

The probabilistic traveling salesman problem with time windows

Stacy A. Voccia · Ann M. Campbell · Barrett W. Thomas

Received: 30 July 2012 / Accepted: 16 January 2013 / Published online: 14 February 2013
© Springer-Verlag Berlin Heidelberg and EURO - The Association of European Operational Research Societies 2013

Abstract With time-definite services occupying a large part of the delivery business, the explicit consideration of time windows into a route design has the potential to reduce transportation costs and the penalty costs associated with late deliveries. In this paper, we incorporate time windows into a priori routes by introducing the probabilistic traveling salesman problem with time windows (PTSPTW). The PTSPTW is an extension of the well-known probabilistic traveling salesman problem, where in addition to stochastic customer presence, each customer has an associated time window during which deliveries must be made. We present a recourse model and a variable neighborhood search with variable neighborhood descent algorithm to solve problem instances. We also present computational experiments that demonstrate the value of incorporating stochasticity into the problem.

Keywords Traveling salesman problem · Probabilistic · Time windows

Electronic supplementary material The online version of this article (doi:[10.1007/s13676-013-0018-0](https://doi.org/10.1007/s13676-013-0018-0)) contains supplementary material, which is available to authorized users.

S. A. Voccia · A. M. Campbell (✉) · B. W. Thomas
Department Of Management Sciences, University Of Iowa, 108 John Pappajohn Business Building,
Iowa City, IA 52242-1994, USA
e-mail: ann-campbell@uiowa.edu

S. A. Voccia
e-mail: stacy-voccia@uiowa.edu

B. W. Thomas
e-mail: barrett-thomas@uiowa.edu

Introduction

Time-definite delivery plays a crucial role in the shipping industry. Now more than ever, businesses are operating with lean production strategies and just-in-time inventories. With this trend, freight is shipped in smaller lot sizes and the predictability of arrival times is critical (Schultz 2008). Randy Guidry, communications coordinator for Averitt Express, notes “Every year, more and more of our customers are requesting appointments within a delivery window” (Terreri 2011). In 2010, time-definite, day definite and same day delivery services accounted for 51.4 % of the United States shipping industry’s total market value (Datamonitor 2010b). In the European market, these segments accounted for 62.3 % in 2010, a 10.6 % increase from 2009 (Datamonitor 2010a, 2009).

With time-definite services occupying a large part of the delivery business, the explicit consideration of time windows into a route design has the potential to reduce transportation costs and the penalty costs associated with late deliveries. Currently, many companies employ a pre-planned, or a priori, route which identifies an ordering of all possible customers that a particular driver may need to visit. The driver then skips those customers on the route who do not require a delivery on that day. A priori routes can be implemented with relative ease and are an alternative to the high cost of re-optimization. In addition, a priori routes offer both drivers and customers consistency and help to improve driver efficiency as the driver becomes familiar with the route. Although businesses place high importance on delivery time windows, the consideration of time windows in a priori routing has received little attention in the literature. Additionally, a priori policies can be important subproblems in dynamic routing problems (Goodson et al. 2012; Manni 2009). For these reasons, we examine the impact of time windows on a priori route design.

We incorporate time windows into a priori routes by introducing the probabilistic traveling salesman problem with time windows (PTSPTW). While methods have been developed to address such elements as stochastic travel or service times within a time window constrained environment, the literature does not address how to deal with stochastic customer presence within this same environment. For this reason, we maintain the presence of the customer as the only stochastic element in our model.

The PTSPTW can be considered an extension of the probabilistic traveling salesman problem (PTSP). The PTSP is the problem of finding a minimum expected cost a priori tour through a set of customers $N = \{i \mid 1, \dots, n\}$ with probabilities $P = \{p_i \mid 1, \dots, n\}$ of requiring service on any given day. The travel time between any two customers i and j is given by d_{ij} , where $d_{ij} = d_{ji}$. These travel times also serve as the cost to traverse each arc. In the PTSPTW, also associated with each customer $i \in N$ is a time window with an earliest time of service e_i and latest time of service l_i , where service must begin at or after e_i but should not begin later than l_i . If arrival at customer i occurs prior to e_i , then service is postponed until e_i .

Time windows present particularly challenging modeling issues when considered in conjunction with stochastic customer presence. This paper’s primary contribution is the presentation of a recourse model that incorporates time windows into a routing problem with stochastic customer presence. As in previous work on the probabilistic traveling salesman problem with deadlines (PTSPD), our recourse model accounts

for late deadline violations by penalizing any violations in the objective function. Unlike the PTSPD however, our model for the PTSPTW also accounts for early arrivals. This is an extension that requires nontrivial calculations to find the probability of early arrivals at each customer. A secondary contribution of this paper is the results from computational experiments that identify the circumstances under which PTSPTW solutions differ from the solutions for the TSPTW. The TSPTW is the deterministic analog of the PTSPTW. Ultimately, this knowledge can help practitioners to determine when the added computational burden of incorporating stochasticity into the model is justified.

This paper is organized as follows: In “[Literature review](#)”, we survey the related literature. In “[Recourse Model](#)” and “[Solution Approach](#)”, we present a recourse model and an algorithm to solve problem instances. We describe our experimental design in “[Experimental Design](#)” and present results of computational experiments in “[Results](#)”. “[Conclusions](#)” summarizes our work and offers directions for future research.

Literature review

The PTSPTW is related to both the PTSP and stochastic vehicle routing problems with time constraints. Jaillet (1988) introduces the PTSP. Campbell and Thomas (2008) introduce and propose models for the PTSPD, and Campbell and Thomas (2009) offer tractable ways to solve the PTSPD. These papers on the PTSPD summarize related literature prior to 2008.

A number of recent papers have been published on the PTSP and related problems. A subset of these papers focus on solving the PTSP through local search methods (Marinakis and Marinaki 2010; Balaprakash et al. 2009; Marinakis and Marinaki 2009; Liu 2008; Marinakis et al. 2008). Birattari et al. (2008) and Weyland et al. (2009) use local search methods but also employ sampling techniques for the evaluation of the objective function. Mohan et al. (2008) and Chen et al. (2009) introduce arc-routing variants of the PTSP.

Weyland et al. (2011) introduce a new method for reducing the computational complexity of the objective function of the PTSPD. An approximation for the objective function based on Monte Carlo Sampling is combined with a quasi-parallel evaluation of the samples. The authors show that heuristics which use this new method outperform previous approaches for the PTSPD both in terms of computational time and solution value. Weyland et al. (2012) show that the evaluation of the objective function of the PTSPD is $\#P$ -hard. The complexity class $\#P$ captures the set of problems where one is not only interested in whether a solution exists, but also the number of solutions that exist (Weyland et al. 2012).

