Available online at www.sciencedirect.com
SCIENCE DIRECT®
Transportation Research Part E 41 (2005) 131–144
TRANSPORTATION RESEARCH PART E
www.elsevier.com/locate/tre

# A vehicle routing problem with backhauls and time windows: a guided local search solution

Yingjie Zhong [a,1], Michael H. Cole [b,*,1]

[a] *Motorola, Inc., Schaumburg, IL 60196, USA*
[b] *Department of Mechanical and Industrial Engineering, Montana State University, Bozeman, MT 59717, USA*

Received 20 February 2003; received in revised form 16 October 2003; accepted 9 December 2003

**Abstract**

This paper presents a guided local search heuristic to solve a vehicle routing problem with backhauls and time windows. The VRPBTW with and without customer precedence are both considered. Customer precedence requires that all linehaul customers be visited before any backhaul customer. The basic approach is to construct an initial infeasible solution and then use a guided local search to improve the solution feasibility and quality. A new technique—section planning—is used to enhance the feasibility. Computational results show that the new heuristic can solve problems in which customers are distributed normally or in clusters. Some of the results are better than the best solutions that have appeared in the literature.


*Keywords:* Vehicle routing; Time windows; Metaheuristic

## 1. Introduction

The Vehicle Routing Problem with Backhauls and Time Windows (VRPBTW) can be stated as follows: A set of customers with deterministic demand, deterministic time windows, and certain types of service requirements (pick up and/or delivery) must be served by a homogeneous fleet of vehicles with fixed capacity starting from and ending at a central depot, which also has a certain time horizon. For each service requirement (pickup or delivery), each customer must be assigned exactly once to a vehicle. The customers are classified into two groups: linehaul customers whose

* Corresponding author. Tel.: +1-406-994-1399; fax: +1-406-994-6292.
*E-mail address:* mcole@ie.montana.edu (M.H. Cole).
[1] Formerly at the University of Arkansas.

1366-5545/$ - see front matter 
doi:10.1016/j.tre.2003.12.003

demand needs to be delivered, and backhaul customers whose demand needs to be picked up (if a customer requires both pickup and delivery, it is modeled as two separate customers). In this paper, a vehicle is allowed to arrive at a customer before the relevant time window, but cannot service the customer until the time window opens. The VRPBTW has two objectives: minimize the number of routes and minimize the sum total distance of all the routes. The VRPBTW can be classified into two cases. In the VRPBTW with customer precedence, all of the linehaul customers must be served before any backhaul customers in each route. In the VRPBTW without customer precedence, linehaul and backhaul customers can be interspersed on a given route.

This research develops a heuristic approach to the VRPBTW. Optimal methods are not appropriate since the traditional VRP is an NP-hard problem and, in general, adding time windows and backhauls does not simplify the problem. The new heuristic is a cluster-first, route-second algorithm, with most effort spent on routing. The two phases are

- *Phase 1*—Use an adapted sweep algorithm (Fisher and Jaikumar, 1981) to generate an initial infeasible solution. Use GLSA, a guided local search heuristic (based partly on Voudouris and Tsang, 1999) to manage 2-opt, 1-move, and 1-exchange (a new method) to improve the initial solution.
- *Phase 2*—Enhance feasibility using a new technique called section planning. This technique inserts new routes until feasibility is attained, and arranges customers within routes to reduce travel distance. This phase is iterative with feasibility constraints being "soft" in early iterations and "hard" in later iterations.

The new heuristic was originally developed for the VRPBTW without customer precedence. After promising results for problems without customer precedence, we extended the heuristic to consider the VRPBTW with customer precedence.

## 2. Literature review

A comprehensive review of the VRP can be found in Bodin et al. (1983) and Ball et al. (1995). Useful techniques for the general VRP are outlined in Golden and Assad (1988) and Aarts and Lenstra (1997). Reeves (1993) covers modern techniques such as simulated annealing, tabu search, and genetic algorithms. Various heuristic methods may be found in the literature for both the VRPTW (Potvin et al., 1996b; Potvin and Bengio, 1996; Russell, 1995; Chiang and Russell, 1997) and the VRPB (Casco et al., 1988; Deif and Bodin, 1984; Golden et al., 1985; Goetschalckx and Jacobs-Blecha, 1989; Jacobs-Blecha and Goetschalckx, 1993; Toth and Vigo, 1996).

