{j+1}} \right) \leq t. \tag{5}$$

Note that $g(\pi_S, i_1, \ldots, i_{r-1})$ is the total travel cost of the r routes $(v_0, v_1, \ldots, v_{i_1}, v_0)$, $(v_0, v_{i_1+1}, \ldots, v_{i_2}, v_0)$, $\ldots$, $(v_0, v_{i_{r-1}+1}, \ldots, v_{i_r}, v_0)$, where $v_0 = 0$ denotes the depot.

In other words, (k, ℓ)-completeness implies that if the customers are partitioned such that the total expected demand in each district is no greater than the capacity of k vehicles, feasibility is guaranteed. Furthermore, regardless of the outcome of the demand, inequalities (4)-(5) imply that the routing problem

in each district can be solved by no more than ℓ non-empty routes. Given a partition $\{S_1, \ldots, S_m\}$ of $\mathcal{V} \setminus \{0\}$, where all of $S_1, \ldots, S_m$ satisfy (3), the completeness property means that only travel plans with this few routes need to be considered when computing the expected recourse cost. In practice checking whether for given k and ℓ a tentative districting plan satisfying (3) will also satisfy (4) and (5) is $\mathcal{NP}$-complete. Our algorithm tests this condition heuristically. Whenever the test is negative the corresponding solution is deemed infeasible.

3 Approximation of the expected cost

Since computing the expected cost of a given district S requires the solution of a number of routing problems, this is in its own right a resource demanding operation. By relying on an approximate solution to the subproblems, the heuristic can be guided by upper bounds rather than the expected cost itself. In Section 3.1 we show how to compute such bounds.

3.1 Approximating the expected recourse cost

Consider a district $S \subseteq \mathcal{V} \setminus \{0\}$ satisfying (3), and let $\pi_S = (v_1, \ldots, v_{|S|})$ be a permutation of its customers. For a given $d \in \Omega$, we define

$$
\begin{aligned}
g^*(\pi_S, d) \quad &= \min\nolimits_{r, i_0, \ldots, i_r} g(\pi_S, i_1, \ldots, i_{r-1}) && && (6)\\
\text{s.t.} \quad & 0 = i_0 < i_1 < \cdots < i_r = |S| && && (7)\\
& r \in \{1, \ldots, \ell\} && && (8)\\
& \textstyle\sum_{j=i_{h-1}+1}^{i_h} d_{v_j} \leq q && \forall h = 1, \ldots, r && (9)
\end{aligned}
$$

which is computable in $\mathcal{O}\left(|S|^{\ell+1}\right)$ time.

Since the set of routes $\left\{\left(0, v_{i_{h-1}+1}, \ldots, v_{i_h}, 0\right)\right\}_{h=1}^{r}$ constitutes a feasible solution to $VRP(S, d)$, it follows that $f(S, d) \leq g^*(\pi_S, d)$. By (k, ℓ)-completeness, we also have $f(S, d) \leq t$, and hence the upper bound

$$
E[f(S, D)] \leq B(\pi_S) = \sum_{d \in \Omega} P(D = d) \min\left\{t, g^*(\pi_S, d)\right\}.
$$

Similarly, for $u \in S$ and $v \in \mathcal{V} \setminus \{0\} \setminus S$, we define $\pi_S - u$ and $\pi_S + v$ to be the permutations of $S \setminus \{u\}$ and $S \cup \{v\}$, respectively, where the order in π_S is preserved, and where $\pi_S + v$ is formed by inserting v in a position in π_S such that $B(\pi_S + v)$ is minimized.

3.2 The case of (1,2)-complete recourse

Assuming that the recourse is (1,2)-complete, the upper bound on $E[f(S_j, D)]$ can be written as $B(\pi_S) = P\left(\sum_{i \in S} D_i \leq q\right) \sum_{h=0}^{|S|-1} c_{v_h v_{h+1}} + \sum_{i=1}^{|S|-1} P(\mathcal{E}_i) g(\pi_S, i)$, where $\mathcal{E}_i$ denotes the event that v_i is the *feasible return point* on π_S minimizing

the saving $\delta_i = c_{0v_i} + c_{0v_{i+1}} - c_{v_i v_{i+1}}$. Note that $\mathcal{E}_0 = \mathcal{E}_{|S|}$, and that this event is identical to $\sum_{i \in S} D_i \leq q$. The computation of $B(\pi_S)$ states that the sequence $v_1, \ldots, v_{|S|}$ can be followed as planned if the demand of S does not exceed q; otherwise this sequence will have to be broken at suitable intermediate points i, each defining the last customer of a feasible vehicle route. For simplicity, we assume distinct savings on route π_S (a consistent ordering can be introduced to break ties). A customer $v_i \in S$ is a feasible return point for the demand d if $\sum_{j=1}^{i} d_{v_j} \leq q$ and $\sum_{j=i+1}^{|S|} d_{v_j} \leq q$.