While our problem relates to the PTSP in that it deals with stochastic customer presence, problems in the literature can also be characterized by other stochastic elements such as stochastic demands or travel times. The common feature among these different stochastic elements is that they all lead to stochastic arrival times. The remainder of this section focuses on a variety of stochastic problems with time constraints.

Jaillet et al. (2012) seek to minimize a lateness index for single-vehicle problems with stochastic travel times and deadlines. However, this objective is not appropriate when a payment is made based on late delivery, as is the situation in our problem.

Chang et al. (2009) developed a heuristic for a version of the time-dependent, stochastic traveling salesman problem with time windows (STSPTW) where travel and service times are stochastic. The authors use an n -path relaxation of a deterministic traveling salesman problem (TSP) and a convolution-propagation approach to approximate arrival times at each customer.

A related problem to the TSP is the orienteering problem (OP). In the orienteering problem, a profit is associated with each customer. The objective is to construct a tour that maximizes the total profit while observing a specified time limit. Campbell et al. (2011) and Tang and Miller-Hooks (2005) introduce stochastic versions of the OP where travel and service times are random. Campbell et al. formulate a recourse model and present a variable neighborhood search to solve instances of the orienteering problem. Tang and Miller-Hooks formulate their problem as a chance-constrained stochastic program and develop an exact method that solves small to medium-size problems as well as a heuristic method for larger instances.

Another related routing problem is the stochastic vehicle routing problem (SVRP) where customers have demands and vehicles have capacity constraints. Several recent papers highlight the SVRP with time windows (SVRPTW). Lei et al. (2011) model a version of the SVRPTW with stochastic demands as a stochastic program with recourse and propose an adaptive large neighborhood search heuristic. Li et al. (2010) introduce a problem with stochastic travel and service times and present two different model formulations. The first model is a chance constrained programming model where the objective is to minimize transportation cost when some constraints hold under a certain confidence level. The second model is a stochastic programming model with recourse where the objective is to minimize the expected transportation cost of a priori solutions. A tabu search-based method is used to solve both models.

Erera et al. (2009) present a stochastic dynamic vehicle routing problem with time windows (SDVRPTW). The SDVRPTW is a problem in which customers are dynamically inserted into routes. The problem is similar to ours in that customer presence is stochastic and each customer has an associated time window. However, Erera et al. maintain time window feasibility and use a paired-vehicle recourse action while we penalize late time window violations and use a single-vehicle recourse action.

Recourse model

In this section, we present a recourse model for the PTSPTW. A recourse model is characterized by two stages. In stage one, an a priori solution is determined. Then the random variables are realized and a recourse action is applied to the original solution. In this case, the a priori solution is a route that identifies the order in which

customers will be visited. After information becomes available about which customers need to be visited, the recourse action is applied. This recourse action consists of adjusting the route so that customers who do not need to be visited are skipped, but the remaining customers are still visited in the a priori route order.

In our model, service at each customer cannot begin before e_i . An early arrival requires the driver to wait until e_i . However, the vehicle is permitted to visit a customer after l_i has passed, but a penalty is incurred for doing so. We utilize a per-unit-time penalty charge for arriving after the close of the time window. The penalty is represented by λ . The per-unit-time charge represents cases where the delivery company is charged per unit time of lateness. For instance, FedEx Custom Critical refunds varying percentages of the cost of a shipment based on how late the shipment is delivered (FedEx 2005). For additional examples, see Charnirisirakskul et al.(2004) and Slotnick and Sobel (2005). Thus, the objective of the PTSPTW, formulated as a recourse model, is to identify an a priori route where the sum of the expected travel costs between customers and the expected penalty term is minimum.

In the case of our recourse model, we let τ define an order, or tour, in which the customers $i = 1, \dots, n$ are to be visited. We assume that the customers are indexed according to their position in τ . We assume a tour always begins and ends at a fixed depot, and the depot is indexed as $i = 0$. We assume integer travel times and deadlines and that all tours start at time $t = 0$. Our recourse model provides a formula for evaluating the cost of a given tour τ .

As a result of our assumptions, the expected travel costs can be calculated as they are for the well-known PTSP (see Jaillet (1988) for further reference) with a straightforward modification for the fixed depot (Campbell and Thomas 2008):

$$\sum_{j=1}^n p_j d_{0,j} \prod_{k=1}^{j-1} (1 - p_k) + \sum_{i=1}^{n-1} \sum_{j=i+1}^n p_i p_j d_{i,j} \prod_{k=i+1}^{j-1} (1 - p_k) + \sum_{i=1}^n p_i d_{i,0} \prod_{k=i+1}^n (1 - p_k). \tag{1}$$

This formula calculates the probability and the resulting expected cost of any arc that may appear in the tour. The expected cost of an arc (i, j) depends on customers i and j being realized and no customers $k, k = i + 1, \dots, j - 1$, being realized.

For both the issue of early and late arrivals, we need to determine the probability that arrival at a customer occurs at a particular time t . To begin, let the random variable

$$X_i = \begin{cases} 0 & \text{if customer } i \text{ is not realized} \\ 1 & \text{if customer } i \text{ is realized.} \end{cases}$$

Also, let A_i be a random variable representing the time of arrival at the customer in position i in τ . We assume that arrival at a customer i cannot be early or late if customer i is not realized. Hence, we are left to compute $P(A_i = t \mid X_i = 1)$, where t represents the time of arrival at customer i . For notational convenience, let $g(i, t) = P(A_i = t \mid X_i = 1)$. We also let $G(i, t) = P(A_i \leq t \mid X_i = 1)$, which can be computed as $G(i, t) = \sum_{k=0}^t g(i, k) = G(i, t - 1) + g(i, t)$. We assume that arrival at the depot occurs at time $t = 0$. Therefore, $G(0, t) = 1$ for all values of t . Because

there is no time window or penalty associated with the depot, it is not necessary to consider the probability that the arrival at the depot on the return trip occurs at any particular time.

To account for the fact that arrival at customer i depends on the departure times at previous customers, we let $h(i, t)$ be the probability of departing from customer i at time t . We can then compute the g and h values in an iterative, recursive fashion. Note that arrival and departure times for a customer differ when waiting for the opening of the time window occurs. Before describing the computation procedure in Algorithm 1, we first introduce bounds on the earliest and latest times that a customer can be visited.

If the triangle inequality holds for the travel time data, then the times between the earliest possible arrival time and the latest possible departure time that need to be considered for each customer i can be limited based on the location of customer i in the tour. The earliest arrival time at customer i , expressed as T_i^{min} , occurs when all customers prior to i are not realized. Thus, $T_i^{min} = d_{0,i}$ for all i . The latest possible arrival time at customer i is computed by assuming that every customer prior to i is realized. If arrival occurs prior to the start of the time window for any customer prior to i , then this additional waiting time must be accounted for in the computation. Given $T_1^{max} = d_{0,1}$, we can compute the other values recursively:

$$T_i^{max} = \max(T_{i-1}^{max}, e_{i-1}) + d_{i-1,i}. \tag{2}$$

All values of t outside of the range of T_i^{min} to T_i^{max} result in g and h values of zero unless $T_i^{max} < e_i$, in which case $h(i, e_i)$ has a value of one.

