TRANSPORTATION SCIENCE
Vol. 38, No. 3, August 2004, pp. 369–378
ISSN 0041-1655 | EISSN 1526-5447 | 04 | 3803 | 0369

informs®

DOI 10.1287/trsc.1030.0046
© 2004 INFORMS

Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems

Ann Melissa Campbell
Department of Management Sciences, Henry B. Tippie College of Business,
University of Iowa, Iowa City, Iowa 52242-1000, ann-campbell@uiowa.edu

Martin Savelsbergh
Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0205,
martin.savelsbergh@isye.gatech.edu

Insertion heuristics have proven to be popular methods for solving a variety of vehicle routing and scheduling problems. In this paper, we focus on the impact of incorporating complicating constraints on the efficiency of insertion heuristics. The basic insertion heuristic for the standard vehicle routing problem has a time complexity of $O(n^3)$. However, straightforward implementations of handling complicating constraints lead to an undesirable time complexity of $O(n^4)$. We demonstrate that with careful implementation it is possible, in most cases, to maintain the $O(n^3)$ complexity or, in a few cases, increase the time complexity to $O(n^3 \log n)$. The complicating constraints we consider in this paper are time windows, shift time limits, variable delivery quantities, fixed and variable delivery times, and multiple routes per vehicle. Little attention has been given to some of these complexities (with time windows being the notable exception), which are common in practice and have a significant impact on the feasibility of a schedule as well as the efficiency of insertion heuristics.

Key words: vehicle routing and scheduling; insertion heuristics; time windows; variable delivery times; shifts
History: Received: February 2002; revision received: June 2002; accepted: June 2002.

Insertion heuristics have proven to be popular methods for solving a variety of vehicle routing and scheduling problems. Insertion heuristics were first introduced and analyzed, as were so many other popular optimization techniques, for the traveling salesman problem (Rosenkrantz et al. 1977).

Insertion heuristics construct a feasible solution, i.e., a set of feasible routes, by repeatedly and greedily inserting an as of yet unrouted customer into a partially constructed feasible solution. Different variants of the insertion heuristic arise as a result of how the two key decisions that are made at every iteration are answered: which unrouted customer to insert, and where to insert it in the partial solution?

Insertion heuristics are popular because they are fast, they produce decent solutions, they are easy to implement, and they can easily be extended to handle complicating constraints. The literature contains many examples of insertion heuristics applied to various routing and scheduling problems. These include the vehicle routing problem with time windows (Solomon 1987), the asymmetric capacitated vehicle routing problem (Vigo 1996), the fleet size and mix vehicle routing problem with time window constraints (Liu and Shen 1999), and the vehicle routing problem with backhauling (Salhi and Nagy 1999).

Insertion heuristics are also typically the method of choice for constructing an initial feasible solution in local search and metaheuristics for vehicle routing and scheduling problems (Desrosiers et al. 1995).

In this paper, we focus on the impact of incorporating complicating constraints on efficiency. Efficient implementations of insertion heuristics are of interest for two reasons:

• Technological developments have created an environment in which we are able at every moment in time to know where our transportation assets are and to communicate with them. As a result, there is a growing interest in and demand for real-time routing and scheduling technology. In real-time routing and scheduling environments, decisions typically have to be made in a very short amount of time, often in a matter of seconds, and fast insertion heuristics may provide one of the few viable options for decision making. There may not be enough time to employ local search and metaheuristics.

• Many commercial routing and scheduling software packages rely on insertion heuristics as the core decision technology, the reason being that most routing and scheduling software packages are interactive, i.e., support user interaction. In such situations, it is often the case that solution quality is sacrificed in favor of fast response times.

The basic insertion heuristic for the standard vehicle routing problem has a time complexity of $O(n^3)$. However, straightforward implementations of handling complicating constraints lead to an undesirable time complexity of $O(n^4)$. We demonstrate that with careful implementation it is possible, in most cases, to maintain the $O(n^3)$ complexity or, in a few cases, increase the time complexity to $O(n^3 \log n)$. These same techniques should be applicable to complexities beyond those discussed here.

As mentioned earlier, insertion heuristics are often succeeded by iterative improvement heuristics. Various papers have been written on how to efficiently implement iterative improvement heuristics in the presence of complicating constraints. For a survey of these techniques, see Kindervater and Savelsbergh (1997).

The complicating constraints we consider in this paper are time windows, shift time limits, variable delivery quantities, fixed and variable delivery times, and multiple routes per vehicle. Some of these complexities have received considerable attention over the years (particularly time windows), whereas others have received little or no attention (e.g., variable delivery quantities and variable delivery times). However, all of them are common in practice and have a significant impact on the feasibility of a schedule as well as the efficiency of insertion heuristics. We have opted to discuss all of them, turning this paper into one that is part survey and part new material, because it allows us to illustrate a general framework in which all of these techniques can be applied. This may better serve researchers and practitioners who want or need to develop insertion heuristics for complex routing and scheduling situations where several of these complicating constraints exist.

1. Basic Insertion Heuristic

In this section, we analyze the complexity of the basic insertion heuristic for the vehicle routing problem. We assume that the reader is familiar with the vehicle routing problem, but if not, the recent book edited by Toth and Vigo (2002) provides an excellent introduction to the vehicle routing problem, its variants, and existing solution procedures. The basic insertion heuristic is presented next in great detail to introduce the reader to the various steps of the analysis that we will carry out for each of the more complex variants.

