TRANSPORTATION SCIENCE
Vol. 42, No. 1, February 2008, pp. 22–31
ISSN 0041-1655 | EISSN 1526-5447 | 08 | 4201 | 0022

informs

DOI 10.1287/trsc.1070.0204

An Optimization-Based Heuristic for the Split Delivery Vehicle Routing Problem

Claudia Archetti, M. Grazia Speranza
Department of Quantitative Methods, University of Brescia, 25122 Brescia, Italy
{archetti@eco.unibs.it, speranza@eco.unibs.it}

Martin W. P. Savelsbergh
School of Industrial and Systems Engineering, Georgia Institute of Technology,
Atlanta, Georgia 30332, mwps@isye.gatech.edu

The split delivery vehicle routing problem is concerned with serving the demand of a set of customers with a fleet of capacitated vehicles at minimum cost. Contrary to what is assumed in the classical vehicle routing problem, a customer can be served by more than one vehicle, if convenient. We present a solution approach that integrates heuristic search with optimization by using an integer program to explore promising parts of the search space identified by a tabu search heuristic. Computational results show that the method improves the solution of the tabu search in all but one instance of a large test set.

Key words: vehicle routing; split delivery; tabu search; integer programming
History: Received: December 2005; revision received: January 2007; accepted: April 2007.

1. Introduction

In vehicle routing problems (VRPs) a set of customers needs to be served and a fleet of capacitated vehicles is available to serve them. The objective is to minimize costs, which usually means minimizing the total distance traveled. In most VRPs it is assumed that the demand of a customer is less than or equal to the capacity of a vehicle and that each customer has to be served by exactly one vehicle; i.e., there is a single-visit assumption. It is obvious that when a customer's demand exceeds the vehicle capacity, it is necessary to visit that customer more than once, but it requires only a little more thought to see that even when all customer demands are less than or equal to the vehicle capacity, it may be beneficial to use more than one vehicle to serve a customer. In the split delivery vehicle routing problem (SDVRP) the single-visit assumption is relaxed and each customer may be served by more than one vehicle.

Although the SDVRP has received little attention in the past compared to other variants of the VRP, it has recently been studied by a number of researchers. The SDVRP was introduced by Dror and Trudeau (1989, 1990), who defined the problem, derived some structural properties, and proposed a local search heuristic. An integer programming formulation was presented and valid inequalities were derived by Dror, Laporte, and Trudeau (1994), and real-life applications validating the interest in the SDVRP were discussed by Mullaseril, Dror, and Leung (1997) and Sierksma and Tijssen (1998). A special case was formulated and heuristically solved by Frizzell and Giffin (1995). The computational complexity of the SDVRP was analyzed by Archetti, Mansini, and Speranza (2005). A tight bound on the cost reduction that can be obtained by allowing split deliveries was given by Archetti, Savelsbergh, and Speranza (2006). A lower bound was proposed and computationally tested by Belenguer, Martinez, and Mota (2000). Exact solution approaches were investigated by Gueguen (1999) and Gendreau et al. (2006). In the latter, customers have delivery time windows. A tabu search algorithm was proposed and evaluated by Archetti, Hertz, and Speranza (2006).

We present a solution approach that integrates heuristic search with optimization. The proposed approach is based on two main ideas. The first is to use the information provided by a tabu search heuristic to identify parts of the solution space that most likely contain good solutions. The second idea is to explore this part of the solution space by means of a suitable integer programming model. The computational results we have obtained are encouraging and validate the interest in nontraditional uses of integer programming. The proposed optimization-based heuristic was able to improve the solution produced by the tabu heuristic in all but one instance in our test set. Recently, Chen, Golden, and Wasil (2007) also proposed a solution approach for the SDVRP that incorporates heuristic as well as integer programming components.

The use of integer programming (or other optimization techniques) as a component of heuristic solution approaches is an increasingly popular technique.

De Franceschi, Fischetti, and Toth (2006) present a superb example. A well-known and useful improvement technique for VRPs is to remove and optimally reinsert a customer. De Franceschi, Fischetti, and Toth convert this idea into a much more powerful improvement technique by simultaneously extracting sequences of customers and reinserting derived sequences, where derived sequences are combinations of extracted sequences. To be able to do this, a small set partitioning problem has to be solved in each improvement step. Espinoza et al. (2008a, 2008b) develop a parallel local search scheme for solving a per-seat, on-demand air transportation problem. Because passengers are not allowed to change planes, a partial flight schedule involving a subset of planes can be viewed as a small instance of the problem, for which optimal or near optimal solutions can be found using a customized integer multicommodity flow solver. What sets these methods apart from earlier work on embedding optimization in heuristic solution approaches is the willingness to embed difficult NP-hard problems in a heuristic solution scheme.

The paper is organized as follows. In §2 the SDVRP is introduced. In §3 an integer programming model for the SDVRP is presented. In §4 the optimization-based solution approach is described, and, finally, in §5 computational results are shown and discussed.

2. The SDVRP

In the SDVRP, a set of customers C has to be served by a fleet M of capacitated vehicles. Each vehicle $v \in M$ has capacity Q and has to start and finish its tour at the depot, which we denote by 0. Each customer $i \in C$ has demand d_i, which can be less than, equal to, or greater than the vehicle capacity Q. A customer may be visited more than once. The cost to travel between locations i and j is c_{ij}. We assume that the costs c_{ij} satisfy the triangle inequality. The objective is to serve customers' demands at minimum cost.

At present, the tabu search heuristic developed by Archetti, Hertz, and Speranza (2006) is the method of choice for solving SDVRP instances. Because it is one of the components of the solution approach discussed in this paper, we briefly describe it here. Given a solution s, the tabu search explores

- Neighboring solutions obtained by removing a customer i from all routes in s that visit i and introducing i into a route in s that has sufficient residual capacity to accommodate i or into a new route by itself
- Neighboring solutions obtained by moving a portion of the demand of a customer to another route.