Algorithm 1 Pseudocode for computation of g and h values for Recourse Model

```

Initialization:
     $g(0,0) \leftarrow 1$  and  $g(i,t) \leftarrow 0$  for  $(i,t) \neq (0,0)$ ,  $i = 0, 1, \dots, n$  and  $t = 0, 1, \dots, T_n^{max}$ .
     $h(0,0) \leftarrow 1$  and  $h(i,t) \leftarrow 0$  for  $(i,t) \neq (0,0)$ ,  $i = 0, 1, \dots, n$  and  $t = 0, 1, \dots, T_n^{max}$ .
     $G(0,t) \leftarrow 1$  for  $t = 0, 1, \dots, T_n^{max}$  and  $G(i,t) \leftarrow 0$  for  $i = 1, \dots, n$  and  $t = 0, 1, \dots, T_n^{max}$ .
    for  $i = 1, \dots, n$  do
        if  $T_i^{max} < e_i$  then
             $h(i, e_i) \leftarrow 1$ 
        end if
    end for
    for  $i = 1, \dots, n$  do
        for  $t = T_i^{min}, \dots, T_i^{max}$  do
            for  $v = 0, \dots, i - 1$  do
                if  $t \geq d_{v,i}$  then
                     $g(i, t) \leftarrow p_v h(v, t - d_{v,i}) \prod_{k=v+1}^{i-1} (1 - p_k) + g(i, t)$ 
                end if
            end for
             $G(i, t) \leftarrow G(i, t - 1) + g(i, t)$ 
            if  $t < e_i$  then
                 $h(i, t) \leftarrow 0$ 
            else if  $t = e_i$  then
                 $h(i, t) \leftarrow G(i, t)$ 
            else
                 $h(i, t) \leftarrow g(i, t)$ 
            end if
        end for
    end for

```

We now clarify the initialization procedure for Algorithm 1. The values $g(i, t)$, $h(i, t)$, and $G(i, t)$ are set to zero for all combinations of i and t , where $i = 0, 1, \dots, n$ and $t = 0, 1, \dots, T_n^{max}$, unless explicitly noted. The values $g(0,0)$ and $h(0,0)$ are set to one to represent the probabilities of both arriving and departing from the depot at time zero. Also, $h(i, e_i)$ is set to one if $T_i^{max} < e_i$. Finally, $G(0,t)$ is set to one for all values of t .

Using the previously described bounds and initialization, Algorithm 1 recursively computes h and g values. Recall that $g(i, t)$ represents the probability of arriving at customer i at a particular time t , given that customer i is realized. Therefore, the algorithm considers the probability that previous customers are realized, as well as the probability of departing at a particular time from these previous customers. Thus $g(i, t) = \sum_{v=0}^{i-1} p_v h(v, t - d_{v,i}) \prod_{k=v+1}^{i-1} (1 - p_k)$, and is computed recursively in Algorithm 1. In words, $p_v h(v, t - d_{v,i})$ represents the probability that customer v is realized times the probability that departure from customer v occurs at the current time minus the travel time from customer v to customer i . This departure time implies that customers v and i are realized but no customers $k, k = v + 1, \dots, i - 1$, are realized. The probability that these customers are not realized is represented by the term $\prod_{k=v+1}^{i-1} (1 - p_k)$ in the formula. Note also that when $t < d_{v,i}$, $t - d_{v,i}$ is negative, and thus represents an impossible situation when considering the original assumptions of the problem. Thus $h(v, t - d_{v,i})$ is set to zero when $t < d_{v,i}$.

Because $G(i, t)$ is the cumulative function for $g(i, t)$, $G(i, t)$ can be computed recursively using the formula $G(i, t - 1) + g(i, t)$. We are left to compute $h(i, t)$, which depends on three situations. Either

1. time t is before the start of the time window,
2. time t is equivalent to the start of the time window, or
3. time t is after the start of the time window.

If the first situation occurs, then the probability of departing from customer i at time t is zero. That is, $h(i,t) = 0$. If the second situation occurs, then the probability of departing from customer i at time t is equivalent to the probability of arriving at customer i at or before time t . Therefore, $h(i, t) = G(i, t)$. If the third situation occurs, $h(i, t) = g(i, t)$. This completes the description of Algorithm 1.

To account for the per-unit-time penalty, we complete the objective by adding the following to Eq. 1:

$$\sum_{i=1}^n p_i \sum_{t=l_i+1}^{T_i^{max}} \lambda g(i, t)(t - l_i). \tag{3}$$

Thus we have the following objective function:

$$\begin{aligned} & \sum_{j=1}^n p_j d_{0,j} \prod_{k=1}^{j-1} (1 - p_k) + \sum_{i=1}^{n-1} \sum_{j=i+1}^n p_i p_j d_{i,j} \prod_{k=i+1}^{j-1} (1 - p_k) \\ & + \sum_{i=1}^n p_i d_{i,0} \prod_{k=i+1}^n (1 - p_k) + \sum_{i=1}^n p_i \sum_{t=l_i+1}^{T_i^{max}} \lambda g(i, t)(t - l_i). \end{aligned} \tag{4}$$

The computation of the objective function is more complex than that of the PTSP, but is similar to the complexity of the PTSPD. The initialization of Algorithm 3 is computed in $O(n T_n^{\max})$ time, while the body of the algorithm is computed in $O(n^2 \max_i \{T_i^{\max} - T_i^{\min}\})$ time. Without restrictions on distances, this is not necessarily polynomial in n . Once the g and h values are known, the per-unit-time penalty portion of the objective can be computed in $O(n \max_i \{T_i^{\max} - l_i\})$ time, and the distance portion of the objective function can be computed in $O(n^2)$ time. The entire objective calculation is dominated by the g and h computations, and thus takes $O(n^2 \max_i \{T_i^{\max} - T_i^{\min}\})$ time. In addition, Weyland et al.'s (2012) characterization of the PTSPD objective function as #P-hard can be trivially extended to the PTSPTW.

Solution approach

Our solution approach is motivated by the recent work of da Silva and Urrutia (2010) on the TSPTW, the deterministic version of the problem discussed in this paper. Da Silva and Urrutia combine a Variable Neighborhood Search (VNS) with a Variable Neighborhood Decent (VND) as a local search. The algorithm yields high quality solutions for the data sets from which those used in this paper are derived and achieves best-known results on other benchmark data sets. Thus, we chose to implement a VNS/VND heuristic as well.