Before describing the basic insertion heuristic, we define the relevant terminology and notation that will be used. There are n customers and the delivery volume at customer j is denoted by D_j. We assume that each D_j is less than the truck capacity Q, and that we have a homogeneous fleet of vehicles. The travel time between two customers i and j is denoted by $T_{i,j}$, and initially there is no additional time required for delivery at a customer beyond the travel time. A *route* is a trip from the depot to a sequence of customers and back to the depot. For presentational convenience, we abuse notation and represent a route as $(0, 1, 2, \ldots, i, \ldots, n+1)$, where 0 and $n+1$ both refer to the depot and where we will refer to customer i, when in fact we refer to the customer currently in position i in the route. In describing an insertion heuristic, we will always consider inserting (unrouted) customer j between customers $i-1$ and i on the route $(0, 1, 2, \ldots, i-1, i, \ldots, n+1)$.

The basic insertion heuristic for the standard vehicle routing problem can be found in Algorithm 1. It is a parallel insertion heuristic, as discussed in Potvin and Rousseau (1993), where multiple routes are being built at the same time. In a sequential version, each route is completed before another one is created. We do not make any assumptions about selecting seed points to initialize any of the routes. Any seed selection method can be used here without changing the complexity of the algorithm as long it its complexity is less than $O(n^3)$. No seed selection is required, though, for the insertion algorithm to work. When it is cheapest to insert an uninserted customer on an empty route rather than an existing route, the customer will be inserted on the new route.

Algorithm 1. Insertion Heuristic

```
1.  $N$ = set of unassigned customers
2.  $R$ = set of routes; always contains the empty
        route; initially contains only the empty route
3.  while $N \neq \emptyset$ do
4.    $p* = -\infty$
5.    for $j \in N$ do
6.      for $r \in R$ do
7.        for $(i-1, i) \in r$ do
8.          if $Feasible(i, j)$ and $Profit(i, j) > p*$ then
9.            $r* = r$
10.           $i* = i$
11.           $j* = j$
12.           $p* = Profit(i, j)$
13.         end if
14.       end for
15.     end for
16.   end for
17.   $Insert(i*, j*)$
18.   $N = N \setminus j*$
19.   $Update(r*)$
20. end while
```

The complexity of the procedure is $O(n^3)$ under the assumption that checking the feasibility of an insertion (function *Feasible*( )) and computing the profitability of an insertion (function *Profit*( )) can be done in constant time and that updating a route (function

Update( )) can be done in $O(n^2)$ time (the latter is usually no issue).

Information Maintained
To achieve a $O(n^3)$ complexity for the basic insertion heuristic for the standard vehicle routing problem, we need to maintain for every route the sum of the delivery quantities currently assigned to that route, q_r.

Checking Feasibility
In the standard vehicle routing problem, the only constraint that needs to be verified is the vehicle capacity constraint. If we know the sum of the delivery quantities currently assigned to a route, it is easy to verify the feasibility of inserting customer j into the route; i.e., $D_j < Q - q_r$, and this takes constant time.

Computing Profitability
For the time being, we assume that our objective is to minimize the total travel time. Therefore, we assign the following profit to an insertion $-(T_{i-1,j} + T_{j,i} - T_{i-1,i})$. This quantity is the negative of the extra travel time introduced for the route by inserting j between $i-1$ and i. The larger this value, the smaller the extra travel time. Again, it is easy to see that computing the profitability can be done in constant time.

Updating the Route
After the customer to be inserted has been selected and it has been decided where to insert the customer, the affected route needs to be updated. In terms of the information we maintain to facilitate feasibility checking, all we need to do is update the sum of the delivery volumes currently assigned to the route; i.e., $q_r = q_r + D_j$. Of course, we also need to update the data structures used to maintain the current set of routes.

In each major iteration of the insertion heuristic, a customer is selected and inserted into a partial route, which is subsequently updated. Because there are n customers, there are $O(n)$ major iterations. In each major iteration, we evaluate every unrouted customer at every possible insertion point. Because there are $O(n)$ unrouted customers and there are $O(n)$ possible insertions, and because evaluating an insertion, i.e., checking its feasibility and calculating its profit, can be done in constant time, the selection of a customer and its best insertion place takes $O(n^2)$ time. Updating the affected route takes $O(1)$ time. Consequently, each major iteration takes $O(n^2)$ time. This gives an overall time complexity of $O(n^3)$.

From the above discussion it is clear that to ensure an overall complexity of $O(n^3)$ when incorporating complicating constraints, we have to focus on efficiently checking feasibility and computing profitability. Straightforward implementations perform these two functions by physically inserting j between $i-1$ and i (temporarily) and traversing the route to check feasibility, which takes $O(n)$ time and increases the overall time complexity to $O(n^4)$. We will show that by maintaining appropriate information about the current partial feasible solution, it is often possible to perform these two functions in constant time or in at most $O(\log n)$ time, leading to an overall complexity of $O(n^3)$ and $O(n^3 \log n)$, respectively.

2. Vehicle Routing Problem with Time Windows

In the vehicle routing problem with time windows (VRPTW), a time window (E_j, L_j) is specified for each customer j, with E_j denoting the earliest time a delivery can take place and L_j denoting the latest time a delivery can take place. How to handle time windows efficiently in insertion heuristics is well known, and included here mainly for completeness and for illustrative purposes. The same notation and methodology will be expanded for the complexities discussed later.

