2-2018

Risk-sensitive stochastic orienteering problems for trip optimization in urban environments

Pradeep VARAKANTHAM
Singapore Management University, pradeepv@smu.edu.sg

Akshat KUMAR
Singapore Management University, akshatkumar@smu.edu.sg

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

William YEOH
New Mexico State University

DOI: https://doi.org/10.1145/3080575

Published in ACM Transactions on Intelligent Systems and Technology, 2018, 9 (3), Article 24.
https://doi.org/10.1145/3080575

Risk-Sensitive Stochastic Orienteering Problems for Trip Optimization in Urban Environments

PRADEEP VARAKANTHAM, AKSHAT KUMAR, and HOONG CHUIN LAU,
Singapore Management University
WILLIAM YEOH, New Mexico State University

Orienteering Problems (OPs) are used to model many routing and trip planning problems. OPs are a variant of the well-known traveling salesman problem where the goal is to compute the highest reward path that includes a subset of vertices and has an overall travel time less than a specified deadline. However, the applicability of OPs is limited due to the assumption of deterministic and static travel times. To that end, Campbell et al. extended OPs to Stochastic OPs (SOPs) to represent uncertain travel times (Campbell et al. 2011). In this article, we make the following key contributions: (1) We extend SOPs to Dynamic SOPs (DSOPs), which allow for time-dependent travel times; (2) we introduce a new objective criterion for SOPs and DSOPs to represent a percentile measure of risk; (3) we provide non-linear optimization formulations along with their linear equivalents for solving the risk-sensitive SOPs and DSOPs; (4) we provide a local search mechanism for solving the risk-sensitive SOPs and DSOPs; and (5) we provide results on existing benchmark problems and a real-world theme park trip planning problem.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Planning and scheduling**; **Planning under uncertainty**;

Additional Key Words and Phrases: Orienteering problems, risk-sensitive optimization, sample average approximation

1 INTRODUCTION

In competitive orienteering sports, each individual starts at a specified control point, tries to visit as many checkpoints as possible, and returns to the starting control point within a given time frame. Each checkpoint has a certain score and the objective is to maximize the total collected score. Motivated by such orienteering sports, *Orienteering Problems* (OPs) (Tsiligrides 1984) represent the problem of selecting the maximum reward path involving a subset of vertices that can be traversed within the given deadline. OPs have been studied extensively, and for a detailed survey of existing

Authors' addresses: P. Varakantham, A. Kumar, and H. C. Lau, 80 Stamford Road, School of Information Systems, Singapore Management University, Singapore - 178902; emails: {pradeepv, akshatkumar, hclau}@smu.edu.sg; W. Yeoh, Department of Computer Science and Engineering, College of Engineering, Washington University in St Louis, St. Louis, MO 63130, United States of America; email: wyeoh@wustl.edu.

work, refer to the survey by Gunawan et al. (2016). Oversubscription planning problems (Smith 2004) are another category of problems where OPs can potentially be employed.

Unfortunately, OPs assume that travel times are deterministic and static, an assumption that is not justifiable in most vehicle routing or trip design problems. This is because travel time between two locations is not only dependent on distance traveled but also on congestion levels, type of road segments, and other such factors. To that end, we are interested in OPs where the travel time between any two vertices is not only uncertain (Stochastic OPs (SOPs)) but also time dependent or dynamic (DSOPs). The key argument for dynamic travel times in DSOPs is that traffic is usually time dependent (e.g., road congestion is high during peak hours and low at other times, traffic at large roller coaster rides is low immediately after lunch).

Our research is motivated by a number of applications[1] that have such uncertain and time-dependent travel times:

(1) Trip design problems (Archetti et al. 2008; Gavalas et al. 2014; Gunawan et al. 2016) that are of relevance in large cities, theme parks, museums, large expos, and so forth, provide another compelling category of use cases for SOPs and DSOPs. Once again the time available is limited, and travel time (that can include a significant portion of waiting time) is uncertain and time dependent due to varying congestion levels at various points of interest on different days. By analyzing past data, models for travel time can be constructed. Reward represents the utility/preference for that point of interest.
(2) Businesses that involve deliveries (food, equipment, clothing, home fuel, etc.) or service (technical service associated with repairs, plumbing, television, utilities, etc.) to customer locations (Golden et al. 1987) provide an ideal category of use cases for SOPs and DSOPs. The payments that customers make for deliveries or services can be considered as the reward for visiting a certain vertex. In high demand settings, not all requests can be catered to in the time available (representing the budget). In addition, since deliveries have to be made by vehicles traveling on roads, there is uncertainty and time dependence associated with travel times.[2]
(3) Another category of applications is for traveling sales persons who do not have enough time to visit all possible locations (Tsiligrides 1984). The sales person knows the expected number of sales in each city and wants to maximize total sales in the time available. SOPs and DSOPs are an ideal model to represent the guidance problem for traveling salespersons that have to deal with uncertain and dynamic traffic conditions.

Due to stochasticity and dynamism, there is a risk associated with violating the deadline for any strategy. Considering robust objectives (i.e., the worst case) yields very conservative solutions. Therefore, to achieve the right balance between completely avoiding risk and being overly conservative, we choose a risk-sensitive criterion, where we compute the maximum reward path where the probability of violating the deadline is less than a given risk parameter α. Overall, we make the following categories of contributions:[3]

[1]Please note that each of the points in the list provides a category of applications and not just one application.
[2]The travel time models required by SOP and DSOP can be constructed by analyzing existing traffic data.
[3]This article combines our previous work (Varakantham and Kumar 2013; Lau et al. 2012) and extends them in three major ways: (a) We formulate risk-sensitive DSOPs as a mixed integer linear program. Providing a linear formulation for DSOPs is significantly more difficult than the one for SOPs and hence this is a significant improvement over the two papers mentioned earlier; (b) we provide a general purpose extension to SAA that is applicable for domains with continuous valued uncertainties. This helps improve scalability of the optimization formulation in (a) considerably; (c) we provide experimental results for risk-sensitive DSOPs on the real-world theme park problem.