Several papers deal specifically with the VRPBTW. Gelinas et al. (1995) propose a new branching strategy for branch-and-bound approaches based on column generation. This algorithm finds optimal solutions to different test problems with up to 100 customers. Potvin et al. (1996a) design a genetic algorithm to identify an ordering of customers that produces good routes. Duhamel et al. (1997) design a tabu search heuristic for the VRPBTW with customer precedence. First a feasible solution is constructed by an adapted version of Solomon's (1987) insertion heuristic. Then tabu search and local search and improvement algorithms are used to improve the solution. All three of these papers detail experiments on problems with customers distributed

normally within the service area. Kontoravdis and Bard (1995) describe a greedy randomized adaptive search procedure (GRASP) for VRPTW. This algorithm is also capable of solving VRPBTW without customer precedence. The authors report experimental results for problems with customers distributed in clusters. Cheung and Hang (2003) develop label matching algorithms for solving the VRPBTW. Their heuristic approach can handle the addition of complex real-world constraints, such as vehicles of different capacities and penalties for vehicles that arrive early.

Voudouris and Tsang (1999) develop the GLS (guided local search) metaheuristic and apply it to the traveling salesman problem. The GLS penalizes solution features (e.g., active arcs) during each iteration based on the value of a utility function. The penalty acts as a disturbance to an augmented objective function which is adjusted during each iteration. This reduces the chance that the solution procedure will get stuck in a local optimum.

## 3. Algorithm description

### *3.1. Solution feasibility consideration*

In recent years, VRP researchers investigating additional constraints, such as time windows or backhauls, have concentrated on postponing solution feasibility (Chiang and Russell, 1997). Logically, this concern is a perfect match for simulated annealing. However, our consideration of solution infeasibility is different from that of some researchers. Often, infeasibility is taken to mean that there exist unrouted customers, but that all customers in routes are in feasible routes. In this paper, the routes themselves may be infeasible. The local search heuristics in Phase 1 enhance the feasibility of the solution. The section planning technique in Phase 2 enforces feasibility by inserting a new route into the solution during each iteration. In the most extreme case, the heuristic would generate a separate route for each customer. Thus, a feasible solution is guaranteed.

### *3.2. Guided local search approach (GLSA)*

This subsection develops an augmented objective function and penalty function for the VRPBTW.

#### *3.2.1. Augmented objective function*

The augmented VRPBTW objective function comprises the usual VRP objective function, Lagrangean multipliers that penalize constraint violations, and a disturbance item derived from the GLS. Eq. (1) is the augmented objective function for the VRPBTW without customer precedence. The augmented objective function for the VRPBTW with customer precedence is identical except for an added term related to precedence violations, $\lambda_p v_p$.

$$h(S) = g(S) + \lambda_t v_t + \lambda_c v_c + \lambda \sum_{ij \in S} p_{ij} I_{ij}(S) \tag{1}$$

$g(S)$ typical VRP cost function
$v_t$ violation of due-time constraints
$v_c$ violation of capacity constraints
$\lambda_t$ and $\lambda_c$ Lagrangean multipliers
$\lambda$ multiplier indicating the magnitude of the total disturbance due to arc penalties
$p_{ij}$ penalty associated with arc $ij$
$I_{ij}(S)$ 1 if arc $ij$ is in solution $S$, 0 otherwise

Parameters $\lambda_t$ and $\lambda_c$ should increase after each iteration in order to more strongly encourage feasibility. The value of $p_{ij} \cdot I_{ij}(S)$ should eventually decrease to zero. Parameter $\lambda$ should decrease after each iteration, as well. Note that the augmented objective function does not penalize violations of ready-time constraints. Although customers can only be served within their time windows, vehicles are allowed to arrive early and wait without penalty.

### 3.2.2. Penalty function

The penalty $p_{ij}$ depends on three factors: the arc length, the arc time precedence, and a dynamic utility function. An arc's length $d_{ij}$ is the travel distance between nodes $i$ and $j$. An arc's time precedence $t_{ij}$ is calculated as described in Fig. 1 (the hard-coded numbers are included to give a sense of scale).