Information Maintained
For every customer i already assigned to a route, we maintain two quantities: the earliest time a delivery can be made at i, denoted by e_i, and the latest time a delivery can be made at i, denoted by l_i. For convenience, we set $e_0 = 0$ and $l_{n+1} = T$, where T is the end of the planning period. Note that initially $e_j = E_j$ and $l_j = L_j$, but that e_j and l_j are specifically introduced to capture interactions between customers on the same route.

Checking Feasibility
The feasibility of inserting customer j between $i-1$ and i can now be checked as follows. First compute the earliest time a delivery can take place at j:

$$e_j = \max(E_j, e_{i-1} + T_{i-1,j}),$$

and the latest time a delivery can take place at j:

$$l_j = \min(L_j, l_i - T_{j,i}).$$

Given these quantities, checking the feasibility of the insertion amounts to verifying whether $D_j < Q - q_r$ and $e_j \le l_j$.

Evaluating profitability works the same as in the basic version of the algorithm because the time windows do not impact the extra travel time. To consider an objective related to route duration instead, it is necessary to maintain additional information (representing travel time from a node to the end of the route). This modification is discussed in detail in the next section on shift time limits.

Updating the Route

An inserted customer can impact the deliveries to customers that come both before it and after it on the route. More specifically, the late values for the prior customers and the early values for the later customers may be affected by the insertion. Due to the travel time required to visit the new customer, there might not be as much room to postpone prior deliveries and subsequent deliveries may have to start later. Updating these early and late values can be done in $O(n)$ time as follows, where we use the already computed values e_j and l_j.

- For $k = i - 1$ to 0, the new delivery impacts the latest time that deliveries at these locations may occur:

$$l_k = \min(l_k, l_{k+1} - T_{k,k+1}).$$

- For $k = i$ to $n+1$, the insertion alters the earliest time that these deliveries can begin:

$$e_k = \max(e_k, e_{k-1} + T_{k-1,k}).$$

In practice, we only need to continue the updates of late values backward (early values forward) as long as the late (early) value for the most recently evaluated customer changed. For example, in working backwards from $i-1$ to 0, if l_{i-1} does not change, then there is no need to reevaluate l_{i-2}. Because we *may* have to update the l values for all prior deliveries and the e values for all subsequent deliveries, updating has complexity $O(n)$.

Finalizing the Solution

After a feasible set of routes has been created, there is often some flexibility remaining with regard to delivery times. It is easy, however, to establish a feasible solution from the values we have maintained about the route. By construction, it is always feasible to deliver the requested volume to all customers at the earliest time, e_i, or to all customers at the latest time, l_i. Both of these solutions clearly require the same amount of travel time. To minimize waiting time for a given route, which is often a consideration, it is best to begin each route at the latest time possible (l_0). After this, begin delivery to the following customers at the earliest feasible time. This minimizes the duration of the route, which in turn minimizes the waiting time.

3. Shift Time Limit

The Department of Transportation places limits on how many hours drivers can work, including a 16-hour limit on how long a driver can be on duty (shift time limit), a 10-hour limit on the amount of time spent driving (drive time limit), and a limit of 70 total hours in any 8 consecutive days. If a driver works five 14-hour work days in a row, this driver then cannot work at all for the next three days because of this last restriction. Often companies place even more restrictive time limits on how long drivers can be on the road each day for either safety reasons, scheduling reasons, or both. We will discuss here how to modify insertion heuristics to consider a shift time limit. A drive time limit can be included similarly.

Note that the basic insertion heuristic will not necessarily produce a solution that satisfies shift time limits. To consider this variation, we will let S represent the shift time limit. We will continue to assume that all customers have delivery time windows. Others have considered a shift time limit in a version of the vehicle routing problem that minimizes completion time of the routes, including de Jong et al. (1996) and Potvin and Bengio (1996), where our methodology will maintain time limit feasibility with a variety of objectives.

Information Maintained

The values of e_0 and l_{n+1} will be used to reflect the shift time limit. We define $e_0 = \max(0, e_{n+1} - S)$; i.e., the difference between the earliest ending time of the route and the earliest start time of the route will be within the shift time limit, and $l_{n+1} = \min(l_0 + S, T)$. That is, the difference between the latest completion time of the route and the latest start time of the route will be within the shift time limit. Furthermore, we maintain a quantity a_i representing the cumulative travel time from customer i forward to the end of the route.

Checking Feasibility

In addition to computing the earliest and latest times the delivery can take place at j, we also compute the earliest completion time of the shift if j is inserted, the implied earliest departure time, the latest departure time of the shift if j is inserted, and the implied latest completion time. The first two values, computed as described in the previous section, help determine whether the insertion is feasible with regard to the time windows, and the latter values help decide whether the insertion is feasible with regard to the shift time limit.

The earliest completion time of the shift if j is inserted is computed as:

$$\bar{e}_{n+1} = \max\big(e_{n+1}, e_j + T_{j,i} + a_i\big).$$

The first term of the maximum can dominate if the current shift has substantial waiting time on the portion of the route from i forward. If, for example, e_i is determined by E_i, the opening of the customer-defined delivery window, rather than travel time from preceding customers, then e_i may exceed e_{i-1} by significantly more than the travel time between these two stops. In this case, if $e_j + T_{j,i}$ is still less than e_i, then the early completion time of the shift may be unchanged. The implied earliest departure of the shift is $\bar{e}_0 = \max(e_0, \bar{e}_{n+1} - S)$. If $\bar{e}_0$ is defined by the second term, the added time required to visit j will reduce the amount of time available before the delivery because of the strict time limit.