The tabu search selects the best solution among nontabu neighbors (except when a tabu neighbor exists that improves the current best solution), updates the current solution, and continues until a maximum number of iterations without improvement is reached.

3. An Integer Programming Model

We present a route-based model for the SDVRP. A similar model has been independently developed and presented in Gendreau et al. (2006) for the SDVRP with time windows. Let R represent a set of routes, and let c_r denote the cost of route r. The set of routes R may contain all feasible routes or some restricted set of the feasible routes. The model chooses a minimum-cost subset of routes in R and decides on the quantities to be delivered to the customers on the selected routes. The formulation has two sets of variables. The variable x_r represents the number of times a route is executed in an optimal solution. For routes that visit more than one customer, the variable x_r can be assumed to be binary, as it is never optimal to execute such a route more than once. However, when a route visits a single customer, the variable x_r has to be a nonnegative integer to accommodate situations in which there are customers with demand greater than the vehicle capacity. The continuous variable y_r^i represents the quantity delivered to customer i on route r. Let $C(r)$ denote the set of customers visited on route r. The integer programming model is presented below.

$$\min \sum_{r \in R} c_r x_r \tag{1}$$

$$\sum_{i \in C(r)} y_r^i \le Q x_r \quad r \in R, \tag{2}$$

$$\sum_{r \in R:\, i \in C(r)} y_r^i \ge d_i \quad i \in C, \tag{3}$$

$$x_r \in \{0, 1\} \quad r \in R \text{ and } |C(r)| > 1, \tag{4}$$

$$x_r \in \mathbb{Z}_+ \quad r \in R \text{ and } |C(r)| = 1, \tag{5}$$

$$y_r^i \ge 0 \quad r \in R,\ i \in C. \tag{6}$$

The objective function (1) minimizes the total cost of the selected routes. Constraints (2) impose that a delivery to a customer i on route r can only take place if route r is selected and that the total quantity delivered on a selected route cannot exceed the vehicle capacity. Constraints (3) ensure that the demand d_i of customer i is completely satisfied.

The formulation presented above needs to be strengthened in a variety of ways to make it computationally effective.

First, we observe that it is possible that the total quantity that can be delivered on route r, i.e., $\sum_{i \in r} d_i$, may be less than the vehicle capacity. Therefore, constraints (2) can be strengthened to

$$\sum_{i \in r} y_r^i \le \min\left(\sum_{i \in r} d_i, Q\right) x_r \quad r \in R. \tag{7}$$

These constraints can in turn be disaggregated to give

$$y_r^i \le d_i x_r \quad r \in R,\ i \in C. \tag{8}$$

Next, we observe that the minimum number of routes necessary to serve all the customers is $\lceil \sum_i d_i / Q \rceil$. Thus we have the following inequality:

$$\sum_{r \in R} x_r \geq \left\lceil \frac{\sum_i d_i}{Q} \right\rceil. \tag{9}$$

The next set of inequalities is inspired by the fact that there exists an optimal SDVRP solution where no two routes have more than one customer with a split delivery in common (Dror and Trudeau 1990). We include the following inequalities:

$$\sum_{r \in R: \{i,j\} \in r} x_r \leq 1 \quad i, j \in C. \tag{10}$$

For each pair of customers, the inequalities enforce that there is at most one route in the solution that visits both of them. This is more restrictive than what is implied by the property shown in Dror and Trudeau (1990), because we enforce the constraint even if the customers are not split. However, these inequalities have proven to be effective in speeding up the solution of the integer program; therefore, we decided to keep them, even though they may cut off some feasible solutions.

Finally, as each customer must be visited at least once, we can add the following redundant inequalities:

$$\sum_{r \in R: i \in r} x_r \geq 1 \quad i \in C. \tag{11}$$

Even though an optimal integer solution will always satisfy these inequalities, they may be violated by the linear programming relaxation. We do not introduce all inequalities (11)—only those relating to customers visited by at most three routes. This to avoid burdening the model with useless inequalities.

Even with all the strengthenings introduced above, the integer program remains difficult to solve. One of the main reasons is the fact that the quantity delivered to a customer on a route is not fixed, because the customer may be served by more than one vehicle, but the quantity has to be determined by the optimization. However, for many instances it may be possible to identify a set of customers C' who are very unlikely to be served more than once in an optimal solution, e.g., isolated customers far away from the depot. We can improve solutions times, sometimes significantly, by enforcing that certain customers cannot be split. This can be done by changing constraints (11) to

$$\sum_{r \in R: i \in r} x_r \geq 1 \quad i \in C \setminus C' \tag{12}$$

and

$$\sum_{r \in R: i \in r} x_r = 1 \quad i \in C'. \tag{13}$$

In fact, because the set R will not contain all possible routes, it may be better to replace constraints (13) with conditions on the variables y^{i_r}, i.e.,

$$y^{i_r} = 0 \quad \text{or} \quad y^{i_r} = d_i \quad r \in R,\ i \in C'. \tag{14}$$

That is, the variables y^{i_r} become semicontinuous variables. Most state-of-the-art integer programming solvers have special features to effectively handle semicontinuous variables.

We will refer to the integer program given by (1), (3)–(6), (8)–(10), (12), and (14) as the *route-optimization integer program*. The key to its successful use is identifying sets R and C'.

4. A Solution Approach

One of the key ideas underlying our solution approach is that tabu search can identify parts of the solution space that are likely to contain high-quality solutions.