The initial value of each arc's penalty, $p_{ij}$, is set equal to the arc's time-precedence. It is updated in each iteration as described in Fig. 2 (the hard-coded numbers are included to give a sense of scale). If an arc develops a very large penalty, it is deleted from future consideration. This eases the computational burden associated with evaluating the inclusion of the arc into future solutions.

## 3.3. Solution approach

This section introduces the solution approach for both the VRPBTW with precedence and the VRPBTW without precedence. The algorithm has two phases.

```
Set t = d_ij / vehicle speed

If (t + customer i ready time > customer j due time)
      t_ij = 200

Else if (t + customer i ready time > customer j ready time)
      t_ij = 0

Else
      t_ij =  20
End if

If (t + customer i due time < customer j due time)
      t_ij -= 20

Else if (t + customer i due time > customer j due time)
      t_z  += 20

End if
```

Fig. 1. Pseudocode for calculating time precedence of arc $ij$.

```
Set  u = I_ij(S) · (d_ij · t_ij) / (1 + p_ij)

If u > 9
      p_ij += 5
End if

if p_ij > 60
      Set I_ij = 0 for remainder of algorithm
End if
```

Fig. 2. Pseudocode for calculating penalty value of arc *ij* during each iteration.

*3.3.1. Phase 1*

Phase 1 has four stages:

1. Clustering and initial route construction.
2. Intra-route improvement: 2-opt.
3. Inter-route improvement: 1-move.
4. Inter-route improvement: 1-exchange.

*3.3.1.1. Clustering and initial route construction*. Clustering and initial route construction are detailed in Fig. 3 (based, in part, on Fisher and Jaikumar, 1981).

*3.3.1.2. Route improvement*. This section describes how GLSA guides local search heuristics. The composite heuristic improves the initial infeasible solution into a much more ordered structure. A local search heuristic is designed for searching for a high quality solution in a shrunken solution

```
For i = 1 to NumberOfCustomers
   Determine polar angle in relation to depot
Next i

For i = 1 to 50
   Select random start angle from U[0,pi/2]
   Select random cluster capacity from U[0.6,1.0]*vehicle capacity

   Sweep customers into clusters

   For j = 1 to NumberOfClusters
      Divide cluster into two zones of equal polar angle
      Assign each customer to the route, alternating between zones
   Next j
Next i

Select initial solution from the 50 candidates such that the number
of routes is low, and violations of customer due times and
precedence are low
```

Fig. 3. Pseudocode for initial route construction.

space—a neighborhood. In solving a complicated problem, a set of local search heuristics based on different neighborhoods is necessary to increase the search efficiency and effectiveness. The general idea is to use simple local search heuristics followed by more sophisticated local search heuristics to yield a better solution quality within a reasonable run time.

This section discusses three local search heuristics that can be categorized into two classes: intra-route exchange and inter-route exchange. 2-opt and 1-move are common arc exchange heuristics that are frequently used in solving the VRP, while 1-exchange is a new heuristic designed particularly for the VRPBTW.

*3.3.1.3. Intra-route exchange: 2-opt*. 2-opt (Croes, 1958) is a local search heuristic for the traveling salesman problem (TSP) and other combinatorial problems. 2-opt is commonly used in the VRP without time windows. Potvin et al. (1996b) develop 2-opt* in order to maintain time window feasibility when solving the VRPTW. In Phase 1, 2-opt is sufficient because maintaining solution feasibility is not necessary. 2-opt is used to decrease the length, time precedence violation, and customer precedence violations (for the VRPBTW with customer precedence) of each route. 2-opt does not use the augmented VRPBTW objective function. The neighborhood of 2-opt is all pairs of arcs in a particular route.

*3.3.1.4. Inter-route exchange: 1-move, 1-exchange*. Inter-route exchange involves the augmented objective function. The GLS first sets the initial value of the Lagrangean multipliers, and then performs 1-move and 1-exchange for a predetermined number of iterations.

*3.3.1.5. 1-move*. 1-move is the same heuristic as operators $(1,0)$ and $(0,1)$ proposed by Chiang and Russell (1997). The 1-move procedure effectively improves the quality and feasibility of poor solutions. 1-move deletes one node from a route and inserts it into another route. The evaluation of 1-move is decided by the length and time precedence of the arc, and the time and capacity violations change caused by 1-move. The neighborhood of 1-move is all the customers and any potential customer positions for all pairs of routes in the current solution.