In order to compute $B(\pi_S)$, we derive explicit formulae for $P(\mathcal{E}_i)$ for all $i = 1, \ldots, |S| - 1$. For $i_0 \in \{1, \ldots, |S|\}$, define the integers $\alpha, \beta \in \{1, \ldots, |S|\}$ and $0 = i_{-\alpha} < i_{-\alpha+1} < \cdots < i_{-1} < i_0 < i_1 < \cdots < i_\beta = |S|$ such that $\{i_{-\alpha}, \ldots, i_{-1}, i_1, \ldots, i_\beta\} = \{i = 0, \ldots, |S| : \delta_i < \delta_{i_0}\}$. Let $\mathcal{E}_i^-$ and $\mathcal{E}_i^+$ denote the events $\sum_{j=1}^{i} D_{v_j} \leq q$ and $\sum_{j=i+1}^{|S|} D_{v_j} \leq q$, respectively, and $\overline{\mathcal{E}}_i^-$ and $\overline{\mathcal{E}}_i^+$ their complements.

Proposition 1. $P(\mathcal{E}_{i_0}) = P(\mathcal{E}_{i_0}^-)P(\mathcal{E}_{i_0}^+) - P(\mathcal{E}_{i_{-1}}^+ \cap \mathcal{E}_{i_0}^-) - P(\mathcal{E}_{i_1}^- \cap \mathcal{E}_{i_0}^+) + P(\mathcal{E}_{i_{-1}}^+ \cap \mathcal{E}_{i_1}^-)$

Proof. Since $\mathcal{E}_{i_j}^+ \subseteq \mathcal{E}_{i_{j+1}}^+$ and $\mathcal{E}_{i_{j+1}}^- \subseteq \mathcal{E}_{i_j}^-$ $(j = -\alpha, \ldots, \beta - 1)$, we have

$$P(\mathcal{E}_{i_0}) = P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_0}^+ \cap \bigcap_{\substack{j=-\alpha \\ j \neq 0}}^{\beta} \left(\overline{\mathcal{E}}_{i_j}^- \cup \overline{\mathcal{E}}_{i_j}^+\right)\right)$$

$$= P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_0}^+ \cap \bigcap_{j=1}^{\alpha} \overline{\mathcal{E}}_{i_{-j}}^+ \cap \bigcap_{j=1}^{\beta} \overline{\mathcal{E}}_{i_j}^-\right)$$

$$= P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_0}^+ \cap \overline{\mathcal{E}}_{i_{-1}}^+ \cap \overline{\mathcal{E}}_{i_1}^-\right)$$

$$= P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_0}^+ \cap \overline{\mathcal{E}}_{i_1}^-\right) - P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_{-1}}^+ \cap \overline{\mathcal{E}}_{i_1}^-\right)$$

$$= P\left(\mathcal{E}_{i_0}^-\right) P\left(\mathcal{E}_{i_0}^+\right) - P\left(\mathcal{E}_{i_1}^- \cap \mathcal{E}_{i_0}^+\right) - P\left(\mathcal{E}_{i_0}^- \cap \mathcal{E}_{i_{-1}}^+\right) + P\left(\mathcal{E}_{i_{-1}}^+ \cap \mathcal{E}_{i_1}^-\right).$$

□

Note that if $\delta_i = \min\{\delta_j : j = 1, \ldots, |S| - 1\}$, that is v_i is the best return point in π_S apart from $v_{|S|}$, then Proposition 1 states that $P(\mathcal{E}_i) = P\left(\mathcal{E}_i^-\right) P\left(\mathcal{E}_i^+\right) - P\left(\sum_{i \in S} D_i \leq q\right)$.

4 Tabu search heuristic

Tabu search is a memory-based search strategy, originally proposed by Glover (1986), that guides the local search process beyond a local optimum. One way of achieving this is to keep track of recent moves or solutions made in the past. A tabu list records recently performed moves or visited solutions. Whenever the algorithm attempts to move to a solution forbidden by the tabu list, the move is banned. This rule forces other solutions to be explored. However, this feature is not strict, as it can be overridden when some *aspiration criterion* is satisfied. A common criterion is that the target function value be the best ever seen. If this is the case, it is obvious that this solution has never been encountered before.

4.1 Preprocessing

Typical districting problems usually consist of partitioning a large territory (composed of a number of units) into districts. In a standard political districting problem, it is rather straightforward to determine the adjacent units of a unit. However, our problem is not that simple, as we are partitioning customer locations (which are associated with points in the plane) into districts and the notion of adjacency is not well defined. As opposed to routing problems where any vertex of a route can be moved to another route, we will only move a vertex from a district to another if it lies close to the border between the two districts. The intent is to avoid creating disconnected districts or enclaves (i.e. districts within districts), which are unlikely to be optimal. Thus, vertices centrally located within a district should not be moved at all. In order to define vertex adjacency we construct a graph $\mathcal{G}' = (\mathcal{V}, \mathcal{A}')$ and declare vertex j adjacent to vertex i if $(i, j) \in \mathcal{A}'$. The arc set $\mathcal{A}'$ is constructed by comparing pairs of arcs in $\mathcal{A}$, and if they intersect, the most costly one is removed from the set. This process is described in Algorithm 1.

Algorithm 1 AdjacencyGraph

Require: $\mathcal{G} = (\mathcal{V}, \mathcal{A})$
  Set $\mathcal{A}' = \mathcal{A}$.
  for $\forall (i, j) \in \mathcal{A} : i > 0, j > 0$ **do**
    if $(i, j) \in \mathcal{A}'$ **then**
      for $\forall (u, v) \in \mathcal{A}' \setminus \{(i, j)\} : u > 0, v > 0$ **do**
        if (u, v) intersects (i, j) and $c_{uv} > c_{ij}$ **then**
          Set $\mathcal{A}' = \mathcal{A}' \setminus \{(u, v)\}$.
  Set $\mathcal{G}' = (\mathcal{V}, \mathcal{A}')$.
  return $\mathcal{G}'$

The resulting graph $\mathcal{G}'$ is a planar graph. An example of $\mathcal{G}'$ for a 50 vertex instance is provided in Figure 1. The circles represent customer locations, and a district is composed of a number of these locations and is a connected component of $\mathcal{G}'$. We refer to vertex j as a neighbor of vertex i if $(i, j) \in \mathcal{A}'$.