The latest departure time of the shift is computed as:

$$\bar{l}_0 = \min\big(l_0, l_j - T_{i-1,j} - (a_0 - a_{i-1})\big).$$

The first term of the minimum will dominate if the current route has waiting time on the portion of the route preceding $i-1$ due to customer-defined time windows. The implied latest completion time of the shift is $\bar{l}_{n+1} = \min(l_{n+1}, \bar{l}_0 + S)$. If $\bar{l}_{n+1}$ is defined by the second term, the added time required to visit j will reduce the amount of time available after the delivery because of the strict time limit.

Verifying feasibility now amounts to checking whether or not $D_j \le Q - q_r$, $e_j \le l_j$, $\bar{e}_0 \le \bar{l}_0$, and $\bar{l}_{n+1} \ge \bar{e}_{n+1}$. The insertion will be infeasible if any of these are violated.

Updating the Route

As before, we must update the e values for the deliveries following the inserted customer j (or until no change occurs). However, if the computed $\bar{l}_{n+1}$ is less than l_{n+1} or $\bar{e}_0 > e_0$ due to the shift time limit, we have to perform additional updates. If $\bar{e}_0 > e_0$, then after we update the e values for the positions after the insertion, we need to update e values from e_0 forward to just prior to the point of insertion. If $\bar{l}_{n+1} < l_{n+1}$, then after we update the l values for the positions prior to the insertion, we need to update l values from l_{n+1} backward to just after the point of insertion.

Obviously, we also need to update the cumulative travel time information; i.e., $a_j = T_{j,i} + a_i$. To update the a values for the customers preceeding j, we compute the change in time Δ caused by the insertion, i.e., $\Delta = T_{i-1,j} + T_{i,j} - T_{i-1,i}$, and then update for $k = i-1, \ldots, 1$ as follows: $a_k = a_k + \Delta$. All of these updates can clearly be accomplished in linear time, preserving the $O(n^3)$ complexity.

4. Variable Delivery Volume

The delivery volume requested by a customer is typically based on consumption by the customer from the time of the last delivery until the estimated time of arrival of the next delivery. In many environments, customers specify a fairly large delivery window and are often willing and able to receive a larger delivery volume later in the window. For example, if a grocery store places an order for a specified number of loaves of bread to arrive between 8:00 am and 12:00 pm on a given day, the grocery store has more room on the shelves the closer to 12:00 the truck arrives, due to the bread sales during the morning. In the inventory routing problem (Campbell 2000, Dror and Ball 1987, Dror et al. 1985), the vendor has negotiated the right to decide the day, time, and volume of all deliveries to its customers. This complete transfer of inventory management responsibility to the vendor allows vendors to choose delivery quantities that combine well to make efficient, full-truckload deliveries. In the more restrictive case we consider in this paper, a customer specifies a minimum delivery volume, but allows the volume to increase over time according to a customer-specific usage rate. The customer still maintains responsibility for inventory management, but allows some flexibility. Allowing a larger delivery volume gives the customers added protection against running out of product, and the flexibility gives the vendors the ability to make better use of their resources.

We can model this complexity with a usage rate for each customer j, represented by U_j, and the assumption that we can deliver up to a certain capacity indicated by C_j. We will use I_j to represent initial inventory at customer j at the beginning of the planning period, and D_j will now represent the minimum delivery volume to each customer.

We will use time windows implied by usage and inventory status, rather than customer-defined delivery windows. These will show the influence that usage and inventory can have on delivery times. Customer-specified delivery windows can easily be included as well.

Information Maintained

We replace q_r with $q_r^{\min}$ to represent the minimum volume deliverable on a route ($q_r^{\min} = \sum_{i=1,\ldots,n} D_i$). For profitability, not feasibility, we also introduce $f_i(t)$, a piecewise linear concave function yielding the maximum volume deliverable to all customers up to and including i on the route if delivery at i takes place at time t, g_i the maximum volume deliverable to i and all customers succeeding i on the route, and $q_r^{\max}$ the maximum volume deliverable on the route.

Checking Feasibility

The feasibility check is similar to what we have seen before, with small changes to reflect the impact that usage and inventory have on feasible delivery times.

We compute the earliest time a delivery can take place considering the usage rate as:

$$e_j = \max\left(e_{i-1} + T_{i-1,j}, \frac{D_j - C_j + I_j}{U_j}\right).$$

The first term of the maximum represents the earliest arrival after the previous delivery, as before, and the second part represents the earliest time the quantity D_j can be delivered to the customer given its capacity constraints. The value $C_j - I_j$ represents the available capacity at j at the beginning of the planning period, so $D_j - (C_j - I_j)$ is the extra capacity needed for D_j to "fit" at j. Capacity is added at rate U_j, so the second term of the maximum gives the point in time when there is sufficient available capacity at j to receive the quantity D_j. Note that we can view the second term as the opening of an implied time window; i.e., $E_j = (D_j - C_j + I_j)/U_j$.

We compute the latest time a delivery can take place as follows:

$$l_j = \min\left(l_i - T_{j,i}, \frac{I_j}{U_j}\right).$$

The first term in the minimum represents the latest feasible departure time from j to reach i by its latest feasible time, and the second term represents the time when customer j runs out of product given the initial inventory level. Note that we can view the second term as the closing of the implied time window; i.e., $L_j = I_j/U_j$.

Now checking feasibility amounts to verifying $D_j < Q - q_r^{\min}$ and $e_j \le l_j$.

Computing Profitability

The costs typically associated with an insertion in vehicle routing and scheduling problems involve extra travel time, extra mileage, or extra waiting time. With flexibility in delivery volume, however, the objective should reflect the trade-off between the increased cost and increased revenue associated with each insertion. All previous variants had the same total delivery volume and thus the same revenue.