In broad terms, the VNS we use can be described as a systematic change of neighborhoods during a two-phase procedure. The first phase is a perturbation of the current best solution, while the second phase is a descent to find a new local minimum. The purpose of the perturbation is to move the solution out of a local minimum. Note that we do not need the initial construction phase implemented by da Silva and Urrutia (2010) because we penalize time window violations, thus making all solutions feasible.

We now describe the algorithm by beginning with a description of the VNS. As input, the VNS takes an initial seed tour x , a value k_{\max} that controls the maximum level of perturbation in the Shake function, and an objective function $f(\cdot)$ that maps a tour to the set of positive real numbers. Specifically, $f(\cdot)$ is the objective function presented in Eq. 4. After initializing k to 1 and *improvement* to true, the algorithm perturbs the current solution x by calling the Shake function. The perturbed solution, x' , is then passed to the VND to find a local minimum x^* . If the objective value of x^* is less than the objective value of the current solution x , then x is set to x^* , k is reset to 1, and the procedure repeats. If, on the other hand, the objective value of x^* is not less than the objective value of x and k does not equal k_{\max} , then k is incremented by 1 and the procedure repeats. When k reaches k_{\max} , then no improving solution has been found within the last k_{\max} iterations. The parameter *improvement* is set to false, the current solution x is set to *best*, and the algorithm terminates returning *best*, the best tour found. The pseudo code for the VNS algorithm is presented in Algorithm 2.

Algorithm 2 VNS

```

Input:  $x, k_{max}, f(\cdot)$ 
Output:  $best$ 
Initialization:
 $k \leftarrow 1$ 
 $improvement \leftarrow \text{True}$ 
while  $improvement = \text{True}$  do
   $x' \leftarrow Shake(k, x)$ 
   $x^* \leftarrow VND(x', j_{max})$ 
  if  $f(x^*) < f(x)$  then
     $x \leftarrow x^*$ 
     $k \leftarrow 1$ 
  else if  $k < k_{max}$  then
     $k \leftarrow k + 1$ 
  else
     $improvement \leftarrow \text{False}$ 
  end if
end while
 $best \leftarrow x$ 

```

We now describe the Shake function that is used to perturb the current best solution. As input, the Shake function requires a tour x and a neighborhood specification k . We denote a set of neighborhoods for the Shake function with $N_k^{Shake} = \{1, \dots, k_{max}\}$ and with $N_k^{Shake}(x)$ the set of solutions in the k^{th} neighborhood. Specifically, $N_k^{Shake}(x)$ contains all tours that differ from x by a combination of k 1-shift moves. A 1-shift move removes a single customer from a tour and reinserts the customer in a new location on the tour. We assume the solutions from $N_k^{Shake}(x)$ are ordered such that $\{x^1, \dots, x^{|N_k^{Shake}(x)|}\}$. The perturbation selects a random tour $x^w \in N_k^{Shake}(x)$, sets x' to x^w , and outputs the perturbed solution x' . The pseudo code for the algorithm is presented in Algorithm 3.

Algorithm 3 Shake

```

Input:  $x, k$ 
Output:  $x'$ 
 $w \leftarrow [1 + Rand(0, 1) \times |N_k^{Shake}(x)|]$ 
 $x' \leftarrow x^w$ 

```

In the algorithm presented by da Silva and Urrutia, k_{max} is set to 30. However, such large neighborhoods lead to many iterations of the VNS and VND. Because the computational complexity of the PTSPTW objective calculation is greater than that of the TSPTW objective calculation, a large number of iterations causes the running time to be very large. Additionally, during preliminary experiments, we found that the algorithm often converged to the same solution for values of k_{max} set to 30 and set to 5. For these reasons, we set k_{max} to 5.

Hansen et al. (2010) propose an implementation of VNS/VND for combinatorial optimization problems which differs from the implementation proposed by da Silva and Urrutia in the procedure for changing neighborhoods within the VND. In preliminary experiments, we found that both algorithms generally converged to the same solutions, but the algorithm proposed by Hansen et al. converged faster. For this reason, we base our VND implementation on the algorithm proposed by Hansen et al.

The VND is used to optimize the recently perturbed solution. The input for the VND includes a tour x , a value j_{max} , and the previously described objective function $f(\cdot)$. We denote a set of neighborhoods for the VND with $N_j^{VND} = \{1, \dots, j_{max}\}$ and with $N_j^{VND}(x)$ the set of solutions in the j^{th} neighborhood. The value j_{max} represents the last neighborhood that the VND iterates through. We set j to 1 to specify the previously discussed 1-shift neighborhood and we set j to 2 to specify the 2-Opt neighborhood. A 2-Opt move deletes two edges from a tour so that the tour is broken into two paths, then reconnects the paths in the only other possible way. Because we specify two neighborhoods for the VND, we set j_{max} to 2. Our choices of the 1-shift and 2-Opt neighborhoods are also motivated by the work of da Silva and Urrutia and the descriptions of the neighborhoods can be found therein.

After initializing j to 1 and *improvement* to true, a locally optimal solution x' is found by iterating through the entire neighborhood N_j^{VND} . If the objective value of x' is less than the objective value of the current solution x , then x is set to x' , j is reset to 1, and the procedure repeats. If, on the other hand, the objective value of x' is not less than the objective value of x and j does not equal j_{max} , then j is incremented by 1 and the procedure repeats. When j reaches j_{max} , then no improving solution has been found in either neighborhood. The parameter *improvement* is set to false, the current solution x is set to x^* , and the algorithm terminates returning x^* , the best locally optimal tour found. The pseudo code for the VND algorithm is presented in Algorithm 4.

Algorithm 4 VND

```

Input:  $x, j_{max}, f(\cdot)$ 
Output:  $x^*$ 
Initialization:
   $j \leftarrow 1$ 
  improvement  $\leftarrow$  True
  while improvement = True do
     $x' \leftarrow \arg \min_y \{f(x^y)\}$ 
    if  $f(x') < f(x)$  then
       $x \leftarrow x'$ 
       $j \leftarrow 1$ 
    else if  $j < j_{max}$  then
       $j \leftarrow j + 1$ 
    else
      improvement  $\leftarrow$  False
    end if
  end while
   $x^* \leftarrow x$ 

```

Experimental design

Next, we investigate what instance characteristics result in different tours when customer presence is modeled stochastically instead of deterministically. To address this question, we focus on the effects of customer realization probabilities, time window widths, per-unit-time penalties, and the number of customers.

