Runtime Reduction Techniques for the Probabilistic Traveling Salesman Problem with Deadlines

Ann Melissa Campbell *, Barrett W. Thomas

Department of Management Sciences, University of Iowa
108 John Pappajohn Business Building, Iowa City, Iowa 52242-1994

Abstract

The probabilistic traveling salesman problem with deadlines (PTSPD) is an extension of the well-known probabilistic traveling salesman problem in which, in addition to stochastic presence, customers must also be visited before a known deadline. For realistically sized instances, the problem is impossible to solve exactly, and local search methods struggle due to the time required to evaluate the objective function. Because computing the deadline violations is the most time consuming part of the objective, we focus on developing approximations for the computation of deadline violations. These approximations can be imbedded in a variety of local-search methods, and we perform experiments comparing their performance using a 1-shift neighborhood. These computational experiments show that the approximation methods lead to significant run time improvements without loss in quality.

Key words: Traveling salesman problem, deadlines, heuristics, stochastic

* Corresponding author.
Email addresses: ann-campbell@uiowa.edu (Ann Melissa Campbell), barrett-thomas@uiowa.edu (Barrett W. Thomas).

1. Introduction

Whether they are called expedited, express, or time-definite delivery, there is no question that these services now dominate the delivery business. The most common example of these services is the next-day package delivery featured by United Parcel Service (UPS) and FedEx, who offer a choice of deadlines such as 10 am, noon, or 3 pm for these deliveries. Such time-definite services have grown from just 4% of the parcel delivery market in 1977 to over 60% in 2002 [1]. The market for all time-definite cargo was expected to grow by 7.6% in 2006 [2], and the growth is expected to continue. The growth in time-definite services has created a challenge for many delivery providers. Many are struggling with how to create routes that satisfy time guarantees yet remain cost efficient. In [3], we introduced the Probabilistic Traveling Salesman Problem with Deadlines (PTSPD) to address the need for time-definite routing in an a priori routing environment.

An a priori, or pre-planned, route is a route which specifies 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 receive a delivery. These a priori routes are commonly used in the package express industry because of technology limitations as well as hidden costs associated with daily reoptimization [4]. Even for companies with the appropriate information and computing technologies, a priori routes are appealing because they are easily implementable. A priori routes offer both drivers and customers consistency and help to improve driver efficiency as the driver becomes increasingly familiar with the route [5, 6].

Formally, the PTSPD is the problem of finding a minimum expected cost a priori tour through a set of customers $N = \{i \mid 1, \ldots, n\}$ with probabilities $P = \{p_i \mid 1, \ldots, n\}$ of requiring service on any given day. Associated with each customer $i \in N$ is a known deadline l_i. The travel time between any

two customers i and j is given by d_{ij}, where $d_{ij} = d_{ji}$, and travel times are assumed to satisfy the triangle inequality.

While our computational experiments in [3] demonstrate the expected cost savings possible from modeling a time-constrained a priori routing problem with probabilistic information, they also show that, like many stochastic problems, the probabilistic model requires significant computation time to solve. To solve problems such as these, many researchers turn to local search heuristics. Due to the expense of evaluating the objective function, however, using these local search techniques alone are not computationally efficient mechanisms for solving the PTSPD. As shown in [3], in computing the expected cost of a tour, the greatest portion of the computational effort is in computing the deadline violations. Due to the probabilistic nature of the objective function, evaluating a small change in an a priori route, such as in local search procedures, can easily be as expensive as evaluating the cost of the full objective function. Consequently, we focus on techniques for approximating the violation portion of the objective function. These approximations can be imbedded in a variety of local-search methods. These ideas differ from many of the solution approaches for the PTSP that rely on exact computation of expected solution values, even when they are obtained using heuristics.

We present three methods for reducing the computation time required for the determining deadlines violations while still preserving solution quality. It is important to note that the presented approximations methods are not specific to the PTSPD. Rather, they have application in any discrete-time problem in which the timing of events is stochastic. Our three approximation methods are expected arrival time, temporal aggregation, and truncation. As the name implies, the expected-arrival-time approximation measures deadline violations only by the expected arrival time to a customer. The temporal aggregation scheme simplifies computation by measuring time in units larger than those in which the penalty is defined. Finally, the truncation approximation computes

arrival time probabilities, and thus violations, for a limited set of customers when evaluating the cost of neighboring tours.

The remainder of this paper is organized as follows. Section 2 summarizes related literature. Section 3 introduces a formal model for the PTSPD, and Section 4 discusses the proposed approaches for improving computational efficiency within a local-search framework. In Section 5, we detail the design of our computational experiments, and in Section 6, we present the results of our experiments. Section 7 considers the effectiveness of our solution approaches in the context of a fixed charge penalty for lateness. Finally, Section 8 presents insights and future work.

2. Literature Review

The research presented in this paper focuses on a problem in which customer presence on the tour is random. In an early treatment of stochastic routing, Bartholdi et al. [7] introduce a space-filling curve heuristic for constructing a priori tours for meals-on-wheels routing. Jaillet [8] formally introduces the PTSP and demonstrates some interesting properties of optimal tours including the fact that such a tour may intersect itself. Jaillet [9] provides a formulation for the expected value of a tour and bounds the relationship between optimal PTSP and TSP solutions. Laporte et al. [10] provide an exact algorithm for the PTSP. As indicated in Section 1, this exact approach is limited to small problem sizes, so much of the PTSP literature focuses on heuristic approaches. Bertsimas et al. [11] discuss space-filling curve and iterative heuristics. Bertsimas and Howell [12] and Chervi [13] introduce equations for efficiently evaluating the cost of local-search moves for the PTSP. Bianchi et al. [14] and Bianchi and Campbell [15] provide corrections for the equations in Bertsimas and Howell and in Chervi, respectively. Recent work by Campbell [16] and Tang and Miller-Hooks [17] focuses on approximations for

the PTSP. Overviews of the research in this area can be found in Powell et al. [18], Bertsimas and Simchi-Levi [19], and Gendreau et al. [20].

While the literature contains research into many constrained versions of the TSP, there is limited research into constrained versions of the PTSP. The best known of the constrained versions is the stochastic vehicle routing problem (SVRP). The SVRP requires the consideration of vehicle capacity in the formation of the tours, and rather than customer presence, customer demand is usually the stochastic element of the problem. The first mention of this problem can be found in Tillman [21]. Bertsimas [22, 23] introduces an analytical framework and bounds for the SVRP. Other work can be divided into consideration of chance constrained and recourse model formulations. Stewart and Golden [24], Laporte et al. [25], and Bastian and Rinnooy Kan [26] provide chance constrained formulations and demonstrate how they can be transformed into deterministic problems. Dror et al. [27], Dror [28], and Bastian and Rinnooy Kan [26] present stochastic programming solutions to various recourse models for the SVRP. Many offer heuristics for the SVRP, including Dror and Trudeau [29], Bramel et al. [30], Bertsimas et al. [31], Savelsbergh and Goetschalckx [32], and Yang et al. [33]. Gendreau et al. [34] offer a stochastic programming approach for an SVRP variant in which both customer presence and customer demand is stochastic. Gendreau et al. [35] offer a tabu-search heuristic for the same problem.

As noted earlier, Campbell and Thomas [3] introduce the PTSPD, providing two recourse models and a chance-constrained model for the problem. In addition, computational experiments demonstrate situations in which it is important to model the problem stochastically versus situations in which deterministic models are sufficient. The authors know of only a few other papers that address routing under uncertainty with time constraints. These papers consider time constraints in the context of stochastic travel times. Teng et al. [36] apply the L-shaped algorithm to the time-constrained traveling salesman

problem (TCTSP) with stochastic travel and service times. In the TCTSP, the time constraint is on the length of the tour, which contrasts with this paper where the time constraints control when individual customers can be visited. Wong et al. [37] introduce a 2-stage stochastic integer program with recourse for a problem where customers have time windows and travel times are stochastic. Finally, Wang and Regan [38] consider truckload assignments for time-constrained deliveries in the case that service and travel times are stochastic.