(1) We provide heuristic approaches that employ local search to solve SOPs and DSOPs with a risk-sensitive objective. Specifically, in this technique, we start from an initial solution and improve upon it iteratively.
(2) We provide a principled approximation technique based on *Sample Average Approximation* (SAA) (Pagnoncelli et al. 2009) to formulate risk-sensitive SOPs and DSOPs. In addition to the application of the basic SAA method, we also provide an improvement to the SAA that is based on aggregation of samples and is applicable in problems with durational uncertainty.

In order to illustrate the utility of our approaches, we evaluate them on a synthetic benchmark set introduced by Campbell et al. (2011) and also on a real-world theme park navigation problem, where the travel times are computed from a year-long dataset of travel times at a popular theme park in Singapore. We observe the following: (1) In small- to medium-scale problems, linear optimization formulations for solving SOPs and DSOPs provide significant and consistent improvement in solution quality compared to the local search approach (more than 50% in some synthetic benchmarks and more than 100% in some real-world problem instances); and (2) on large-scale problems, our local search approach is able to solve DSOPs quickly, while our linear optimization-based formulations are unable to generate any good quality solutions.

We now provide the outline for the rest of this article. In Section 2, we provide a background on the formal description of OPs, SOPs, and SAA. We then describe formal models for DSOPs along with the risk-sensitive criterion in Section 3. We provide linear optimization formulations and local search-based approaches for solving SOPs and DSOPs in Sections 4 and 5, respectively. Finally, we provide empirical results of our approaches on benchmark and real-world SOPs and DSOPs in Section 6 before discussing related work in Section 7 and summarizing our work in Section 8.

2 BACKGROUND

In this section, we first provide a formal model for OPs and SOPs.

2.1 Orienteering Problems (OPs)

An OP (Tsiligrides 1984) is defined by a tuple $\langle G, T, R, v_1, v_n, H\rangle$, where $G = \langle V, E\rangle$ is a graph with sets of vertices V and edges E; $T : v_i \times v_j \to \mathbb{R}^+ \cup \{0, \infty\}$ specifies a finite non-negative travel time between vertices v_i and v_j if $(v_i, v_j) \in E$ and ∞ otherwise; and $R : v_i \to \mathbb{R}^+ \cup \{0\}$ specifies a finite non-negative reward for each vertex $v_i \in V$. H refers to the deadline or the time horizon; v_1 and v_n are the starting and ending vertices.

A simplified version of our motivating theme park navigation problem, where travel and queueing times are deterministic and static, can be modeled as an OP. The vertex v_1 corresponds to the entrance of the park, while the rest of the vertices v_i correspond to attractions in the park and sink v_n can be any arbitrary vertex in V. Travel times $T(v_i, v_j)$ correspond to the sum of the travel time between attractions v_i and v_j and the queueing time at attraction v_j.

A solution in an OP is a Hamiltonian path over a subset of vertices including source vertex v_1 and sink vertex v_n and whose total travel time is no larger than H. Optimally solving OPs entails finding a solution that maximizes the sum of rewards of vertices in its path. The source and sink vertices in OPs are often distinct vertices. In the special case where they are the same vertex, the problem is called an *Orienteering Tour Problem* (OTP) (Ramesh et al. 1992). The difference between both formulations is small. It is always possible to add a dummy edge with zero travel time between the source and sink vertices to convert an OP to an OTP.

Researchers have proposed several branch-and-bound methods to solve OPs (Laporte and Martello 1990) including optimizations with cutting plane methods (Leifer and Rosenwein 1994; Fischetti et al. 1998). However, since OPs are NP-hard (Golden et al. 1987), exact algorithms often

suffer from scalability issues. Thus, constant-factor approximation algorithms (Blum et al. 2007) are necessary for scalability. Researchers also proposed a wide variety of heuristics to address this issue including sampling-based algorithms (Tsiligrides 1984), local search algorithms (Golden et al. 1987; Chao et al. 1996), neural network-based algorithms (Wang et al. 1995), and genetic algorithms (Tasgetiren 2001). More recently, Schilde et al. developed an ant colony optimization algorithm to solve a bi-objective variant of OPs (Schilde et al. 2009).

2.2 Stochastic OPs (SOPs)

The assumption of deterministic travel times is not valid in many real-world settings. Using our motivating theme park navigation problem as an example, the travel time of patrons depends on numerous factors like fatigue, natural speed of walking, distractions such as food places, or traveling with children or senior citizens. Representing such factors accurately and obtaining deterministic travel times is not possible with current methods. Hence, researchers have extended OPs to *Stochastic OPs* (SOPs) (Campbell et al. 2011), where travel times, $T(v_i, v_j)$ are now random variables that follow a given distribution, and the goal is to find a path that maximizes the sum of expected utilities from vertices in the path. The random variables are assumed to be independent of each other.

Given a strategy π (sequence of vertices to visit), the expected utility of a vertex is the difference between the expected reward and the expected penalty of the vertex. The expected reward (or penalty) of a vertex is the reward (or penalty) of the vertex times the probability that the travel time along the path thus far is no larger (or larger) than H. Formally, the expected utility $U_\pi(v_i)$ of a vertex v_i given a strategy π is given by

$$U_\pi(v_i) = \Pr_\pi(a_i \leq H)\, R(v_i) - \Pr_\pi(a_i > H)\, C(v_i), \tag{1}$$

where the random variable a_i is the arrival time at vertex v_i (i.e., the travel time from v_1 to v_i), $R(v_i)$ is the reward of arriving at vertex v_i before or at H, and $C(v_i)$ is the penalty of arriving at vertex v_i after H.