*3.3.1.6. 1-exchange*. The 1-exchange heuristic deletes two customers from two different routes and inserts each customer into the best position in the other route. In Fig. 4, customer $x$ in route 1 connects with arcs $(x_1,x)$ and $(x,x_2)$, and customer $y$ in route 2 connects with arcs $(y_1,y)$ and $(y,y_2)$.

1-exchange first deletes the two customers, $x$ and $y$, from route 1 and route 2 respectively. Then, customer $x$ searches for a good position in route 2 and customer $y$ searches for a good position in route 1. The best combination of these two insertions (if the result has a better objective function value than the original configuration) is the result of 1-exchange. The best position for customer $x$ in route 2 is after customer $y_2$ and the best position for customer $y$ in route 1 is before customer $x_1$.

1-exchange intensifies the local search. The neighborhood of 1-exchange is larger than that of 1-move since each of the customers needs to find its best position in the other route. To reduce the computational burden, if the length of the new arcs is more than twice that of the deleted arcs, the candidate exchange is cancelled before extensive evaluation of time windows and capacity constraints.

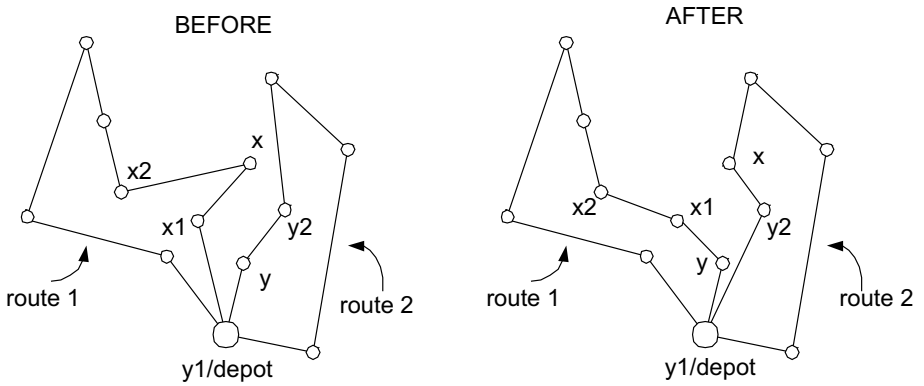


Fig. 4. Before and after 1-exchange of nodes $x$ and $y$.

*3.3.2. Phase 2*

Phase 2 is an iterative procedure that adds a user-specified number of new routes in an effort to achieve feasibility (see Fig. 5). Each iteration has two main steps: route feasibility enhancement and route improvement. The feasibility enhancement step deletes customers that create infeasibilities and adds a new route comprising the deleted customers. Section Planning (described below) is a key part of feasibility enhancement. The improvement step applies the 2-opt, 1-move, and 1-exchange heuristics.

*3.3.2.1. Section Planning for route feasibility enhancement*. Section Planning divides each route into several sections and tries to delete the customer in each section that contributes most to the violation of due-time and capacity constraints (see Fig. 6). Considering a route section by section is computationally easier than considering the route as a whole.

```
For k = 1 to NumberOfNewRoutes(a parameter)
      For i = 1 to NumberOfRoutes
            Delete precedence violating customers
            Delete "big" violators of capacity and time windows
            Do Section Planning
      Next i
      Add one new route comprising deleted customers
      Update info

      For j = 1 to NumberOfIterations (a parameter)
            Do route improvement:
                2-opt
                1-move
                1-exchange (every other iteration)
      Next j
Next k
```

Fig. 5. Pseudocode for Phase 2.

```
If a capacity-violating customer exists then

  Divide the route into sections divided by the violating customer(s)

  For each section

   If (no customers violate due time in this section or next section)

      Delete the linehaul or backhaul customer in the section that best
      reduces the capacity violation

   Else If (a customer violates due time in this section)

     Let n = customer with largest due time violation in this section

     If (find backhaul customer before n that best reduces capacity
         violation)
       Delete the backhaul customer
     End If

   Else If (a customer violates due time in the next section)

      Let n = customer with largest due time violation in next section

     If (find linehaul customer before n that best reduces capacity
         violation)
       Delete the linehaul customer
     End If

   End If

  Next

Else If a ready-time-violating customer exists then

  Divide the route into sections divided by the violating customer(s)

  For each section

   If (a customer violates due time in this section)
     Let n = customer with largest due time violation in this section
     Delete the customer before n that best reduces the due time violation
   End If

  Next

Else If a due-time-violating customer exists then

   [treat entire route as a single section]

   Let n = customer with largest due time violation in the entire route
   Delete the customer before n that best reduces the due time violation

Else
  This route is feasible

End If
```

Fig. 6. Pseudocode for section planning a route.