3. Model Formulation

In this paper, we seek to improve the computational efficiency of local-search methods for the PTSPD. We focus on the Recourse I model for the PTSPD with a per-unit-time penalty which was introduced in [3]. 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 [39].

In this model, the vehicle is allowed to visit a customer i after the delivery deadline has passed, but incurs a per-unit-time penalty, λ_i, for doing so. The tour departs from the depot at time 0 and returns to the depot after visiting all realized customers on the tour. For a given tour τ, we can compute the latest possible arrival time at each customer i, T_i, and the probability, $g(i,t)$, that the vehicle arrives at customer i at time t ($t < T_i$). The expected cost of tour τ is then:

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

$$+ \sum_{i=1}^{n} p_i d_{i0} \prod_{k=i+1}^{n}(1-p_k) \tag{1}$$

$$+ \sum_{i=1}^{n} p_i \sum_{t=l_i+1}^{T_i} \lambda_i g(i,t)(t-l_i). \tag{2}$$

Equation 1 represents the expected travel cost, and Equation 2 defines the expected penalty cost associated with late arrivals.

The probability $g(i,t)$ can be computed recursively using the following equations. When $t < d_{0i}$, $g(i,t)$ values will always be zero, since arrival cannot occur any earlier than with a direct trip from the depot. When $t = d_{0i}$, arrival at t can occur only if no prior customers are realized:

$$g(i,t) = \prod_{k=1}^{i-1}(1-p_k). \tag{3}$$

When $t > d_{0i}$, arrival at i is based on all of the possible preceding customers and their possible departure times:

$$g(i,t) = \sum_{h=1, t>d_{0h}}^{i-1} p_h g(h, t-d_{hi}) \prod_{k=h+1}^{i-1}(1-p_k). \tag{4}$$

Note that $g(i,t)$ values are not necessarily increasing or decreasing with t, but are strictly based on the combinations of travel times that may occur. A detailed discussion of this model as well as others for the PTSPD can be found in [3].

The complexity of the function evaluation is dominated by the computation of the g values and is $O(n^2 \max_i\{T_i\})$. In doing local search to find PTSPD solutions, we must evaluate the change in the expected cost for each neighboring solution considered. Thus, for each new solution generated in the search, the cost of evaluating the expected cost of the solution can be $O(n^2 \max_i\{T_i\})$. For the PTSP, Bianchi et al. [14] and Bianchi and Campbell [15] propose a

recursive approach for evaluating a local search neighborhood that greatly reduces complexity. Unfortunately, these same recursive ideas cannot be applied to the computation of the g values. Thus, even when using a simple 1-shift neighborhood, a neighborhood in which one customer i is moved to a later position on the tour, the $g(i,t)$ values for all customers after i must be recomputed.

4. Penalty Approximations

In this section, we introduce methods for approximating Equation 2. As discussed previously, Campbell and Thomas [3] demonstrate that the computation of the penalty, through the g values, is the most computationally expensive portion of evaluating the PTSPD objective. In fact, experiments show the computation of the penalty portion can account for over 99% of the computation time.

The following approximations are used to evaluate the cost of a new tour in a local search approach. Even though it is possible to also approximate Equation 1, we still exactly compute the travel cost portion of the objective (Equation 1) when evaluating a tour since it requires so little time relative to the penalty cost portion (Equation 2).

Three approximations follow: expected value, temporal aggregation, and a truncation approximation.

4.1 Expected Value

The g values in the above equations reflect the probability of arrival at i at each of its possible arrival times. Considering each of the possible arrival times is what makes the function evaluation so expensive. Our first approximation method is based on computing an expected arrival time at each customer and

using this time to compute our penalty value. The complexity of computing this expected arrival time is based only on the preceding customers (up to $n-1$) and not on T_i.

To begin, let the random variable

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

We let A_i be a random variable representing the arrival time to customer i for a given tour. Then, $\tilde{A}_i = E[A_i \mid X_i = 1]$, where E is the expectation operator. That is, $\tilde{A}_i$ is the expected arrival time to customer i given customer i is realized. We can assume that $E[A_i \mid X_i = 0] = 0$. We can calculate $\tilde{A}_i$ through the following straightforward recursion:

$$\tilde{A}_i = \sum_{j=1}^{i-1} p_j(\tilde{A}_j + d_{ji}) \prod_{k=j+1}^{i-1} (1 - p_k),$$

noting that $\tilde{A}_1 = 0$ by assumption.

We can estimate the size of the penalty incurred at i based on the size of $\tilde{A}_i$ relative to l_i. Using the per-unit-time penalty λ_i, we can replace Equation 2 with the following term:

$$\sum_{i=1}^{n} p_i \lambda_i \max(\tilde{A}_i - l_i, 0).$$

This approximation method allows us to complete a function evaluation now in $O(n^2)$ time. This method is simple to implement, in that it does not require any parameter selection as in the succeeding approaches, and is quite intuitive. Using expected arrival times, though, is unlikely to be sufficient in itself for solving the PTSPD. If it is used in a local search approach and the local search converges, several improving moves likely still exist with regard to the original

version of the objective function. Thus, we recommend using the expected-value approach in two phases. In the first phase, we perform the search using the described expected-value approach to evaluate the penalty cost. When the search under the penalty approximation converges, we use the newly found solution to seed a second phase of the search in which we evaluate the penalty portion directly. We use this two-phase approach in Section 6.

Implementation choices exist with regard to whether or not the expected value approach should be used until the local search converges or until improvements found with respect to the approximate objective are no longer improving with respect to the original objective function. Although verifying each move with respect to the original objective function requires an added full function evaluation at each iteration, our experiments show that it is a more dependable approach. We found that using the approximate objective function without this extra check often led to cycling in the local search procedure. Thus, we recommend the additional check with regard to the full objective function in order to ensure convergence.

4.2 Temporal Aggregation

If penalties are assessed based on the number of minutes that a delivery is late, it is necessary for accuracy to compute the $g(i,t)$ values with the t indices representing minutes. If there are a large number of customers or if the travel times between some customers are quite long, the T_i values can easily become quite large and exponential in n. This makes the objective function very expensive and time consuming to evaluate.

In a local search scheme, it is typical to choose the change to the current solution that makes the largest improvement in the objective value. In this context, we can think of a changes that creates a large reduction in travel cost or reduces the lateness at customers by hours rather than minutes. This idea

motivates our temporal aggregation scheme. Instead of making the t values represent minutes, or whatever the final time discretization that is required for the penalty calculation, we consider larger time discretizations and gradually refine it until the final time discretization is used. In this way, the objective function will reflect the expected travel cost and an estimate of the expected penalty cost.

Aggregation has a long history in the literature. Rogers et al. [40] offer a framework for using aggregation as well as an extensive review of early work. Multigrid techniques, a form of aggregation, have been used extensively in the solutions of partial differential equations [41], and recently multigrid approaches have been used to accelerate nonlinear programming algorithms [42]. Aggregation has been frequently used in facility location problems to reduce the number of of customer locations [43] and has been applied recently in routing problems, including the PTSP [16]. Temporal aggregation has been applied in economic models [44] and in integer programming [45]. Most applications of temporal aggregation in integer programming are based on increasing the size of time periods for which decisions are made in an attempt to reduce the number of decision variables.

In a temporal aggregation scheme, the units of the T_i, l_i, and d values will be changed, and these new values will be used in the penalty calculation. We will refer to the new values by T_i^*, l_i^*, and d^*, respectively. These constants dictate the largest t index that needs to be evaluated for each i in creating the g values.

The first step in a temporal aggregation scheme is to decide the new larger time units that will be used in evaluating the penalty function. The largest time discretization used should be large enough to gain computational advantage in the penalty calculation, but small enough such that penalty improvements can be found. If the original time discretization is in minutes, the larger dis-

cretization does not need to be hours but could be, for example, 3 minutes, 40 minutes, or 180 minutes. The largest time discretization that is used will vary and will clearly have a relationship to the customer dataset being considered.