The overall objective of solving SOPs in this case can be formally summarized as

$$\max_\pi \sum_i U_\pi(v_i).$$

Campbell et al. have extended OP algorithms to solve SOPs including an exact branch-and-bound method and a local search method based on variable neighborhood search (Campbell et al. 2011). Gupta et al. introduced a constant-factor approximation algorithm for a special case of SOPs, where there is no penalty for arriving at a vertex after H (Gupta et al. 2012).

3 MODELS

We now formally describe our extensions to SOPs along with a definition of the risk-sensitive criterion.

3.1 Dynamic Stochastic OPs (DSOPs)

SOPs assume independence of travel time distributions across different edges. However, in many problems, there is a considerable dependence of travel times on the arrival time at a vertex. Using our motivating theme park navigation problem again as an example, the travel time of a patron depends on factors like fatigue, and the level of fatigue of a patron increases as the patron spends more time in the park. Furthermore, waiting time at attractions, which is a key component of travel time, is dependent on the time of the day. For instance, large roller coasters are not preferred immediately after lunch.

To capture dependencies between travel time distributions and represent time-dependent travel time distributions, we introduce an extension to SOPs called *Dynamic SOPs* (DSOPs). The key difference from SOPs is that the travel time distribution in a DSOP for moving from vertex v_i to vertex v_j depends on the arrival time a_i at vertex v_i. In this article, we will assume $T_{i,j}$ to be a discrete set of distributions, where each element of the set corresponds to a range of values for a_i. Notationally, the travel time distribution for an arrival time of a_i is represented as $T_{i,j}^{a_i}$ and, hence, the probability that travel time is u is given by $T_{i,j}^{a_i}(u)$.

3.2 Risk-Sensitive Criterion

While expected utility is a good metric in general, the approach by Campbell et al. (2011) suffers from many limitations. Firstly, it is a point estimate solution that does not consider the "risk" profile of the patron. By "risk," we do not refer to the term used in a financial sense, but rather the level of conservativeness measured in terms of the probability of completing the path within the deadline. In other words, a risk-seeking patron will be prepared to choose a sequence of attractions that have a large utility, but with a high probability of not completing the path within the deadline, compared to a risk-averse patron who might choose a more "relaxed" path with lower utility. Secondly, the underlying measurement of expected utility is not intuitive in the sense that a utility value accrued at each attraction does not usually depend on the probability that the patron arrives at the attraction by a certain time; but, rather, the utility is accrued when the attraction is visited, and the patron is concerned with visiting all the attractions (i.e., sum of utilities) within a certain time threshold.

Given the above considerations, we are interested in a problem that allows the patron to trade off the level of conservativeness (or risk) against the total utility. More precisely, given a value $0 < \alpha < 1$, we are interested in obtaining a path π that maximizes the reward obtained while ensuring that the probability of reaching the destination vertex v_n after the deadline H is no larger than α. Or, more precisely,

$$\Pr_{\pi}(a_n \geq H) \leq \alpha, \tag{2}$$

where a_n is the arrival time at the last vertex of the path.

4 SOLVING RISK-SENSITIVE SOPS AND DSOPS USING LINEAR OPTIMIZATION

In this section, we provide linear optimization formulations that approximately represent risk-sensitive SOPs and DSOPs.

4.1 Solving Risk-Sensitive SOPs

We first formulate an SOP with the risk-sensitive criterion (see Section 3.2) as an optimization problem. We then employ SAA to get a deterministic approximation and we refer to this formulation as MILP-SAA.

For each directed edge (v_i, v_j), the binary variable $\pi_{i,j}$ denotes whether the edge (v_i, v_j) is in the final path. The random variable $T_{i,j}$ denotes the travel time for traversing the directed edge (v_i, v_j). We assume that the underlying distribution for each variable $T_{i,j}$ is provided as input. The parameter R_i represents the reward of arriving at vertex v_i.

Table 1 shows a risk-sensitive SOP formulated as a non-linear chance-constrained mathematical program. We now describe its structure. We designate the source vertex with id 1 and the sink vertex with id n. The objective function seeks to maximize the overall reward obtained based on vertices visited. Constraints 4 and 5 specify that there is a single incoming and outgoing active edge for each vertex. We refer to constraints 4–6 as flow preservation constraints and are henceforth represented as $\mathbf{F}_{\pi} \leq 0$.

Table 1. A Risk-Sensitive SOP Formulated as a Chance-Constrained Mathematical Program

$$\max_{\pi} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \tag{3}$$

$$\sum_{j} \pi_{j,i} \leq 1 \qquad \forall v_i \in V \tag{4}$$

$$\sum_{j} \pi_{i,j} \leq 1 \qquad \forall v_i \in V \tag{5}$$

$$\sum_{j} \pi_{i,j} - \sum_{j} \pi_{j,i} = \begin{cases} 1 & \text{if } i = 1 \\ -1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \qquad \forall v_i \in V \tag{6}$$

$$r_1 = 1 \tag{7}$$

$$r_n = n \tag{8}$$

$$r_i \leq r_j - 1 + (1 - \pi_{i,j})\, M \qquad \forall v_i, v_j \in V \tag{9}$$

$$\Pr\left(\sum_{i,j} \pi_{i,j}\, T_{i,j} > H \right) \leq \alpha \tag{10}$$

$$\pi_{i,j} \in \{0, 1\} \qquad \forall v_i, v_j \in V \tag{11}$$

$$r_i \in [1, n] \qquad \forall v_i \in V \tag{12}$$

To ensure that there are no cycles in the path, we introduce a new set of variables r_i for each vertex v_i to denote its rank in the final path. For instance, if the rank of the source vertex is 1, then any vertex connected immediately from the source will be ranked greater than 1, and so on. This monotonically increasing ranking of vertices will ensure that no cycles are generated. Constraint 9 models this ranking scheme. The parameter M is a large constant used to maintain the consistency of the constraint. We refer to constraints 7–9 as cycle prevention or sub-tour elimination constraints and are henceforth represented as $\mathbf{C}_r \leq 0$. These ranking-based constraints for elimination of sub-tours were first introduced by Miller et al. (1960) in their Miller-Tucker-Zemlin (MTZ) path sequencing formulation. We can also use the separation algorithms (Fischetti and Toth 1997) that incrementally introduce Subtour Elimination Constraints (SECs) based on the violations in the current iteration. However, not all models of SECs are applicable for OPs due to the budget constraint, which entails that not all vertices will be included in the final solution.