The data sets we use are a subset of the TSPTW instances first proposed by Dumas et al. (1995). We use the 20-, 40- and 60-customer instances with time-

window widths of 20, 60, and 100 U. These instances are labeled “Feasible” in the results tables. We also generate new data sets from each of the existing instances. These new instances differ in the starting and ending times of the time windows, and represent the situations where feasible solutions with respect to time windows are unlikely to exist if all customers are realized. These situations are likely to occur as the demand for time-definite services grows. Our results, which are presented in the next section, indicate that large cost savings are possible when these situations are modeled stochastically compared to deterministically. We denote the new early and late deadlines as e'_i and l'_i , respectively. In general, we set l'_i equal to the early deadline. Then we set e'_i equal to l'_i minus the width of the corresponding time window, unless that time is negative in which case e'_i is set to zero. As an example, consider an instance with a time window width of 20 where $e_i = 15$ and $l_i = 35$. Then $l'_i = 15$, and since $15 - 20 = -5$, set $e'_i = 0$. Sometimes, however, l'_i is equal to zero. In this case, we instead let l'_i equal the late deadline and e'_i equal l'_i minus the width of the corresponding time window. Suppose we again have a time window width of 20. If $e_i = 0$ and $l_i = 20$, then since l'_i would equal 0, we let $l'_i = 20$ and $e'_i = 0$. These instances will be referred to in the tables by the label “Infeasible.”

Similar to the design of the PTSPD experiments in Campbell and Thomas (2008) (see Campbell and Thomas (2006) for PTSPD benchmark data sets), we consider four different probability settings for each instance. Three of these settings are homogeneous with customer presence probabilities set to 0.1, 0.5, and 0.9, respectively. These instances are referred to as 0.1, 0.5, and 0.9 in the results tables. The probabilities represent the likelihood that each customer will be realized. A probability of 0.1 may be suitable to represent a home or small business because packages are unlikely to be delivered every day. Probabilities of 0.5 may be suitable to represent small or medium businesses that receive packages a few days a week, while probabilities of 0.9 may be suitable to represent medium or large businesses that receive packages almost every day. The fourth probability setting is heterogeneous, where the probability of each customer is randomly assigned a probability of either 0.1 or 1. This case represents the situation in which both small and large businesses are served by the same vehicle. This data set will be referred to in the tables by the label “Mixed.”

For each instance with 20, 40 and 60 customers, we consider two different per-unit-time penalties. As was done in the PTSPD work of Campbell and Thomas (2008), one penalty is set to 5, while in the other case the penalty is set to 50. These penalties represent small and large costs of failing to satisfy customer time windows. In total, we solve 720 different instances.

Because our goal is to determine differences between TSPTW and PTSPTW solutions, for feasible instances we seed the VNS/VND heuristic described in “[Solution Approach](#)” with the best-known TSPTW solution [see da Silva and Urrutia (2010)] and solve the corresponding PTSPTW instance with homogeneous and mixed probability settings. For the infeasible instances, we construct corresponding TSPTW solutions to use as seed solutions for the VNS/VND heuristic. To construct these TSPTW solutions, we assign the same per-unit-time penalties as the corresponding PTSPTW set, and set all probabilities to one. We run

the VNS/VND heuristic 10 times using the corresponding best-known TSPTW solution with feasible time windows, converted to infeasible time windows, as a starting solution. We select the best TSPTW solution out of the 10 runs for each instance to use as the seed solutions for the VNS/VND algorithm to solve the corresponding PTSPTW infeasible instances with homogeneous and mixed probability settings.

In evaluating our computational results, it is necessary to be able to compare two solutions. One useful metric of comparison is the Hamming distance introduced by Bierwirth et al. (1996) for the Job Shop Scheduling problem. Ehmke et al. (2012) use this measure with vehicle routing problems. In terms of tours, the Hamming distance is the ratio of the precedence relationships of customers being preserved between two tours. In order to calculate the Hamming distance, a tour must be converted to a bit-string representation. A tour with n customers has a bit-string representation of $\frac{n^2-n}{2}$ bits. If customer i is located before customer j on a tour, then the bit is set to one. On the other hand, if customer j is located prior to i , then the bit is set to zero. The normalized Hamming distance for two tours represented as bit-strings x and y is calculated by $d_{x,y} = \frac{1}{l} \sum_{k=1}^l xor(x_k, y_k)$, where l is the length of the bit-strings. The “exclusive-or” operator is represented by $xor(\cdot)$ and results in one if the bits x_k and y_k differ, and zero otherwise. In this paper, a Hamming distance of zero indicates two identical tours, while a Hamming distance of one indicates that no customers share the same precedence relationship between two tours. A Hamming distance of one occurs only in the case where one tour is the exact inverse of the other tour.

Another metric that we use to compare solutions is the percentage of change in objective value. In the case of feasible instances, the initial solution is the best-known TSPTW solution. For infeasible instances, the initial solution is the best TSPTW solution we found using our heuristic technique. Both the initial and final solutions are evaluated with the PTSPTW objective function. The percentage of change in objective value is calculated as

$$\left(\frac{\text{Initial objective value} - \text{Final objective value}}{\text{Initial objective value}} \right) \times 100\%. \quad (5)$$

Because the initial objective value is either equal to or greater than the final objective value, the percentage of change always represents a decrease in value from the initial seed tour.

Because of the probabilistic nature of the algorithm, each instance is run 10 times. The online resource for this paper provides results for all instances. In the next section, we provide a summary of those results in Tables 1 and 2. In order to clearly understand the abbreviated labels in the summary tables, we first present a description of some of the labels found in the online resource. “Avg. Δ (%)” represents the average percentage of change in objective value for each instance across the 10 runs. Similarly, “Avg. Ham.” and “Avg. Time” represent the average Hamming distance and the average CPU seconds of runtime, respectively. In the summary tables, the abbreviation “AVG.” represents the calculation of an average of an average. “AVG. Δ (%)” is the average of “Avg. Δ (%)” over each of the five