The simplest use of this idea is the identification of a set C' of customers that are likely to be served by a single vehicle in high-quality SDVRP solutions. If a customer is never, or rarely, split in the solutions encountered during the tabu search, we interpret this as an indication that it is likely that the customer will be served by a single vehicle in high-quality SDVRP solutions (and the customer therefore should be in the set C'). We have implemented this idea as follows. Let S denote the set of all SDVRP solutions encountered during the tabu search. For each customer i, we calculate the *node counter* n_i, the number of times customer i is split in the solutions in S, where we say that a customer is split $k-1$ times if the customer is served by k routes in a solution $s \in S$. Let $n_{\max} = \max_i n_i$. We include customer i in C' if $n_i < 0.1 \times n_{\max}$ and if i is not split in the final solution of the tabu search.

The use of this idea in the identification of the set R of promising routes is more involved. For each edge $\{i, j\}$, we calculate n_{ij}, the number of times edge $\{i, j\}$ appears in any of the routes of the solutions in S. We will refer to n_{ij} as the *edge counter* of edge $\{i, j\}$. Again, we interpret a large value n_{ij} as an indication that edge $\{i, j\}$ will likely be included in high-quality SDVRP solutions. The edge counters n_{ij} guide the construction of a set of *promising routes* $\bar{R}$. The set $\bar{R}$ is not used directly in the route optimization *IP*, because it is usually too large, but the route optimization *IP* is solved several times with subsets R of $\bar{R}$.

An overview of the proposed approach can be found in Algorithm 1.

Algorithm 1. Solution approach:
Calculate the node counters n_i for all $i \in C$ and determine C'.
Calculate the edge counters n_{ij} for all $\{i, j\}$.
Initialize the best known solution s^* with the solution produced by the tabu search.

Generate a set of promising routes $\bar{R}$ guided by the edge counters n_{ij}.
Sort the routes in $\bar{R}$ based on a desirability measure.
while a time limit has not been reached or a maximum number of *IP*s has not been solved **do**
  Select a subset of routes R of $\bar{R}$
  Solve the route optimization *IP* over the set R
  if the solution found by the route optimization *IP* improves s^*, **then**
    Update s^*.
  end if
end while

4.1. Route Generation

The set of *promising routes* $\bar{R}$ is generated by starting from a set B of base edges. For each edge $\{i, j\} \in B$, a set of routes $R'_{\{i,j\}}$ is generated. Let $R' = \bigcup_{\{i,j\} \in B} R'_{\{i,j\}}$ be the union of these sets of routes. Then a dominance checking procedure is performed on R' and the set $\bar{R}$ is obtained.

The set B of base edges includes those edges $\{i, j\}$ with an edge counter that is greater than or equal to a given percentage β_B of the maximum edge counter; i.e., $\{i, j\} \in B$ if and only if $n_{ij} \geq \beta_B \times \max_{\{i,j\}} n_{ij}$. For each base edge $\{i, j\}$ a set of routes $R'_{\{i,j\}}$ is generated. This is done as follows. We generate a set of paths P_i from customer i to the depot, none of which includes edge $\{i, j\}$, and a set of paths P_j from customer j to the depot, none of which includes edge $\{i, j\}$. We can take any path $p_i \in P_i$, the edge $\{i, j\}$, and any path $p_j \in P_j$ and convert these into a route. The route is inserted into the set $R'_{\{i,j\}}$ if the paths p_i and p_j do not have any vertex in common and the total demand of the route is not greater than $Q + \delta/2$, where δ is the average customer demand. The generation of the sets P_i and P_j is done using a scheme that resembles depth-first search and is controlled by the edge counters. Consider the generation of the paths in P_i. Let p denote the active path, i.e., the path under consideration. The first node of p is i and the last node of p is u.

Initially, path p contains no edges and the last node is equal to the first node i. We extend path p by examining the edges incident to u, i.e., edges $\{u, v\}$ with $v \notin p$. Whenever the edge counter of an edge $\{u, v\}$ is greater than some threshold L, v is the depot, or the edge $\{u, v\}$ belongs to at least one route of an improving solution found during the tabu search (where an improving solution is a solution that improved the current best solution), we extend path p with edge $\{u, v\}$. Consequently, we may not be able to extend p, we may be able to extend p with a single edge, or we may be able to extend p with several edges. The threshold L is calculated as a percentage of the maximum counter of an edge incident to u; i.e., $L = \beta_I \times \max_v n_{uv}$.

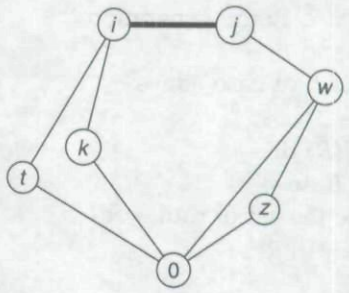

Figure 1 **Route Generation for Base Edge** $\{i, j\}$

When path p reaches the depot, i.e., path p is extended with the edge $\{u, depot\}$, path p is added to the set P_i. A similar scheme is used to generate the paths in P_j. An example is shown in Figure 1. Edge $\{i, j\}$ is the base edge and the other edges depicted are edges with an edge counter greater than threshold L and thus the only ones considered by the path generation procedure. The set P_i contains paths $\{i, t, 0\}$ and $\{i, k, 0\}$, and the set P_j contains paths $\{j, w, 0\}$ and $\{j, w, z, 0\}$. The set of routes $R'_{\{i,j\}}$ contains paths $\{0, t, i, j, w, 0\}$, $\{0, t, i, j, w, z, 0\}$, $\{0, k, i, j, w, 0\}$, and $\{0, k, i, j, w, z, 0\}$, provided that the demand of each route is not greater than $Q + \delta/2$.

Through a series of preliminary experiments we observed that the out-and-back routes from the depot to each customer often appear in high-quality solutions, especially in instances where customer demands are large; therefore, we added those routes to R'. Since R' can include duplicated routes or routes that visit the same customers in different orders, we eliminate from R' all dominated routes. That is, whenever two routes are found that visit the same set of customers, only the route with the smallest cost is kept. The set of *promising routes* $\bar{R}$ is the set of remaining routes.