To include this flexibility, we may want to evaluate the maximum delivery volume for customer j if it is inserted between $i - 1$ and i in a route, which can be computed as follows:

$$d_j^{\max} = \min\left(Q - q_r^{\min}, C_j - I_j + U_j l_j\right).$$

The first term of the minimum is the capacity remaining in the vehicle if we assume all other customers on the route will receive their minimum delivery quantities, and the second term represents the volume that will fit at the latest time a delivery can be made at j.

Unfortunately, the above quantity does not allow us to compute the maximum volume that can be delivered on the route (or the change in maximum delivery volume for the route). The delivery volume to j may just be "taken away" from the delivery quantities to one or more other customers on the route. Because it is not possible to determine the impact a delivery at j has on the maximum delivery volume to customers preceding j on a route in constant time (due to the possibility of waiting time between deliveries), we cannot compute the exact change in total delivery volume on the route with the insertion of j in constant time. However, we will show that by maintaining additional information (f and g), we can compute the exact change in total delivery volume associated with an insertion in $O(\log n)$ time, which results in a total complexity for the insertion heuristic of $O(n^3 \log n)$.

As stated earlier, for each customer i, we will maintain a piecewise linear concave function $f_i(t)$ that yields the maximum total delivery volume to all customers up to and including i on the route if delivery at i takes place at time t. This function is not a simple linear function with slope equal to the sum of the usage rates of the preceding customers if there are waiting times between the scheduled deliveries. Waiting times can occur, for example, when the first customer on a route has low inventory and thus requires a delivery soon, but the minimum delivery volume to the second customer is large and will not fit in inventory until much later in the day.

The changes in the slope of the piecewise linear functions occur exactly at delivery times corresponding with a change in the number of customers considered in computing the maximum delivery volume. The first "piece" of this function for each customer k starts at the earliest delivery time (e_k) and has the steepest slope of all segments because the largest number of preceding customers are included, where the last piece is the flattest because the least number of customers are included. The piecewise function will be concave because the total volume deliverable will clearly be nondecreasing with time. Because each of the changes in slope corresponds with the behavior of a preceding customer, each piecewise function will have at most $O(n)$ points defining it. An example of such a function can be found in the bottom portion of Figure 1. The graph in the top portion represents the

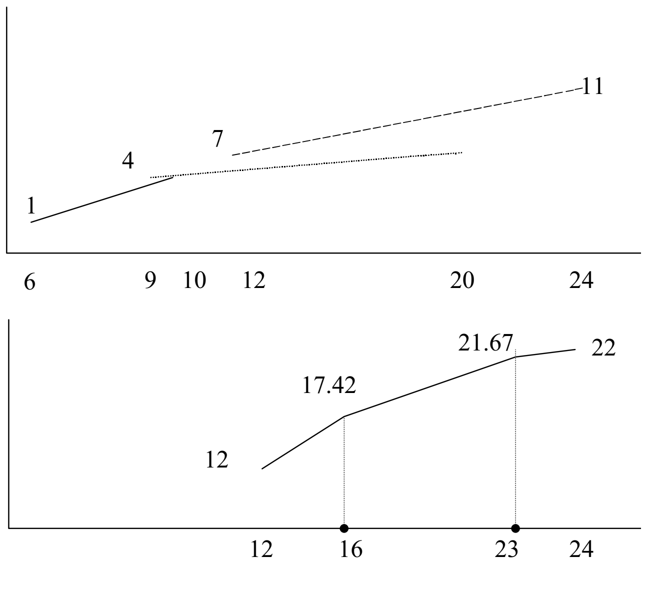

Figure 1 Time vs. Volume Deliverable

quantities deliverable to three individual customers. We will assume the travel time between each customer is three time units, such that delivery at the first at 6 leads to arrival at the second at 9, and a delivery at the second at 9 leads to an arrival at the third at 12. The bottom graph serves to summarize the maximum volume deliverable to all three given a delivery time at the third (i.e., it serves as f_3). Beginning delivery at the latest time for the third (at 24) creates wait before this delivery because the latest delivery time for the second is at 20 ($24-3-20=1$ unit of waiting time). The graph changes slope at 23 because this is where the delivery time at the last customer starts to impact the delivery volume at the second customer (because $23-3=20$). A delivery to the second customer at 20 still leaves wait after the first customer because $20-3$ is greater than 10. Only when the second customer starts delivery at 13 (third customer at 16) is the wait removed from after the first customer. From 12 to 16, the usage rate at all three customers impacts the delivery volume, which is why this portion of the cumulative graph has the largest slope.

For each customer i, we also maintain g_i, the maximum delivery volume possible to i and all of the customers succeeding i on the route. Given functions $f_i(t)$ and values g_i for each customer on a route, we can determine the profitability of an insertion in $O(\log n)$ time. To find the volume deliverable on the route ($\bar{q}_r^{\max}$) after customer j is inserted between $i-1$ and i, we compute:

$$g_j = \min\left(Q - \sum_{k=1,\ldots,i-1} D_k, C_j - I_j + U_j l_j + g_i\right).$$

The first term of the minimum restricts the largest volume deliverable to j and succeeding customers to allow for the minimum volume to be delivered to the preceding customers. The second term of the minimum limits the largest volume deliverable to j to the available capacity at l_j plus the largest volume deliverable to succeeding customers. (Note that the largest volume deliverable at a customer always occurs at the late time and that the late times are such that it is always possible to travel to the next customer and arrive at or before its late time.) Given this, we compute:

$$\bar{q}_r^{\max} = \min\big(Q, f_{i-1}(\min(l_{i-1}, l_j - t_{i-1,j})) + g_j\big).$$

The g_j computation takes constant time, but the $\bar{q}_r^{\max}$ calculation requires $O(\log n)$ time because it involves a function evaluation of a piecewise linear function with $O(n)$ pieces (f_{i-1}).