Our formulation for SOPs and DSOPs works with both the above methods of sub-tour elimination. We chose the ranking method due to ease of implementation. Constraint 10 is a chance constraint. The total duration of the SOP is denoted as $\sum_{i,j} \pi_{i,j}\, T_{i,j}$, which is a random variable as each $T_{i,j}$ is a random variable. The parameter H denotes the input deadline. The chance constraint states that the probability of violating the deadline should be no greater than $\alpha \in (0, 1)$, which is another input parameter. This constraint is non-linear and, in general, a closed-form expression is not readily available. We next show how to compute a deterministic equivalent of this constraint using SAA in a mixed-integer program.

For each edge of the graph, we generate $|Q|$ samples for the random variable $T_{i,j}$, where $t^q_{i,j}$ denotes the q-th sample. We represent the chance constraint using the following linear constraints:

$$z^q \geq \frac{\sum_{i,j} \pi_{i,j}\, t^q_{i,j} - H}{M} \qquad \forall q \in Q, \tag{13}$$

$$z^q \in \{0,1\} \qquad \forall q \in Q, \tag{14}$$

where we have introduced auxiliary integer variables z^q for each sample q. Using these auxiliary variables, we can represent violation of the chance constraint for samples as

$$\frac{\sum_q z^q}{|Q|} \leq \alpha', \tag{15}$$

where α' is a parameter that is set by the user and is generally smaller than the parameter α as used in constraint 10. The setting of α' is critical and we will provide a detailed discussion about it in our experimental results section. To summarize, we get a deterministic mixed-integer program corresponding to the stochastic program of Table 1 by using $|Q|$ SAA samples for each random variable corresponding to an edge, introducing auxiliary integer variables z^q for each SAA sample, and replacing the stochastic constraint 10 with linear constraints 13–15. The following theoretical results establish the convergence guarantees for the SAA technique.

Theorem 4.1 (Pagnoncelli et al. 2009). *Let $x^\star$ be the optimal solution and $v^\star$ be its quality, $\hat{x}_N$ be the solution found with SAA using N samples and $\hat{v}_N$ be its quality, and the parameter $\alpha' = \alpha$. Then, $\hat{v}_N \to v^\star$ and $\hat{x}_N \to x^\star$ as $N \to \infty$.*

The next theorem provides convergence results regarding the feasibility of the solution with respect to the chance constraint.

Theorem 4.2 (Pagnoncelli et al. 2009). *If $\hat{x}_N$ is a feasible solution of the SAA problem and $\alpha' < \alpha$, then the probability that $\hat{x}_N$ is a feasible solution of the actual problem approaches 1 exponentially fast with increasing number of samples N.*

4.2 Solving Risk-Sensitive DSOPs

We now provide two optimization formulations, MILP-SAA and MILP-Percentile, that approximately represent a DSOP with the risk-sensitive objective. MILP-SAA is based on SAA and, thus, has theoretical convergence guarantees. MILP-Percentile is a heuristic approximation of MILP-SAA that considerably improves its scalability.

4.2.1 MILP-SAA. Similar to SOPs, for each directed edge (v_i, v_j), we use the binary variable $\pi_{i,j}$ to denote whether the edge (v_i, v_j) is in the final path and, for each vertex v_i, we use R_i to represent the reward of arriving at that vertex. However, unlike SOPs, the travel time for traversing the directed edge (v_i, v_j) depends on the arrival time a_i at the source vertex v_i. We thus use $T_{i,j}^{a_i}$ to denote this travel time distribution. To better represent the real world, for each vertex v_i, instead of assuming that every time point of arrival at v_i leads to a different travel time distribution, we assume that there exist P intervals of arrival times at a vertex (the P intervals can be different for different vertices) which lead to a different travel time distribution. We use $[\check{sl}_i^p, \hat{sl}_i^p]$ to denote the p-th interval. Additionally, for each interval p and vertex v_j (where $(v_i, v_j) \in E$) pair, there is a travel time distribution associated with it.

Table 2 shows a risk-sensitive DSOP formulated as a non-linear chance-constrained mathematical program. The constraints associated with ensuring the feasibility of path π and prevention of cycles are similar to the ones presented for solving risk-sensitive SOPs in Table 1 and are represented as $\mathbf{F}_\pi \leq 0$ and $\mathbf{C}_r \leq 0$, respectively. The two sources of non-linearity present in this formulation are constraints 19 and 21. We now describe how we linearize these two sets of constraints:

—Constraint 19: We first address the non-linearity presented in constraint 19. For ease of explanation, we provide the linearization by assuming that $T_{j,i}^{a_j}$ is a continuous variable and not a random variable. We will later relax this assumption by considering samples of the

