Journal of the Operational Research Society (2005) 56, 439–452

www.palgrave-journals.com/jors

Algorithms for a stochastic selective travelling salesperson problem

H Tang[1] and E Miller-Hooks[2]*

[1]*Operations Research and Spatial Applications, FedEx Express Corporation, Memphis, TN, USA; and*
[2]*The University of Maryland, College Park, MD 20742, USA*

In this paper, the selective travelling salesperson problem with stochastic service times, travel times, and travel costs (SSTSP) is addressed. In the SSTSP, service times, travel times and travel costs are known *a priori* only probabilistically. A non-negative value of reward for providing service is associated with each customer and there is a pre-specified limit on the duration of the solution tour. It is assumed that not all potential customers can be visited within this tour duration limit, even under the best circumstances. And, thus, a subset of customers must be selected. The objective of the SSTSP is to design an *a priori* tour that visits each chosen customer once such that the total profit (total reward collected by servicing customers minus travel costs) is maximized and the probability that the total actual tour duration exceeds a given threshold is no larger than a chosen probability value. We formulate the SSTSP as a chance-constrained stochastic program and propose both exact and heuristic approaches for solving it. Computational experiments indicate that the exact algorithm is able to solve small- and moderate-size problems to optimality and the heuristic can provide near-optimal solutions in significantly reduced computing time.
Journal of the Operational Research Society (2005) **56,** 439–452. doi:10.1057/palgrave.jors.2601831
Published online 27 October 2004

Keywords: selective travelling salesperson problem; stochastic travelling salesman; chance constraints; orienteering problem

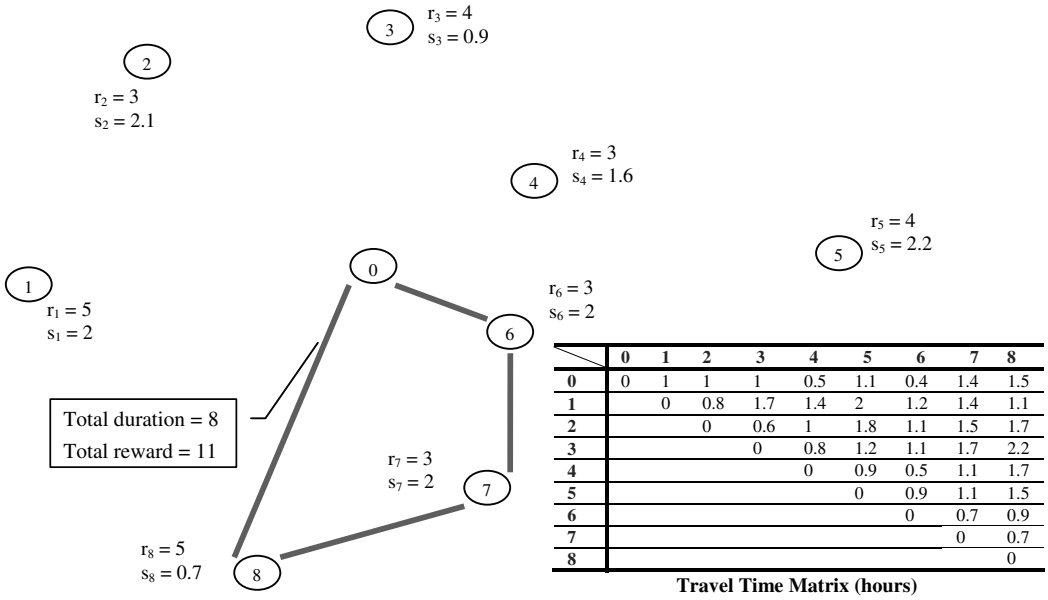
Introduction

This paper considers the problem of determining an optimal tour for servicing a subset of customers subject to a tour duration restriction and stochastic service times, travel times and travel costs. This problem is a variant of the deterministic selective travelling salesperson problem (STSP), where service times, travel times and travel costs are known with certainty. Thus, we refer to this problem as the stochastic selective travelling salesperson problem, or SSTSP. Given a complete graph $G=(V, E)$, where V is the vertex set with associated service time matrix (s_i) and reward matrix (r_i), $E=\{(i, j)|i, j\in V\}$ is the edge set with associated travel cost matrix (c_{ij}), the SSTSP seeks the maximum profit (total reward minus total travel cost) tour that traverses a subset of vertices such that the probability that the actual tour duration exceeds a pre-specified limit L must be no greater than a probability threshold α.

To illustrate this problem, we first consider an example of the deterministic STSP as given in Figure 1. Let vertex 0 represent the depot and vertices 1–8 represent customers that need to be visited. The location of each vertex corresponds with the associated customer location. In this example, the maximum tour duration L (travel time plus service time) is 8 h. Service time s_i and reward r_i for vertex i, $i\in V$, and travel times between vertices are shown in the figure. Travel times are assumed to be symmetric. We assume that the travel cost on each link is equal to its link travel time. The goal of this deterministic STSP is then to find the maximum profit tour that traverses a subset of customers such that the total tour duration does not exceed 8 h. (Note that unlike in this work, most published STSP studies have an objective of maximizing only total tour reward and, thus, the resulting optimal solution may not have a minimum total travel time. The problem addressed in this paper simultaneously considers reward and total travel time, ie a generalization of the problem studied in these other works.) In Figure 1, tour 0-6-7-8-0 is the best solution for the STSP with total profit of 7.7.

Now, assume that the service times shown in Figure 1 are the expected service time values and for each customer i, there is a 0.5 probability that the service time is $s_i/2$ and a 0.5 probability that it is $3s_i/2$. By assuming the existence of stochastic service times, the total duration of solution tour 0-6-7-8-0 is no longer a deterministic value and is not guaranteed to meet the 8-h duration limit for all service time realizations. Further, the probability that the actual tour duration of tour 0-6-7-8-0 exceeds 8 h is 0.5 (ie $P(s_6=1, s_7=3, s_8=1.05)+P(s_6=3, s_7=1, s_8=1.05)+P(s_6=3, s_7=3)=0.5$). Thus, if the probability threshold for this stochastic problem is less than 0.5, tour 0-6-7-8-0 will not be

*Correspondence: E Miller-Hooks, Department of Civil and Environmental Engineering, The University of Maryland, 1173 Glenn L. Martin Hall, College Park, MD 20742, USA.
E-mail: elisemh@umd.edu



	0	1	2	3	4	5	6	7	8
0	0	1	1	1	0.5	1.1	0.4	1.4	1.5
1		0	0.8	1.7	1.4	2	1.2	1.4	1.1
2			0	0.6	1	1.8	1.1	1.5	1.7
3				0	0.8	1.2	1.1	1.7	2.2
4					0	0.9	0.5	1.1	1.7
5						0	0.9	1.1	1.5
6							0	0.7	0.9
7								0	0.7
8									0

Travel Time Matrix (hours)

Figure 1 An illustrative example for the STSP and the SSTSP.

feasible for the SSTSP defined on this network. While 0-6-7-8-0 is feasible at a probability threshold of 0.5, it may not be optimal. In fact, the optimal tour is 0-1-8-6-0 with total profit of 9.6 units. Existing solution approaches for the deterministic version of the STSP would require that the random service times be replaced by their expected values to address the SSTSP. However, such an approach may result in a suboptimal or infeasible solution to the SSTSP as is illustrated in our example. Specifically, tour 0-1-8-6-0, the optimal solution to the SSTSP, has an expected duration of 8.1 h, which is greater than the 8-h limit. Therefore, this solution will appear to be infeasible and an alternative solution that is suboptimal will be identified.