Obviously, the number of promising routes depends on the values of the parameters β_B and β_I. The larger the values of the parameters are, the smaller the number of generated routes is. We observed that the "right" values of the parameters depend on the characteristics of the instance and that this makes the setting of the parameters difficult. For this reason, we decided to proceed by defining an acceptable range $[l(\bar{R}), u(\bar{R})]$ for the size of the set $\bar{R}$ and to iteratively change the values of the parameters until the size of the set $\bar{R}$ falls within this range.

We proceed as follows. We set initial values for β_I and β_B and generate a set of promising routes $\bar{R}$. If $|\bar{R}| < l(\bar{R})$, then we reduce the value of β_B by 0.05 and the value of β_I by 0.02. We repeat the generation as many times as necessary until the size of the set $\bar{R}$ is greater than or equal to $l(\bar{R})$. If $|\bar{R}| > u(\bar{R})$, then we randomly eliminate routes from $\bar{R}$ until its size is $u(\bar{R})$. An overview of the proposed route generation can be found in Algorithm 2.

ALGORITHM 2. Route generation:
Initialize β_B and β_I
Create the set B of base edges
$\bar{R} \leftarrow \varnothing$, $R' \leftarrow \varnothing$
while $|\bar{R}| \leq l(\bar{R})$ **do**
  for $\{i, j\} \in B$ **do**
    Generate the set of routes $R'_{\{i,j\}}$
    Add $R'_{\{i,j\}}$ to R'
  end for
  Eliminate dominated routes from R'. Let $\bar{R}$ be the set of remaining routes.
  if $|\bar{R}| < l(\bar{R})$ **do**
    $\beta_B \leftarrow \beta_B - 0.05$, $\beta_I \leftarrow \beta_I - 0.02$, and $\bar{R} \leftarrow \varnothing$.
  else
    if $|\bar{R}| > u(\bar{R})$ **then**
      Randomly eliminate $|\bar{R}| - u(\bar{R})$ routes from $\bar{R}$
    end if
  end if
end while

4.2. Route Sorting

To be successful, the set $\bar{R}$ of promising routes needs to contain routes that can be combined into high-quality, near-optimal solutions. Preliminary experiments have shown that to guarantee the existence of such high-quality solutions, $\bar{R}$ needs to contain a reasonably large number of routes. As it is impossible to solve the route-optimization model with the entire set of routes $\bar{R}$, we have to select "good" subsets of routes to pass to the route optimization *IP*.

The identification of good subsets is based on three ideas. First, we always include the routes of the best known solution. That way it is possible for the route optimization *IP* to improve just a portion of the best solution, i.e., to perform a local improvement. Second, we include the routes with a positive value in the solution to the linear programming relaxation of the route optimization *IP* over the entire set $\bar{R}$. As the linear programming relaxation considers the entire set of routes $\bar{R}$, it may be able to identify sets of complimentary routes from a global perspective. Finally, we include routes based on a desirability criterion. We tested various "desirability" criteria. Let λ_i denote the dual variables associated with constraints (3); then we define the following "desirability" criteria:

Criterion 1. Order the customers visited by route r by increasing values of λ_i. Let k be the first customer for which

$$\sum_{1 \leq i \leq k} d_{[i]} > Q,$$

where $[\cdot]$ gives the reordered sequence of customers. Change the demand of customer k to

$$d_{[k]} = Q - \sum_{1 \leq i \leq k-1} d_{[i]}.$$

Then the desirability of route r is calculated as:

$$c_r - \sum_{1 \leq i \leq k} d_{[i]} \lambda_{[i]}.$$

Criterion 2. The desirability of route r is calculated as

$$c_r - \sum_{i \in r} d_i \lambda_i.$$

Criterion 3: The desirability of route r is calculated as

$$c_r - \frac{\sum_{i \in r} d_i \lambda_i}{\sum_{i \in r} d_i}.$$

Criterion 4. Choose an equal number of routes with each of the previous criteria.

Once the criterion is chosen, the routes are ordered on the basis of an increasing "desirability" value.

To summarize, the set of routes R that feeds the route optimization *IP* consists of three sets of routes:

- the routes of the best known solution;
- the routes with positive values in the linear programming relaxation of the route optimization *IP* over the entire set $\bar{R}$;
- the routes selected with the desirability criterion.

4.3. Route-Optimization Phase

Let $r_{\max}$ be the maximum number of routes we allow in R, and let $IP_{\max}$ be the maximum number of *IP*s we want to solve. This parameter is set to ensure that the route-optimization *IP* will solve in a reasonably short time. The routes of the best known solution are always included in R. The first *IP* also includes the routes with positive values in the linear programming relaxation of the route-optimization *IP* over the entire set $\bar{R}$. These routes are complemented with desirable routes that have not been used before; i.e., starting from the first not-yet-used desirable route and following the order determined by the desirability criterion, routes are added until the maximum number of routes $r_{\max}$ is reached (or all the desirable routes have been used). An overview of the proposed route optimization can be found in Algorithm 3.