Table 2. Risk-Sensitive DSOP Formulated as a Chance-Constrained Mathematical Program

$$\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \tag{16}$$

$$\mathbf{F}_{\pi} \leq 0 \tag{17}$$

$$\mathbf{C}_{r} \leq 0 \tag{18}$$

$$a_i = \sum_{j} \left[\pi_{j,i}\, a_j + \pi_{j,i}\, T_{j,i}^{a_j} \right] \qquad \forall v_i \in V \setminus \{v_1\} \tag{19}$$

$$a_1 = 0 \tag{20}$$

$$\Pr(a_n > H) \leq \alpha \tag{21}$$

$$a_i \in [0, M] \tag{22}$$

random variable $T_{j,i}^{a_j}$. Within constraint 19, we have two non-linear terms, namely, $\pi_{j,i}\, a_j$ and $\pi_{j,i}\, T_{j,i}^{a_j}$. We account for $\pi_{j,i}\, a_j$ by introducing a new variable $b_{j,i}$ that is defined as

$$b_{j,i} = \pi_{j,i} \cdot a_j.$$

Intuitively, this refers to the following logical constraints:
—If $\pi_{j,i} = 1$, then $b_{j,i} = a_j$.
—If $\pi_{j,i} = 0$, then $b_{j,i} = 0$.
This definition of $b_{j,i}$ can be linearized as follows:

$$b_{j,i} \leq a_j \qquad \forall (v_j, v_i) \in E, \tag{23}$$

$$b_{j,i} \leq \pi_{j,i}\, M \qquad \forall (v_j, v_i) \in E, \tag{24}$$

$$a_j \leq b_{j,i} + (1 - \pi_{j,i})\, M \qquad \forall (v_j, v_i) \in E, \tag{25}$$

where M is a large number.

Finding a linear equivalent for the term $\pi_{j,i}\, T_{j,i}^{a_j}$ is more difficult as $T_{j,i}^{a_j}$ is dependent on the arrival time a_j:

$$\hat{T}_{j,i} = \pi_{j,i} \cdot T_{j,i}^{a_j}. \tag{26}$$

We exploit the intervals (represented using p) in travel time distribution at each vertex to find linear equivalent constraints.

$$\hat{T}_{j,i} = \sum_{m} T_{j,i}^{p} \qquad \forall (v_j, v_i) \in E. \tag{27}$$

Intuitively, $T_{j,i}^{p}$ should satisfy the following logical constraints ($sl_{j,i}^{p} = 1$ indicates interval of arrival at j is p):
—If $\pi_{j,i} = 1 \wedge sl_{j,i}^{p} = 1$, then $T_{j,i}^{p} = D_{j,i}^{p}$.
—If $\pi_{j,i} = 0 \vee sl_{j,i}^{p} = 1$, then $T_{j,i}^{p} = 0$.
We linearize these logical constraints as follows:

$$T_{j,i}^{p} \leq \pi_{j,i}\, D_{j,i}^{p} \qquad \forall p \in P, (v_j, v_i) \in E, \tag{28}$$

$$T_{j,i}^{p} \leq sl_{j}^{p}\, D_{j,i}^{p} \qquad \forall p \in P, (v_j, v_i) \in E, \tag{29}$$

$$D_{j,i}^{p} - T_{j,i}^{p} \leq (2 - sl_{j}^{p} - \pi_{j,i})\, D_{j,i}^{p} \qquad \forall p \in P, (v_j, v_i) \in E, \tag{30}$$

where $T_{j,i}^{p}$ is a variable that is set to the constant travel time $D_{j,i}^{p}$ between vertices v_j and v_i—if the arrival time at v_j is in the interval p and policy $\pi_{j,i}$ is set to 1—and 0 otherwise. The

sl_j^p variables indicate whether the arrival time at v_j belongs to interval p, which is achieved through the following linear constraints:

$$1 - sl_j^p \geq \frac{\check{sl}_j^p - a_j}{M} \qquad \forall p \in P, v_j \in V, \tag{31}$$

$$1 - sl_j^p \geq \frac{a_j - \hat{sl}_j^p}{M} \qquad \forall p \in P, v_j \in V, \tag{32}$$

$$\sum_p sl_j^p = 1 \qquad \forall v_j \in V. \tag{33}$$

The overall Mixed Integer Linear Program (MILP) without considering the uncertainty distributions for $T_{j,i}^{a_j}$ is provided in Table 7 in the online appendix. We next prove the equivalence of constraints 27–33 to constraint 26.

Proposition 4.3. *When we have intervals in travel time distribution, for a given realization of uncertainties* $\boldsymbol{D}$*, constraints 27–33 are equivalent to constraint 26.*

Proof. When we have intervals in travel time distribution, for a given realization of uncertainties, $T_{j,i}^{a_j}$ is a piecewise constant function. That is to say,

$$T_{j,i}^{a_j} = \begin{cases} D_{j,i}^1, \textbf{if} \quad \check{sl}_j^1 \leq a_j \leq \hat{sl}_j^1 \\ D_{j,i}^2, \textbf{if} \quad \check{sl}_j^2 \leq a_j \leq \hat{sl}_j^2 \\ \dots \\ D_{j,i}^p, \textbf{if} \quad \check{sl}_j^p \leq a_j \leq \hat{sl}_j^p \\ \dots \end{cases}.$$

Given this piecewise constant representation,

$$\hat{T}_{j,i} = \pi_{j,i} \cdot T_{j,i}^{a_j} = \begin{cases} 0, & \textbf{if} \quad \pi_{j,i} = 0 \\ D_{j,i}^1, & \textbf{if} \quad \check{sl}_j^1 \leq a_j \leq \hat{sl}_j^1 \ \wedge \ \pi_{j,i} = 1 \\ D_{j,i}^2, & \textbf{if} \quad \check{sl}_j^2 \leq a_j \leq \hat{sl}_j^2 \ \wedge \ \pi_{j,i} = 1 \\ \dots \\ D_{j,i}^p, & \textbf{if} \quad \check{sl}_j^p \leq a_j \leq \hat{sl}_j^p \ \wedge \ \pi_{j,i} = 1 \\ \dots \end{cases}.$$

Constraints 31–33 ensure right interval for a_j amongst

$$\{[\check{sl}_j^1, \hat{sl}_j^1], [\check{sl}_j^2, \hat{sl}_j^2], \dots\}.$$

Constraints 29 and 30 ensure right assignment for the interval variable, sl_j^p.