We propose one general suggestion for the choice of the largest time discretization. Since the complexity of evaluating the objective function is potentially non-polynomial in n due to the T values, we propose the use of time units of size v where $v = \frac{\max_i\{T_i\}}{n}$. Now, the largest T_i^* can be is $\frac{\max_i\{T_i\}}{v}$ which is now n, making an objective function evaluation possible now in $O(n^3)$ time.

For a given level of aggregation, the next step in developing a temporal aggregation scheme is transforming the d^*, T_i^*, and l_i^* values to reflect the new time discretization. We do this by dividing the original d, T_i, and l_i values by v, our chosen level of aggregation. Thus, for every customer i, $l_i^* = \frac{l_i}{v}$. Because the l_i^* values are used as indices in the g functions, they must obtain integer values. Thus, a simple transformation using a particular time discretization v is to round each $\frac{l_i}{v}$ to its nearest integer to obtain l_i^*. For all customers i and j, $d_{ij}^* = \frac{d_{ij}}{v}$. We repeat the rounding process to create the new distance values d^*. With the d^* values, for each i, we can compute $T_i^* = \sum_{k=0}^{i-1} d_{k,k-1}^*$.

Computation of the penalty cost approximation requires computing new g^* values. Note that Equation 3 does not change, just the value of d_{0i} to d_{0i}^*. Equation 4, though, becomes Equation 5:

$$g^*(i,t) = \sum_{h=1, t>d_{0h}^*}^{i-1} p_h g(h, t - d_{hi}^*) \prod_{k=h+1}^{i-1} (1 - p_k). \tag{5}$$

Now, Equation 2 becomes:

$$\sum_{i=1}^{n} p_i \sum_{t=l_{i+1}^*}^{T_i^*} \lambda_i g^*(i,t)(t - l_i^*)v. \tag{6}$$

Note that v is included in the new penalty calculation, so that the magnitude of the penalty is preserved. For time units of size v, a function evaluation now

requires $O(n^2 \max_i\{\frac{T_i}{v}\})$.

Another component of the design of a temporal aggregation scheme is to decide when and how the level of aggregation is changed. It makes sense to keep a particular level of discretization until the local search converges. We have the choice, though, of whether or not to allow moves that are improving only at the coarse time discretization but are not improving at the final level of aggregation. We recommend verifying that the move selected at each iteration is improving with regard to the final level of discretization, and changing the level of discretization if it is not. As with the expected value approach, our computational experiments found that cycling often occurred when doing local search at a particular level of discretization without this verification step.

Once the decision has been made to change the level of discretization, the choice is what level of aggregation to use next. This decision can be tuned to various datasets, but an obvious choice is to keep dividing the discretization level until the final level of discretization is reached. This approach is used in Section 6. This dividing method does not always lead to the final level of discretization, though. Consider for example, choosing 30 minutes as a coarse representation of time rather than individual minutes. The v values would go from 30 to 15 to 7.5 to 3.75 to 1.875 to 0.9375 minutes. In such a case, when v becomes less than one minute, the final level of discretization (one minute) should be used.

4.3 Truncation Approximation

In the PTSPD, even a small change in the tour involving position i will impact the expected arrival times, and thus expected penalties, at all customers succeeding i on the route. Due to the probabilistic nature of the problem, though, the change in penalty will be largest for the customers that are served just after i on the tour. Based on this idea, we propose evaluating the change

in the penalty portion of the objective associated with a local search move by considering only the q nearest neighbors to each customer.

Analogous truncation schemes have some history in the literature. Gendreau et al. [35] use a truncation approximation to compute the travel portion of the expected cost in an SVRP. For the PTSP, Tang and Miller-Hooks [17] embed approximation within a threshhold accepting heuristic. Campbell and Savelsbergh [46] use a similar idea to estimate the cost of new service requests for home-delivery routing.

In the full g calculation (Equation 4), we consider travel to i from all of its possible preceding customers. Direct travel to i from customer $i-q$ is very unlikely if q is high and the customers between $i-q$ and i have a reasonable probability of occurring. In fact, the probability that none of the customers between $i-q$ and i require a delivery is

$$\prod_{j=i-q+1}^{i-1}(1-p_j). \tag{7}$$

For many values of q and p, Equation 7 has a value near zero. For example, if all customers have probability of 0.5, there is only a 3% chance that direct travel will occur between customers six stops apart on the same tour. Our truncation approximation is based on replacing these "near zero" probabilities strictly with zero rather spending the time to compute values that make little difference in the penalty portion of the objective.

To compute the penalty portion of the objective, Equation 2 remains the same, but now involves g^* terms rather than g. Compared to g, fewer g^* terms will have nonzero values. Our truncation approximation will only compute a value for $g^*(i,t)$ when $t = d_{0i}$ only if $i \leq q$. In other words, for $t = d_{0i}$ and $i > q$, we will set $g^*(i,t) = 0$. If $i \leq q$, Equation 3 is used. Next we modify Equation 4 to only consider the closest q customers to i:

$$g^*(i,t) = \sum_{h=\max\{i-q,1\},t>(d_{hi}+d_{0h})}^{i-1} p_h g^*(h, t-d_{hi}) \prod_{k=h+1}^{i-1} (1-p_k). \quad (8)$$

Note that this approximation makes it such that no $g^*(i,t)$ calculation requires more than $O(q)$ complexity, assuming that the product portion values are stored and computed ahead of time. Now the full penalty cost can be computed in $O(nq \max_i\{T_i\})$ time rather than $O(n^2 \max_i\{T_i\})$. Thus, the size of q relative to n impacts the reduction in run time.

As with v, how to initialize and increment q is an important question. To prevent cycling, we recommend using a particular q until a non-improving move with regard to the exact objective function is selected by the local search procedure. Recall that increasing q will refine the level of aggregation. A straightforward method for initializing and incrementing q is to initialize $q = 1$ and then double q each time that q is to be incremented. When $q > n$, we can replace q with n and complete the search.

Note that with truncation, the computed violation is always an underestimate of the full penalty cost. Even though it is an underestimate, it should still be of a similar magnitude to the full penalty cost. More importantly, truncation will yield an estimate of the penalty cost that will reflect the relative impact of various changes in the solution considered by a local search procedure.

5. Experimental Design

To test the effectiveness of the proposed solution approaches, we perform a set of computational experiments using realistically-sized datasets. We use the 40- and 60-customer sets introduced in [3] for the PTSPD and introduce analogous 100-customer sets. All sets, 5 of each size, are derived from the time-window width 20 units TSPTW instances first proposed by Dumas et al. [47] and can be found at `http://myweb.uiowa.edu/bthoa/research.html`. For each of the 15 sets, we generate two instances which differ only in the deadlines. For

the first instance, we set the deadline equal to the opening time of the time window unless that time is zero in which case the deadline is set to the closing time of the time window. For the second instance, we set all deadlines equal to the closing time of each customers' time window. These instances will hereafter be referred to as "early" and "late" deadlines, respectively. The early deadline instances are used to represent situations where feasible solutions with respect to deadlines are very unlikely to exist if all customers are realized. With the later deadlines, even when all customers are realized, there exist solutions that do not violate any deadlines.

For each of the now 30 instances, we then consider four different probability settings. Two of these settings are homogeneous settings in which all probabilities are set to 0.1 and 0.9, respectively. These two settings represent when each customer is unlikely to be realized or, alternately, very likely to be realized. The other two settings are heterogeneous. In the first case, the probabilities are randomly assigned to each customer with the probabilities ranging between 0 and 1. This helps us understand how the results change when there are more options in terms of customer probabilities. In the second case, we randomly assign probabilities of either 0.1 or 1. This case addresses the situation in which large and small businesses are served by the same vehicle. These two data sets will be referred to in the tables by the labels "range" and "mixed," respectively.

Finally, for the 40- and 60-customer sets, we test per-unit-time penalties of 5 and 50. These choices of penalties represent small versus large costs for failure to satisfy the customer deadlines. For the 100-customer sets, we limit ourselves to penalties of 5 due to the longer run times of these larger datasets. The result is a total of 200 instances, 80 each with 40- and 60-customers, and 40 with 100-customers.