ALGORITHM 3. Route optimization:
$counter_{ip} = 1$.
while the elapsed time is less than $T_{\max}$ and $counter_{ip} \leq IP_{\max}$ **do**
  $R \leftarrow \varnothing$.
  Add the routes of best known solution to R.
  If $counter_{ip} = 1$, add the routes with a positive value in the LP relaxation to R.
  Add desirable routes to R until $|R| = r_{\max}$ (or no more desirable routes exist).
  Delete the selected desirable routes from the set of desirable routes.
  Solve the route-optimization *IP* over R.
  if the solution found improves the best known solution s^*, **then**

Update s^*.
end if
$counter_{ip} \leftarrow counter_{ip} + 1$.
end while

5. Computational Experiments

To evaluate the merits of the proposed optimization-based heuristic, we tested it on a set of 42 instances with varying demand characteristics and we compare the solution values to the results obtained with the tabu search algorithm of Archetti, Hertz, and Speranza (2006). Their tabu search algorithm stops after $400|C|$ iterations without improvements. Archetti, Hertz, and Speranza (2006) compared their tabu search algorithm with the local search algorithm of Dror and Trudeau (1990), which is, to the best of our knowledge, the only other algorithm proposed for the SDVRP. Because the tabu search algorithm *always* outperformed the local search algorithm of Dror and Trudeau, we compare our optimization-based heuristic with the tabu search algorithm only. Moreover, Archetti, Hertz, and Speranza (2006) evaluated the performance of the tabu search algorithm also by comparing its results to optimal solutions obtained for small instances (up to 15 vertices). The tabu search algorithm found optimal solutions for all tested instances in only a few seconds. These results suggest that the tabu search algorithm is quite powerful and that it may not be easy for our optimization-based heuristic to obtain even better results.

The 42 instances used for our tests are derived from seven basic instances; the same instances used to test the tabu search algorithm of Archetti, Hertz, and Speranza (2006). These basic instances vary in terms of the number of customers (ranging from 50 to 199) and in terms of vehicle capacity (ranging from 140 to 200). Five additional set of instances are created by changing the demand of the customers in the basic instances but keeping all other characteristics the same. Each of the new sets of instances is characterized by a lower bound on the demand at the customers, α, and by an upper bound on the demand at the customers, γ, expressed as a fraction of the vehicle capacity Q, i.e., $\alpha, \gamma \in [0, 1]$ with $\alpha \leq \gamma$. The demand d_i of customer i is set to

$$d_i = \lfloor \alpha Q + \delta(\gamma - \alpha)Q \rfloor$$

for some random δ in $[0, 1]$, i.e., the demand d_i of customer i is chosen randomly in the interval $[\alpha Q, \gamma Q]$. The five additional sets of instances are created with the following lower and upper bound combinations (α, γ): $(0.1, 0.3)$, $(0.1, 0.5)$, $(0.1, 0.9)$, $(0.3, 0.7)$, and $(0.7, 0.9)$ (following Dror and Trudeau 1989). A set of preliminary computational experiments was conducted to identify a set of reasonable choices for the parameters controlling the algorithm. On the basis of these experiments, we decided on the following values:

- The value of parameter β_B, which controls the selection of base edges, is set to 0.8 when $(\alpha, \gamma) = (0.3, 0.7)$ and $(0.7, 0.9)$ and to 0.5 in all other cases.
- The value of parameter β_I, which controls the selection of the edges considered for each vertex during path extension, is set to 0.25 when $(\alpha, \gamma) = (0.3, 0.7)$ and $(0.7, 0.9)$ and to 0.10 in all other cases.
- The minimum number of routes to be inserted in the LP, i.e., parameter $l(\bar{R})$, is set to 10,000.
- The maximum number of routes to be inserted in the LP, i.e., parameter $u(\bar{R})$, is set to 20,000 for the basic instances and when $(\alpha, \gamma) = (0.7, 0.9)$ and to 30,000 in all other cases.
- Criterion 1 has been chosen to measure the "desirability" of the routes.
- The maximum number of routes in each *IP*, r_{max}, is set to 300.
- The time limit T_{max} is set to one hour, and we do not impose a limit on the number of *IP*s solved; i.e., the parameter IP_{max} is not set.

In Table 1, we present the results obtained by the optimization-based heuristic for these parameter settings. Each row of the table is associated with a particular instance. The first four columns provide information on the instance characteristics, i.e., the name of the instance, the number of customers in the instance, the lower bound on customer demand (α), and the upper bound on customer demand (γ). For the basic instances, no value is provided for the lower and upper bound on demand. The second set of four columns provides information on the solution process and the quality of the solutions obtained, i.e., the number of integer programs solved, the total running time, the percentage improvement over the solution obtained with the tabu search of Archetti, Hertz, and Speranza (2006), and the percentage gap between the value of the solution obtained by the optimization-based heuristic and the value of the LP relaxation over *all* routes generated (i.e., the entire set of promising routes). Finally, the last two columns show the largest improvement and the smallest gap obtained during a variety of experiments with different desirability criteria and different integer program sizes. The results show that the optimization-based heuristic was able to find an improved solution for all but one instance (the only exception is basic instance p11). The average improvement is a little over 0.5 percent. Even though the improvements are relatively small, we believe this is primarily because the tabu solutions are already very good. Although the value of the linear program over the entire set of promising routes is not a true lower bound, as we are not optimizing over the entire