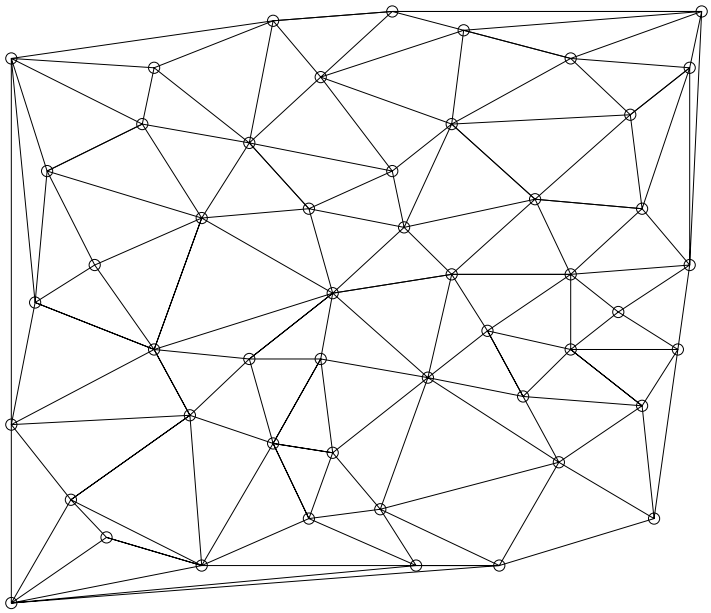

Figure 1: An adjacency graph.

Since $\mathcal{G}'$ is planar, a vertex with several neighbors in both district S_1 and district S_2 indicates that it lies close to the border between S_1 and S_2. Furthermore, if all neighbors $u \in S_1$ of a vertex $i \in S_1$ have at least two neighbors in $S_1 \setminus \{i\}$, which are themselves connected, it indicates that i can leave S_1 without disconnecting the district. These criteria are used to restrict the set of valid moves (see Section 4.4).

4.2 Testing feasibility

The definition of (k, ℓ)-complete recourse and the approximation for the cost of expected recourse, presented in Sections 2 and 3, are embedded within our algorithms to test the feasibility of a solution. Definition 1 is general and applies to k vehicles and ℓ routes. In our implementation, we only make use of (1,2)-complete recourse, i.e., only one vehicle is assigned to each district and each vehicle is allowed to at most one return trip to the depot. A permutation π_S of a district is feasible if

(i) condition (3) is satisfied, i.e., $\sum_{i \in S} E[D_i] \leq q$, and

(ii) for all $d \in \Omega$ either $\sum_{i \in S} d_i \leq q$, or there exists some $i_1 \in S$ such that $\sum_{j=1}^{i_1} d_{v_j} \leq q$, $\sum_{j=i_1+1}^{|S|} d_{v_j} \leq q$ and $g(\pi_S, i_1) \leq t$.

The computation of $E[f(S_j, D)]$ relies on the approximation discussed in Section 3.

4.3 Initial solution

When designing the initial solution (see Algorithm 2), we use the adjacency graph $\mathcal{G}' = (\mathcal{V}, \mathcal{A}')$.

Algorithm 2 InitialSolution

```
Set j = 0 and k = 1.
while j < n do
  i = seed(k)
  Assign vertex i to district k.
  Set j = j + 1, P = ∅ and ok = 1.
  while j < n and ok = 1 do
    Identify the excluded neighbors of vertex i, and denote this set as P_i.
    Set P = P ∪ P_i.
    if P ≠ ∅ then
      Extract a vertex i from P
      if assigning vertex i to district k violates the capacity or travel length
      constraints then
        Set ok = 0.
      else
        Assign vertex i to district k, and set j = j + 1.
    else
      Set ok = 0.
  Set k = k + 1.
```

The criteria for choosing the seed vertex of each district are as follows:

- If it is the first district, then the seed is chosen to be $i \in \arg\max_{j \in \mathcal{V}}\{c_{0j}\}$.
- Otherwise, if a district cannot be expanded further due to capacity or maximum travel length restrictions, then the next district is initialized by a vertex determined as follows: consider the excluded neighbors of the last inserted vertex of the latest district, and choose the one with the least number of arcs to other excluded neighbors. If there is a tie, choose the one closest to the last inserted vertex.
- Otherwise, if the last inserted vertex has no excluded neighbors, select among all excluded vertices the vertex with lowest travel cost from the last inserted vertex.

A neighbor vertex to be included in a district is chosen as follows:

- If there is some vertex in $\mathcal{P}$ with no excluded neighbors, let i be the closest one to the last inserted vertex.

- Otherwise choose $i \in \mathcal{P}$ with $\max\{\xi_i\}_{i\in\mathcal{P}}$; break ties by choosing the one with the least number of excluded neighbors; in case of a tie, choose the one closest to the last inserted vertex.