Table 1 Summary of feasible experiments

	Probability	AVG. Δ (%)			AVG. Ham.			AVG. Time					
		0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed	0.1	0.5	0.9	Mixed
$n = 20$	Penalty = 50												
	$w = 20$	0.03	0.04	NC	0.02	0.01	0.00	NC	0.00	6.09	5.36	5.45	5.07
	$w = 60$	0.50	2.00	0.46	0.47	0.05	0.03	0.01	0.02	6.08	5.99	5.79	4.79
	$w = 100$	0.46	0.20	0.06	0.98	0.10	0.01	0.01	0.05	6.31	4.66	4.81	4.88
$n = 20$	Penalty = 50												
	$w = 20$	0.03	NC	NC	0.01	0.01	NC	NC	0.00	5.84	5.12	5.45	4.89
	$w = 60$	0.40	1.01	0.13	0.42	0.04	0.02	0.01	0.01	6.28	6.15	5.40	4.80
	$w = 100$	0.31	0.11	0.02	0.97	0.09	0.01	0.01	0.04	6.68	5.00	4.89	4.94
$n = 40$	Penalty = 5												
	$w = 20$	0.12	0.30	0.03	0.57	0.01	0.00	0.00	0.00	156.08	138.82	132.23	120.88
	$w = 60$	0.58	0.96	0.01	2.65	0.03	0.01	0.00	0.02	167.98	124.56	113.12	120.56
	$w = 100$	1.73	1.13	0.00	4.98	0.05	0.02	0.00	0.04	164.72	124.49	103.83	153.78
$n = 40$	Penalty = 50												
	$w = 20$	0.11	0.20	0.04	0.37	0.01	0.00	0.00	0.00	154.54	131.82	135.48	127.85
	$w = 60$	0.50	0.69	0.01	2.62	0.03	0.01	0.00	0.02	170.77	120.24	115.17	120.41
	$w = 100$	1.43	1.11	0.00	3.87	0.05	0.02	0.00	0.03	169.63	120.33	104.94	133.84
$n = 60$	Penalty = 5												
	$w = 20$	0.19	0.62	0.06	2.65	0.01	0.00	0.00	0.01	823.93	705.98	668.30	960.30
	$w = 60$	0.59	0.96	0.78	3.65	0.02	0.01	0.00	0.01	932.91	716.23	677.40	850.16
	$w = 100$	1.52	1.88	0.17	4.54	0.04	0.02	0.00	0.03	1,013.68	783.61	632.49	970.53
$n = 60$	Penalty = 50												
	$w = 20$	0.16	0.33	0.00	1.61	0.01	0.00	0.00	0.00	990.25	827.87	819.30	936.87
	$w = 60$	0.45	0.45	0.03	2.76	0.02	0.00	0.00	0.01	1,192.77	868.17	766.85	810.23
	$w = 100$	1.31	1.31	0.17	4.41	0.04	0.02	0.00	0.02	1,303.77	937.40	741.76	989.51

Table 2 Summary of infeasible experiments

Probability	AVG. Δ (%)			AVG. Ham.			AVG. Time			
	0.1	0.5	0.9	0.1	0.5	0.9	0.1	0.5	0.9	
	Mixed	Mixed	Mixed	Mixed	Mixed	Mixed	Mixed	Mixed	Mixed	
$n = 20$										
Penalty = 5										
$w = 20$	2.93	0.60	NC	0.01	0.00	NC	0.00	6.02	5.38	5.23
$w = 60$	3.82	3.40	0.04	0.07	0.06	0.00	0.04	6.02	5.24	4.36
$w = 100$	5.46	6.00	0.51	22.58	0.13	0.06	0.01	6.78	4.53	3.85
$n = 20$										
Penalty = 50										
$w = 20$	13.03	2.20	NC	5.52	0.02	0.01	NC	5.99	5.61	5.06
$w = 60$	21.30	8.93	2.74	6.61	0.08	0.05	0.01	6.70	5.09	4.63
$w = 100$	28.98	13.89	1.74	39.60	0.17	0.08	0.01	6.93	5.06	3.99
$n = 40$										
Penalty = 5										
$w = 20$	0.80	3.93	1.75	3.42	0.02	0.01	0.00	158.54	129.41	120.97
$w = 60$	4.53	7.16	0.20	3.16	0.05	0.02	0.00	165.50	110.67	94.76
$w = 100$	13.16	14.14	1.53	24.99	0.08	0.04	0.01	163.05	113.88	91.78
$n = 40$										
Penalty = 50										
$w = 20$	5.54	12.77	3.58	8.54	0.01	0.01	0.00	155.76	133.02	123.90
$w = 60$	24.45	12.77	0.30	3.71	0.06	0.01	0.00	196.67	110.97	95.29
$w = 100$	25.51	20.95	0.44	3.94	0.09	0.04	0.01	182.84	119.47	91.93
$n = 60$										
Penalty=5										
$w = 20$	0.66	1.12	0.05	4.79	0.01	0.00	0.00	833.44	705.66	686.59
$w = 60$	18.94	23.46	12.22	37.42	0.05	0.03	0.04	1,073.15	825.71	843.04
$w = 100$	6.67	6.87	0.72	22.46	0.06	0.02	0.00	1,098.85	703.37	561.38
$n = 60$										
Penalty = 50										
$w = 20$	7.44	6.02	0.02	14.31	0.01	0.00	0.00	999.66	832.76	769.01
$w = 60$	40.86	33.69	18.68	54.01	0.04	0.02	0.03	1,307.59	930.00	890.20
$w = 100$	48.12	36.69	4.04	33.28	0.06	0.02	0.01	1,655.80	876.45	671.45

instances at each setting from the tables in the online resource. “AVG. Ham.” represents the average of “Avg. Ham.” over the five instances at each setting. Similarly, “AVG. Time” is the average of “Avg. Time“ for each of the five instances at each setting. For simplification of language, when discussing results from the summary tables, the term “average” is assumed to refer to the “AVG.” calculation. The abbreviation “NC” within the tables stands for “No Change,“ as opposed to “0.00” which means that a very small change has occurred.

Results

The summarized results in Tables 1 and 2 indicate that the 0.1 homogeneous setting has greater differences in tour structure from the TSPTW solution than the other homogeneous and mixed settings. Meanwhile, the 0.9 setting displays the smallest differences. This outcome is logical, because as customer realization probabilities increase towards one, the problem gets closer to the deterministic TSPTW. An example of this trend is displayed in Fig. 1. The PTSPTW solution for the 0.9 probability instance is visually similar to the TSPTW solution and has a Hamming distance of 0.02. On the other hand, the 0.1 instance looks very different from the TSPTW solution and has a larger Hamming distance of 0.14. The key

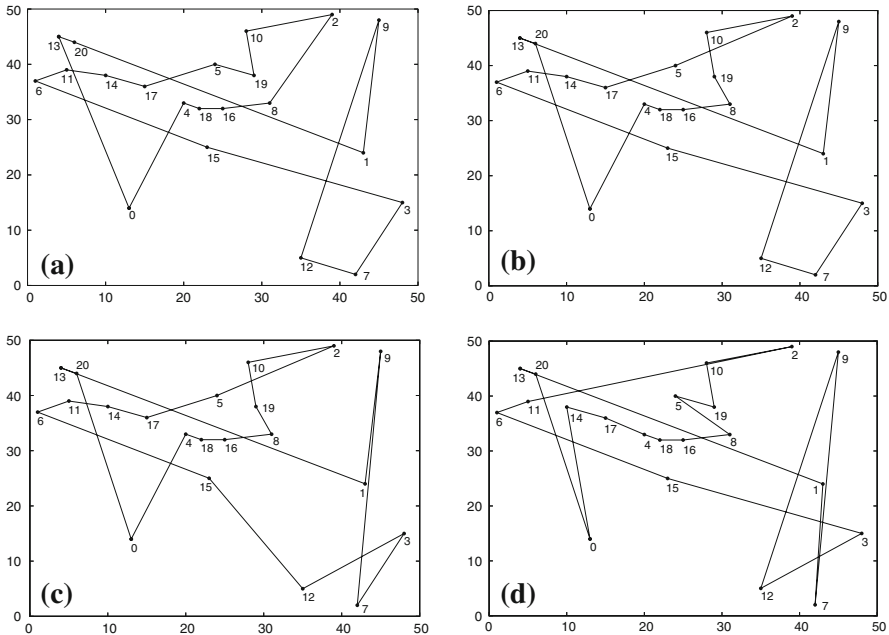


Fig. 1 Solution tours for homogeneous probabilities for the third feasible instance with 20 customers, time window widths of 100, and a per-unit-time penalty of 50. **a** TSPTW solution, **b** PTSPTW solution for 0.9 probabilities, Hamming distance: 0.02, change in objective value: 0.06 %. **c** PTSPTW solution for 0.5 probabilities, Hamming distance: 0.03, change in objective value: 0.48 %, **d** PTSPTW solution for 0.1 probabilities, Hamming distance: 0.14, change in objective value: 0.39 %