Table 1 Performance of Optimization-Based Heuristic

Instance	n	α	γ	No. of IPs	time	Percentage improvement	Percentage gap	Percentage improvement	Percentage gap
p01.cri	50	—	—	68	97	0.59	0.53	0.59	0.53
p02.cri	75	—	—	36	52	0.08	1.17	0.08	1.17
p03.cri	100	—	—	1	51	0.01	0.01	0.01	0.01
p04.cri	150	—	—	70	298	0.68	0.30	0.70	0.28
p05.cri	199	—	—	71	297	0.15	0.12	0.15	0.12
p10.cri	199	—	—	71	298	0.15	0.12	0.15	0.12
p11.cri	120	—	—	69	262	0.00	0.14	0.00	0.14
p01.cri	50	0.1	0.3	45	256	0.00	2.36	0.37	1.99
p02.cri	75	0.1	0.3	56	161	0.99	1.23	0.99	1.23
p03.cri	100	0.1	0.3	40	159	0.54	1.75	0.65	1.65
p04.cri	150	0.1	0.3	113	1,152	0.24	2.20	0.24	2.20
p05.cri	199	0.1	0.3	39	567	0.10	0.89	0.13	0.85
p10.cri	199	0.1	0.3	39	545	0.10	0.89	0.13	0.85
p11.cri	120	0.1	0.3	110	585	0.83	4.80	1.41	4.20
p01.cri	50	0.1	0.5	96	866	0.17	3.17	0.30	3.04
p02.cri	75	0.1	0.5	109	646	0.48	1.90	0.53	1.85
p03.cri	100	0.1	0.5	39	201	1.41	1.88	1.46	1.83
p04.cri	150	0.1	0.5	80	517	0.33	3.25	0.49	3.08
p05.cri	199	0.1	0.5	128	1,138	0.49	2.30	0.84	1.95
p10.cri	199	0.1	0.5	128	1,114	0.49	2.30	0.84	1.95
p11.cri	120	0.1	0.5	64	365	0.12	4.62	0.59	4.14
p01.cri	50	0.1	0.9	110	2,939	0.00	2.94	0.08	2.86
p02.cri	75	0.1	0.9	39	361	0.03	1.84	0.04	1.83
p03.cri	100	0.1	0.9	82	620	0.44	1.18	0.60	1.01
p04.cri	150	0.1	0.9	109	592	0.25	2.61	0.31	2.55
p05.cri	199	0.1	0.9	52	806	0.05	0.92	0.10	0.88
p10.cri	199	0.1	0.9	52	813	0.05	0.92	0.10	0.88
p11.cri	120	0.1	0.9	9	4,882	1.48	5.36	3.27	3.54
p01.cri	50	0.3	0.7	12	1,684	0.13	3.06	0.13	3.06
p02.cri	75	0.3	0.7	63	2,551	1.06	2.97	1.08	2.95
p03.cri	100	0.3	0.7	122	1,605	0.50	2.73	0.50	2.73
p04.cri	150	0.3	0.7	52	251	0.68	2.55	0.70	2.53
p05.cri	199	0.3	0.7	103	1,702	0.31	3.00	0.38	2.93
p10.cri	199	0.3	0.7	103	1,704	0.31	3.00	0.38	2.93
p11.cri	120	0.3	0.7	4	7,147	0.58	6.69	0.58	6.69
p01.cri	50	0.7	0.9	6	834	0.06	2.25	0.06	2.25
p02.cri	75	0.7	0.9	47	1,872	0.34	2.89	0.34	2.89
p03.cri	100	0.7	0.9	10	2,433	0.29	3.17	0.41	3.05
p04.cri	150	0.7	0.9	39	2,460	0.30	2.40	0.30	2.40
p05.cri	199	0.7	0.9	65	656	0.44	1.62	0.44	1.62
p10.cri	199	0.7	0.9	65	656	0.44	1.62	0.44	1.62
p11.cri	120	0.7	0.9	4	3,948	1.17	3.42	2.34	2.25

set of routes, it is likely to be very close. Therefore, the small gaps observed substantiate the fact that the solutions obtained are likely to be close to optimal.

Our optimization-based heuristic starts from the solution found by the tabu search algorithm. Therefore, it is important to understand whether the time spent on the optimization-based heuristic is well spent or whether we would have been better off continuing the tabu search algorithm. We performed this comparison and found that for only 16 instances (of 42), the tabu search algorithm was able to further improve the solution with a maximum improvement of 1.77%, while the optimization-based heuristics improves the solution for 41 instances (all but one) with a maximum improvement of 3.27%. Comparing the final solutions, we see that for 33 instances the solution produced by the optimization-based heuristic is the best; for eight instances the solution produced by the tabu search algorithm is best; and only for one instance are the solutions the same. These results support our belief that the use of optimization techniques in conjunction with heuristic search can indeed lead to better solutions than focusing entirely on heuristic search. The results also point to the fact that iterating between the two schemes, i.e., iterating between the tabu search algorithm and the optimization-based heuristic, may result in even better performance. This is a topic for further research.

Table 2 Impact of Integer Program Size

IP size	No. of *IP*s	Time	Percentage improvement	Percentage gap	Percentage improvement	Percentage gap
200	108	288	1.56	5.29	1.56	5.29
300	9	4,882	1.48	5.36	1.48	5.36
400	1	3,671	1.46	5.39	2.18	4.64
500	1	3,601	1.43	5.42	2.97	3.84
600	1	3,601	1.46	5.39	2.68	4.14

In such a set up, the optimization-based heuristic can be viewed as a sophisticated diversification scheme.

Next, we analyze the impact of the choice of the size of the integer programs. For instance p11 with $(\alpha, \gamma) = (0.1, 0.9)$ we varied the integer program size from 200 to 600. The results are presented in Table 2. With integer program size 200, the integer programs are solved quickly, as 108 integer programs can be solved with 288 seconds. This also indicates that after constructing 108 integer programs we have run out of promising columns. With integer program size 300, we can only solve nine integer programs because the time to solve the integer programs increases rapidly with its size. (The reason that the total time is greater than 3,600 seconds is that each individual integer program has a time limit of one hour as well.) With integer program sizes greater than 300 we can solve at most one integer program.

The other observation that can be made is that for some other criterion an increased size of the integer program did lead to better solutions, but that this improvement is not monotonic with the increase in integer program size.