In some cases the expansion of a district stops not because of capacity or maximum travel length restrictions, but because of the non-existence of excluded neighbors available for inclusion. Hence, these districts may be rather small in size. Thus, if the number of vertices in a district is at most three, the algorithm tries to eliminate this district by relocating each of its vertices to another district. See Section 4.9 for more details.

4.4 Neighborhoods

The neighborhoods of solution x are given by $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$. In $\mathcal{N}_1(x)$, solutions are obtained by moving a vertex i from its current district S_{l_1} to another district S_{l_2}, and the corresponding move is denoted by (i, l_1, l_2). The move (i, l_1, l_2) is valid if the following two conditions are satisfied:

1. Border condition: There exist arcs $(i, j_1) \in \mathcal{A}'$ and $(i, j_2) \in \mathcal{A}'$ where $j_1 \in S_{l_2}$, $j_2 \in S_{l_2}$ and $j_1 \neq j_2$.
2. Connectivity condition: Let $\mathcal{J} = \{j \in S_{l_1} : (i, j) \in \mathcal{A}'\}$. For all $u \in \mathcal{J}$ there exist vertices $v_1, v_2 \in S_{l_1} \backslash \{i\}$ where $v_1 \neq v_2$ and $(u, v_1), (u, v_2), (v_1, v_2) \in \mathcal{A}'$.

The border condition ensures that the move (i, l_1, l_2) is valid only if i is located closely to S_{l_2}, in the sense that i must be adjacent to at least two distinct vertices in S_{l_2}. Consider the subgraph of $\mathcal{G}'$ induced by S_{l_1}, as well as some embedding of it in the plane. Since i is adjacent to some vertex outside S_{l_1}, it follows from planarity of $\mathcal{G}'$ that it must be on the boundary of the embedding of the subgraph. Furthermore, the connectivity condidtion prohibits all moves that will leave some vertex with no more than one neighbor in S_{l_1}.

The second neighborhood $\mathcal{N}_2(x)$ consists of solutions obtained by swapping vertices i and j between their respective districts S_{l_1} and S_{l_2}. This move is denoted by (i, j, l_1, l_2). The move is valid only if both moves (i, l_1, l_2) and (j, l_2, l_1) are valid.

All valid moves are considered, and the best is chosen. The reason for using two neighborhoods is that relocation moves do not, in general, provide very good solutions (being limited to the feasible part of the solution space and being so restrictive as in our case). Swap moves are more powerful, and can identify solutions where relocation moves fail due to limitations induced by the constraints.

4.5 Recency-based memory and tabu tenure

To avoid cycling, whenever a move (i, l_1, l_2) or (i, j, l_1, l_2) is performed, any move that transfers vertex i back into district l_1 or vertex j back into district l_2, is

declared tabu for θ iterations, where θ is a user-defined parameter. A forbidden move may still be chosen if it yields the best solution encountered so far (i.e. aspiration criterion).

The tabu list is represented by a two-dimensional array with the first index representing districts and the second representing vertices. For each such pair, the array contains the number of iterations to expiry of the tabu status of the corresponding move.

4.6 Frequency-based memory

A simple way of achieving diversification in search is to penalize frequently made moves by adding a penalty term to the objective function value. This will induce the search to explore a wider part of the solution space. Let $E[C(x)]$ be the expected cost of solution x. For any solution $\overline{x} \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$, whenever $E[C(\overline{x})] \geq E[C(x)]$, a penalty is added to $E[C(\overline{x})]$. We only penalize non-improving moves since improving ones have to be encouraged. The penalty $\psi(\overline{x})$ is defined as $\lambda E[C(\overline{x})]\sqrt{nm'}\phi$. The factor ϕ is the most important one as it represents the frequency of a move made in the previous iterations. For relocation moves, the frequency factor $\phi(i, l_2)$ is the number of times vertex i has been moved to district l_2 from some other district. For exchange moves, the frequency factor $\phi(i, j, l_1, l_2)$ is defined as $[(\phi(i, l_2) + \phi(j, l_1))/2]$, where $[y]$ is the nearest integer to y. The factor λ is a user-defined parameter which controls the intensity of diversification, and m' denotes the number of non-empty districts in solution $\overline{x}$. The third factor $\sqrt{nm'}$ is related to the size of the problem; using a square-root factor seems to somehow compensate for the problem size. The use of such a factor was first suggested by Taillard (1993), and has since been employed in other work (see e.g. Cordeau et al., 1997; Cordeau et al., 2001; Ho & Gendreau, 2005).

4.7 Search process

The tabu search heuristic starts with the initial solution described in Section 4.3. The district optimization procedure (see Section 4.8) and the district elimination procedure (see Section 4.9) are applied to this solution. At each iteration, it selects a solution $\overline{x} \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$ minimizing $E[C(\overline{x})] + \psi(\overline{x})$, which is non-tabu or satisfies the aspiration criterion which is the best solution encountered so far. At every κ_{TS} iterations the district optimization procedure is performed, followed by the district elimination procedure. The tabu search process terminates after γ iterations and the best solution found during the search is the final solution. This process is summarized in Algorithm 3.

4.8 District optimization

Since $E[C(\overline{x})]$ is computed with respect to the current ordering of vertices in the districts, it is essential that this ordering be as good as possible. When a vertex is removed from a district or inserted into a district, no local reoptimization of the

sequence of vertices is made. Hence, every κ_{TS} iterations, 2-opt is performed for each district. This procedure consists of replacing two arcs $(i, i+1)$ and $(j, j+1)$ with the two other arcs (i, j) and $(i+1, j+1)$. All valid combinations of replacements are considered, and the best improving one among those is chosen. This is repeated until no improvement can be achieved.