There are many applications for which the SSTSP would be an appropriate model. For instance, consider many public and private agencies whose service technicians must visit geographically dispersed customers. In such applications, there is typically a limitation on the number of hours each technician can be scheduled to work in any given day. Given a limited number of technicians and a high demand for service, it may not be possible to service all customers requesting service on a particular day and only a subset of customers will be serviced on that day. In choosing this subset of customers, one may consider priorities for specific customer visits as well as the costs (ie travel and service times) incurred by including them. Further, since neither service nor travel times can be estimated *a priori* with certainty, the tour duration constraints are complicated by the fact that it may not be possible to determine *a priori* exactly how many customers can be visited on the given day. Rather, customer visits can be assigned to a technician in such a way that, while the total profit (or expected profit) is maximized, the probability that the total duration of the scheduled work exceeds the tour duration limit (here, the workday) is no greater than a given value. A related example of such a service application is given in Golden *et al.*[1] They consider the application of home fuel delivery, where both service and travel times are deterministically known. One can better model that problem by considering the inherent stochastic nature of the service and travel times.

A number of works have addressed the deterministic STSP or the related Orienteering Problem (abbreviated as OP where, unlike the STSP, the starting and terminus points may not be the same), several of which propose exact algorithms based on branch-and-bound[2,3] and branch-and-cut[4,5] procedures. The most successful implementation was due to the branch-and-cut procedure,[4] which solved STSP instances with up to 500 vertices.

As shown by Golden *et al*,[1] the OP (and thus, the STSP) is NP-hard. Therefore, most research on these problems has focused on providing heuristic approaches. Tsiligirides[6] proposed deterministic and stochastic heuristics for the problem. The deterministic heuristic uses a variant of the heuristic procedure proposed by Wren and Holliday[7] for vehicle routing and the stochastic method relies on Monte Carlo techniques (ie where a large number of tours are randomly generated from which the best will be selected). A centre-of-gravity heuristic was introduced by Golden *et al*,[1] where the solution tour is constructed by the cheapest insertion procedure according to a combined measure for vertex selection. Incorporating the concept of a learning measure, Golden *et al*[8] improved the centre-of-gravity

heuristic by rewarding vertices associated with 'above-average' tours while penalizing those associated with 'below-average' tours. Ramesh and Brown[9] proposed a four-phase heuristic. After choosing the best solution from iterations over a set of three phases (vertex insertion, arc exchange and vertex deletion), a fourth phase is entered, where one attempts to insert unvisited vertices into the tour.

Chao[10] and Chao *et al*[11] proposed a general-purpose five-step heuristic for a series of multi-level routing problems, including the STSP. This general-purpose algorithm takes advantage of a probabilistic criterion for candidate solution selection and can be considered as a deterministic variant of simulated annealing, as pointed out by Golden *et al.*[12] Fischetti *et al*[4] used a heuristic approach to obtain the initial lower bound for an exact algorithm. The heuristic method is similar to the four-phase heuristic by Ramesh and Brown;[9] however, information derived through solutions to a linear programming (LP) relaxation of the problem is used in guiding the search.

Gendreau *et al*[5] developed two approximate algorithms for the STSP. The first, H1, gradually constructs a solution tour by inserting a single vertex or a pair of vertices into the current tour. The second, H2, constructs a tour using all vertices, then gradually removes some vertices from the tour. Both heuristics make use of the GENIUS composite heuristic for the travelling salesperson problem (TSP).[13] A tabu search procedure that incorporates the proximity measure on clusters of vertices in candidate moves, rather than on single vertices, was proposed by Gendreau *et al.*[14]

While there have been numerous related works on the deterministic STSP or related problems, to the best of our knowledge, no published works on the STSP (or its variants) have considered the stochasticity in service or travel times or travel costs. The most relevant work in the literature addresses the vehicle routing problem (VRP) with stochastic travel times. Laporte *et al*[15] proposed three stochastic programming formulations: one is a chance-constrained model and the other two are recourse models. They further provided a framework based on branch-and-cut that was later described as the integer L-shaped method[16] to solve the recourse models. Results from computational experiments on problem instances with up to 20 vertices for one of the recourse formulations are given. Unfortunately, solution techniques developed for the VRP with stochastic travel times cannot be directly applied to solve the SSTSP. This is because, for both deterministic and stochastic VRPs, it is assumed that a feasible solution will include all customers and, thus, solution techniques designed for the VRP do not address the issue of customer selection. Furthermore, the objective of the stochastic VRP is to find the set of tours that minimizes total travel cost; whereas, the objective of the SSTSP is to find the tour that maximizes total profit.

In this paper, we provide a chance-constrained stochastic programming formulation for the SSTSP and propose algorithmic approaches to solve it. In this problem, customer service times are known *a priori* only probabilistically. Without loss of generality, travel times are assumed to be known with certainty. The main contribution of this paper is the development of exact and heuristic procedures for addressing the SSTSP. The exact algorithm is based on the branch-and-cut framework that employs implicit enumeration and valid inequalities (valid cuts) that are identified during the search process to obtain an optimal solution. The heuristic, on the other hand, constructs and adjusts solution tours by a series of greedy procedures.

Problem definition and formulation

For a given undirected complete graph $G=(V,E)$, let $V=\{0,1,\ldots,n\}$ be the vertex set and $E=\{(i,j)|i,j\in V\}$ be the edge set with associated travel cost matrix $C=(c_{ij})$ and travel time matrix $T=(t_{ij})$. Every vertex of set $V\backslash\{0\}$, where vertex 0 represents the depot, corresponds to a customer. Let r_i and s_i be the reward and expected service time associated with servicing customer (vertex) i, $i\in V$, respectively, and L be the maximum tour duration. The tour duration is computed from both travel times and service times. For each customer i, we assume that the service time can take one of m states, that is $s_{i1}, s_{i2}, \ldots, s_{im}$ with associated probabilities $p_{i1}, p_{i2}, \ldots, p_{im}$, which satisfy $\sum_{j=1}^{m} p_{ij}=1$ and $\sum_{j=1}^{m} s_{ij}p_{ij}=s_i$ for all i $\in$ V$\backslash\{0\}$.

Without loss of generality, let $s_{i1}\leqslant s_{i2}\leqslant\cdots\leqslant s_{im}$. The SSTSP is to determine an *a priori* tour τ^* with the maximum profit (ie total reward less travel costs) such that the probability that the total tour duration $D(\tau^*)$ exceeds L is no greater than a given threshold value α, or $P[D(\tau^*)>L]\leqslant\alpha$. The *a priori* tour must originate and terminate at depot 0 and pass through each chosen customer once. We require that the sum of the probability that the tour duration exceeds L over all service time realizations be less than or equal to α. Given this, it is possible that the optimal tour could have a duration that exceeds L for some realization of the service times. One could, instead, require that the tour duration not exceed L for any realization with greater than α probability. The summation approach is stronger and ensures that the latter requirement is also true. Thus, we solve for the *a priori* solution that maximizes total profit, but simultaneously ensures that for any realization there is small likelihood (given appropriate definition of α) of exceeding tour duration restriction L.

The SSTSP is formulated as a chance-constrained stochastic integer program using the following notation:

v_i	a vertex, $v_i\in V$
Ω	set of vertices, $\Omega\subset V\backslash\{0\}$
x_{ij}	1, if edge (i,j) is on the *a priori* tour; 0, otherwise
y_i	1, if vertex i is on the *a priori* tour; 0, otherwise
c_{ij}	travel cost on edge (i,j)
t_{ij}	travel time incurred along edge (i,j).
r_i	reward collected by visiting vertex i

S_i service time vector $\{s_{i1}, s_{i2}, \ldots\}$ associated with vertex i ($E[S_i] = s_i$)

P_i probability vector $\{p_{i1}, p_{i2}, \ldots\}$ of service time for vertex i

τ a tour that starts and ends at depot 0

α threshold on the probability of having the total tour duration in excess of L, $0 \leqslant \alpha < 1$

$D(\tau)$ Duration of tour τ. Note that $D(\tau)$ is a random variable

The formulation (P) of the *a priori* SSTSP ($n > 2$) is as follows.

$$\text{(P)} \quad \text{Maximize} \sum_{i \in V \setminus \{0\}} r_i y_i - \sum_i \sum_{j>i} c_{ij} x_{ij} \tag{1}$$