Constraint 28 captures the assignment corresponding to the value of $\pi_{j,i}$.

Constraint 27 combines the output from ensuring all conditions are met.

We can verify that constraints 27–33 represent the above piecewise constant function by simple substitution of values for sl_j^p and $\pi_{j,i}$ variables. □

We now relax the assumption that $T_{j,i}^{a_j}$ is a realization and not a random variable. For this, we draw a set of $|Q|$ samples, $\xi = \{\xi_1, \dots, \xi_q, \dots, \xi_{|Q|}\}$, where $\xi_q = \{T_{j,i}^{p,q}\}_{v_j \in V, v_i \in V, p \in P}$ comes from the distributions $\{T_{j,i}^p\}_{v_j \in V, v_i \in V, p \in P}$. In this representation, $T_{j,i}^{p,q}$ is a number indicating the travel time if you arrive in interval p at vertex v_j according to sample q. Intuitively, each sample ξ_q contains a travel time value obtained from each of the distributions $T_{j,i}(p)$

corresponding to every edge (v_j, v_i) and arrival interval p at vertex v_j. To account for the samples, all variable groups associated with a_j, $b_{j,i}$, $T^p_{j,i}$ will now have an index associated with the sample q.

— Constraint 21: As with SOPs, we linearize the chance constraint by finding the deterministic equivalent using SAA. More specifically,

$$z^q \geq \frac{a^q_n - H}{M} \qquad \forall q \in Q, \tag{34}$$

$$z^q \in \{0, 1\} \qquad \forall q \in Q, \tag{35}$$

where we have introduced auxiliary integer variables z^q for each sample q. Using these auxiliary variables, constraint 21 is represented as

$$\frac{\sum_q z^q}{|Q|} \leq \alpha', \tag{36}$$

where α' is the parameter that is set by the user and is smaller than the parameter α used in constraint 21. The updated MILP is provided in Table 8 in the online appendix. We call this formulation MILP-SAA.

4.2.2 MILP-Percentile. While MILP-SAA is a principled mechanism to solve a risk-sensitive DSOP, it cannot scale to the real-world theme park problems of interest in this article.[4] We now describe a general purpose extension to SAA that can be employed in problem domains where uncertainty is associated with continuous values such as travel times, activity durations, and so forth. The broad idea is to summarize the set of samples ξ used in SAA with a few summary samples. We now describe the MILP-Percentile, where we summarize the sample set ξ using a $(1 - \alpha')$ percentile sample. That is to say, instead of solving the MILP with $|Q|$ samples, we solve it for one sample, in which travel times on edges are obtained by computing $(1 - \alpha')$ percentile duration over all $|Q|$ samples. MILP-Percentile is equivalent to the one provided in Table 7 with the travel times $D^p_{j,i}$ obtained by computing $(1 - \alpha')$ percentile travel times on edge (v_j, v_i) in the sample set ξ. That is to say,

$$D^p_{j,i} = \textbf{Percentile}(\{D^{p,1}_{j,i}, D^{p,2}_{j,i}, \ldots, D^{p,|Q|}_{j,i}\}, (1 - \alpha')) \qquad \forall j, i, p.$$

The key intuition for considering "Percentile" as the summarization criterion is to ensure that the chance constraint (also a percentile) is not violated. Since the percentile is taken at the level of individual edges, it does not theoretically guarantee satisfaction of the percentile constraint at the level of the entire problem. However, as we demonstrate in our experimental results, MILP-Percentile was able to scale to our real-world problem instances and also obtained solutions that were significantly better than the local search mechanism.

5 SOLVING RISK-SENSITIVE SOPS AND DSOPS USING LOCAL SEARCH

We now describe a local search algorithm to solve SOPs and DSOPs.We start from a greedy solution and iteratively make incremental updates to the solution and evaluate the new solution until there are no more updates possible or a maximum number of iterations has been reached. While this is not a principled approximation approach, local search approaches typically converge to good quality solutions efficiently (unlike optimization methods).

[4]We were unable to generate a feasible solution within the threshold time limit of 1,000 seconds.

Table 3. Metrics Corresponding to Different "Neighborhoods" in Variable Neighborhood Search

Metric Name	M1	M2	M3	M4	M5
Explanation	$\frac{\Delta R}{1+\Delta Pr}$	$\frac{1}{1+\Delta Pr}$	ΔR	$\frac{(\Delta R)^2}{1+\Delta Pr}$	$\frac{\Delta R}{\sqrt{1+\Delta Pr}}$

5.1 Solving Risk-Sensitive SOPs

For ease of explanation of the local search algorithm, we first describe a brute force optimal approach for solving SOPs. We consider a depth-first branch-and-bound algorithm, where the root of the search tree is the source vertex and the children of a vertex are all the unvisited vertices minus the sink vertex. The branch of an arbitrary vertex thus represents the path from the source vertex to that vertex. The value of a vertex is the sum of rewards of all vertices along its branch. The algorithm prunes the subtree of a vertex if it fails to satisfy our risk-sensitive criterion. For example, assume that a vertex v_k is on the branch $\pi = \langle v_1, v_2, \ldots, v_k \rangle$, where vertex v_i is on the i-th position on the branch. The algorithm prunes the subtree rooted at vertex v_k if the condition in constraint 2 is not satisfied if one appends the sink vertex to the end of the path. The algorithm returns the vertex with the largest value and the branch of that vertex with the sink vertex appended at the end of the path as the best solution that satisfies the risk-sensitive criterion.

As expected, the branch-and-bound algorithm suffers from scalability issues as the size of the search tree is exponential in the number of vertices in the graph. We thus introduce a local search algorithm that is based on the standard two-phase approach—a construction heuristic to generate an initial solution followed by local improvements on that solution.