The difference between $\bar{q}_r^{\max}$ and $q_r^{\max}$ for the route yields the change in delivery volume associated with a particular insertion. The change in travel time can be computed as discussed earlier, and the two values can be combined in a variety of ways to define a measure of the desirability of an insertion, e.g.,

$$\alpha(\bar{q}^{\max} - q^{\max}) - \beta\big(T_{i-1,j} + T_{j,i} - T_{i-1,i}\big),$$

where α and β can be chosen to reflect the trade-off between revenue and cost.

Note that if D_i values are such that deliveries are feasible for the full planning period, then we do not need these piecewise linear functions. The objective evaluations can be made in constant time, and the overall complexity for the insertion heuristic remains $O(n^3)$.

Updating the Route

The updating procedures work the same as before except that now we need to additionally update the g_i values and the piecewise linear concave functions $f_i(t)$. The g_i values must be updated for customers prior to the insertion, and the piecewise linear concave functions change for customers after the insertion. The ordering of the updates is critical for the values to be updated correctly. *After* we have updated the late values for all customers preceding j, we compute for $k = i-1, \ldots, 1$:

$$g_k = \min\left(Q - \sum_{s=1,\ldots,k-1} D_s, C_k - I_k + U_k l_k + g_{k+1}\right).$$

After updating all g_k values, we update $q_r^{\max} = g_1$.

We must create the piecewise linear concave function $f_j(t)$ and update $f_k(t)$ for $k = i, \ldots, n$. We start with $f_j(t)$ being a simple linear function between e_j and l_j. The function values associated with the endpoints will be the maximum deliverable to j at these times, and the slope will be equal to U_j. Next, we iteratively refine the function $f_j(t)$ based on the shape of the function $f_{i-1}(t)$. For each breakpoint s of $f_{i-1}(t)$, we compute $\hat{s} = s + T_{i-1,j}$ and see if $\hat{s}$ falls between e_j and l_j. If it does, we have found a new breakpoint for $f_j(t)$. The function value $f_j(\hat{s})$ is increased by $f_{i-1}(s)$; i.e., $f_j(\hat{s})$ is set to $f_{i-1}(s) + f_j(\hat{s})$. The original value $f_j(\hat{s})$ can be computed in constant time given $f_j(e_j)$, the volume deliverable at e_j, and the usage rate U_j. The slopes of the two segments of $f_j(t)$ created by the introduction of a breakpoint at $\hat{s}$ are obtained similarly by adding the slope of the corresponding segment in $f_{i-1}(t)$ and the slope of $f_j(t)$ at $\hat{s}$. We process the breakpoints of $f_{i-1}(t)$ in time order until $\hat{s} > l_j$. The number of points added to $f_j(t)$ this way is at most $O(n)$. Because we update the piecewise function for at most $O(n)$ customers, the updating procedure has complexity $O(n^2)$.

Summarizing, the selection of which customer to insert, and where, now requires $O(n^2 \log n)$ time, and updating after an insertion can be done in $O(n^2)$, resulting in a total complexity of $O(n^3 \log n)$. Note that we do increase the memory requirements because we have to store the functions $f_j(t)$, which requires $O(n^2)$ space.

Finalizing the Solution
When all insertions have been made, there may not only be some flexibility in the timing, as in the previous versions, but also in the delivery quantities. To optimize delivery quantities, deliveries should start as late as possible. This allows more usage to occur, which increases the volume deliverable. Set all deliveries to begin at their late time, delivering the smaller of $C_i - I_i + U_i l_i$ or the current available vehicle capacity. The current available vehicle capacity is the truck capacity, Q, minus the delivery quantities already fixed, minus the minimum delivery quantities to the remaining (unfixed) customers. This enables a full-truckload delivery where possible, while ensuring that the minimum volume is delivered to all customers.

5. Fixed and Variable Delivery Time

Although it is often modeled as such, the delivery of a product is never instantaneous. There are usually check-in procedures at each customer as well as time needed to take pallets of product from a truck or to pump product into a customer's tank. In other words, there is often a *fixed* amount of delivery time where the amount of time required is independent of the size of the delivery (often called service or dwell time) and a *variable* portion that is dependent on the size of the delivery.

We let S_i represent the fixed stop time at customer i, and we model the variable portion of the delivery time with a delivery rate P_i specifying the rate at which a unit of product can be delivered from the truck to customer i. The S_i values, though influential in determining the final solution, do not impact the solution methodology. We can include the S_i values through a modification of the travel time values:

$$\overline{T}_{i,j} = T_{i,j} + S_j \quad \forall (i,j).$$

Similarly, there are often routine truck inspections done at the beginning and end of each route which can be included in the travel times to and from the depot.

Delivery rates can also be added quite easily when the delivery volume is fixed, as in the first few sections of this paper. Again, we can include the delivery time for the prescribed volume in the time it takes to reach the next customer:

$$\overline{T}_{i,j} = T_{i,j} + P_i D_i \quad \forall (i,j).$$

Therefore, if the delivery volume is fixed and there is both a fixed and variable delivery time, we can use:

$$\overline{T}_{i,j} = P_i d_i + T_{i,j} + S_j \quad \forall (i,j).$$

In this way, the e_i and l_i values will now represent the earliest and latest times that a delivery can literally *begin* at i.