The algorithm first defines sections such that each section includes exactly one customer that violates capacity restrictions. If any customer in the current section or following section also

violates due-time, the algorithm deletes a customer that alleviates both the capacity violation and the due-time violation. Otherwise, the algorithm deletes a customer that alleviates only the capacity violation. If the route has no capacity violations, the algorithm defines sections according to ready-time or due-time violations, and deletes the single customer that best alleviates the worst violation of due-time.

## 4. Computational experiments

The two versions of the GLSA heuristic (without customer precedence and with customer precedence) were programmed in Microsoft Visual C++ 6.0 and executed on a 450 MHz PC. GLSA was tested on problems sets derived from Solomon's data sets R1, R2, C2, and RC2. In Solomon's original data sets, all the problems are Euclidean with a single service type. As described by Kontoravdis and Bard (1995):

- Each data set contains between eight and twelve 100-node problems over a service area defined on a 100 by 100 grid.
- For R1 and R2, the customers are located uniformly over the service area. Set C2 has clustered customers and set RC2 has a combination of clustered and randomly placed customers.
- R1 has a short scheduling horizon and a vehicle capacity of 200 units; R2, C2 and RC2 have a long scheduling horizon and vehicle capacity of 1000, 700, and 1000 units respectively.
- The time window constraints in problem set R1 allow only a small number of customers to be served by each vehicle. The opposite is true for R2, C2, and RC2. In both cases, the vehicle capacity constraints are fairly loose.

### *4.1. VRPBTW without customer precedence*

For the VRPBTW without customer precedence, GLSA was tested on data sets developed by Kontoravdis and Bard (1995). Kontoravdis constructed data sets MR2, MC2, and MRC2 by designating each customer in Solomon's R2, C2, and RC2 data sets as either linehaul or backhaul with equal probability. Their experiments are restricted to derivatives of R2, C2, and RC2 because "time window constraints are dominant for R1, C1 and RC1, and thus the solution state space would not be greatly affected after changing half of the customers to backhaul." They reduced the vehicle capacity to 250 units to "assure that the capacity constraint had the primary influence on feasibility."

Tables 1–3 compare the performance of GLSA algorithm to the performance of the GRASP algorithm developed by Kontoravdis and Bard (1995). Two different GLSA runs were performed on each problem and the best solution was selected at the end. The tables show three measures of performance: number of routes, total route distance, and CPU time (in seconds). Note that the individual GRASP solution measures are not available, and are thus represented by averages. The CPU times are not directly comparable, since computing power has increased considerably since Kontoravdis and Bard implemented their algorithm on a Sun Sparc-10 workstation in 1995.

Table 1
Results for MC2 data set

| Problem | GRASP | | | GLSA | | |
|---|---|---|---|---|---|---|
| | Routes | Distance | CPU time | Routes | Distance | CPU time |
| MC201 | – | – | – | 5 | 763.88 | 57 |
| MC202 | – | – | – | 4 | 1186.24 | 347 |
| MC203 | – | – | – | 4 | 1096.31 | 323 |
| MC204 | – | – | – | 4 | 885.73 | 431 |
| MC205 | – | – | – | 5 | 781.70 | 169 |
| MC206 | – | – | – | 5 | 860.74 | 352 |
| MC207 | – | – | – | 5 | 792.96 | 476 |
| MC208 | – | – | – | 5 | 859.92 | 393 |
| Average | 4 | 1094.94 | 130.1 | 4.625 | 903.56 | 319 |