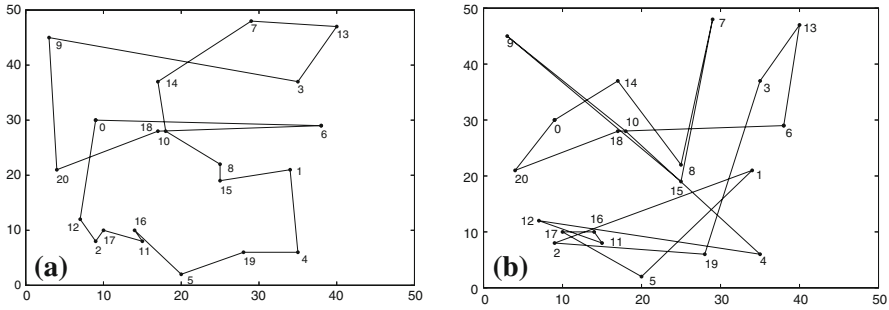


Fig. 2 Solution tour for 0.1 probabilities for the second infeasible instance with 20 customers, time window widths of 100, and a per-unit-time penalty of 50. **a** TSPTW solution, **b** PTSPTW solution for 0.1 probabilities, Hamming distance: 0.43, change in objective value: 53.41 %

difference is the number of times that the tour crosses itself. The increased crossings in instances with smaller probabilities occur as a way to improve routing cost when it is possible to skip customers.

Out of all the probability settings, the mixed setting displays the largest differences from the TSPTW solutions for cost. This is because of the PTSPTW’s ability to prioritize customers with larger probabilities in the tour. As noted earlier, however, the mixed setting does not display the largest differences in structure. This indicates that even small changes in tour structure can produce large changes in cost.

It is intuitive that time windows drive the tour construction for the TSPTW. Our results support the idea that time windows are also a significant contributor to tour construction for the PTSPTW. In a majority of cases, instances with time window widths of 100 have greater percentage of changes in objective value and also greater Hamming distances than instances with time window widths of 20. This is because larger time windows allow greater flexibility in the construction of the tours.

Another result comes from comparing the percentage of changes in objective value of the feasible instances to those of the infeasible instances. The average percentage of changes in objective value for the feasible instances are much smaller than those of the infeasible instances. The average change across all feasible instances is 0.92 %, while the average for the infeasible instances is 11.86 %. An example of the type of change in tour structure that creates a large decrease in objective value is displayed in Fig. 2. The PTSPTW tour chooses to serve customers 7 and 9, which are far from the other customers, in a very different way. If customers 7 and 9 are not realized, the PTSPTW tour will be much shorter and allow more customers to be visited feasibly than the TSPTW tour. This result indicates that when infeasibility exists in terms of time windows, it may be very beneficial to use the PTSPTW formulation as opposed to the TSPTW formulation.

Per-unit-time penalty increasing from 5 to 50 affects the solution tours for all the instances as expected. For the feasible instances, the average percentage of change in objective value decreases as the penalty increases. Because these are feasible instances, it is possible to serve many customers within their respective time

windows. It is likely that incurring a large penalty cost for missing a deadline is more costly than traveling a longer distance to arrive within the respective time window. Thus, larger penalties push the solutions toward the optimal TSPTW solutions. All of the Hamming distances either decrease or do not demonstrate change within two significant digits.

For infeasible instances, the opposite trend occurs. The average percentage of change in objective value generally increases as the penalty increases from 5 to 50. By construction of these instances, it is unlikely that a tour exists such that the driver will be able to serve all customers within their respective time windows. Therefore, penalty charges are unavoidable. As the charge increases, the cost of the tour increases. Any percentage of change in objective value that occurs at a low penalty level is multiplied at the higher level.

An increase in the number of customers also causes an increase in the average percentage of change in objective value for both feasible and infeasible instances. Hamming distance, on the other hand, tends to decrease and is most evident with time window widths of 100. It is possible that because of the larger number of customers on the tour, more precedence relationships are preserved even when customer positions differ and cause changes in the objective value.

Lastly, it is worthwhile to mention two factors contributing to increased runtimes. The most noticeable factor is problem size. As the number of customers increases, so does the runtime. This is due to the fact that as the solution space grows, the number of times that the objective value is calculated increases. The second factor is the use of small or mixed probabilities. Instances with these two probability types deviate more from their TSPTW solutions than instances with larger probabilities. This implies that the increased runtime is caused by more local search moves.

Conclusions

In this paper, we have presented a recourse model for the PTSPTW that requires construction of a tour that visits each realized customer but pays a penalty if the late deadline of the time window constraint is violated. We described a VNS/VND heuristic to solve instances of the PTSPTW. Finally, we presented computational results that offer the following key insights when comparing PTSPTW solutions to their TSPTW counterparts:

- Large probabilities of customer presence have less impact on the solution than small probabilities.
- When customers have a combination of high and low probabilities, solutions are significantly impacted by modeling customers stochastically. This is because of the stochastic model's ability to prioritize customers with larger probabilities in the solution.
- Large time window widths have a greater impact on the solution than small time window widths when customers are modeled stochastically. An explanation is that as time window widths increase, the algorithm has more flexibility for placing customers on the tour.

- Solutions to the PTSPTW come with large computation times due to the complexity of the objective calculation. Factors that contribute to larger runtimes are large problem sizes and problems with low or mixed probabilities.

Because the PTSPTW is a new problem in vehicle routing, there are many directions for future research. One such research direction would be to apply a fixed-charge penalty for arriving after the late time window deadline. The fixed-charge recourse represents the case where the delivery company reimburses the customer for the cost of the delivery in the event that the deadline is violated. Well-known examples of such penalties are FedEx's and UPS' money-back guarantees (FedEx 2011; United Parcel Service 2011). Our solution method is capable of handling such an extension of the problem. For the fixed-charge penalty, the following term would need to be added to Eq. 1:

$$\sum_{i=1}^n p_i \Lambda \bar{G}(i, l_i), \quad (6)$$

where $\bar{G}(i, l_i) = 1 - G(i, l_i)$ and Λ is the fixed-charge penalty.