Subject to

$$\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} - 2y_k = 0, \ (0 \leqslant k \leqslant n) \tag{2}$$

$$\sum_{i \in \Omega} \sum_{\substack{j \in V \setminus \Omega \\ j>i}} x_{ij} + \sum_{j \in V \setminus \Omega} \sum_{\substack{i \in \Omega \\ j<i}} x_{ji} \geqslant 2y_k \ (\Omega \subset V \setminus \{0\}, k \in \Omega) \tag{3}$$

$$P[D(\tau) > L] \leqslant \alpha, \ \forall \tau = (v_0 = 0, v_1, v_2, \ldots, v_m, v_{m+1} = 0) \tag{4}$$

$$y_0 = 1 \tag{5}$$

$$x_{ij} = \{0, 1\} (0 \leqslant i < j \leqslant n) \tag{6}$$

$$y_j = \{0, 1\} (0 \leqslant j \leqslant n) \tag{7}$$

In this formulation, the objective (1) is to maximize the total profit (ie the total reward by visiting customers minus the total travel cost). Constraints (2) ensure the connectivity of the solution tour, while subtours are prohibited by constraints (3). Constraints (4) exclude the tours where the probability that the actual total duration is greater than L exceeds threshold α. The requirement that depot 0 is always in the *a priori* tour is respected by (5). Constraints (6) and (7) set integral requirements. This formulation assumes that at least one of the optimal solutions includes at least two customer vertices (other than depot 0), because decision variable x_{ij} is only defined for $i < j$ ($0 \leqslant i < j \leqslant n$). For notation simplicity, in the remainder of this paper, it is assumed that $c_{ij} = t_{ij}$, that is travel cost on link (i, j) is equal to link travel time t_{ij}.

For some applications, the cost associated with travel is insignificant and it may not be necessary to deduct it from the total reward. In such circumstances, objective (1) can be substituted with

$$\sum_{i \in V \setminus \{0\}} r_i y_i \tag{1'}$$

We will see in the next section that this model (with objective (1′)) can also be solved through the same techniques developed herein for the model with objective (1).

One could view formulation (P) as identical to that of the deterministic STSP, where deterministic tour duration constraints

$$D(\tau) \leqslant L \text{ or } \sum_i \sum_{j>i} \left(t_{ij} + \frac{s_i}{2} + \frac{s_j}{2} \right) x_{ij} \leqslant L \tag{8}$$

(ie t_{ij} and s_i are deterministic), are replaced with chance constraints (4). While the proposed formulation (P) may differ only in one set of constraints from its deterministic counterpart for which available solution techniques exist, significant added difficulty is incurred as a consequence of this replacement. The added difficulty arises from the following. In a branch-and-bound (or branch-and-cut) solution framework, chance constraints (4), must be relaxed, since they cannot be expressed in closed form. This relaxation results in a greatly enlarged gap between the initial relaxation problem at the root of the branch-and-bound tree and an initial lower bound (eg a heuristic solution). Moreover, the number of chance constraints (4) increases exponentially with problem size. In addition, service or travel times can at best be estimated *a priori* only with uncertainty. This uncertainty results in much greater complexity in evaluating a given solution for feasibility and in employing successive bounding techniques during a branch-and-bound search. Even with the additional complexity, one cannot employ existing techniques to solve the stochastic problem where chance constraints (4) are given in place of the deterministic tour duration constraints (8). This is because existing techniques for solving deterministic problems do not consider the likelihood of more than one tour duration realization.

Although there is no obvious closed-form expression for the computation of the probability that the tour duration exceeds α, this probability can be directly computed as long as the discrete stochastic service times have finite support of small cardinality. This is also true for continuous stochastic service times if the distributions have additive probability distributions.[15] In this study, we employ a recursive procedure to compute the chance probabilities that examines edge by edge the additive probability that the actual tour duration exceeds L. For applications where tour cardinality is large and service time distributions are complex (eg do not have finite support), simulation techniques (eg Monte Carlo simulation) can be used to approximate this probability.

Throughout this paper, we assume that service times are known *a priori* only probabilistically, while the travel times are known deterministically. However, solution techniques developed in this paper can be readily extended to the case where both service and travel times are stochastic.

Exact algorithm

The model (P) formulated by (1)–(7) can be solved exactly by a branch-and-cut algorithm that is described in this section.

Algorithm overview

In this branch-and-cut algorithm, a LP relaxation and augmentation (P′) of the original problem formulation (P) is solved at the root of a search tree. If the solution of this relaxation and augmentation problem is integral and satisfies chance constraints (4), then the algorithm terminates with the optimal solution (note that the solution for the LP problem (P′) at the root of the search tree is greater than or equal to the optimal solution value). If this solution is not integral, valid inequalities that are violated by the LP solution to (P′) will be identified and added to (P′) and the resulting LP problem will be re-solved. If no violated inequalities can be identified and the LP solution is fractional, the search tree is extended by branching on a selected fractional variable. The outline of this algorithm follows. Text in italic will be discussed in detail in the following subsections.

Algorithm SSTSP

Step 1: Let the iteration index $q=0$ and create a list of sub-problems containing only the *relaxation and augmentation problem* (P'). Set the current best solution value $\underline{Z}$ to that found by applying a heuristic solution approach (one such approach is given in the next section) to solve (P).

Step 2: If the list of sub-problems is empty, stop; otherwise, choose the last sub-problem in the list and remove it. Call it the current sub-problem.

Step 3: Let $q=q+1$. Solve the current LP sub-problem and denote the optimal solution to this sub-problem as (X^q, Y^q).

Step 4: If $RY^q-TX^q \leqslant \underline{Z}$, fathom the current problem and return to Step 2; otherwise, check violated *valid inequalities for the deterministic STSP*. If one or more can be identified, append these violated inequalities to the current sub-problem and return to Step 2. Otherwise, continue to Step 5.

Step 5: Check if solutions X^q and Y^q are integral. If at least one variable is fractional, create two new sub-problems by *branching on a selected variable*. Add *conditional inequalities* and *improved bounding constraints* to the sub-problems. Add these sub-problems to the sub-problem list and return to Step 2. If no variables are fractional, continue to Step 6.

Step 6: Denote the current solution as τ (suppose that $\tau=(0, v_1, v_2, \ldots, v_m, 0)$). If τ does not violate chance constraints (4), update $\underline{Z}$ (ie $\underline{Z}=RY^q-TX^q$), fathom the current sub-problem and return to Step 2. If τ violates chance constraints (4), introduce an *illegal tour elimination cut* to the sub-problem and *examine neighbourhood solutions of* τ. If a feasible solution can be found (from the neighbourhood solutions of τ) with an objective value greater than the current lower bound $\underline{Z}$, update $\underline{Z}$ with the objective value of this new solution. (Otherwise, $\underline{Z}$ remains unchanged.) Return to Step 3.

Algorithm details

Details of the exact algorithm outlined previously are provided in this subsection. Properties and valid inequalities of the SSTSP that have been employed in the algorithm are also presented.

Relaxation and augmentation problem (P′). The LP relaxation and augmentation (P′) is formulated as follows:

$$(\mathrm{P}') \quad \text{Maximize} \quad \sum_{i\in V\setminus\{0\}} r_i y_i - \sum_i \sum_{j>i} t_{ij} x_{ij} \qquad (1^*)$$

Subject to

constraints (2) and (5)

$$\sum_i \sum_{j>i} t_{ij} x_{ij} + \sum_i s_{i1} y_i \leqslant L \qquad (9)$$

$$0 \leqslant x_{ij} \leqslant 1 \ (0 \leqslant i < j \leqslant n) \qquad (10)$$

$$0 \leqslant y_i \leqslant 1 \ (0 \leqslant i \leqslant n) \qquad (11)$$

We refer to formulation (P′) as both a relaxation and augmentation with respect to formulation (P). This is because the integrality, sub-tour elimination and chance constraints of (P) are relaxed in (P′) and valid inequality (bounding constraint) (9) is added. This addition of inequality (9) is viewed as an augmentation of (P). Inequality (9) is useful in (P′) because it can exclude from consideration many solutions which would violate chance constraints (4) that have been relaxed in (P′). The correctness of inequality (9) is demonstrated by the following proposition.