5.1.1 Construction Heuristic. The construction heuristic is a greedy insertion algorithm that inserts the best unvisited vertex at the best position in the current path according to a given metric. The algorithm begins with the path that starts at the source vertex and immediately exits at sink vertex, and it terminates when it can no longer insert any vertex at any position without violating the condition in constraint 2.

We use the following metric to evaluate the value of inserting vertex v_i at position p: $\frac{\Delta R}{1+\Delta Pr}$, where ΔR and ΔPr are the gain in reward and probability, respectively, for inserting vertex v_i at position p. Thus, $\Delta R = R_i$, which is the reward of vertex v_i, and $\Delta Pr = \Pr'(a_n \leq H) - \Pr(a_n \leq H)$, where $\Pr'(a_n \leq H)$ and $\Pr(a_n \leq H)$ are the probabilities of arriving at the sink vertex before and after insertion, respectively. Finally, we add 1 to the gain in probabilities such that the denominator is greater than 0.

This metric is motivated by similar metrics in knapsack problems, namely, the utility of an item is the ratio between the reward and size of that item (Nauss 1976). We also tried four other variants of the above metric, namely, (1) $\frac{1}{1+\Delta Pr}$, (2) ΔR, (3) $\frac{(\Delta R)^2}{1+\Delta Pr}$, and (4) $\frac{\Delta R}{\sqrt{1+\Delta Pr}}$, where we ignored the effects of rewards in (1) and probabilities in (2), and we amplified the effects of rewards in (3) and probabilities in (4). However, our chosen metric was shown to outperform these four variants empirically. For easy accessibility, we provide all five metrics in Table 3.

5.1.2 Local Improvements. We use a hybrid approach that consists of a variable neighborhood search combined with simulated annealing to locally improve our initial solution found by the construction heuristic. Algorithm shows the pseudocode of this algorithm. After constructing the initial solution (line 1), the algorithm iteratively runs the following four phases until the maximum number of iterations is reached (line 6):

ALGORITHM 1: Local Search Algorithm

```
   /* Generate Initial Solution                                              */
1  currentPath = ConstructionHeuristic()
   /* Make Local Improvements                                                */
2  bestPath = currentPath
3  numIterNoImprove = 0
4  currentMetric = pick randomly from {M1, M2, M3, M4, M5}
5  T = starting temperature
6  for iterations = 1 to maxIterations do
7      T = T · ΔT
8      Z = numIterNoImprove / (2 · maxIterNoImprove)
       /* Perform 2-Exchange Operation on currentPath                        */
9      currentPath = 2-Exchange(currentPath)
       /* Remove Vertices from currentPath                                   */
10     while currentPath is infeasible OR rand() ≤ Z do
11         remove the second last vertex from currentPath
12     end
       /* Insert Vertices to currentPath                                     */
13     neighborPath = Insert(currentPath, currentMetric)
       /* Update currentPath and bestPath                                    */
14     ΔR = neighborPath.reward − currentPath.reward
15     if ΔR > 0 OR rand() ≤ e^(ΔR/T) then
16         currentPath = neighborPath
17     end
18     if currentPath.reward > bestPath.reward then
19         bestPath = currentPath
20         numIterNoImprove = 0
21     else
22         numIterNoImprove = numIterNoImprove + 1
23         if numIterNoImprove > maxIterNoImprove then
24             currentMetric = pick randomly from [{M1, M2, M3, M4, M5} \ currentMetric]
25             numIterNoImprove = 0
26         end
27     end
28 end
29 return bestPath
```

Phase 1: If the path contains at least two vertices (not including the source and sink vertices), then the algorithm performs a 2-Exchange operation, that is, it randomly swaps two of these vertices (line 9).

Phase 2: If the path is not feasible, that is, it does not satisfy constraint 2, then the algorithm repeatedly removes the second last vertex until the path is feasible. (The algorithm does

not remove the last vertex because it is the sink vertex.) Once the path is feasible, the algorithm repeatedly removes the second to last vertex probabilistically (lines 10–12).[5]

Phase 3: The algorithm repeatedly inserts unvisited vertices greedily similar to the construction heuristic (line 13). The difference here is that the metric used can be one of five different metrics, either the metric chosen for the construction heuristics or one of its four variants described above. The algorithm starts by choosing one of the five metrics randomly (line 4). If there are no improvements in *maxIterNoImprove* iterations, the algorithm chooses a new different metric randomly (lines 24 and 25). These different metrics correspond to the different "neighborhoods" in our variable neighborhood search.

Phase 4: The algorithm then updates the current path to the new neighboring path, which is a result from inserting unvisited vertices in Phase 3, if the new path is a better path or with a probability that depends on the simulated annealing temperature (lines 14–17).

5.1.3 Approximating the Completion Probability of a Path. In a SOP, distribution for the completion probability of a path is equivalent to the sum of the probability distributions for travel times on the edges in the path. For the probability distributions (associated with travel times on individual edges) of interest in this article, namely, normal and gamma distribution, the sum of distributions over the edges in a path remains normal and gamma distributions, respectively. Hence, computing the completion probability for a path is a trivial operation. For a normal distribution,

$$\sum_i \mathcal{N}(\mu_i, \sigma_i^2) = \mathcal{N}\left(\sum_i \mu_i, \sum_i \sigma_i^2\right).$$

Similarly, for a gamma distribution,

$$\sum_i \Gamma(k_i, \theta) = \Gamma\left(\sum_i k_i, \theta\right).$$

For other complex distributions, including the case for gamma distribution, where θ for individual edges is different, we can employ a sampling-based approach. That is to say, we generate a large number of samples from the distributions and check for the completion probability within the deadline by aggregating the result over a large number of samples.

5.2 Solving Risk-Sensitive DSOPs

The local search algorithm described for risk-sensitive SOPs can also be used to solve risk-sensitive DSOPs. The only change necessary is the computation of the completion probability of a path, which we now elaborate.