The situation changes, however, when there are variable delivery quantities. With variable delivery quantities, we argued earlier that it was best to begin each delivery as late as possible so as to maximize usage and thus maximize total delivery volume. As we will see, this is not necessarily true when delivery rates are added. For simplicity and ease of presentation, we discuss the case where all customers have the same delivery rate, P.

Information Maintained
We maintain all of the values introduced earlier when discussing variable delivery quantities, plus one additional piece of information: the latest delivery time at i, t_i^g, associated with the maximum volume deliverable to i and all customers succeeding i on the route (g_i).

Checking Feasibility
To evaluate the feasibility of inserting j between $i-1$ and i, we first compute the earliest time a delivery can take place at j:

$$e_j = \max\left(e_{i-1} + D_{i-1}P + T_{i-1,j}, \frac{D_j - C_j + I_j}{U_j}\right).$$

The first term of the maximum represents the earliest start time at customer $i-1$ plus the minimum delivery time required at $i-1$ plus the travel time from $i-1$ to j. The second term of the maximum represents the time when the minimum required delivery at j will fit.

Next, we compute the latest time a delivery can begin at j:

$$l_j = \min\left(l_i - T_{j,i} - D_j P, \frac{I_j}{U_j}\right). \tag{1}$$

The late value l_j must allow for at least the minimum required volume to be delivered at j.

Feasibility is guaranteed if $D_j < Q - q_r^{\min}$ and $e_j \leq l_j$.

Computing Profitability
When considering variable delivery times and variable delivery quantities, it is again natural to have an objective based on profit. For each feasible insertion, we need to consider the increased revenue from the delivery volume as well as the related costs. Again, we cannot determine the change in maximum delivery volume for a route resulting from an insertion in constant time, but only in $O(\log n)$ time. However,

the definition of $f_i(t)$ will have to be slightly altered to account for the dueling effects of the delivery and usage rates. The function $f_i(t)$ will now represent the maximum volume deliverable at i and all preceding customers, given that the delivery at i can be completed by t.

The computation of $d_j^{\max}$, i.e., the largest volume deliverable to customer j, illustrates the difficulties arising when simultaneously considering a usage rate and a delivery rate. If l_j is determined by the first term in the minimum in Equation (1), then the volume that can be feasibly delivered at l_j is only D_j. If delivery to j begins earlier than l_j, however, more may be deliverable because of the increased time for delivery. The dueling effects of usage rate and delivery time are portrayed in Figure 2. The line with positive slope represents increasing space available due to usage; the line with negative slope represents decreasing amount deliverable due to delivery time. The value of $d_j^{\max}$ is found at the intersection of the two lines. This intersection occurs at time:

$$\frac{l_i - T_{j,i} - C_jP + I_jP}{1 + U_jP}.$$

Because the two lines may not always intersect between e_j and l_j, the full equation defining $d_j^{\max}$ is:

$$d_j^{\max} = \min\left(Q - q_r^{\min},\ C_j - I_j + \left(\frac{l_i - T_{j,i} - C_jP + I_jP}{1 + U_jP}\right)U_j,\ C_j - I_j + U_jl_j,\ (l_i - T_{j,i} - e_j)P\right).$$

The second term of the minimum represents the volume deliverable at the intersection, but the third term is the volume deliverable at l_j given the usage rate, and the fourth term is the volume at e_j given the service time available. Let $t_j^{\max}$ represent the time where $d_j^{\max}$ is deliverable.

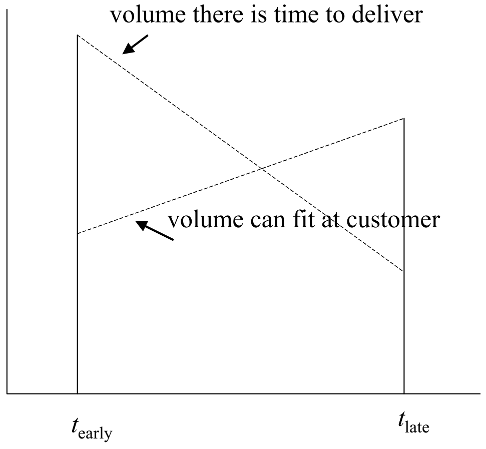

Figure 2 **Time vs. Volume Deliverable**

Given the functions $f_i(t)$ and the values g_i and t_i^g, we can compute the increase in delivery volume on a route associated with an insertion in $O(\log n)$ time. We first compute g_j. If $e_j + D_jP + T_{j,i} > t_i^g$, then

$$g_j = D_j + g_i - (e_j + D_jP + T_{j,i} - t_i^g)P,$$

because the volume deliverable to the succeeding customers will decrease at rate P after t_i^g. If $e_j + D_jP + T_{j,i} \le t_i^g$, then g_j is determined by adding g_i to the largest feasible volume deliverable at j that can be completed by $t_j^g = t_i^g - T_{j,i}$ (so that delivery at i can start at t_i^g). Summing g_j and $f_{i-1}(t_j^g - T_{i-1,j})$ will yield $\hat{q}_r^{\max}$, the new maximum delivery volume resulting from the insertion. The difference between $\hat{q}_r^{\max}$ and $q_r^{\max}$ yields the change in revenue. Because of the function evaluation involved, the objective evaluation requires $O(\log n)$ time.