Proposition 1 *Inequality*

$$\sum_i \sum_{i<j} t_{ij} x_{ij} + \sum_i s_{i1} y_i \leqslant L \qquad (9')$$

is valid for the SSTSP.

Proof Recall that s_{i1} is the smallest possible realization of the service time at vertex i. If the smallest duration realization of any tour (ie the left-hand side of expression (9)) is greater than L, then the chance constraint is violated

for all values of probability threshold $\alpha<1$. Thus, inequality (9′) is valid. □

Note that if the objective function is given by expression (1′), one can replace that objective function by

$$max\ M\left(\sum_{i\in V\backslash\{0\}} r_i y_i\right) - \sum_i \sum_{j>i} t_{ij} x_{ij} \qquad (1'')$$

in (P′), where $M>0$ and is sufficiently large. This adjusted objective function will enable the use of illegal tour elimination cuts that require that solution tours have minimum travel time, as will be described later in this subsection. The optimal solution of this adjusted objective given in (1″) is also optimal for (1′). This is because by selecting a sufficiently large M, the optimal solution to objective (1″) will also be optimal for objective (1′). By including the second component in (1″), the optimal solution with respect to (1″) will also have the minimum total tour travel time of all tours with the same total reward.

Valid inequalities for the deterministic STSP. Some valid inequalities derived for the deterministic STSP (eg Fischetti *et al*[4] and Gendreau *et al*[5]) are also feasible for the SSTSP. For example, violated sub-tour elimination constraints (3) and the violated following covering and 2-matching inequalities can be identified in Step 4 of the branch-and-cut algorithm.

Proposition 2 *The covering constraints*

$$x_{ij}\leqslant y_i \text{ and } x_{ij}\leqslant y_j\ (i,j\in V) \qquad (12)$$

are valid inequalities for the SSTSP.

Covering constraints (12) have been applied to the deterministic STSP[4,5] and other related combinatorial problems, for example, the generalized travelling salesperson problem[17] and the covering tour problem.[18]

Proposition 3 *The following 2-matching inequalities are valid for the SSTSP:*

$$\sum_{i,j\in\Omega} x_{ij} + \sum_{(i,j)\in H} x_{ij} \leqslant \sum_{i\in\Omega} y_i + \frac{|H|-1}{2} \qquad (13)$$

where $\Omega\subset V$ and $H\subset E(|H|\geqslant 3)$ such that H is a collection of edges of odd cardinality with mutually distinct endpoints in Ω, and $|(i,j)\cap\Omega|=1$, $\forall(i,j)\in H$.

Inequality (13) is a strengthened version of the 2-matching inequality for the TSP and was proposed by Gendreau *et al*[5] and Fischetti *et al*[4] for the STSP.

Identification of violated covering constraints (12) requires only the examination of all edges (i,j) for which $x_{ij}>0$. If any of the covering constraints (12) are violated, these violated constraints are appended to the current LP sub-problem. Violated 2-matching inequalities may be identified through heuristic procedures based on those used by Padberg and Hong.[19]

In addition, an efficient heuristic is designed to identify violated sub-tour elimination constraints (3). The heuristic begins by constructing a sub-graph $G'=(V',E')$, consisting of all edges with $x_{ij}>0$. Next, vertex k is selected such that $k=\text{argmax}_{\forall i\in V'\backslash\{0\}}\{y_i\}$. Starting from vertex k, a maximum spanning tree T is constructed (eg using Prim's algorithm, where the arc weights are set to the arc flow values, x_{ij}) by adding edges, one by one, that connect T with G′\T. During the construction of tree T, once a new edge is added to T, we examine cut $x(T,G'\backslash T)$ between T and G′\T, that is,

$$x(T,G'\backslash T)=\sum_{i\in T}\sum_{\substack{i\in G'\backslash T\\ j>i}} x_{ij} + \sum_{j\in G'\backslash T}\sum_{\substack{i\in T\\ j<i}} x_{ji}$$

If $x(T,G'\backslash T)<2y_k$, a violated sub-tour constraint is identified. This constraint will be appended to the current LP sub-problem and edge flow x_{0k} will be increased to $2y_k$ (or larger) to exclude this vertex from further consideration. This procedure terminates when all vertices $k\in V'\backslash\{0\}$ with $y_k>0$ have been examined (note that vertices in V′\{0} are examined in descending order).

Branching on a selected variable. For the branching in Step 5 in this work, priority is given to the y_i variables. The first fractional y_i closest to 0.5 is chosen to branch. If all y_is are integral, then a fractional x_{ij} is selected such that x_{ij} is closest to 0.5 (or most fractional). Ties are broken arbitrarily. This branching approach is used because it is observed that in many cases when the most fractional variables are fixed, the less fractional ones may become integer-valued without further branching. Resulting sub-problems will be appended to the sub-problem list.

Conditional inequalities. Conditional inequalities are valid only when specific conditions are satisfied. Computational experience in an STSP study shows that such inequalities can be effective in closing the integrality gap.[4] We consider the following type of conditional inequalities.

Proposition 4 *The following conditional inequality*

$$\sum_{i,j\in\Omega} x_{ij} \leqslant \sum_{i\in\Omega} y_i - 1 \qquad (14)$$

is valid for the SSTSP if set $\Omega\subset V\backslash\{0\}$ is chosen such that at least one vertex in Ω is included in the solution tour.

Proof For any feasible solution tour τ we have $\sum_{i,j\in\tau} x_{ij}=\sum_{i\in\tau} y_i$. Since depot 0 is not contained in set Ω,

there are at least two edges each with an endpoint in Ω that connect set Ω to vertices in $V\backslash\Omega$. Thus, in any feasible solution, edges whose end points are induced by set Ω cannot constitute a tour because a tour must include the depot. It follows that inequality (14) is valid. □

This is similar to the conditional cut proposed by Fischetti *et al*,[4] but we suggest an alternative implementation. Fischetti *et al* look for violations of this conditional inequality in an edge set, where depot 0 is always included and no feasible solution with value strictly greater than the current lower bound (found in the algorithm) is contained in that set. On the contrary, we seek to identify violations of this inequality for set Ω, where depot 0 is not included and at least one vertex in set Ω is included in the solution tour. This modification makes the identification of the conditional inequalities easier, as long as we know which vertex (vertices) will be included in the solution tour. This is due to the fact that there is no need to enumerate all possible feasible solutions in Ω, because (14) is always valid for that sub-problem. Since some x or y variables are fixed to 1 during the branching process, whether or not the associated vertices are included in the tour under consideration is readily known and valid conditional constraints (14) can be easily identified. Note that inequality (14) may be further strengthened if more information on fixed variables is known during the branch-and-bound process. For example, if there are more than two edges each with an endpoint in Ω that connect set Ω to vertices in $V\backslash\Omega$, we have

$$\sum_{i,j\in\Omega} x_{ij} \leqslant \sum_{i\in\Omega} y_i - 2.$$

Rather than appending the valid conditional inequalities in Step 4, where other violated inequalities for the deterministic STSP are also identified, they are appended during the branching process. This takes advantage of fixed x and y variables. For example, if y_p is fixed to 1, vertex p belongs to the solution tour. Similarly, if x_{pq} is fixed to 1, then vertices p and q belong to the solution tour. The identification of possible violated conditional inequalities (ie inequality (14) is violated) is conducted as follows. First, solve the current LP sub-problem with the additional fixed variable (x or y) and let Ω include only vertex p (if $y_p = 1$) or vertices p and q (if $x_{pq} = 1$). Then, for a given value of ε ($\varepsilon = 0.1$, 0.3, 0.5 and 0.7), let $\Omega = \Omega \cup \{i | y_i \geqslant \varepsilon\}$ and check if set Ω violates conditional inequality (14). If one or more violated inequalities can be identified, append these conditional inequalities to the current sub-problem; otherwise, repeat for additional values of ε. This identification procedure may be further simplified for Euclidean problems, because it may not be necessary to resolve the current LP sub-problem with the additional fixed variable due to symmetry.[20]