A dataset class is defined by the number of customers, deadline (early or late),

penalty (5 or 50), and probability setting (0.1, 0.9, range, or mixed). Each class will consist of 5 datasets.

The ultimate aim of our computational experiments is to demonstrate that the approximations can reduce computation time without degrading solution quality. An approximation scheme that is computationally effective but provides poor solutions is of dubious value. For this purpose, we focus on a proven solution methodology which offers reasonable ease in implementation. In particular, we focus on local-search heuristics. Because Ohlmann and Thomas [48], Cheh et al. [49], and Carlton and Barnes [50] use the 1-shift neighborhood for the TSPTW, we use this neighborhood to solve our closely related problem. For initial solutions, we use solutions to the deterministic version of the PTSPD, the TSPD, generated by the approach outlined in [3]. By starting from the solutions to the TSPD, the run times become an indicator of how different the optimal solutions are to the probabilistic and deterministic versions of the problem. To test the effectiveness of the new approximation procedures on the proposed datasets, we implement a best improvement local search using this 1-shift neighborhood (for additional discussion of best improvement local search, we refer the reader to [51]). For all tests, the travel cost portion (Equation 1) of the expected cost is determined by computing the exact change in cost between the neighboring and incumbent solution. Within this search framework, we then examine each of the three approximations as well as an exact computation of the g values and penalties. For all approximations, we do not allow a non-improving move with respect to the full objective function.

In the case of the expected-value approximation, we begin the search by computing the penalty via the expected cost. For each iteration, the expected cost of the best found solution is computed. If the solution is found to be non-improving, the search is re-calibrated, and the g values are then computed directly rather than via the expected value.

We follow analogous search methodologies for the temporal and truncation approximations. For the temporal approximation, we initialize the discretization as suggested in Section 4.2. When the search returns a best found solution for a particular value of v such that the solution is non-improving, we divide v by two and continue the search. The search continues until no improving solution is found for v equal to the original level of discretization. Similarly, for the truncation approximation, we initialize q at 1 and increment q as discussed in Section 4.3. The search continues in this fashion until no improving solution is found for $q = n$, where n is the number of customers.

6. Computational Comparisons

In order to demonstrate the effectiveness of the proposed approximations, we test the approximations both in terms of solution quality and run time. In the tables that follow, we refer to the truncation approximation as "TC," temporal aggregation as "TA," and the expected-value approximation as "EA."

To test the solution quality of the approximations, we compare the solution values returned by using each of the three approximations. For the $n = 40$ and $n = 60$ datasets, we compare these values to the solution values returned by computing the exact change in solution cost for each neighboring solution in the search. We label this last local search simply PTSPD. Table 1 presents the results of the comparisons. For each approximation, we present the percentage difference between the average PTSPD solution value and average solution value found using the approximation for each class of datasets. The percentage difference for each class is calculated as $\frac{(\text{Average PTSPD Value} - \text{Average Approximation Value})}{\text{Average PTSPD Value}}$. A positive percentage indicates that the approximation has returned better solution values than the PTSPD algorithm, and negative values indicate that the approximation yields worse solution values.

Table 1
Comparison of Solution Values between PTSPD Algorithm and Approximations

Prob			0.1			0.9			Range			Mixed		
Dataset			TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
$n =$	Deadline	Penalty												
40	Early	5	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	Late	5	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	-2%	0%
	Early	50	0%	0%	0%	0%	0%	0%	1%	-12%	0%	0%	0%	0%
	Late	50	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
60	Early	5	0%	0%	0%	0%	0%	0%	-1%	0%	0%	0%	0%	0%
	Late	5	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	Early	50	0%	0%	0%	0%	0%	0%	-1%	0%	0%	0%	0%	0%
	Late	50	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

As Table 1 shows, the approximations almost always return the same values as the PTSPD algorithm. The temporal approximation performs particularly poorly on the instances with 40 customers, early deadlines, a penalty of 50, and the range of probabilities. However, the average is driven by a single instance. Such occasions of the approximations leading to premature convergence is rare.

Run times for the PTSPD algorithm were prohibitive for the $n = 100$ sets, so for these sets, we made comparisons only among the approximations. Table 2 presents comparisons of the solution values returned by the approximation methods for the 100-customer instances. For each deadline category (early and late), we present the percentage difference between the temporal aggregation and the expected-value approximation, temporal aggregation and the truncation approximation, and the expected-value approximation and the truncation approximation. The percentage difference for each class and approximation pair is calculated as $\frac{(\text{Average Approximation 1 Value} - \text{Average Approximation 2 Value})}{\text{Average Approximation 1 Value}}$. A positive percentage indicates that the second approximation yields lower cost solutions than the first while a negative percentage indicates the first approximation performs better. As the table indicates, other than some small percentage differences on the sets with homogeneous probabilities of 0.1, the approxima-

tions return the same values.

Next, we will examine how the approximations impact run times. Table 3 presents the run times for the PTSPD, the truncation, temporal, and expected-value solution approaches, respectively, for the $n = 40$ and $n = 60$ datasets. Table 4 compares the run times of the approximations to those of the PTSPD in order to help evaluate the relative performance of the different approaches. As in Table 1, Table 4 presents the percentage difference between the averages of the PTSPD algorithm and each of the approximations. From Table 3, we can observe that the run times increase dramatically from the 40- to 60-customer sets. This run time growth emphasizes the need to reduce run times as the number of customers increases.

When the probability is homogeneously set to 0.1, we find that all of the approximation methods are, on average, faster than the PTSPD algorithm. The explanation for this difference is the large differences in the seed routes and the routes returned by the PTSPD algorithm and the approximations. Figure 1 visually demonstrates this difference for one of the $n = 40$, early deadlines, penalty of 5, and $p = 0.1$ instances. Because some of the changes lead to dramatic changes in the contribution of the penalty factor to the expected cost, the approximation procedures are successful in identifying improving moves and can converge more quickly than the PTSPD algorithm. For example, for the instance depicted in Figure 1, the PTSPD algorithm requires 23 iterations of full computations to converge. The expected-value approximation had only 6 of its 27 iterations involving full computation of the objective, the temporal approximation had 3 of 27 iterations, and the truncation approximation only had 1 of 23 iterations. Among the approximations, the expected-value approach is never, on average, the most successful approximation. Instead, we see that temporal aggregation is always the best for the 40 customer dataset classes and one of the 60 customer dataset classes, and truncation is best for the remaining 60 customer datasets. The relative success of temporal aggre-

Table 2. Comparison of Solution Values between Approximations for 100-Customer Instances

Prob	0.1			0.9			Range			Mixed		
Deadline	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC
Early	1%	-2%	-2%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Late	0%	0%	-1%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Table 3. Average Run Times in CPU Seconds of Approximations on 40- and 60-Customer Instances