Next, we present, in Figures 2 and 3, the solution obtained by the tabu search and the solution found by the optimization-based heuristic for instance p11 with $(\alpha, \gamma) = (0.1, 0.9)$. Examining these solutions reveals how difficult split delivery vehicle routing problems can be. The difference in solution quality between the two solutions displayed is more than 3%. The figures

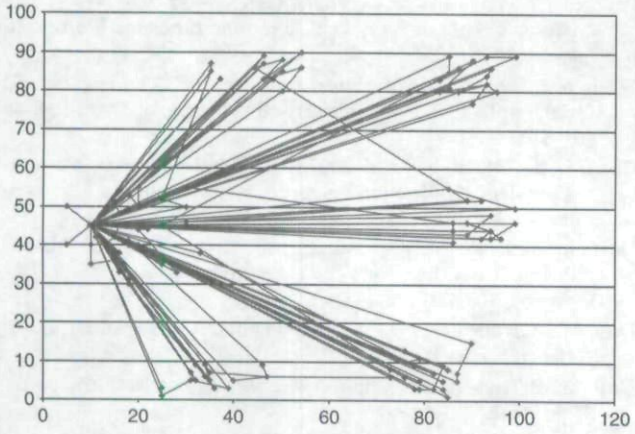

Figure 2 Tabu Solution for Instance p11 with $(\alpha, \gamma) = (0.1, 0.9)$

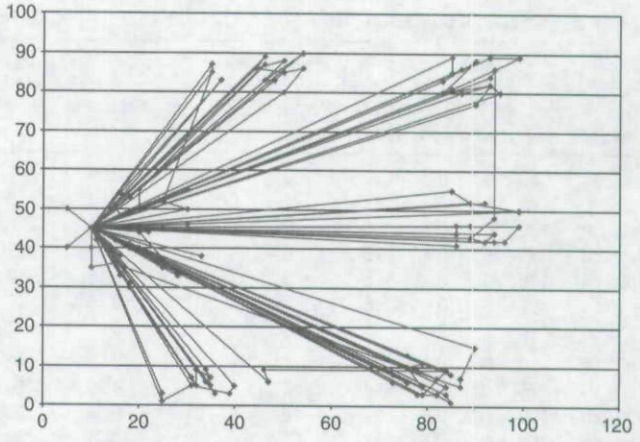

Figure 3 Optimization-Based Heuristic Solution for Instance p11 with $(\alpha, \gamma) = (0.1, 0.9)$

clearly demonstrate that it will be nearly impossible for a human planner to construct high quality solutions.

Additional insights into the characteristics of split delivery vehicle routing problem solutions can be obtained from the information presented in Table 3. The table provides information for three solutions: the solution obtained by the tabu search heuristic, the solution obtained by our optimization-based heuristic, and the best solution obtained by our optimization-based heuristic when we run it with different desirability criteria. We present the number of routes in the solution, the percentage of customers that receive split deliveries, and the maximum number of deliveries received by any customer. The last column of the table reports the best solution found over all tests for each instance.

Especially for instances with customers with relatively large demands—i.e., $(\alpha, \gamma) = (0.1, 0.9)$, (0.3, 0.7), and (0.7, 0.9)—we see that the percentage of customers receiving split deliveries and the maximum number of deliveries at a customer are large, about 40% and around five, respectively. This information reinforces the notion that high-quality solutions will be nearly impossible to construct by human planners as it is unlikely that a human planner will envision and consider solutions in which a customer is visited five times. This further suggests that it may be interesting to look at a variant of the split delivery vehicle routing problem in which the number of visits to a customer is limited. Our optimization-based heuristic can easily accommodate a limit on the number of visits to a customer by introducing constraints

$$\sum_{r \in R: i \in r} x_r \leq k \quad i \in C, \tag{15}$$

where k is the imposed limit.

Finally, it is interesting to observe that for the optimization-based heuristic solutions with the largest