Table 2
Results for MR2 data set

| Problem | GRASP | | | GLSA | | |
|---|---|---|---|---|---|---|
| | Routes | Distance | CPU time | Routes | Distance | CPU time |
| MR201 | – | – | – | 4 | 1388.73 | 108 |
| MR202 | – | – | – | 4 | 1198.99 | 422 |
| MR203 | – | – | – | 4 | 988.92 | 544 |
| MR204 | – | – | – | 4 | 858.32 | 450 |
| MR205 | – | – | – | 4 | 1172.53 | 253 |
| MR206 | – | – | – | 4 | 979.50 | 406 |
| MR207 | – | – | – | 4 | 912.69 | 451 |
| MR208 | – | – | – | 4 | 764.52 | 408 |
| MR209 | – | – | – | 4 | 978.82 | 516 |
| MR210 | – | – | – | 4 | 1061.36 | 557 |
| MR211 | – | – | – | 4 | 878.81 | 692 |
| Average | 4 | 1168.53 | 122.3 | 4 | 1016.66 | 437 |

Table 3
Results for MRC2 data set

| Problem | GRASP | | | GLSA | | |
|---|---|---|---|---|---|---|
| | Routes | Distance | CPU time | Routes | Distance | CPU time |
| MRC201 | – | – | – | 5 | 1498.90 | 73 |
| MRC202 | – | – | – | 4 | 1539.41 | 493 |
| MRC203 | – | – | – | 4 | 1303.48 | 713 |
| MRC204 | – | – | – | 4 | 932.48 | 472 |
| MRC205 | – | – | – | 4 | 1632.04 | 362 |
| MRC206 | – | – | – | 4 | 1433.43 | 208 |
| MRC207 | – | – | – | 4 | 1217.20 | 599 |
| MRC208 | – | – | – | 4 | 1085.57 | 1134 |
| Average | 4.5 | 1496.91 | 135.7 | 4.125 | 1330.31 | 507 |

For data set MC2 (Table 1), GLSA generated more routes than GRASP, but did get a total distance that is around 19% shorter. For set MR2 (Table 2), GLSA got the same number of routes as GRASP, and saved about 15% in total route distance. For set MRC2 (Table 3), GLSA generated fewer routes and also reduced the total distance about 10%. So, for these three data sets, GLSA outperforms GRASP with respect to route distance, and seems to perform at approximately the same level as GRASP with respect to number of routes.

GRASP was primarily designed to minimize the number of routes, so it is not overly surprising that the new GLSA algorithm outperforms GRASP with respect to total route distance for all three sample data sets. Whether GLSA's additional computation time is worth the reduction in total distance depends on the particular case. More interestingly, neither GLSA nor GRASP was consistently better at minimizing the number of routes. This indicates that both algorithms should be run if the primary goal is to minimize the total number of routes while secondarily minimizing the total route distance.

### 4.2. VRPBTW with customer precedence

For the VRPBTW with customer precedence, GLSA was tested on data sets developed by Gelinas et al. (1995). As discussed in Duhamel et al. (1997), Gelinas et al. modified the first five problems of Solomon's R1 data set by randomly choosing 10%, 30%, and 50% of the 100 clients to be backhaul customers, leaving other attributes unchanged. They generated additional test problems by considering only the first 25 and first 50 nodes. The total number of test problems for the VRPBTW with customer precedence is 45.

Representative results are reported in Tables 4 and 5. Five different runs were performed on each problem and the best solution was selected at the end. Potvin et al. (1996a) results with a genetic algorithm GA (run on a Sun Sparc-10 workstation) are also reported. The tables show three measures of performance: number of routes, total route distance, and CPU time (in seconds). Note that the computation times are not directly comparable since computing power has increased significantly since 1996.

Table 4
Results for problems derived from R101

| Number of customers | Backhaul% | Optimal solution | Potvin's GA | | | GLSA | | |
|---|---|---|---|---|---|---|---|---|
| | | | Routes | Distance | CPU time | Routes | Distance | CPU time |
| 25 | 10 | 643.4 | 9 | 643.4 | 5.9 | 9 | 665.24 | 2 |
| | 30 | 711.1 | 10 | 721.8 | 5.6 | 10 | 723.19 | 4 |
| | 50 | 674.5 | 10 | 682.3 | 5.6 | 10 | 684.52 | 3 |
| 50 | 10 | 1122.3 | 14 | 1138.1 | 16.2 | 14 | 1178.73 | 6 |
| | 30 | 1191.5 | 16 | 1192.7 | 16.9 | 16 | 1332.44 | 8 |
| | 50 | 1168.6 | 16 | 1183.9 | 17.5 | 16 | 1237.65 | 6 |
| 100 | 10 | 1767.9 | 23 | 1815.0 | 169.6 | 24 | 1848.04 | 25 |
| | 30 | 1877.6 | 23 | 1896.6 | 163.3 | 24 | 2034.61 | 27 |
| | 50 | 1895.1 | 24 | 1905.9 | 211.6 | 25 | 2057.05 | 27 |