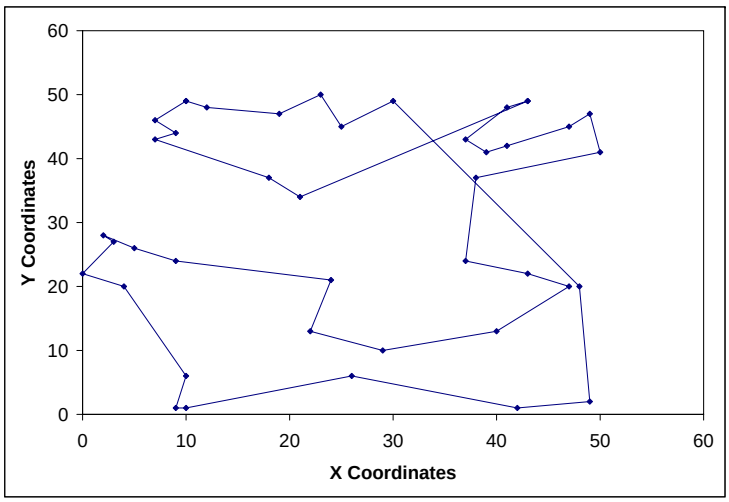
Prob			0.1				0.9				Range				Mixed			
Dataset			PTSPD	TC	TA	EA	PTSPD	TC	TA	EA	PTSPD	TC	TA	EA	PTSPD	TC	TA	EA
$n =$	Deadline	Penalty																
40	Early	5	58.4	36.4	23.0	34.0	9.2	20.2	12.4	10.0	29.4	23.2	18.8	28.8	37.0	25.2	26.0	33.6
	Late	5	35.4	24.0	14.2	15.8	7.8	16.6	11.0	5.6	22.8	23.6	13.6	16.0	24.8	23.6	14.6	19.4
	Early	50	111.4	57.4	42.4	95.0	59.2	57.2	33.0	58.8	89.6	63.6	72.4	92.0	97.8	78.0	66.0	90.2
	Late	50	41.8	24.6	16.8	19.0	8.8	16.4	10.4	4.8	35.8	21.8	15.6	11.2	26.4	19.8	15.4	19.0
60	Early	5	831.4	362.2	350.6	617.6	111.4	138.4	137.6	102.2	374.6	180.0	222.6	351.2	400.4	211.0	284.4	370.8
	Late	5	1114.0	313.2	466.8	521.4	63.8	90.0	101.6	51.4	297.0	113.2	128.6	65.8	241.4	120.0	129.8	48.6
	Early	50	848.4	396.6	536.2	773.8	207.8	197.6	232.0	200.6	367.0	186.2	299.0	340.6	396.6	207.4	348.0	371.2
	Late	50	784.8	24.6	330.6	474.0	63.8	16.4	79.4	35.0	305.4	21.8	172.4	139.6	235.4	19.8	128.4	67.6

Table 4
Comparison of Run Times between PTSPD Algorithm and Approximations on 40- and 60-Customer Instances

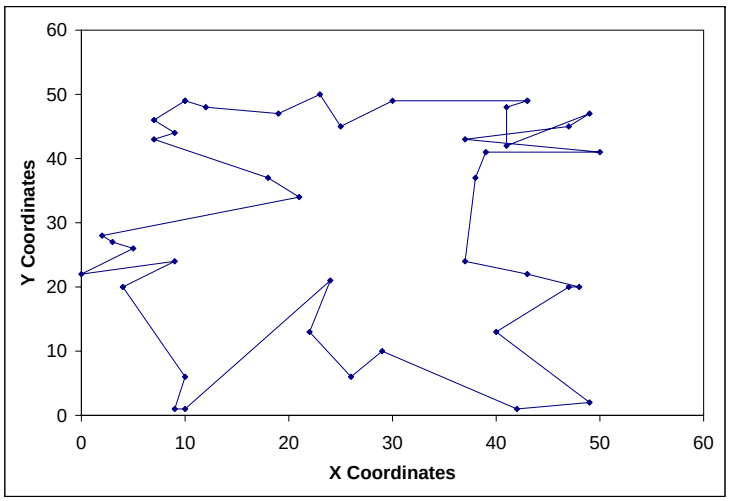
Prob			0.1			0.9			Range			Mixed		
Dataset			TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
$n =$	Deadline	Penalty												
40	Early	5	38%	61%	42%	-120%	-35%	-9%	21%	36%	2%	32%	30%	9%
	Late	5	32%	60%	55%	-113%	-28%	28%	-4%	40%	30%	5%	41%	22%
	Early	50	48%	62%	15%	3%	44%	1%	29%	19%	-3%	20%	33%	8%
	Late	50	41%	60%	55%	-86%	-18%	45%	39%	56%	69%	25%	42%	28%
60	Early	5	56%	58%	26%	-24%	-24%	8%	52%	41%	6%	47%	29%	7%
	Late	5	72%	58%	53%	-41%	-59%	19%	62%	57%	78%	50%	46%	80%
	Early	50	53%	37%	9%	5%	-12%	3%	49%	19%	7%	48%	12%	6%
	Late	50	97%	58%	40%	74%	-24%	45%	93%	44%	54%	92%	45%	71%

gation and truncation results from their ability to not only quickly identify the moves which lead to large changes, but to also offer reduced computation time for moves that lead to smaller changes.

For the instances where the probability is homogeneously set to 0.9, the solutions tend to require fewer changes from the deterministic starting solution due to the larger probabilities. The result is a significant reduction in run times relative to the 0.1 probability instances across all approximation approaches. We also see that each type of approximation outperforms the others, on average, for at least one dataset class. At the same time, with 40 customers, early deadlines, and penalty equal to 5, all three approximations are slower than the PTSPD. This behavior is clearly unusual. We can also see that expected value performs very well for most dataset classes, where it performed much worse, relatively, with the lower probabilities. We conjecture that this is due to the nature of the expected-value approximation in combination with the structure of these datasets. Both the truncation and temporal aggregation require several "rounds" of a local search procedure, with each round corresponding to a particular level of truncation or discretization. Thus, truncation and temporal aggregation tend to be more successful when the overhead of each round is able to yield improvements in the solution values.



(a) Seed Routes



(b) PTSPD Route

Fig. 1. Comparison of Seed Route and PTSPD Route for $n = 40$, early deadlines, penalty of 5, and $p = 0.1$

As an example of the effect of the overhead in the truncation and temporal approximations, we present Table 5. For the dataset class in which $n = 40$, the penalty is 5, the deadlines are late, and the probabilities are 0.9, Table 5 shows the number of iterations performed at each round of the PTSPD, EA, TA, and TC heuristics, respectively. Round 0 is always the first level of approximation. Thus, for the expected-value approximation, Round 0 is the round in which the expected penalty cost is used to approximate the penalty

cost. For the temporal approximation, Round 0 is the case in which v is largest ($v = \frac{\max_i\{T_i\}}{n}$). For the truncation approximation, Round 0 is the case in which $q = 1$. For each heuristic, the highest numbered round for which there is an entry is the round at which the heuristic has reverted to fully evaluating the objective function.

As Table 5 shows, on the second and third instance, the EA heuristic requires 3 and 2 iterations, respectively, using the expected-value approximation of the penalty. The advantage of these iterations with the approximate penalty is demonstrated by the runtime reduction offered relative to the PTSPD. In the other three instances, the EA heuristic matches the PTSPD algorithm in the number of full objective evaluations required. Because the single iteration using the approximated penalty takes relatively no time, the two heuristics require almost exactly the same runtime on these three other instances. On the other hand, the reason for the lack of runtime improvement for TA and TC heuristics is visible in the numerous non-improving rounds (indicated by entries of 1 for the rounds). The runtime required for these non-improving rounds negates any advantage gained by not having to run a number of iterations of full objective computations.

Given the number of non-improving iterations run by the TA and TC heuristics, it is interesting to consider whether or not the algorithms can be terminated before convergence at the last round to reduce the computation time. The results in Table 5 suggest that a general rule would be to terminate after the first round in which no improvements were made after an improving round has occurred. For example, the first instance with TC has an improving second round, but then no improvements occur in Round 3. Hence, this rule would suggest terminating the search after Round 3. However, such a rule would often fail to capture a number of improving rounds for other instances. For example, for the fifth instance where $n = 60$ and the probability setting is 0.1, this rule would cause the TC heuristic to terminate after Round 1 and

Table 5
Number of Iterations at Each Round of Heuristic and Runtimes for Each Instance of $n = 40$, Late Deadlines, Penalty of 5, and Probabilities of 0.9

		Round							
Approximation	Instance	0	1	2	3	4	5	6	Runtime
PTSPD	1	2							9
	2	4							10
	3	2							9
	4	1							4
	5	1							7
EA	1	1	2						9
	2	3	1						3
	3	2	1						4
	4	1	1						4
	5	1	1						8
TA	1	2	1	1	1				10
	2	1	3	1	1				9
	3	2	1	1	1				9
	4	1	1	1	1				11
	5	1	1	1	1				16
TC	1	1	1	2	1	1	1	1	21
	2	3	1	1	1	1	1	1	16
	3	2	1	1	1	1	1	1	16
	4	1	1	1	1	1	1	1	15
	5	1	1	1	1	1	1	1	15

lose the 5.0% improvement that occurs in the subsequent five rounds. In the fourth instance of the dataset class with $n = 60$ and range probabilities, the improvement is 3.3% between the end of Round 1 (non-improving) and the end of Round 4. The results across these datasets highlight the instance specific behavior of the approximations and the difficulty of finding a general rule for early termination.