4.9 District elimination

As mentioned in Section 4.3 some districts may be rather small in size, and an attempt will be made to eliminate these. Hence, every κ_{TS} iterations, for districts of size at most three, relocation moves are performed. Every vertex $v_1, \ldots, v_{|S|}$ is removed from its current district S_{l_1} and reinserted into a new district S_{l_2}, if v_i has at least two neighbors belonging to district S_{l_2} and the capacity and travel length constraints are not violated.

Algorithm 3 TABU SEARCH heuristic

Require: x
  Set $x^* = x$.
  for $i = 1, \ldots, \gamma$ **do**
    Select a solution $\overline{x} \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$ that minimizes $E[C(\overline{x})] + \psi(\overline{x})$ and is non-tabu or satisfies the aspiration criterion.
    if no solution found **then**
      return x^*
    if $E[C(\overline{x})] < E[C(x^*)]$ **then**
      Set $x^* = \overline{x}$.
    Set the reverse move tabu for θ iterations.
    Set $x = \overline{x}$.
    if $\mod(i, \kappa_{TS}) = 0$ **then**
      $r =$ `DistrictElimination`(x)
      $r =$ `DistrictOptimization`(r)
      if $E[C(r)] < E[C(x^*)]$ **then**
        Set $x^* = r$.
      Set $x = r$.
  return x^*

5 Multistart heuristic

In this section we introduce a multistart heuristic and apply it to the districting problem. This approach is a two-phase method, where in each iteration a randomized initial solution is constructed before local search is applied to it. The best solution obtained from all of the iterations is returned as the best overall solution. The multistart heuristic terminates after τ iterations. Its description is given in Algorithm 4.

Algorithm 4 MULTISTART heuristic

Set $E[C(x^*)] = \infty$.
for $i = 1, \ldots, \tau$ **do**
  $x =$ `RandomizedBuild`()
  $x =$ `LocalSearch`(x)
  if $E[C(x)] < E[C(x^*)]$ **then**
    Set $x^* = x$.
return x^*

5.1 Randomized heuristic

This heuristic follows the guidelines provided in Section 4.3. The only difference lies in choosing the seed vertex of the first district. Instead of choosing the seed vertex farthest away from the depot, we employ some randomness when making the decision: randomly select ν vertices, narrow the selection down to the μ vertices farthest away from the depot, and choose the seed vertex at random.

5.2 Local search

The local search phase works by steepest descent and is based on relocating and swapping border vertices between their respective districts. All valid relocations and swaps are considered, and the best among those is chosen. Its outline is given in Algorithm 5. It is run until it reaches a local optimum. The procedures of optimizing the district and elimination of small districts are also performed every κ_M iterations.

Algorithm 5 LocalSearch

Require: x
Set $x^* = x$.
Set $improvement = 1$ and $i = 1$.
while $improvement = 1$ **do**
  Select a solution $\overline{x} \in \mathcal{N}_1(x) \cup \mathcal{N}_2(x)$ that minimizes $E[C(\overline{x})]$.
  if $E[C(\overline{x})] < E[C(x^*)]$ **then**
    Set $x^* = \overline{x}$ and $x = \overline{x}$.
    if $\mod (i, \kappa_M) = 0$ **then**
      $r =$ `DistrictElimination`(x)
      $r =$ `DistrictOptimization`(r)
      if $E[C(\overline{r})] < E[C(x^*)]$ **then**
        Set $x^* = r$.
      Set $x = r$.
    Set $i = i + 1$.
  else
    Set $improvement = 0$.
return x^*

6 Computational experiments

To our knowledge, no benchmark instances exist for our problem. We have therefore used some of the benchmark instances designed for the *Vehicle Routing Problem* by Christofides & Eilon (1969) and Christofides et al. (1979), as well as some of the benchmark instances designed for the *Vehicle Routing Problem with Time Windows* (VRPTW) by Solomon (1987). However, only the depot and customer locations are used. A lower bound a, and an upper bound b, are first given as fractions of the vehicle capacity for the demands. For customer v, a lower bound $\underline{d}_v$ is chosen in the interval $[aq, bq]$, and an upper bound $\overline{d}_v$, is chosen in the interval $[\underline{d}_v, bq]$, where $a = 0.05$ and $b = 0.1$. The characteristics of the instances used in this experimentation are given in Table 1. For each problem instance, the table indicates the number of customers (n), the number of districts (m), the vehicle capacity (q) and the route maximum length (t). The table also gives for each problem instance the reference (*Ref.*) to where the coordinates of customer locations can be found (e.g. pr1 (VRP) refers to VRP problem instance 1, while r101.50 (VRPTW) refers to the 50 first customers of VRPTW benchmark instance r101).