Improved bounding constraints. During the branch-and-bound process, one can use fixed decision variables to improve bounding functionals (see, eg Laporte *et al*[21] for the probabilistic TSP). Here, the left-hand-side of the bounding constraints (9) can be improved during the branch-and-bound process, taking advantage of fixed decision variables. At each node of the branch-and-bound tree, we have a set of vertices fixed to 1 or 0, denoted by y_{f1} and Y_{f0}, respectively. One example for defining sets Y_{f1} and Y_{f0} is as follows:

$$Y_{f1} = \{i | y_i = 1, \forall i \in V\} \cup \{i, j | x_{ij} = 1, \forall (i, j) \in E\}$$

$$\text{and } Y_{f0} = \{i | y_i = 0, \forall i \in V\}$$

Note that the definitions for sets Y_{f1} and Y_{f0} may be further improved. Suppose that all possible service time realizations of any tour constructed from all and only those vertices in Y_{f1} are sorted in non-decreasing order with respect to the total service time. For instance, the smallest total service time realization for a tour constructed from the vertices in Y_{f1} comes from the combination containing all smallest possible service times, that is, with total actual service time $\sum_{i\in Y_{f1}} s_{i1}$. The probability of this realization is denoted as P_1. Similarly, the probability of the ith smallest total service time realization is denoted as P_i. Suppose that there are B such realizations in total. Let D be the largest integer number such that $1-\sum_{i=1}^{D-1} P_i > \alpha$ ($D \leqslant B$ and $P_0 = 0$) and S_D be the total service time of vertices in Y_{f1} in the Dth smallest service time realization. We have

$$\sum_{\substack{(i,j)\in E \\ i,j\notin Y_{f0}}} t_{ij} x_{ij} + \sum_{i\in V\backslash Y_{f1}} s_{i1} y_i + S_D \leqslant L \qquad (15)$$

When (15) is violated, regardless of service time realizations of those vertices that have not yet been fixed, the probability that the duration of the actual solution tour of this sub-problem exceeds L is at least $1-\sum_{i=1}^{D-1} P_i$. Since $1-\sum_{i=1}^{D-1} P_i > \alpha$, this solution tour is infeasible. Note that for all $1 \leqslant A < D$ (A is integer), the following inequality is also valid:

$$\sum_{\substack{(i,j)\in E \\ i,j\notin Y_{f0}}} t_{ij} x_{ij} + \sum_{i\in V\backslash Y_{f1}} S_{i1} y_i + S_A \leqslant L \qquad (15')$$

However, it will always be more effective to use (15) than (15′).

Thus, bounding constraints (9) can be improved during the branching process by implementing inequality (15). However, implementation of (15) is time consuming, because it involves enumerating and sorting possible service time realizations of fixed vertices, especially when the number of fixed vertices is large. We choose to implement a very simple procedure to identify possible improvements based on the largest service time realizations of fixed vertices. For example, if the probability of this largest realization is

greater than the probability threshold α, S_D can be set as large as the sum of largest possible service times on these fixed vertices, that is S_D is set to S_B. This simple procedure is effective when the number of fixed vertices and probability threshold α are small (note that the problems with small α are considered 'hard' problems, because many identified solutions may be infeasible).

Illegal tour elimination cuts. The illegal solutions that violate chance constraints (4) can be eliminated by the illegal tour elimination cuts described in Proposition 5 and Corollary 1. Note that the cuts discussed in Proposition 5 assume the existence of deterministic travel times.

Proposition 5 *If tour* $\tau = (v_0 = 0, v_1, v_2, \ldots, v_m, v_{m+1} = 0)$ *violates chance constraints* (4), *the illegal tour elimination cut*

$$\sum_{i \in \tau \setminus \{0\}} y_i \leqslant m - 1 \tag{16}$$

is a valid inequality for the SSTSP if travel times are deterministic.

Proof The validity of (16) is based on the fact that if τ is an illegal tour, a feasible tour must not contain all vertices of τ (ie, at least one of $v_1, v_2, \ldots,$ or v_m must be excluded in the feasible solution). This is because tour τ is the minimum expected duration tour if $v_1, v_2, \ldots,$ and v_m are selected, as the exact algorithm also minimizes the total travel time of a solution tour. If the triangular inequality holds and travel times are deterministic (as assumed here), the actual duration of any tour containing all $v_1, v_2, \ldots,$ and v_m must be at least as large as the actual duration of τ given service time realizations of these vertices. Therefore, no tour that includes all vertices $v_1, v_2, \ldots,$ and v_m can be feasible since τ is not feasible and any feasible solution to the SSTSP must satisfy (16). □

Additional illegal tour elimination cuts can be derived as follows.

Corollary 1 *If tour* $\tau = (v_0 = 0, v_1, v_2, \ldots, v_m, v_{m+1} = 0)$ *violates chance constraints* (4), *this tour can be eliminated by imposing inequality*

$$\sum_{i=0}^{m} x_{v_i v_{i+1}} \leqslant m \tag{17}$$

Proof If tour $\tau' \neq \tau$, then τ' contains at least one edge that does not belong to τ. Since for $x \in \tau$, $\sum_{i=0}^{m} x_{v_i v_{i+1}} = m + 1$, imposing inequality (17) will prevent future consideration of tour τ. Note that any tour τ' that differs from τ by at least one edge will not be eliminated by inequality (17). The validity of this inequality follows. □

Remark Inequality (17) does not assume deterministic travel times, that is it can be used when both travel and service times are stochastic. This inequality is similar to one developed for the VRP with stochastic travel times in Laporte *et al.*[15] An immediate improvement to inequality (17) is

$$\sum_{i=0}^{m} x_{v_i v_{i+1}} \leqslant m - 1 \tag{17'}$$

if the triangular inequality holds. This is due to the fact that under the triangular inequality, any feasible tour τ' must differ from τ by at least two edges. Inequality (16) employed in the exact algorithm is much stronger than inequalities (17) and (17′), because (16) eliminates not only tour τ, but also tours obtained by permuting the components of τ and tours that contain all vertices of τ. Such tours are clearly infeasible, because τ is the minimum travel time tour for the given set of vertices in τ.

Examine neighbourhood solutions. In Step 6, we try to improve the current lower bound if the current integer solution is not feasible (ie one or more chance constraints are violated). There are several ways to implement this idea. We employ a simple greedy procedure: when the current solution violates the chance constraints, one vertex will be removed from this tour. In order to choose a vertex in the current solution for such removal, we define measure $c(j, \tau)$ as the cost (or negative reward) of including vertex j in tour τ. Recall that in the problem formulation section we assume that the travel cost on each link is equal to the link travel time (similarly, the cost incurred by servicing a vertex is equal to its service time) for simplicity of notation. Thus, a straightforward definition for measure $c(j, \tau)$ can be chosen as $c(j, \tau) = t_{pj} + t_{jq} - t_{pq} + s_j - r_j$, where p and q are predecessor and successor vertices, respectively, of vertex j in tour τ. The simple greedy procedure begins by selecting three consecutive vertices p, j and $q \in \tau$ such that $c(j, \tau)$ is maximum and removing vertex j from τ. If the new tour is feasible and better than the current best solution (lower bound), the lower bound will be updated. Empirical experiments show that even with this simple procedure, new improved solutions can be found during the search process in many cases, especially when probability threshold α is small (ie most solutions generated during the algorithm may violate the chance constraints).

Heuristic approach

In this section, we present a heuristic called the Construct-and-Adjust, or the CA heuristic. While useful in its own right, this heuristic can also be used as a starting point (provides an initial lower bound) for the exact solution approach based on branch-and-cut presented in the last section. The CA heuristic first constructs a solution tour τ

with expected total duration that does not exceed a value that may be greater than L. The probability that the duration of this tour exceeds L, however, may be greater than α. Thus, the chance constraints (4) may be violated. First, an attempt is made to obtain a tour of lower expected duration through the use of a TSP improvement procedure. If the improved tour τ satisfies the chance constraints (4) (when considering actual service time distributions rather than their expected values), additional vertices will be considered for insertion in τ. These additional vertices are included in τ as long as the chance constraints are met. If the improved tour does not satisfy the chance constraints, some vertices of τ will be removed until these constraints are met and τ becomes feasible. Finally, a vertex exchange procedure is employed to improve the total profit of tour τ.