For both the mixed and range probabilities, we again see that each type of approximation outperforms the others, on average, for at least one dataset class. Somewhat to our surprise, the run times for each dataset class are similar between the two probability settings. With the mixed probabilities, as with probabilities of 0.1, all approximation methods are, on average, faster than the PTSPD algorithm. The results for the mixed and range settings also highlight a significant pattern in the run times for the expected-value approach.

We can easily observe here, but it holds for the other probability settings as well, that the performance of expected-value approximation is highly sensitive to the strictness of the deadlines. For the 40 and 60-customer datasets and mixed probability setting, the solutions for instances with early deadlines have average run time improvements of 9, 8, 7, and 6% where late deadlines yield improvements of 22, 28, 80, and 71%. The reason is related to the above discussion. Better performance on the late deadline sets occurs because, with late deadlines, the best-found solutions are likely to not violate any deadlines, and thus the penalties have less impact on the solutions. Hence, the overhead of truncation and temporal aggregation is not useful, and expected-value approximation is more computationally efficient.

Across the experiments, temporal aggregation tends to perform the best for the 40-customer datasets, where truncation performs the best across the 60-customer datasets. We believe this result is partially due to the overhead of the different procedures. For truncation, there tend to be more rounds of the local search procedure than with temporal aggregation. The fewer levels of discretization for temporal aggregation are successful and more efficient with 40 customers, but with 60 customers, the added levels associated with truncation tend to bring more rewards.

Table 6 presents the run times in CPU seconds for the approximations on the 100-customer datasets, and Table 7 presents comparisons of the run time values returned by the approximation methods for the 100-customer instances. In a manner analogous to Table 2, Table 7 presents the percentage difference in runtimes between the temporal aggregation and the expected-value approximation, temporal aggregation and the truncation approximation, and the expected-value approximation and the truncation approximation. Recall that a positive percentage indicates that the second approximation yields lower cost solutions than the first, on average, where a negative percentage indicates the first approximation performs better.

In the 100-customer results, the run times are much larger than with 60-customer datasets. The run time differences among the approximations can often be 30 minutes to an hour of computation time.

As with the 40 and 60-customer datasets, we see the lower run times for the instances with probability 0.9 relative to the instances with probability 0.1. We also see that temporal approximation is always dominated by the truncation approximation. This result can be seen in the fact that all of the "TA - TC" values are positive. A big portion of this is likely due to the fact that truncation should be able to yield significant improvements at each q as the size of the dataset gets larger.

Comparisons between the truncation and expected-value approximations depend on the deadlines of the instance in question. When the deadline is late, the expected-value approximation outperforms the truncation approximation, in same cases significantly so. As was the case in the earlier comparisons, the difference results from how likely the best-found solution is to include violations. When deadlines are late, the best-found solution is less likely to have violations and thus the expected-value approximation is more efficient.

7. Extensions to Fixed Charge Penalties

In this section, we test the effectiveness of our solution approaches in the case where the penalty is a fixed charge rather than per-unit-time penalty. 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 not met. Well-known examples of such penalties are FedEx's and UPS' money-back guarantees [52, 53].

To compute the expected cost of a tour with fixed charge penalties, we introduce $G(i, t)$ which is the probability that arrival at customer i occurs at or

Table 6. Run Times for Approximations on 100-Customer Instances

Prob	0.1			0.9			Range			Mixed		
Deadline	TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
Early	4709.0	13075.6	23689.0	2133.8	2490.4	2232.6	1659.2	3503.6	4189.0	2210.6	7354.8	5687.8
Late	2636.4	8875.4	9257.6	1289.4	2062.8	391.8	2010.4	4222.4	2347.8	3059.4	5486.8	4289.8

Table 7. Comparison of Run Times between Approximations for 100-Customer Instances

Prob	0.1			0.9			Range			Mixed		
Deadline	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC
Early	-81%	64%	80%	10%	14%	4%	-20%	53%	60%	23%	70%	61%
Late	-4%	70%	72%	81%	37%	-229%	44%	52%	14%	22%	44%	29%

before time t. To compute $G(i,t)$, we have

$$G(i,t) = \sum_{k=d_{0i}}^{t} g(i,k) = G(i,t-1) + g(i,t). \tag{9}$$

To compute the expected cost of a tour, we then replace Equation 2 with:

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

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

Implementing the approximations for the fixed-charge PTSPD is straightforward. For the temporal and truncation approximations, we first compute the approximation of g, g^*, as described in Sections 4.2 and 4.3, respectively. For each approximation, we then compute an approximation of G, G^*, by replacing Equation 9 with:

$$G^*(i,t) = \sum_{k=d_{0i}}^{t} g^*(i,k) = G^*(i,t-1) + g^*(i,t).$$

In the case of the expected-value approximation, we simply apply the fixed-charge penalty if the expected arrival time to customer i occurs after its deadline l_i.

For the 40-, 60-, and 100-customer datasets, we consider early and late deadlines and the four probability settings. In order to reduce the computational effort, our computational experiments for the fixed-charge penalty focus on only penalties of value 50. We test each approximation in the previously described manner.

Table 8 presents the comparison of solution values between the approximations. We compare the approximations in the manner in which we presented the results of the 100-customer datasets with per-unit penalties. For each deadline category (early and late), we present the percentage difference be-

tween the temporal aggregation and the expected-value approximation, temporal aggregation and the truncation approximation, and the expected-value approximation and the truncation approximation.

In contrast to the results for the per-unit-charge penalty, Table 8 shows that greater variation in the results returned by the approximations for the fixed-charge penalty. None of the approximations dominates any of the others across all instances. Except for the 100-customer and late deadline datasets with homogeneous 0.9 and range probabilities, however, the temporal and expected-value approximations return better solution values than the truncation approximation. At the same time, the temporal approximation returns better solution values than the expected-value approximation in eight of 24 dataset classes and is equal to the expected-value approximation in 10 others. Yet, the expected-value approximation never performs more than 4% worse than the temporal approximation.

Table 9 presents the run times in CPU seconds for the approximations on the fixed-charge datasets, and Table 10 presents run time comparisons for the proposed approximations. From Table 9, it is clear that run times increase with the number of customers, as they did with per-unit penalties. There is also some decrease in run time associated with the increase in probability from 0.1 to 0.9. We can see that expected value is faster than temporal aggregation in 21 out of the 24 dataset classes. We also see that truncation is faster than temporal aggregation in 19 out of 24 dataset classes. This observation indicates that temporal aggregation is not the best choice with fixed charge penalties. This is not surprising because temporal aggregation is based on giving an estimate of the size of violation at a customer, but all violations incur the same fixed charge in this model. When we compare truncation and expected value, we see that each are the fastest in 12 out of the 24 dataset classes. In Table 10, we do see, though, that truncation can sometimes perform significantly worse than expected value. For example, with probabilities of 0.1, 100 customers,

Table 8. Comparison of Solution Values between Approximations for Fixed Charge Problem

Prob		0.1			0.9			Range			Mixed		
Dataset		TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC
$n =$	Deadline												
40	Early	0%	-2%	-2%	-2%	-5%	-3%	1%	-4%	-5%	0%	-2%	-2%
	Late	0%	-5%	-5%	0%	0%	0%	-2%	-6%	-4%	0%	-1%	-2%
60	Early	-1%	-5%	-5%	0%	-1%	-1%	0%	-4%	-4%	-2%	-2%	0%
	Late	-1%	-6%	-6%	0%	0%	0%	1%	-8%	-8%	0%	-1%	-1%
100	Early	-4%	-6%	-2%	-1%	-1%	0%	2%	-9%	-11%	1%	-11%	-13%
	Late	-1%	-6%	-5%	0%	1%	1%	2%	2%	0%	0%	-1%	-1%

and late deadlines, expected value outperforms truncation by 335%. If we look carefully at the performance of expected value, we see that, like with per-unit penalties, significant speedups occur with later deadlines, and these later deadlines often yield the significant gains over truncation approximation.