Table 5
Results for problems derived from R105

| Number of customers | Backhaul% | Optimal solution | Potvin's GA | | | GLSA | | |
|---|---|---|---|---|---|---|---|---|
| | | | Routes | Distance | CPU time | Routes | Distance | CPU time |
| 25 | 10 | 565.1 | 7 | 565.1 | 5.6 | 7 | 577.85 | 2 |
| | 30 | 623.5 | 8 | 630.2 | 6.8 | 8 | 635.62 | 6 |
| | 50 | 591.1 | 7 | 592.1 | 5.4 | 7 | 618.20 | 3 |
| 50 | 10 | 970.6 | 10 | 1002.5 | 14.9 | 10 | 1041.89 | 2 |
| | 30 | 1007.5 | 11 | 1047.8 | 15.2 | 10 | 1089.00 | 8 |
| | 50 | 993.4 | 11 | 1018.0 | 15.4 | 11 | 1048.87 | 3 |
| 100 | 10 | – | 17 | 1621.0 | 191.4 | 17 | 1590.54 | 34 |
| | 30 | – | 16 | 1652.8 | 210.6 | 17 | 1667.92 | 55 |
| | 50 | – | 18 | 1706.7 | 160.4 | 19 | 1699.88 | 64 |

Table 4 shows results for problems derived from the R101 data set. For this problem set, the GLSA solutions were typically a few percent worse than the GA solutions, which were a few percent worse than optimal. The results were similar for problems derived from data sets R102, R103, and R104.

Table 5 shows results for problems derived from R105 data set. GLSA underperformed Potvin's GA except for two cases. For the 100 customer/10% backhaul case, GLSA yielded a solution with the same number of routes but 1.9% lower total distance. For the 100 customer/50% backhaul case, GLSA yielded a solution with one more route but 0.4% lower total distance.

In contrast to the case of the VRPBTW problem without customer precedence, GLSA did not perform very well compared to existing algorithms for the VRPBTW with customer precedence. However, GLSA did outperform the GA algorithm in one instance. Additional research will be needed before adding GLSA to the toolkit of problem-solving techniques for the VRPBTW with customer precedence.

## 5. Conclusions

GLSA is a simple heuristic for solving the vehicle routing problem with backhauls and time windows (VRPBTW). The first phase of the GLSA heuristic uses guided local search to improve routes, but still allows time and capacity violations. The second phase of GLSA uses a new technique, Section Planning, to eliminate time and capacity violations by selectively moving problematic customers to newly created routes. We compared GLSA to GRASP, developed by Kontoravdis and Bard (1995) and to GA, developed by Potvin et al. (1996a).

GLSA was originally developed for the VRPBTW without customer precedence. On a series of test data sets, GLSA provided solutions with 10–19% lower total distance, but occasionally more routes, than GRASP. GLSA requires more computational effort than GRASP, but the additional effort might be a reasonable tradeoff for better solution quality. Since the computational burden of GLSA and other modern heuristics is relatively small and since no heuristic is guaranteed to be

best for a particular data set, researchers and practitioners are advised to solve problems using several different heuristics.

We extended GLSA to consider the VRPBTW with customer precedence. GLSA did not perform as well as GA on this class of VRPBTW, although its solution values were generally within 5% of published best results. This indicates that customer precedence is a major distinguishing factor of VRPBTW problems, and that problems with customer precedence might require very different solution approaches than problems without customer precedence.

In summary, GLSA makes the following contributions to the VRPBTW literature:

- GLSA introduces the Section Planning technique for eliminating time and capacity violations within a route.
- GLSA outperforms the previous GRASP algorithm in minimizing the total route distance for VRPBTW problems without customer precedence. GLSA does not consistently underperform or outperform GRASP in minimizing the number of routes. This indicates that neither algorithm should be used alone to solve such VRPBTW problems.
- GLSA generally underperforms the existing GA algorithm for VRPBTW problems with customer precedence. However, GLSA occasionally does find a better solution than GA. After additional development, GLSA might be a useful algorithm to run in conjunction with GA.