Table 3 Solution Characteristics

				Tabu search			Optimization-based				Best optimization-based				
Instance	n	α	γ	No. of routes	Percent of split	Max no. of split	Percent of improve	No. of routes	Percent of split	Max no. of split	Percent of improve	No. of routes	Percent of split	Max no. of split	Value of the best solution
p01.cri	50	0	0	5	0	0	0.59	5	0	0	0.59	5	0	0	5,276,751
p02.cri	75	0	0	10	4	1	0.08	10	4	1	0.08	10	4	1	8,536,078
p03.cri	100	0	0	8	0	0	0.01	8	0	0	0.01	8	0	0	8,401,150
p04.cri	150	0	0	12	6	1	0.68	12	5	1	0.70	12	6	1	10,550,759
p05.cri	199	0	0	16	3	1	0.15	16	3	1	0.15	16	3	1	13,383,599
p10.cri	199	0	0	16	3	1	0.15	16	3	1	0.15	16	3	1	13,383,816
p11.cri	120	0	0	7	3	1	0	7	3	1	0	7	3	1	10,569,587
p01.cri	50	0.1	0.3	11	10	1	0	11	10	1	0.37	11	10	1	7,582,003
p02.cri	75	0.1	0.3	16	13	1	0.99	16	11	1	0.99	16	11	1	11,229,145
p03.cri	100	0.1	0.3	22	12	1	0.54	22	11	1	0.65	22	11	1	15,054,586
p04.cri	150	0.1	0.3	32	13	1	0.24	32	13	2	0.40	32	13	2	20,932,806
p05.cri	199	0.1	0.3	41	18	2	0.10	41	16	2	0.13	41	15	2	25,826,172
p10.cri	199	0.1	0.3	41	18	2	0.10	41	16	2	0.13	41	15	2	25,826,172
p11.cri	120	0.1	0.3	26	13	2	0.83	26	15	2	1.41	26	13	1	30,179,211
p01.cri	50	0.1	0.5	16	12	1	0.17	16	14	1	0.30	16	16	1	10,210,207
p02.cri	75	0.1	0.5	24	13	2	0.48	24	16	2	0.53	24	16	2	15,485,438
p03.cri	100	0.1	0.5	33	22	2	1.41	33	21	1	1.46	33	20	1	20,245,791
p04.cri	150	0.1	0.5	49	21	2	0.33	49	23	2	0.49	49	21	2	29,770,034
p05.cri	199	0.1	0.5	63	16	2	0.49	63	17	2	0.84	63	17	2	35,939,960
p10.cri	199	0.1	0.5	63	16	2	0.49	63	16	2	0.84	63	17	2	35,939,960
p11.cri	120	0.1	0.5	40	21	3	0.12	40	15	2	0.59	41	23	2	44,763,774
p01.cri	50	0.1	0.9	26	38	2	0	26	36	2	0.08	26	34	2	14,972,843
p02.cri	75	0.1	0.9	41	29	2	0.03	41	29	2	0.04	41	27	2	23,378,115
p03.cri	100	0.1	0.9	56	32	3	0.44	56	31	3	0.60	56	30	2	31,362,933
p04.cri	150	0.1	0.9	84	35	3	0.25	84	36	2	0.31	84	36	3	46,599,000
p05.cri	199	0.1	0.9	107	29	4	0.05	107	29	3	0.10	107	28	3	57,102,065
p10.cri	199	0.1	0.9	107	29	4	0.05	107	29	3	0.10	107	28	3	57,102,065
p11.cri	120	0.1	0.9	67	33	4	1.48	69	34	2	3.27	70	35	2	71,172,434
p01.cri	50	0.3	0.7	26	32	2	0.13	26	34	2	0.13	26	34	2	15,020,022
p02.cri	75	0.3	0.7	39	27	2	1.06	39	28	3	1.08	39	29	3	22,631,233
p03.cri	100	0.3	0.7	53	30	3	0.50	53	29	2	0.50	53	29	2	30,555,132
p04.cri	150	0.3	0.7	80	34	3	0.68	79	38	3	0.70	79	39	2	44,654,674
p05.cri	199	0.3	0.7	103	35	4	0.31	104	32	3	0.38	103	36	3	55,497,672
p10.cri	199	0.3	0.7	103	35	4	0.31	104	32	3	0.38	103	36	3	55,497,672
p11.cri	120	0.3	0.7	65	33	2	0.58	65	29	2	0.58	65	29	2	71,268,363
p01.cri	50	0.7	0.9	42	32	3	0.06	42	32	3	0.06	42	32	3	21,667,970
p02.cri	75	0.7	0.9	61	40	4	0.34	62	40	4	0.34	62	40	4	32,503,861
p03.cri	100	0.7	0.9	82	39	5	0.29	82	41	5	0.41	82	37	5	44,525,587
p04.cri	150	0.7	0.9	123	43	4	0.30	123	42	4	0.30	123	42	4	64,627,754
p05.cri	199	0.7	0.9	162	42	5	0.44	162	43	5	0.44	162	43	5	83,554,528
p10.cri	199	0.7	0.9	162	42	5	0.44	162	43	5	0.44	162	43	5	83,554,528
p11.cri	120	0.7	0.9	99	39	5	1.17	101	41	3	2.34	101	43	3	104,297,549

improvements over the tabu search heuristic solutions, the number of routes in the solution is larger, e.g., for instance p11 with $(\alpha, \gamma) = (0.1, 0.9)$ and $(0.7, 0.9)$.

Acknowledgments

The authors thank the referees for thoughtful and insightful comments that helped improve both the quality and the presentation of the paper.

References

Archetti, C., A. Hertz, M. G. Speranza. 2006. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Sci.* **40** 64–73.

Archetti, C., R. Mansini, M. G. Speranza. 2005. Complexity and reducibility of the skip delivery problem. *Transportation Sci.* **39** 182–187.

Archetti, C., M. W. P. Savelsbergh, M. G. Speranza. 2006. Worst-case analysis for split delivery vehicle routing problems. *Transportation Sci.* **40** 226–234.

Belenguer, J. M., M. C. Martinez, E. Mota. 2000. A lower bound for the split delivery vehicle routing problem. *Oper. Res.* **48** 801–810.

Chen, S., B. Golden, E. Wasil. 2007. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks* **49** 318–329.

De Franceschi, R., M. Fischetti, P. Toth. 2006. A new ILP-based refinement heuristic for vehicle routing problems. *Math. Programming* **105** 471–499.

Dror, M., P. Trudeau. 1989. Savings by split delivery routing. *Transportation Sci.* **23** 141–145.

Dror, M., P. Trudeau. 1990. Split delivery routing. *Naval Res. Logist.* **37** 383–402.

Dror, M., G. Laporte, P. Trudeau. 1994. Vehicle routing with split deliveries. *Discrete Appl. Math.* **50** 239–254.

Espinoza, D., R. García, M. Goycoolea, G. L. Nemhauser, M. W. P. Savelsbergh. 2008a. Per-seat, on-demand air transportation Part I: Problem description and an integer multi-commodity flow model. *Transportation Sci.* Forthcoming.

Espinoza, D., R. García, M. Goycoolea, G. L. Nemhauser, M. W. P. Savelsbergh. 2008b. Per-seat, on-demand air transportation Part II: Parallel local search. *Transportation Sci.* Forthcoming.

Frizzell, P. W., J. W. Giffin. 1995. The split delivery vehicle scheduling problem with time windows and grid network distances. *Comput. Oper. Res.* **22** 655–667.

Gendreau, M., P. Dejax, D. Feillet, C. Gueguen. 2006. Vehicle routing with time windows and split deliveries. Technical Report 2006-851, Laboratoire d'Informatique d'Avignon, Université d'Avignon, d'Avignon, France.

Gueguen, C. 1999. Méthodes de résolution exacte pour les problèmes de tournées de véhicules. Unpublished doctoral thesis, École Centrale, Paris, Paris.

Mullaseril, P. A., M. Dror, J. Leung. 1997. Split-delivery routing in livestock feed distribution. *J. Oper. Res. Soc.* **48** 107–116.

Sierksma, G., G. A. Tijssen. 1998. Routing helicopters for crew exchanges on off-shore locations. *Ann. Oper. Res.* **76** 261–286.