A list (W) of vertices that are used for constructing solution tours is created and updated in this heuristic. List W is initialized according to expression

$$W = \{j | t_{0j} + t_{j0} \leqslant L \quad \text{and} \quad \sum_{k \in \{i | s_{ji} > L - t_{0j} - t_{j0}, \forall 1 \leqslant i \leqslant m\}} p_{jk} \leqslant \alpha, \ \forall j \in V \backslash \{0\}\} \tag{18}$$

Those vertices whose inclusion in a solution tour results in larger tour duration than the pre-specified limit L (ie $t_{0j} + t_{j0} > L$) or higher duration realization probability than the probability threshold α (ie $\sum_{k \in \{i | s_{ji} > L - t_{0j} - t_{j0}, \forall 1 \leqslant i \leqslant m\}} p_{jk} > \alpha$) are excluded from consideration. Inclusion of any of these vertices in a solution will result in an infeasible solution. Note that this expression can be extended to exclude two- or three-vertex combinations that result in infeasible solution tours.

Empirical results indicate that the quality of the resulting tour constructed by the heuristic is often sensitive to the initial tour that is constructed. Thus, one may wish to restart the heuristic several times given several different initial tours. This approach may aid in finding superior solutions to those that result from poor initial solutions. This approach is similar to jump search.[22] The CA heuristic employs two sets of parameters θ_λ^1 and θ_λ^2 ($0 \leqslant \theta_\lambda^1, \theta_\lambda^2 < 1$, where λ is an iteration index) for adjusting tour duration limit L to create different initial solutions. The description of the CA heuristic follows.

CA heuristic

Step 0: Set the maximum number of iterations to be I_{MAX}. Let iteration index λ be 0.

Step 1: If $\lambda > I_{\mathrm{MAX}}$, output the final solution and stop; otherwise, $\lambda = \lambda + 1$. Create an initial solution tour $\tau = \{0, 0\}$ and let its total duration $D(\tau) = 0$. Let

$$W = \{j | t_{0j} + t_{j0} \leqslant L \text{ and } \sum_{k \in \{i | s_{ji} > L - t_{0j} - t_{j0}, \forall 1 \leqslant i \leqslant m\}} p_{jk} \leqslant \alpha, \ \forall j \in V \backslash \{0\}\}$$

Step 2: If $W = \varnothing$, go to Step 3. Otherwise, select $j \in W$ and two vertices p and q on τ such that evaluation function $c(j, \tau) = t_{pj} + t_{jq} - t_{pq} + s_j - r_j$ is minimum and the updated tour duration for tour τ satisfies inequality: $D(\tau) + t_{pj} + t_{jq} - t_{pq} + s_j \leqslant L(1 + \theta_\lambda^1)$. If τ does not contain other vertices than depot 0, that is, $p = q = 0$, or if $r_j - (t_{pj} + t_{jq} - t_{pq}) > 0$, then insert j between p and q, remove j from W, update $D(\tau)$: $D(\tau) = D(\tau) + t_{pj} + t_{jq} - t_{pq} + s_j$ and repeat Step 2. Otherwise, go to Step 3.

Step 3: Apply Or-opt procedure[23] to improve the sequence of tour τ.

Step 4: If $\mathrm{P}(\mathrm{D}(\tau) > L) \leqslant \alpha$ (it is assumed that there exist two vertices that can be inserted in the tour before this chance constraint is violated), go to Step 5; otherwise, go to Step 6.

Step 5: If $W = \varnothing$, go back to Step 1. Otherwise, select $j \in W$ and two vertices p and q on τ such that evaluation function $c(j, \tau)$ is minimum and the updated tour duration for tour τ satisfies the chance constraints (if such a vertex j cannot be found, use evaluation function $c(j, \tau) = t_{pj} + t_{jq} - t_{pq} + s_j$). If $r_j - (t_{pj} + t_{jq} - t_{pq}) > 0$, insert j between p and q, remove j from W, update $D(\tau)$: $D(\tau) = D(\tau) + t_{pj} + t_{jq} - t_{pq} + s_j$ and repeat Step 5. Otherwise, go to Step 7.

Step 6: Select four consecutive vertices p, i, j and q on tour τ such that $t_{pi} + t_{ij} + t_{jq} - t_{pq} + s_i + s_j - r_i - r_j$ is maximum, remove vertices i and j from τ, and update $D(\tau)$: $D(\tau) = D(\tau) - (t_{pi} + t_{ij} + t_{jq} - t_{pq} + s_i + s_j)$. If the updated tour τ satisfies the modified chance constraints, $P[D(\tau) > L(1 - \theta_\lambda^2)] < \alpha$, go to Step 5; otherwise, repeat Step 6.

Step 7: If $W = \varnothing$, go back to Step 1. Otherwise, select two vertices $i \in \tau$ and $j \in W$ such that simultaneously removing vertex i from tour τ and inserting j in tour τ (at the position that would incur minimum insertion length) will result in maximum profit gain and the updated tour duration satisfies the chance constraints. If such vertices i and j can be found, make this exchange and repeat Step 7; otherwise, stop.

Note that two improvement procedures are employed in the CA heuristic. The first one is the classical Or-opt procedure in Step 3 for improving the sequence of tour τ. Other procedures with similar functionality (eg 2-opt and 3-opt) may also be used for such a purpose. The second one is a vertex substitution procedure (in Step 7) that iteratively increases the total tour profit by replacing a vertex in tour τ with one in set W.

Computational experiments

In this section, the average run time and solution quality of the exact algorithm and the CA heuristic are examined

through numerical experiments conducted on randomly generated problem instances. Both procedures were implemented in C++ and run on a DEC Alpha XP1000 computer, running Digital 4.0E operating system, using Digital's C++ compiler. The exact algorithm was coded around the CPLEX callable library.

Experimental design

The exact and heuristic procedures were tested on randomly generated problem instances with $n = 15$, 20, 30, 40, 50, 75, 100 and 125. For each problem, $n+1$ vertex positions (one depot plus n customer vertices) (a_i, b_i) $(i \in V)$ were generated based on a uniform distribution from $[0, 100]$.[2] Travel time matrix (t_{ij}) was then constructed by setting $t_{ij} = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}$. For each problem size (ie each n), the percentage (η) of vertices with stochastic service times varied from 20 to 100%. Service times and associated probabilities of occurrence were randomly assigned to these vertices, where service times were drawn from a uniform distribution on $[50, 100]$. The associated probabilities were chosen randomly from $(0, 1)$ and the sum of the associated probabilities for each vertex was one. The number of states for stochastic service times is set to three for 15- and 20-vertex problem instances, and two for problem instances with 30 or more vertices. Only relatively small numbers of states are tested, because as the tour length and the number of states increase, exact computation (required in the exact algorithm and the heuristic) to determine if a tour has exceeded the probability threshold requires extensive computation time. Two probability thresholds ($\alpha = 0.5$ and 0.2) were chosen for each problem instance. Each customer vertex was assigned a nonnegative reward value chosen from a uniform distribution [1, 100], similar to the study of Gendreau *et al*[5] for the STSP.