## Acknowledgements

The authors would like to thank the anonymous referees for their helpful comments. This research was supported by the Mack-Blackwell Transportation Center at the University of Arkansas.

## References

Aarts, C., Lenstra, J. (Eds.), 1997. Local Search in Combinatorial Optimization. John Wiley & Sons, New York.

Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (Eds.), 1995. Network Routing. Elsevier Science, Amsterdam.

Bodin, L.D., Golden, B.L., Assad, A., Ball, M.O., 1983. Routing and scheduling of vehicles and crews: the state of the art. Computers & Operations Research 10 (2), 63–211.

Casco, D., Golden, B., Wasil, E., 1988. Vehicle routing with backhauls: models, algorithms and case studies. In: Golden, B., Assad, A. (Eds.), Vehicle Routing: Methods and Studies. North-Holland, Amsterdam, pp. 127–147.

Chiang, W., Russell, R., 1997. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. INFORMS Journal on Computing 9 (4), 417–430.

Cheung, R.K., Hang, D.D., 2003. Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls. IIE Transactions 35 (3), 191–205.

Croes, A., 1958. A method for solving traveling salesman problems. Operations Research 5, 791–812.

Deif, I., Bodin, L.D., 1984. Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In: Didder, A. (Ed.), Proceedings of the Babson Conference on Software Uses in Transportation and Logistic Management, Babson Park, pp. 75–96.

Duhamel, C., Potvin, J., Rousseau, J., 1997. A tabu search heuristic for the vehicle routing problem with backhauls and time windows. Transportation Science 31 (1), 49–59.

Fisher, M., Jaikumar, R., 1981. A generalized assignment heuristic for vehicle routing. Networks 11, 109–124.
Gelinas, S., Desrochers, M., Desrosiers, J., Solomon, M., 1995. A new branching strategy for time constrained routing problems with application to backhauling. Annals of Operations Research 61, 91–109.
Goetschalckx, M., Jacobs-Blecha, C., 1989. The vehicle routing problem with backhauls. European Journal of Operational Research 42, 39–51.
Golden, B., Assad, A. (Eds.), 1988. Vehicle Routing: Methods and Studies. Elsevier Science Publishers, North-Holland, Amsterdam.
Golden, B., Baker, E., Alfaro, J., Schaffer, J., 1985. The vehicle routing problem with backhauling: two approaches. In: Hammesfahr, R. (Ed.), Proceedings of the 21st Annual Meeting of SE TIMS, Myrtle Beach, South Carolina, pp. 90–92.
Jacobs-Blecha, C., Goetschalckx, M., 1993. The vehicle routing problem with backhauls: properties and solution algorithms. Technical report MHRC-TR-88-13, Georgia Institute of Technology, Atlanta, Georgia.
Kontoravdis, G., Bard, J., 1995. A GRASP for the vehicle routing problem with time windows. ORSA Journal on Computing 7 (1), 10–23.
Potvin, J., Bengio, S., 1996. The vehicle routing problem with time windows, part II: genetic search. INFORMS Journal on Computing 8 (2), 165–172.
Potvin, J., Duhamel, C., Guertin, F., 1996a. A genetic algorithm for vehicle routing with backhauling. Applied Intelligence 6, 345–355.
Potvin, J., Kervahut, T., Garcia, B., Rousseau, J., 1996b. The vehicle routing problem with time windows, part I: tabu search. INFORMS Journal on Computing 8 (2), 158–164.
Reeves, C. (Ed.), 1993. Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publications, Oxford.
Russell, R., 1995. Hybrid heuristics for the vehicle routing problem with time windows. Transportation Science 29 (2), 156–166.
Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 35, 254–265.
Toth, P., Vigo, D., 1996. A heuristic algorithm for the vehicle routing problem with backhauls. In: Bianco, L., Toth, P. (Eds.), Advanced Methods in Transportation Analysis: Proceedings of the 2nd TRISTAN Conference. Springer, Berlin.
Voudouris, C., Tsang, E., 1999. Guided local search and its application to the traveling salesman problem. European Journal of Operational Research 113, 469–499.