8. Insights and Future Work

Based on our computational experiments, we can make the following conclusions regarding the performance of the different approximations:

(1) With per-unit penalties, all approximations yield roughly identical solutions.
(2) Greater variation in solution values occurs with fixed charge penalties. Temporal aggregation and expected value tend to yield lower cost solutions than truncation.
(3) Run times increase dramatically as the number of customers increases across all experiments and all approximations.
(4) Run times are less for high probabilities, across all experiments, since fewer changes are required from the seed solution.
(5) Expected value performs better when deadlines are high due to lower total penalties in the objective value.
(6) For per-unit penalties, truncation tends to outperform temporal aggregation in terms of run time as the number of customers increases.
(7) Temporal aggregation is not the best choice with fixed charge penalties.
(8) Expected value is a good choice with fixed charge penalties when later deadlines are used.

There are many remaining questions and opportunities for future work. One obvious research area is to explore different ways in which the approximation parameters should be tuned in the solution process. In this paper, we utilized a 1-shift neighborhood in doing local search. These approximation ideas can be

Table 9. Run Times for Approximations on Fixed Charge Problem

Prob		0.1			0.9			Range			Mixed		
Dataset		TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
$n =$	Deadline												
40	Early	21.6	23.2	22.8	17.4	17.0	9.6	17.2	15.0	13.8	20.6	18.2	12.8
	Late	17.6	26.0	20.0	16.4	15.0	4.2	17.6	23.8	6.6	17.6	14.4	10.2
60	Early	122.0	444.4	652.8	92.0	182.2	61.8	104.0	249.4	213.4	118.0	212.8	231.0
	Late	112.0	434.6	98.8	102.0	121.2	42.0	100.2	212.4	83.4	102.2	127.4	30.8
100	Early	1291.0	14366.2	4204.4	1770.0	4335.0	2405.4	1379.0	3457.0	4936.4	1303.0	3293.0	2125.2
	Late	2006.4	5442.0	460.8	1469.4	2934.6	1781.4	1330.0	3491.4	3016.8	1409.6	4502.4	1873.8

Table 10. Comparison of Run Time between Approximations for Fixed Charge Problem

Prob		0.1			0.9			Range			Mixed		
Dataset		TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC	TA-EA	TA-TC	EA-TC
$n =$	Deadline												
40	Early	2%	7%	5%	44%	-2%	-81%	8%	-15%	-25%	30%	-13%	-61%
	Late	23%	32%	12%	72%	-9%	-290%	72%	26%	-167%	29%	-22%	-73%
60	Early	-47%	73%	81%	66%	50%	-49%	14%	58%	51%	-9%	45%	49%
	Late	77%	74%	-13%	65%	16%	-143%	61%	53%	-20%	76%	20%	-232%
100	Early	71%	91%	69%	45%	59%	26%	-43%	60%	72%	35%	60%	39%
	Late	92%	63%	-335%	39%	50%	18%	14%	62%	56%	58%	69%	25%

embedded, though, within many other local search neighborhoods and solution techniques. Future work may explore if certain approximation ideas work better with certain neighborhood structures or approaches. It would also be interesting to determine if there would be benefits from using some of these approximation ideas in conjunction with each other. It is also not clear if the answers to these questions will be different depending on the geographical distribution of the customers and/or the distribution of the customer deadlines. It will also be interesting to see how the ideas here can be extended to incorporate vehicle capacity, as in the stochastic vehicle routing problem with deadlines [54].

Acknowledgment

The authors acknowledge many helpful comments from two referees, which have led to several improvements in the paper. This work was partially supported by the National Science Foundation through grant number 0237726(Campbell).

Appendix

Tables 11, 12, and 13 present the solution values for the various computational tests. For each category of problem instance (determined by n, deadline value, penalty value, and probability class), the entry in the table is an average over the five instances in that category.

Table 11. Solution Values of Approximations on 40- and 60-Customer Instances

Probability			0.1				0.9				Range				Mixed			
Dataset			PTSPD	TC	TA	EA	PTSPD	TC	TA	EA	PTSPD	TC	TA	EA	PTSPD	TC	TA	EA
$n =$	Deadline	Penalty																
40	Low	5	104.2	104.2	104.2	104.2	449.4	449.4	449.4	449.4	264.6	264.9	264.6	264.6	259.1	259.1	259.1	259.1
	High	5	102.7	102.7	103.1	102.7	271.3	271.3	272.1	271.3	220.7	220.7	221.6	220.7	199.8	199.8	203.3	199.8
	Low	50	111.8	112.1	111.8	111.8	1950.2	1950.2	1950.2	1950.2	479.1	473.0	536.5	481.0	631.7	631.7	631.7	631.7
	High	50	131.6	103.0	131.5	103.0	333.6	273.3	333.6	273.3	258.3	221.5	259.2	221.4	277.5	206.5	277.5	206.5
60	Low	5	146.0	146.1	145.8	146.0	671.3	671.3	671.3	671.3	372.4	374.8	372.4	372.4	394.9	394.9	395.3	394.9
	High	5	131.6	131.4	131.5	131.6	333.6	332.7	333.6	333.6	258.3	258.3	259.2	258.3	277.5	277.5	277.5	277.5
	Low	50	234.5	234.5	234.5	233.5	3542.2	3542.2	3542.2	3542.2	1288.4	1306.7	1288.4	1288.4	1399.6	1399.6	1399.7	1399.6
	High	50	134.9	134.9	134.9	134.9	331.9	331.9	332.0	331.9	259.2	259.2	259.2	259.2	276.6	277.0	276.9	276.9

Table 12. Solution Values of Approximations on 100-Customer Instances

Prob.	0.1			0.9			Range			Mixed		
Deadline	TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
Low	175.2	172.4	171.5	800.4	800.4	799.8	453.8	453.6	453.6	469.7	468.4	468.4
High	161.7	160.9	160.8	428.3	428.0	428.0	346.4	346.1	346.2	352.7	428.0	352.4

Table 13. Solution Values of Approximations on 40-, 60-, 100-Customer Instances for Fixed Charge Problem

Probability		0.1			0.9			Range			Mixed		
Dataset		TC	TA	EA	TC	TA	EA	TC	TA	EA	TC	TA	EA
$n =$	Deadline												
40	Low	113.5	111.0	110.8	382.6	365.5	7249.0	292.5	281.5	278.1	265.3	260.6	260.6
	High	111.8	106.7	106.9	317.3	317.2	317.2	260.8	245.6	250.3	216.0	213.2	212.2
60	Low	150.3	150.4	151.4	459.7	524.4	523.2	339.1	368.6	368.2	333.6	359.6	364.6
	High	146.9	138.5	139.9	383.1	381.3	380.9	307.4	282.6	288.4	303.8	293.9	302.5
100	Low	225.4	190.8	197.3	878.2	745.1	758.0	502.3	515.4	506.4	558.9	537.3	532.9
	High	191.1	176.3	183.5	485.4	490.2	490.2	380.3	388.6	380.9	400.6	396.3	396.8

References

[1] United Parcel Service, About UPS, `http://www.corporate-ir.net/ireye/ir_site.zhtml?ticker=UPS&script=2100&layout=7`, accessed on November 30, 2006 (2002).

[2] B. Carey, Expedited grows on the surface, Traffic World (2006) 1.

[3] A. M. Campbell, B. W. Thomas, The probabilistic traveling salesman problem with deadlines, forthcoming in *Transportation Science* (2007).

[4] W. C. Benton, M. D. Rosetti, The vehicle scheduling problem with intermittent customer demands, Computers and Operations Research 19 (1992) 521–531.

[5] H. Zhong, Territory planning and vehicle dispatching with stochastic customers and demand, Ph.D. thesis, University of Southern California (2001).

[6] H. Zhong, R. W. Hall, M. Dessouky, Territory planning and driver learning in vehicle dispatching, Transportation Science.

[7] J. J. Bartholdi, L. K. Platzman, R. L. Collins, W. H. Warden, A minimal technology routing system for meals on wheels, Interfaces 13 (1983) 1–8.