For each problem instance, the maximum tour duration limit L was generated. The procedure used for generating each such L is as follows. For each problem instance with $n < 75$, the optimal TSP solution, L_{TSP}, was obtained. This was done by employing our exact algorithm for the SSTSP, assuming that $L = \infty$. Each vertex is associated with an equally large reward and all service time realizations are equivalent to their expected values. For problem instances with $n \geqslant 75$, L_{TSP} was approximated through equation $L_{TSP} = \lfloor 0.95 \times UB(TSP) + 0.5 \rfloor$, where $UB(TSP)$ is the TSP solution value obtained by the largest insertion heuristic,[24] similar to the approximation method used in Fischetti *et al*.[4] Next, L was set such that $L = \beta \times L_{TSP}$, where $0 < \beta < 1$. For each problem, the CA heuristic was iterated five times (ie $I_{\mathrm{MAX}} = 5$), each time with a different parameter set $[\theta_\lambda^1, \theta_\lambda^2]$. The five parameter sets used in the computational experiments were $[0, 0.2]$, $[\alpha/10, 0.2]$, $[\alpha/5, 0.2]$, $[\alpha/3, 0.2]$ and $[\alpha, 0.2]$.

Numerical results

Average statistics for the experiments are reported in Tables 1–6. Some column headings are defined as follows:

β	parameter for defining the maximum tour duration limit
η	percentage of vertices with stochastic service times

Table 1 Computational results for 15-vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
15	0.5	0.25	20	5	0.99	0	52.0	5.2	0	0.1
			50	5	1.00	0	60.0	8.2	0	0.2
			100	5	1.00	0	56.6	4.0	1.2	0.1
		0.50	20	5	1.00	0	63.2	1.6	0	0.2
			50	5	1.00	0	123.2	4.6	1.6	0.5
			100	5	0.99	0	150.0	1.6	10.8	1.1
		0.75	20	5	1.00	0	53.6	0.8	0.4	0.2
			50	5	1.00	0	94.4	1.4	2.6	0.4
			100	5	0.94	1.1	19.2	0	0.6	0.1
	0.2	0.25	20	5	0.99	0	52.0	5.2	0	0.1
			50	5	1.00	0	78.0	9.4	0.6	0.2
			100	5	1.00	0	86.8	4.6	4.4	0.3
		0.50	20	5	1.00	0	88.0	2.2	1.2	0.3
			50	5	0.90	0	289.2	8.2	16.2	1.8
			100	5	0.99	0	172.8	2.6	13.6	1.2
		0.75	20	5	1.00	0	55.2	0.8	1.4	0.3
			50	5	0.99	0	141.8	1.6	7.8	0.7
			100	5	0.93	0.8	19.2	0	0.6	0.1

Table 2 Computational results for 20-vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
20	0.5	0.25	20	5	0.94	0	288.2	8.2	2.8	1.3
			50	5	1.00	0	190.8	11.0	2.6	0.9
			100	5	0.91	0	342.0	12.2	8.2	2.0
		0.50	20	5	0.99	0	222.4	1.4	7.4	1.7
			50	5	0.98	0	632.2	9.6	29.4	7.4
			100	5	0.97	0.2	972.2	16.2	68.4	12.0
		0.75	20	5	0.99	0	90.4	0.6	3.0	0.6
			50	5	0.99	0.2	130.2	0.4	6.8	1.1
			100	5	0.98	23.2	95.6	0.8	10.2	26.4
	0.2	0.25	20	5	0.94	0	288.2	8.2	2.8	1.3
			50	5	1.00	0	255.0	14.2	4.2	1.3
			100	5	0.92	0	476.4	17.4	17.8	3.2
		0.50	20	5	0.99	0	361.8	2.4	16.2	3.3
			50	5	0.97	0	1237.2	13.2	95.2	18.4
			100	5	0.95	0.2	1221.8	18.2	95.8	15.8
		0.75	20	5	0.99	0	110.6	1.0	5.0	0.8
			50	5	0.98	0.2	158.2	0.6	10.8	1.4
			100	5	0.97	25.2	125.0	1.0	15.8	21.0

Table 3 Computational results for 30-vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
30	0.5	0.10	20	5	1.00	0	226.8	19.8	0	1.2
			50	5	0.95	0	178.2	15.2	0	0.9
			100	5	1.00	0	179.0	18.8	1.0	1.0
		0.30	20	5	0.99	0	375.0	12.6	0.6	3.7
			50	5	1.00	0	488.0	8.0	1.8	5.3
			100	5	0.98	0	2551.6	36.8	47.2	40.4
		0.50	20	5	1.00	0	161.0	1.8	0.6	2.2
			50	5	1.00	0	1230.8	15.2	12.0	28.1
			100	5	0.99	0.6	2864.4	28.8	145.6	90.3
	0.2	0.10	20	5	1.00	0	226.8	19.8	0	1.2
			50	5	1.00	0	180.8	15.6	0.2	1.0
			100	5	1.00	0	179.0	18.8	1.0	1.0
		0.30	20	5	0.99	0	460.8	14.8	1.2	4.5
			50	5	1.00	0	604.2	7.8	2.4	6.3
			100	5	0.95	0	3183.2	47.4	82.2	59.2
		0.50	20	5	1.00	0	169.6	2.0	0.8	2.4
			50	5	1.00	0	1355.0	17.8	17.0	32.5
			100	5	0.98	0.4	4675.6	47.4	273.8	172.4

Succ number of instances out of 5 solved to optimality within 3600 c.p.u. seconds
CA/Exact ratio of the CA heuristic solution divided by the exact solution
V-cuts number of identified violated constraints (3), (12) and (13)
C-cuts number of identified conditional inequalities (14)
I-cuts number of identified illegal tour cuts (16)
c.p.u. c.p.u. time in seconds

Results in Tables 1–6 are the average solutions of the problem cases (out of a maximum of five runs for each category) that were successfully solved to optimality within 3600 c.p.u. seconds. The results reported in these tables

Table 4 Computational results for 40-vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
40	0.5	0.10	20	5	1.00	0	241.2	18.2	0	1.8
			50	5	0.95	0	369.2	21.8	0.2	3.4
			100	5	0.96	0	629.6	35.6	3	7.4
		0.20	20	5	0.99	0	798.6	7.8	0	10.5
			50	5	0.98	0	1277.2	25.8	3	18.5
			100	5	0.98	0	4308.2	101.8	57.8	115.6
		0.30	20	5	0.97	0	2311.4	13.8	0.2	40.0
			50	5	0.96	0	962.0	8.6	1.2	20.9
			100	5	0.97	0.2	11 861.6	239.4	332.2	718.0
	0.2	0.10	20	5	1.00	0	241.2	18.2	0	1.8
			50	5	0.94	0	369.2	21.8	0.2	3.4
			100	5	0.98	0	632.8	36.0	3.4	7.5
		0.20	20	5	0.96	0	803.6	8.2	0	10.6
			50	5	0.98	0	1471.8	38.0	5.0	21.5
			100	5	0.96	0	6009.6	139.2	100.2	187.9
		0.30	20	5	0.96	0	2371.8	14.0	0.2	41.3
			50	5	0.96	0	1083.0	11.6	2.6	24.3
			100	4	0.98	0.2	16 522.0	264.8	525.3	1193.3

Table 5 Computational results for 50-vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
50	0.5	0.10	20	5	0.97	0	326.0	11.6	0.2	3.8
			50	5	0.93	0	1054.0	44.4	2.8	16.8
			100	5	0.95	0	499.2	11.8	1.0	7.8
		0.20	20	5	1.00	0	1599.2	39.4	4.4	39.3
			50	5	0.95	0	6967.2	101.2	34.8	245.5
			100	5	0.98	0	5698.4	76.6	41	165.0
		0.30	20	5	0.97	0	2963.4	36.6	21.0	110.2
			50	5	0.97	0.1	12 141.8	208.2	106.8	712.8
			100	5	0.97	2.5	15 191.4	241.8	222.4	991.1
	0.2	0.10	20	5	0.97	0	610.4	28.4	2.0	8.0
			50	5	0.90	0	1179.0	48.4	3.6	18.7
			100	5	0.94	0	912.2	30.0	5.8	15.0
		0.20	20	5	0.99	0	1981.6	54.0	8.0	49.4
			50	5	0.97	0	7082.6	97.4	38.8	240.8
			100	5	0.98	0	6941.0	102.4	60.2	204.9
		0.30	20	5	0.95	0	3677.0	54.2	31.0	143.1
			50	5	0.97	0.1	21 566.8	391.6	214.6	1485.9
			100	4	0.96	1.6	11 429.8	115.8	158.8	741.3

indicate that the exact algorithm is able to solve problem instances with up to 125 vertices to optimality, and the heuristic can provide competitive approximate solutions with very limited computing effort. For example, most heuristic results are a few percent away from optimal and the computational times are close to 0 (we report c.p.u. times accurate to the first decimal place, ie if reported as 0 in the tables, it is less than 0.05 s). These results also show that the quality of average heuristic solutions is consistent across all problem categories (ie there is no indication of solution quality degradation with an increase in problem size). There are two cases ($n = 20$, $\beta = 0.75$, $\alpha = 0.5$ and 0.2) reported in Table 2 for which the heuristic c.p.u. times are more significant (about 23 and 25 s). These exceptions are due to the large c.p.u. times used to exactly compute the probability of an actual tour exceeding the pre-specified duration limit.