Table 1: Characteristics of the benchmark instances used for computational experiments

Problem	n	m	q	t	Ref.
pr1a	50	5	160	150	pr1 (VRP)
pr1b	50	5	160	200	pr1 (VRP)
pr1c	50	5	160	250	pr1 (VRP)
pr2a	75	8	140	150	pr2 (VRP)
pr2b	75	8	140	200	pr2 (VRP)
pr2c	75	8	140	250	pr2 (VRP)
pr3a	100	12	160	150	pr3 (VRP)
pr3b	100	12	160	200	pr3 (VRP)
pr3c	100	12	160	250	pr3 (VRP)
pr4a	50	6	160	150	r101.50 (VRPTW)
pr4b	50	6	160	200	r101.50 (VRPTW)
pr4c	50	6	160	250	r101.50 (VRPTW)
pr5a	75	8	140	150	r101.75 (VRPTW)
pr5b	75	8	140	200	r101.75 (VRPTW)
pr5c	75	8	140	250	r101.75 (VRPTW)

In our experiments we assume that for all $v = 1, \ldots, n$, $D_v - \underline{d}_v$ has a binomial distribution $B(\overline{d}_v - \underline{d}_v, p)$. This is true when the demand is composed of a set of at most $\overline{d}_v$ identical items, $\underline{d}_v$ of which are always demanded, and each of the remaining are demanded independently with probability p. We thus have $P(D_v = d) = p^{d-\underline{d}_v}(1-p)^{\overline{d}_v - d} \begin{pmatrix} \overline{d}_v - \underline{d}_v \\ d - \underline{d}_v \end{pmatrix}$, where $p \in [0, 1]$.

6.1 Sensitivity analyses

In this section we present sensitivity analyses on the various parameters of the two algorithms. The heuristics were coded in C++ and all experiments were carried out on a Pentium 4, 2.53 GHz computer.

6.1.1 Tabu search

Based on results from the literature (see, e.g. Cordeau & Laporte, 2003; Ho & Gendreau, 2005), we have first set $\kappa_{TS} = 10$ and $\lambda = 0.015$. Preliminary testing indicated that good results are usually obtained early in the search, thus, the value of γ was set equal to 1000. Running the algorithm for more iterations does not seem to produce better solutions. Sensitivity analyses on the parameters were performed sequentially, leaving the remaining parameters unchanged, and using the following order: θ, λ, κ_{TS}. We believe the tabu tenure θ is the most sensitive and difficult parameter to analyze, therefore it has the highest priority.

Parameter θ We fixed the value of the other parameters, and ran tests on all problem instances using different values of θ in the intervals $[1, m]$, $[1\log_{10} n, 10\log_{10} n]$ and $[1\log_{10} m, 10\log_{10} m]$. The results show that setting θ equal to m produces good results. Similar observations are also obtained with values of θ in the intervals $[3\log_{10} n, 5.5\log_{10} n]$ and $[8\log_{10} m, 10\log_{10} m]$. We have also performed experiments on random values of θ. Moderate results were obtained using tabu tenures randomly chosen in the intervals $[2.5\log_{10} n, 5\log_{10} n]$ and $[5\log_{10} m, 7.5\log_{10} m]$. In our case, the results do not seem to favor the use of random values of θ. The most appropriate value seems to be m, thus the value of θ has been set equal to m.

Parameter λ Having set θ equal to m, we then let parameter λ vary in the interval $[0.001, 0.05]$. The average cost of the solutions seems insensitive to modifications of λ in the interval $[0.004, 0.02]$, which also produced the best average results, thus we let the value of λ be equal to 0.015 (the original value).

Parameter κ_{TS} We let this parameter vary in the interval $[1, 50]$. Using too small a value of κ_{TS} seems to overdo the intensification process, whereas using too large a value seems to dilute the effect of the desired intensification. Results indicate that setting κ_{TS} equal to values in the interval $[10, 20]$ yields promising results, with 10 being the most appropriate value.

The results for TABU SEARCH were obtained with $\kappa_{TS} = 10$, $\gamma = 1000$, $\theta = m$ and $\lambda = 0.015$.

6.1.2 Multistart

As local search will terminate much earlier than tabu search, κ_M is set relatively small compared to κ_{TS}, thus κ_M is set equal to 5. Sensitivity analyses on

the parameters were performed sequentially, leaving the remaining parameters unchanged, and using the following order: (ν, μ), κ_M. The first two parameters were tested jointly.

Parameters ν and μ The choice of the seed vertex of the first district of Section 5.1 obviously depends on the values of parameters ν and μ. We let parameter ν vary in the interval $[0.1n, 0.3n]$ and parameter μ $[0.3\nu, 0.5\nu]$. It seems that giving the algorithm too much freedom of choosing the seed vertex does not always yield good solutions. We noticed promising results are obtained by setting ν equal to $0.1n$ and μ equal to 0.3ν.

Parameter κ_M We let this parameter vary in the interval $[1, 10]$. Using values from the lower half of the interval yields good results. The most appropriate value for κ_M seems to be 5.

For MULTISTART the parameters were $\kappa_M = 5$, $\nu = [0.1n]$ (where $[y]$ is the nearest integer to y), $\mu = [0.3\nu]$ and τ determined such that the heuristics consume an approximately equal amount of computing time (see column "CPU" in Table 2).

6.2 Results

Table 2 shows the results of both the initial and final solutions of TABU SEARCH. The percentage deviation from the best (i.e., the best result of any of the two heuristics) is given within the parentheses of each heuristic. Since no optimal solutions are known for any of these problem instances, using the best solutions obtained by the two proposed methods as benchmarks seems reasonable. The columns labeled m' are the numbers of non-empty districts created for the initial and the final solutions, respectively. The column labeled "return" is the number of districts that require a return trip to the depot. The next column labeled "CPU" is the total computing time in minutes needed to perform the search. Sometimes it is difficult to find a feasible solution, thus there may be situations where the search terminates before the total number of iterations reaches γ. The rightmost column labeled "itr" gives the number of iterations performed by TABU SEARCH.