[8] P. Jaillet, Probabilistic traveling salesman problems, Ph.D. thesis, Massachusetts Institute of Technology (1985).

[9] P. Jaillet, A priori solution of the traveling salesman problem in which a random subset of customers are visited, Operations Research 36 (1988) 929–936.

[10] G. Laporte, F. V. Louveaux, H. Mercure, A priori optimization of the probabilistic traveling salesman problem, Operations Research 42 (1994) 543–549.

[11] D. J. Bertsimas, P. Jaillet, A. R. Odoni, A priori optimization, Operations Research 38 (1990) 1019–1033.

[12] D. J. Bertsimas, L. H. Howell, Further results on the probabilistic traveling salesman problem, European Journal of Operational Research 65 (1993) 68–95.

[13] P. Chervi, A computational approach to probabilistic vehicle routing problems, Master's thesis, Massachusetts Institute of Technology (1988).

[14] L. Bianchi, J. Knowles, N. Bowler, Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms, European Journal of Operational Research 162 (2005) 206–219.

[15] L. Bianchi, A. M. Campbell, Extension of the 2-*p*-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem, European Journal of Operational Research 176 (2007) 131–144.

[16] A. Campbell, Aggregation for the probabilistic traveling salesman problem, Computers & Operations Research 33 (2006) 2703–2724.

[17] H. Tang, E. Miller-Hooks, Approximate procedures for the probabilistic traveling salesman problem, Transportation Research Record 1882 (2004) 27–36.

[18] W. B. Powell, P. Jaillet, A. Odoni, Stochastic and dynamic networks and routing, in: M. O. Ball, T. L. Magnanti, C. L. Monma, G. L. Nemhauser (Eds.), Network Routing, Vol. 8 of Handbooks in Operations Research and Management Science, North-Holland, Amsterdam, 1995, pp. 141–295.

[19] D. J. Bertsimas, D. Simchi-Levi, A new generation of vehicle routing research: Robust algorithms, addressing uncertainty, Operations Research 44 (1996) 286–303.

[20] M. Gendreau, G. Laporte, R. Séguin, Stochastic vehicle routing, European Journal of Operational Research 88 (1996) 3–12.

[21] F. Tillman, The multiple terminal delivery problem with probabilistic demands, Transportation Science 3 (1969) 192–204.
[22] D. J. Bertsimas, Probabilistic combinatorial optimizations problems, Ph.D. thesis, Massachusetts Institute of Technology (1988).
[23] D. J. Bertsimas, A vehicle routing problem with stochastic demand, Operations Research 40 (1992) 574–585.
[24] W. R. Stewart, B. L. Golden, Stochastic vehicle routing: A comprehensive approach, European Journal of Operational Research 14 (1983) 371–385.
[25] G. Laporte, F. V. Louveaux, H. Mercure, Models and exact solutions for a class of stochastic location-routing problems, European Journal of Operational Research 39 (1989) 71–78.
[26] C. Bastian, A. H. G. Rinnooy Kan, The stochastic vehicle routing problem revisited, European Journal of Operational Research 56 (1992) 407–412.
[27] M. Dror, G. Laporte, P. Trudeau, Vehicle routing with stochastic demands: Properties and solution frameworks, Transportation Science 23 (1989) 166–176.
[28] M. Dror, Modeling vehicle routing with uncertain demands as stochastic programs: Properties of the corresponding solution, European Journal of Operational Research 64 (1993) 432–441.
[29] M. Dror, P. Trudeau, Stochastic vehicle routing with modified savings algorithm, European Journal of Operational Research 23 (1986) 228–235.
[30] J. Bramel, E. G. Coffman, P. W. Shor, D. Simchi-Levi, Probabilistic analysis of the capacitated vehicle routing problem with unsplit demands, Operations Research 340 (1992) 1095–1106.
[31] D. J. Bertsimas, P. Chervi, M. Peterson, Computational approaches to stochastic vehicle routing problems, Transportation Science 29 (1995) 342–352.
[32] M. W. P. Savelsbergh, M. Goetschalckx, A comparison of the efficiency of fixed versus variable vehicle routes, Journal of Business Logistics 46 (1995) 474–490.
[33] W.-H. Yang, K. Mather, R. H. Ballou, Stochastic vehicle routing problem with restocking, Transportation Science 34 (2000) 99–112.
[34] M. Gendreau, G. Laporte, R. Séguin, An exact algorithm for the vehicle routing problem with stochastic demands and customers, Transportation Science 29 (1995) 143–155.
[35] M. Gendreau, G. Laporte, R. Séguin, A tabu search heuristic for the vehicle routing problem with stochastic demands and customers, Operations Research 44 (1996) 469–477.
[36] S. Y. Teng, H. L. Ong, H. C. Huang, An integer L-shaped algorithm for the time-constrained traveling salesman problem with stochastic travel times and service times, Asia-Pacific Journal of Operational Research 21 (2004) 241–257.
[37] J. C. F. Wong, J. M. Y. Leung, C. H. Cheng, On a vehicle routing problem with time windows and stochastic travel times: Models, algorithms, and heuristics, Tech. Rep. SEEM2003-03, Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong (2003).
[38] X. Wang, A. C. Regan, Alternative assignment models for time constrained local fleet assignment in which service and travel times are stochastic, Tech. Rep. UCI-ITS-WP-00-13, Institute for Transportation Studies and Department of Civil and Environmental Engineering, University of California, Irvine (November 2000).

[39] FedEx, Rules/accessorial tariff via all motor routes naming rules, regulations and claims procedures applying on surface expedited services between points in north america (except mexico), http://customcritical.fedex.com/us/serviceinfo/documents/pdf/tarifffdcc101g.pdf?link=4, accessed on April 11, 2005 (2003).

[40] D. F. Rogers, R. D. Plante, R. T. Wong, J. R. Evans, Aggregation and disaggreation techniques and methodology in optimization, Operations Research 39 (4) (1991) 553–582.

[41] W. Hackbusch, Multi-grid Methods and Applications, No. 4 in Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 1985.

[42] R. M. Lewis, R. G. Nash, Model problems for multigrid optimization of systems governed by differential equations, SIAM Journal on Scientific Computing 26 (6) (2005) 1811–1837.

[43] R. L. Francis, T. J. Lowe, A. Tamir, Demand point aggregation for location models, in: Facility Location: Applications and Theory, Springer-Verlag, Berlin, 2002, Ch. 7.

[44] J. Mercenier, P. Michel, Discrete-time finite horizon approximation of infinite horizon optimization problems with steady-state variance, Econometrica 62 (3) (1994) 635–656.

[45] A. M. Newman, M. Kuchta, Using aggregation to optimize long-term production planning at an underground mine, European Journal of Operational Research 176 (2007) 1205–1218.

[46] A. Campbell, M. Savelsbergh, Decision support for consumer direct grocery initiatives, Transportation Science 39 (3) (2005) 313–327.

[47] Y. Dumas, J. Desrosiers, E. Gelinas, M. M. Solomon, An optimal algorithm for the traveling salesman problem with time windows, Operations Research 43 (1995) 367–371.

[48] J. W. Ohlmann, B. W. Thomas, A compressed annealing approach to the traveling salesman problem with time windows, INFORMS Journal on Computing.

[49] K. Cheh, J. Goldberg, R. Askin, A note on the effect of neighborhood structure in simulated annealing, Computers & Operations Research 18 (1991) 537–547.

[50] W. B. Carlton, J. W. Barnes, Solving the traveling-salesman problem with time windows using tabu search, IIE Transactions 28 (1996) 617–629.

[51] E. Aarts, J. K. Lenstra (Eds.), Local Search and Combinatorial Optimization, Wiley–Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons, Chichester, U.K., 1997.

[52] FedEx, Service info: Money back guarantee, http://www.fedex.com/us/services/express/, accessed on August 9, 2004 (2004).

[53] United Parcel Service, Calculating time and cost faq, http://www.ups.com/content/us/en/resources/service/, accessed on August 9, 2004 (2004).

[54] A. M. Campbell, B. W. Thomas, The stochastic vehicle routing problem with deadlines, working Paper (2007).