Table 6 Computational results for 75, 100 and 125 vertex problem instances

					Heuristic CA		*Exact algorithm*			
n	α	β	η	*Succ*	*CA/exact*	*c.p.u.*	*V-cuts*	*C-cuts*	*I-cuts*	*c.p.u.*
75	0.5	0.10	20	5	0.97	0	5573.8	121.8	6.0	241.3
			50	5	0.98	0	7177.8	157.6	15.4	373.9
		0.20	20	3	0.97	0	14 940.7	104.7	16.0	1533.5
			50	—	—	—	—	—	—	—
	0.2	0.10	20	5	0.97	0	7246.0	177.4	8.2	326.8
			50	2	0.91	0	20 223.5	559.5	80.0	1507.2
		0.20	20	2	0.95	0	20 102.5	132.0	18.5	2106.2
			50	—	—	—	—	—	—	—
100	0.5	0.10	20	4	0.97	0	7985.0	36.5	0	899.0
			50	—	—	—	—	—	—	—
		0.20	20	1	0.98	0	17 001.0	13.0	0	3053.1
			50	—	—	—	—	—	—	—
	0.2	0.10	20	3	0.99	0	3642.0	43.7	0.3	391.0
			50	—	—	—	—	—	—	—
		0.20	20	1	0.98	0	17 001.0	13.0	0	3039.9
			50	—	—	—	—	—	—	—
125	0.5	0.10	20	3	0.98	0	9214.3	91.0	0.3	1321.3
			50	—	—	—	—	—	—	—
	0.2	0.10	20	1	0.92	0	11 497.0	23.0	0	1310.5
			50	—	—	—	—	—	—	—

Computational results also show that several factors affect the problem difficulty. First, when the network size (n), tour duration parameter (β) and percentage of vertices with stochastic service times (η) increase, the problem becomes more difficult to solve (ie the average computational time for the exact algorithm is significantly higher). This is because when n and β increase, more vertices can be included in a solution tour. Additionally, when η increases, the gap between the initial relaxation and augmentation problem (P′) (ie the upper bound) and the lower bound generated by the CA heuristic increases, resulting in more candidates solutions. This is because the bounding constraint (9) becomes less tight when more customer service times are stochastic. Second, the problem difficulty increases with decreasing probability threshold α. This is due to the fact that when α is small, most integral solutions obtained by solving LP relaxation sub-problems during the branch-and-cut process that are determined before the chance constraints are imposed will be infeasible with respect to the chance constraints. This means that more iterations may be required before these sub-problems can be fathomed.

The number of generated valid cuts (identified violated deterministic STSP cuts, conditional cuts and illegal tour elimination cuts) is also reported in Tables 1–6. Given problem size (n), one can see that, in general, the number of total valid cuts increases for problem instances with low values of α and greater numbers of possible service times, that is with decreasing α and increasing η values. In Table 6, some cells are filled with '—'. This means that none of the five randomly generated instances for that problem category was solved to optimality in under 3600 c.p.u. seconds. Note that if this c.p.u. limit were increased, the proposed exact algorithm might be able to solve additional problem instances to optimality. For example, for problem instances with $n=125$, $\alpha=0.2$, $\beta=0.10$ and $\eta=20$, two out of five instances were solved to optimality within 10 000-s c.p.u. limit, and the average CA/Exact ratio (based on these two solved instances) is 0.96.

Conclusions

In this paper, a chance-constrained stochastic program is proposed to model the SSTSP and both exact and heuristic solution techniques are provided to address it. Computational experiments indicate that the exact algorithm based on branch-and-cut can solve small- and moderate-size problems and the proposed heuristic can provide very competitive approximate solutions with significantly reduced computing effort.

Acknowledgement—This work was partially supported by NSF Grant CMS 9875305. This support is gratefully acknowledged, but implies no endorsement of the findings.

References

1 Golden B, Levy L and Vohra R (1987). The orienteering problem. *Naval Res Logist* **34**: 307–318.

2 Laporte G and Martello S (1990). The selective traveling salesman problem. *Discrete Appl Math* **26**: 193–207.
3 Ramesh R, Yoon Y and Karwan M (1992). An optimal algorithm for the orienteering tour problem. *ORSA J Comput* **4**: 155–165.
4 Fischetti M, Salazar J and Toth P (1998). Solving the orienteering problem through branch-and-cut. *INFORMS J Comput* **10**: 33–148.
5 Gendreau M, Laporte G and Semet F (1998). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks* **32**: 263–273.
6 Tsiligirides T (1984). Heuristic methods applied to orienteering. *J Opl Res Soc* **35**: 797–809.
7 Wren A and Holliday A (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Opl Res Quart* **23**: 333–344.
8 Golden B, Wang Q and Liu L (1988). A multifaceted heuristic for the orienteering problem. *Naval Res Logist* **35**: 359–366.
9 Ramesh R and Brown K (1991). An efficient four-phase heuristic for the generalized orienteering problem. *Comput Opns Res* **18**: 151–165.
10 Chao I (1993). *Algorithms and solutions to multi-level vehicle routing problems*. PhD dissertation, University of Maryland, College Park.
11 Chao I, Golden B and Wasil E (1996). A fast and effective heuristic for the orienteering problem. *Eur J Opl Res* **88**: 475–489.
12 Golden B, Wasil E, Kelly J and Chao I (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Crainic T and Laporte G (eds). *Fleet Management and Logistics*. Kluwer Academic Publishers, Boston, pp 33–56.
13 Gendreau M, Hertz A and Laporte G (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Opns Res* **40**: 1086–1094.
14 Gendreau M, Laporte G and Semet F (1998). A tabu search heuristic for the undirected selective traveling salesman problem. *Eur J Opl Res* **106**: 539–545.
15 Laporte G, Louveaux F and Mercure H (1992). The vehicle routing problem with stochastic travel times. *Transport Sci* **26**: 161–170.
16 Laporte G and Louveaux F (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Opns Res Lett* **13**: 133–142.
17 Fischetti M, Salazar J and Toth P (1997). A branch-and-cut algorithm for the symmetric generalized travelling salesman problem. *Opns Res* **45**: 378–394.
18 Gendreau M, Laporte G and Semet F (1997). The covering tour problem. *Opns Res* **45**: 568–576.
19 Padberg M and Hong S (1980). On the symmetric travelling salesman problem: a computational study. *Math Prog Study* **12**: 78–107.
20 Barnhart C *et al* (1998). Branch-and-cut: column generation for solving huge integer programs. *Opns Res* **46**: 316–329.
21 Laporte G, Louveaux F and Mercure H (1994). *A priori* optimization of the probabilistic traveling salesman problem. *Opns Res* **42**: 543–549.
22 Tsubakitani S and Evans J (1998). An empirical study of a new metaheuristic for the traveling salesman problem. *Eur J Opl Res* **104**: 113–128.
23 Or I (1976). *Travelling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking*. PhD dissertation. Northwestern University, Evanston.
24 Rosenkrantz D, Stearns R and Lewis P (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM J Comput* **6**: 563–581.

Received March 2003;
accepted May 2004 after two revisions