Updating the Route

The updating procedures are the same as those discussed in the section on variable delivery quantities, except for the adaptations required to accommodate the delivery rate as indicated above in the description of checking feasibility and computing profitability. Also, we must update each t_k^g when we update g_k. Each piecewise linear function may now start with one or two pieces, rather than one, as in Figure 2. The total number of pieces for each function remains $O(n)$, preserving the overall $O(n^3 \log n)$ complexity of the algorithm.

Finalizing the Solution

By maintaining the t_k^g values, we always know the time at which to begin delivery to each customer k to maximize the volume deliverable to k and all succeeding deliveries. We can use these times from the last customer to the first to set the final delivery times and volumes. We can set each final delivery volume using the volume deliverable at each customer k at time t_k^g, along with the sum of the committed delivery volumes for the succeeding deliveries, the sum of the minimum delivery volumes to the preceding customers, and the truck capacity to ensure feasibility.

6. Multiple Routes per Vehicle

In practice, it is often the case that vehicles make multiple trips per day (or more generally multiple trips during the planning period). Insertion heuristics can easily be adapted to handle multiple trips per vehicle as has been shown in various articles in the literature. This last section illustrates how all of the earlier complexities we have discussed can easily be used with several trips per vehicle and maintain both feasibility and efficiency.

Suppose vehicle m performs a sequence $(r_1, r_2, \ldots, r_k, \ldots, r_{t(m)})$ of routes, where $t(m)$ is the number of trips made by vehicle m. Up to now, we have assumed that a route can start at time 0, modeled by initializing $e_0 = 0$, and that a route needs to be completed by the end of the planning period T, modeled by initializing $l_{n+1} = T$. This no longer suffices, and we need to maintain for each route the earliest time the route can start, taking into account any prior routes, i.e., $e_0^{r_1} = 0$ and $e_0^{r_k} = l_{n+1}^{r_{k-1}}$ for $k = 2, \ldots, t(m)$, and the latest time a route can be completed, taking into account any subsequent routes; i.e., $l_{n+1}^{r_{t(m)}} = T$ and $l_{n+1}^{r_k} = l_0^{r_{k+1}}$ for $k = 1, \ldots, t(m) - 1$.

Furthermore, we have to realize that a newly inserted customer impacts not only the early and late values for deliveries that come before it and after it on the route, but also the early values for deliveries for the subsequent routes on the same vehicle and the late values for deliveries for the prior routes on the same vehicle.

The ordering of the updates is important. If customer j is inserted between customer $i-1$ and i on route r_k, we first update the e values for the customers coming after j on the route. Next, we set $e_0^{r_{k+1}} = e_{n+1}^{r_k}$ and update the e values for route r_{k+1}, and repeat until we have processed route $r_{t(m)}$. Likewise, we update the l values for the customers coming before j on the route, followed by setting $l_{n+1}^{r_{k-1}} = l_0^{r_k}$, updating l values for the customers on route r_{k-1}, and repeating until we have processed r_1.

7. Conclusions

Insertion heuristics are fairly easily adaptable to a variety of practical complexities such as shift time limit, variable delivery quantities, and variable delivery times. Through careful implementations, which maintain appropriate information about partial routes, we can retain the same time complexity as the basic insertion heuristic for the standard vehicle routing problem, $O(n^3)$, or increase it slightly to $O(n^3 \log n)$. Such careful implementations are important, because most practical vehicle routing and scheduling problems contain one or more of these complexities, or even additional ones, and often involve fairly large numbers of customers.

References

Campbell, A. M. 2000. Inventory routing. Ph.D. dissertation, Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA.

de Jong, C., G. Kant, A. van Vlient. 1996. On finding minimal route duration in the vehicle routing problem with multiple time windows. Manuscript, Department of Computer Science, Utrecht University, Utrecht, The Netherlands.

Desrosiers, J., Y. Dumas, M. M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser, eds. *Network Routing. Handbooks in Operations Research and Management Science*, Vol. 8. Elsevier, Amsterdam, The Netherlands, 35–139.

Dror, M., M. Ball. 1987. Inventory/routing: Reduction from an annual to a short period problem. *Naval Res. Logist. Quart.* **34** 891–905.

Dror, M., M. Ball, B. Golden. 1985. A computational comparison of algorithms for the inventory routing problem. *Ann. Oper. Res.* **4** 3–23.

Kindervater, G. A. P., M. W. P. Savelsbergh. 1997. Vehicle routing: Handling edge exchanges. E. Aarts, J. K. Lenstra, eds. *Local Search in Combinatorial Optimization.* Wiley, Chichester, England, 337–360.

Liu, F. H., S. Y. Shen. 1999. The fleet size and mix vehicle routing problem with time windows. *J. Oper. Res. Soc.* **50** 721–732.

Potvin, J., S. Bengio. 1996. The vehicle routing problem with time windows, Part II: Genetic search. *INFORMS J. Comput.* **8** 165–172.

Potvin, J., J. M. Rousseau. 1993. Parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *Eur. J. Oper. Res.* **66** 331–340.

Rosenkrantz, D. J., R. E. Stearns, P. M. Lewis. 1977. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.* **6** 563–581.

Salhi, S., G. Nagy. 1999. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *J. Oper. Res. Soc.* **50** 1034–1062.

Solomon, M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35** 254–265.

Toth, P., D. Vigo. 2002. *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.

Vigo, D. 1996. Heuristic algorithm for the asymmetric capacitated vehicle routing problem. *Eur. J. Oper. Res.* **89** 108–126.