Fixing all other data, it is expected that the optimal cost decreases with increasing value of t. Within each case set (pr1a-pr1c, pr2a-pr2c, etc.), t is the only unfixed input parameter, and the results are as expected for all sets but pr3a-pr3c. The reason for this may be that the search is restrictive in the sense that it is hard to move vertices around without violating the constraints.

Table 3 shows the final results obtained by the second method, MULTISTART. The rightmost column labeled τ is the number of iterations made by the heuristic during the time provided by TABU SEARCH. This number provides us with the number of local minima encountered. As with Table 2, Table 3 also shows that results improve with an increasing value of t, but this is not the case for instances pr3a-pr3c. Solutions to instances pr3b and pr3c have the same number

Table 2: TABU SEARCH heuristic

Problem	Initial solution		Final solution					
	$E[C(x)]$	m'	$E[C(x^*)]$		m'	return	CPU	itr
pr1a	542.01	5	525.30	(0.00)	5	0	6.08	1000
pr1b	536.79	5	522.98	(0.00)	5	0	10.01	1000
pr1c	536.79	5	522.98	(0.00)	5	0	10.30	1000
pr2a	834.27	8	705.76	(0.00)	7	0	0.89	213
pr2b	820.47	8	675.55	(0.00)	7	2	8.66	1000
pr2c	820.47	8	675.55	(0.00)	7	2	8.76	1000
pr3a	1153.45	12	913.57	(0.00)	10	2	5.31	621
pr3b	1032.81	10	904.55	(0.00)	10	2	11.37	1000
pr3c	1031.55	9	923.42	(0.00)	9	3	1.03	88
pr4a	692.12	6	567.24	(0.00)	6	0	3.70	1000
pr4b	611.57	5	526.79	(0.00)	5	1	2.18	504
pr4c	611.57	5	526.79	(0.00)	5	1	2.49	504
pr5a	870.95	8	787.28	(0.94)	8	0	7.63	1000
pr5b	891.16	8	726.65	(0.00)	7	4	4.73	607
pr5c	853.24	7	708.64	(0.00)	7	1	10.10	1000
Avg.	789.28	7.27	680.87	(0.06)	6.87	1.20	6.22	769.13

of districts, but the former one has a lower expected cost. The reason for this might be that not many local minima were created in the case of pr3c.

Table 3: MULTISTART heuristic

Problem	Final solution					
	$E[C(x^*)]$		m'	return	CPU	τ
pr1a	526.83	(0.29)	5	1	6.10	125
pr1b	526.02	(0.58)	5	1	10.04	162
pr1c	526.02	(0.58)	5	1	10.31	161
pr2a	764.64	(8.34)	8	0	0.92	9
pr2b	733.27	(8.54)	7	3	8.78	75
pr2c	720.38	(6.64)	7	1	8.79	80
pr3a	947.83	(3.75)	10	1	5.37	23
pr3b	917.72	(1.46)	9	3	11.48	59
pr3c	942.98	(2.12)	9	4	1.36	7
pr4a	574.18	(1.22)	5	0	3.72	108
pr4b	564.97	(7.25)	5	1	2.24	33
pr4c	564.97	(7.25)	5	1	2.59	36
pr5a	779.91	(0.00)	8	1	7.70	75
pr5b	752.83	(3.60)	7	4	4.80	34
pr5c	752.83	(6.24)	7	4	10.17	74
Avg.	706.36	(3.86)	6.80	1.73	6.29	70.73

TABU SEARCH produces better results than MULTISTART in every problem instance, except for pr5a. Solutions by TABU SEARCH have a smaller expected cost and a smaller number of planned returned trips to the depot than MULTISTART, while MULTISTART creates solutions with fewer districts. The superiority of TABU SEARCH is also statistically significant. The Wilcoxon signed-rank test has been applied to compare the two heuristics. Following the guidelines provided by Golden & Stewart (1985), the null hypothesis is that both heuristics are equally good, while the alternative hypothesis is that TABU SEARCH is better than MULTISTART. Using a Wilcoxon signed-rank test, the null hypothesis can be rejected at a significance level as low as 0.001.

Both algorithms use diversification mechanisms in the search, but these are employed in different ways. In TABU SEARCH, diversification is achieved by penalizing frequently made moves, which is a long-term memory strategy for TABU SEARCH, while in MULTISTART, different starting points are constructed to explore the solution space more extensively. The results of our experiments indicate that a search process that remembers its past history is capable of generating better solutions than in a search with no memory.

Figure 2 shows the district boundaries before and after performing TABU

SEARCH for problem instance pr1b.