An additional recommendation for future work is to solve larger problem instances. The key challenge with larger instances is the computational complexity of the objective function. Therefore, the solution method will likely need to incorporate sampling methods to evaluate the objective. Investigation of these ideas will help further our understanding of the advantages and disadvantages of incorporating stochasticity into the model.

References

- Balaprakash P, Birattari M, Stützle T, Yuan Z, Dorigo M (2009) Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. *Swarm Intelligence* 3(3):223–242
- Bierwirth C, Mattfeld D, Kopfer H (1996) On permutation representations for scheduling problems. In: Voigt HM, Ebeling W, Rechenberg I, Schwefel HP (eds) *Parallel Problem Solving from Nature PPSN IV*, Lecture Notes in Computer Science, vol 1141. Springer, Berlin, pp 310–318
- Birattari M, Balaprakash P, Stutzle T, Dorigo M (2008) Estimation-based local search for stochastic combinatorial optimization using delta evaluations: A case study on the probabilistic traveling salesman problem. *INFORMS J Comput* 20(4):644–658
- Campbell A, Thomas B (2006) PTSPD benchmark data sets. <http://myweb.uiowa.edu/bthoa/PTSPD BenchmarkDataSets.htm>. Accessed 22 Nov 2011
- Campbell A, Gendreau M, Thomas B (2011) The orienteering problem with stochastic travel and service times. *Annals Operat Res* :1–21
- Campbell AM, Thomas BW (2008) Probabilistic traveling salesman problem with deadlines. *Transp Sci* 42(1):1–21
- Campbell AM, Thomas BW (2009) Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines. *Comput Operat Res* 36(4):1231–1248
- Chang TS, Wan YW, Ooi WT (2009) A stochastic dynamic traveling salesman problem with hard time windows. *Eur J Operat Res* 198(3):748–759
- Charnsirisakskul K, Griffin PM, Keskinocak P (2004) Order selection and scheduling with leadtime flexibility. *IEE Transactions* 36:697–707
- Chen S, Golden B, Wong R, Zhong H (2009) Arc-routing models for small-package local routing. *Transp Sci* 43(1):43–55

- Datamonitor (2009) Express logistics industry profile: Europe. Tech. rep., <http://www.datamonitor.com>. Accessed 3 Nov 2011
- Datamonitor (2010a) Express logistics industry profile: Europe. Tech. rep., <http://www.datamonitor.com>. Accessed 3 Nov 2011
- Datamonitor (2010b) Express logistics industry profile: United States. Tech. rep., <http://www.datamonitor.com>. Accessed 13 May 2011
- Dumas Y, Desrosiers J, Gelinat E, Solomon MM (1995) An optimal algorithm for the traveling salesman problem with time windows. *Operat Res* 43(2):367–371
- Ehmke JF, Steinert A, Mattfeld DC (2012) Advanced routing for city logistics service providers based on time-dependent travel times. *J Comput Sci* 4(3):193–205
- Erera AL, Savelsbergh M, Uyar E (2009) Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks* 54(4):270–283
- FedEx (2005) Surface mode optimization blended services freight forwarder service guide FDFE 201-C. <http://customcritical.fedex.com/us/serviceinfo/documents/pdf/fdff201c.pdf>. Accessed 3 Nov 2011
- FedEx (2011) Service guide: Money-back guarantee. <http://www.fedex.com/us/service-guide/options/money-back-guarantee.html>. Accessed 3 Nov 2011
- Goodson JC, Ohlmann JW, Thomas BW (2012) Rollout policies for dynamic solutions to the multi-vehicle routing problem with stochastic demand and duration limits. *Operations Research Submitted*
- Hansen P, Mladenovic N, Brimberg J, Moreno ~ Perez JA (2010) Variable Neighborhood Search, chap. 3. Springer US, Boston, pp 61–86
- Jaillet P (1988) A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operat Res* 36(6): 929–936
- Jaillet P, Qi J, Sim M (2012) Routing optimization with deadlines under uncertainty Working paper
- Lei H, Laporte G, Guo B (2011) The capacitated vehicle routing problem with stochastic demands and time windows. *Comput Operat Res* 38(12):1775–1783
- Li X, Tian P, Leung SC (2010) Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm. *Int J Prod Econ* 125(1):137–145
- Liu YH (2008) Diversified local search strategy under scatter search framework for the probabilistic traveling salesman problem. *Eur J Operat Res* 191(2):332–346
- Manni E (2009) Topics in real-time fleet management. *4OR: A Quarterly J Operat Res* 7:203–206
- Marinakis Y, Marinaki M (2009) A hybrid honey bees mating optimization algorithm for the probabilistic traveling salesman problem. In: 2009 IEEE Congress on Evolutionary Computation, IEEE, pp 1762–1769
- Marinakis Y, Marinaki M (2010) A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem. *Comput Operat Res* 37(3):432–442
- Marinakis Y, Migdalas A, Pardalos PM (2008) Expanding neighborhood search– GRASP for the probabilistic traveling salesman problem. *Optim Lett* 2(3):351–361
- Mohan S, Gendreau M, Rousseau JM (2008) The stochastic Eulerian tour problem. *Transp Sci* 42(2):166–174
- Schultz JD (2008) 2008 is it trucking’s bounce back year?. *Logistics Manag* 47(3):S65 – S71
- da Silva RF, Urrutia S (2010) A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optim* 7(4):203–211
- da Silva RF, Urrutia S (2010) The traveling salesman problem with time windows—approaches & additional resources. <http://homepages.dcc.ufmg.br/rfsilva/tsptw>. Accessed 22 Nov 2011
- Slotnick SA, Sobel MJ (2005) Manufacturing lead-time rules: Customer retention versus tardiness costs. *Eur J Operat Res* 163(3):825–856
- Tang H, Miller-Hooks E (2005) Algorithms for a stochastic selective traveling salesperson problem. *J Operat Res Soc* 56:439–452
- Terreri A (2011) Technology powers LTL leaders. *World Trade, WT* 100 24(4):18–23
- United Parcel Service (2011) Support FAQ. <http://www.ups.com/content/us/en/resources/sri/ctc1.html>. Accessed 3 Nov 2011
- Weyland D, Bianchi L, Gambardella L (2009) New Approximation-Based Local Search Algorithms for the Probabilistic Traveling Salesman Problem, *Lecture Notes in Computer Science*, vol 5717. Springer, Berlin, pp 681–688
- Weyland D, Montemanni R, Gambardella LM (2011) New heuristics for the probabilistic traveling salesman problem with deadlines based on quasi-parallel monte carlo sampling, submitted
- Weyland D, Montemanni R, Gambardella LM (2012) Hardness results for the probabilistic traveling salesman problem with deadlines Working paper