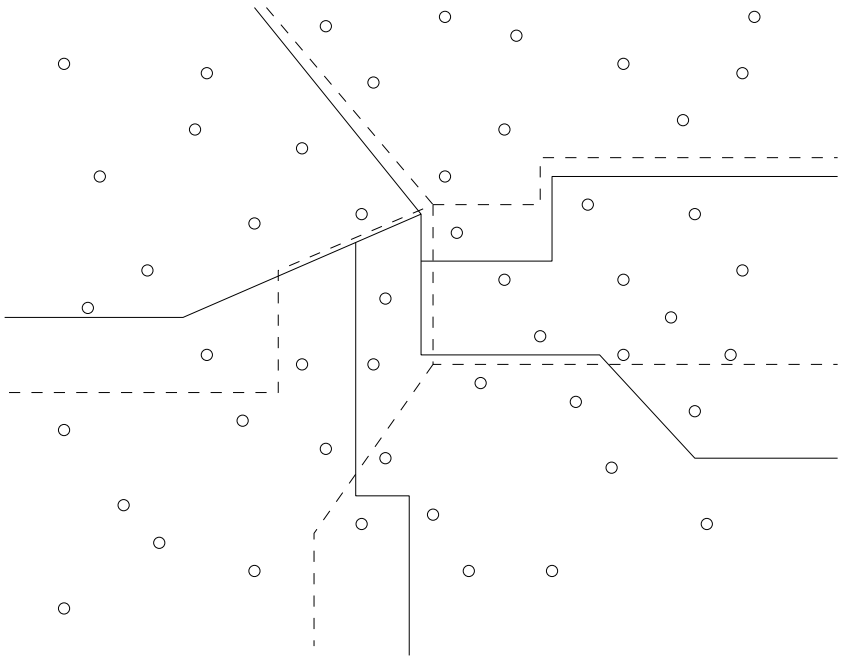

Figure 2: Solution to problem pr1b (TABU SEARCH). The dotted lines correspond to the initial solution and the full lines correspond to the final solution.

In order to investigate the computation time of TABU SEARCH when applied to larger cases, we also solved an instance with 800 customers and 76 vehicles. It took the algorithm 182.45 CPU-minutes to solve this instance.

7 Conclusion

We have introduced and formulated the problem of designing districts for vehicle routing problems with stochastic demands. We have also developed a tabu search heuristic and a multistart heuristic for the problem. Computational results show that tabu search performs better than multistart.

Acknowledgements

This work was supported by the Research Council of Norway under grant 127533/432 and by the Canadian Natural Sciences and Engineering Research Council under grant OGP0039682. This support is gratefully acknowledged.

Thanks are also due to the three anonymous referees for their valuable comments.

References

Blais, M., Lapierre, S. D. & Laporte, G. (2003). Solving a home care districting problem in an urban setting. *Journal of the Operational Research Society* **54**(11), 1141–1147.

Bozkaya, B., Erkut, E. & Laporte, G. (2003). A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research* **144**(1), 12–26.

Christofides, N. & Eilon, S. (1969). An algorithm for the vehicle dispatching problem. *Operational Research Quarterly* **20**(3), 309–318.

Christofides, N., Mingozzi, A. & Toth, P. (1979). The vehicle routing problem. *in* N. Christofides, A. Mingozzi, P. Toth & C. Sandi, eds, 'Combinatorial Optimization'. Wiley. Chichester. pp. 315–338.

Cordeau, J.-F. & Laporte, G. (2003). A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research B* **37**(6), 579–594.

Cordeau, J.-F., Gendreau, M. & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **30**(2), 105–119.

Cordeau, J.-F., Laporte, G. & Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* **52**(8), 928–936.

D'Amico, S. J., Wang, S.-J., Batta, R. & Rump, C. M. (2002). A simulated annealing approach to police district design. *Computers & Operations Research* **29**(6), 667–684.

Drexl, A. & Haase, K. (1999). Fast approximation methods for sales force deployment. *Management Science* **45**(10), 1307–1323.

Dror, M., Laporte, G. & Trudeau, P. (1989). Vehicle routing with stochastic demands: Properties and solution frameworks. *Transportation Science* **23**(3), 166–176.

Ferland, J. A. & Guénette, G. (1990). Decision support system for a school districting problem. *Operations Research* **38**(1), 15–21.

Fleischmann, B. & Paraschis, J. (1988). Solving a large scale districting problem: A case report. *Computers & Operations Research* **15**(6), 521–533.

Gendreau, M., Laporte, G. & Potvin, J.-Y. (2002). Metaheuristics for the capacitated VRP. *in* P. Toth & D. Vigo, eds, 'The Vehicle Routing Problem'. SIAM Society for Industrial and Applied Mathematics. pp. 129–154.

Gendreau, M., Laporte, G. & Séguin, R. (1996). Stochastic vehicle routing. *European Journal of Operational Research* **88**(1), 3–12.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* **13**(5), 533–549.

Golden, B. L. & Stewart, W. R. (1985). Empirical analysis of heuristics. *in* E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan & D. B. Shmoys, eds, 'The Traveling Salesman Problem'. Wiley, Chichester. pp. 207–249.

Ho, S. C. & Gendreau, M. (2005). Path relinking for the vehicle routing problem. *Journal of Heuristics*. Forthcoming.

Kall, P. & Wallace, S. (1994). *Stochastic Programming*. Wiley. Chichester.

Laporte, G. & Semet, F. (2002). Classical heuristics for the capacitated VRP. *in* P. Toth & D. Vigo, eds, 'The Vehicle Routing Problem'. SIAM Society for Industrial and Applied Mathematics. pp. 109–128.

Pezzella, F., Bonanno, R. & Nicoletti, B. (1981). A system approach to the optimal health care districting. *European Journal of Operational Research* **8**(2), 139–146.

Skiera, B. & Albers, S. (1998). Costa: Contribution optimizing sales territory alignment. *Marketing Science* **17**(3), 196–213.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* **35**(2), 254–265.

Taillard, É. D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks* **23**(8), 661–673.