We describe two ways of approximating the completion probability $\Pr(a_n \leq H)$, which is used in constraint 2 and the construction heuristics. Given the order $\pi = \langle v_1, v_2, \ldots, v_k, v_n \rangle$, we can use the following expression to compute $\Pr(a_n \leq H)$:

$$\begin{aligned}\Pr(a_n \leq H) = \\ &\int_{a_n=0}^{H} \int_{a_k=0}^{a_n} \int_{a_{k-1}=0}^{a_k} \cdots \int_{a_1=0}^{a_2} \\ &\times T_{k,n}^{a_k}(a_n - a_k)\, T_{k-1,k}^{a_{k-1}}(a_k - a_{k-1}) \cdots T_{1,2}^{a_1}(a_2 - a_1) \\ &\times d(a_1)\, d(a_2) \ldots d(a_k)\, d(a_n), \end{aligned} \tag{37}$$

[5]The rand() function returns a random number in [0,1].

where a_n is the arrival time at the sink vertex, and we capture the dependencies on arrival times at each of the vertices by reducing the range of feasible arrival times (for the integrals) based on the previous activities in the order of vertices. Unfortunately, the computation of the expression is expensive since the integrals have to be computed sequentially. To provide an intuition for the time complexity, computing triple integrals takes around 30 minutes with the exponential distribution (the most scalable of all distributions with integration) on our machine using the Matlab software. To address this issue of scalability, we employ two approximation approaches: a sampling-based approach and a matrix-based approach.[6] In the sampling-based approximation, we approximate the completion probability $\Pr(a_n \leq H)$ of a path by randomly sampling the travel time distributions for each edge along the path, and checking if the arrival time a_n at the last vertex exceeds H. Alternatively, in matrix-based approximation, we exploit the fact that the dependencies are primarily due to arrival time at a vertex and not on the entire order of vertices before the current vertex. At a higher level, it implies that the underlying problem is Markovian and, hence, we can decompose the expression of Equation (37). We also make conservative estimates of the probability such that we can provide theoretical guarantees on whether constraint 2 is truly satisfied.

6 EXPERIMENTAL RESULTS

We now show empirical comparisons between linear optimization formulations solved using CPLEX and our local search algorithm for both risk-sensitive SOPs and DSOPs on a synthetic benchmark as well as a real-world theme park dataset. We ran our experiments on a 1.8GHz Intel i5 CPU with 8GB memory.

We used the following parameters for the local search algorithm: $maxIterNoImprove = 50$, $maxIterations = 1{,}500$, $T = 0.1$, and $\Delta T = 0.99$. We divided each travel time distribution to 100 ranges for the matrix-based computations and used 1,000 samples for the sampling-based computations. We tried a large number of combinations of parameters and these settings provided the best tradeoff between runtime and solution quality.

We used the following parameters for our optimization-based MILP-SAA algorithm: The number of samples $|Q| = \langle 25, 30, 35, 40 \rangle$, and the number of sample sets generated for each problem is 15. This corresponds to the number of initial random seeds used to sample the travel time from the gamma distribution.

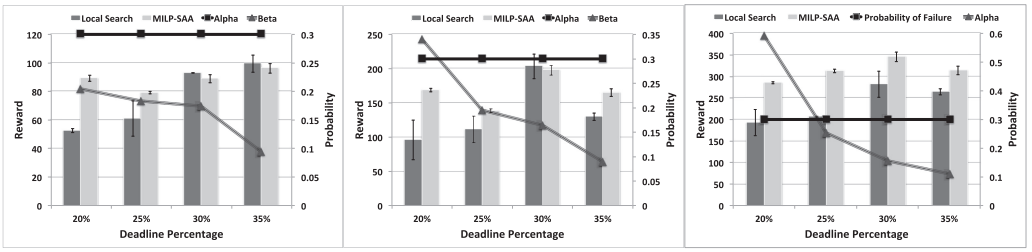
6.1 SOP Results

We measure the performance of our approach with respect to the solution quality and the probability of violating the deadline by varying various problem parameters.

6.1.1 Synthetic Benchmark Set. We use the graph structures introduced by Campbell et al. (2011) and create our synthetic benchmark by varying the following parameters:

- We vary the number of vertices $|V| = \langle 20, 32, 63 \rangle$ and set the reward R_i obtained from visiting a vertex v_i to a random integer between 1 and 10.
- We vary the probability of constraint violation $\alpha = \langle 0.3, 0.25, 0.2, 0.15, 0.11 \rangle$ (see Equation (10)). Corresponding to each setting of α, we use the parameter $\alpha' = \langle 0.2, 0.15, 0.1, 0.05, 0.01 \rangle$ (see Equation (15)).

[6] Details provided in the online appendix.



(a) $|V|$=20 (b) $|V|$=32 (c) $|V|$=63

Fig. 1. Plots showing accumulated reward by our approaches as deadline percentage is varied. In all three graphs: x-axis represents the deadline percentage. Primary y-axis represents the reward accumulated and secondary y-axis represents the probability of failure. The two bars correspond to the reward (primary y-axis) accumulated by Local Search and MILP-SAA methods. The two lines correspond to the desired (Alpha) and actual (Beta) probability of failure (secondary y-axis) values. We also provide error bars representing variance on reward values across multiple runs.

—We employ a gamma distribution $f(x;k,\theta)$ for modeling the travel time of an edge or the random variable $T_{i,j}$, where

$$f(x;k,\theta) = \frac{1}{\theta^k}\frac{1}{\Gamma(k)}x^{k-1}e^{\frac{x}{\theta}}, \quad x > 0,\ k,\theta > 0. \tag{38}$$

We randomly set k for each edge and vary $\theta = \langle 1, 2, 3\rangle$.

—Finally, we vary the deadlines H by setting it to a fraction of the total time required to visit all the vertices. We use the following fractions: $\langle 20\%, 25\%, 30\%, 35\%\rangle$.

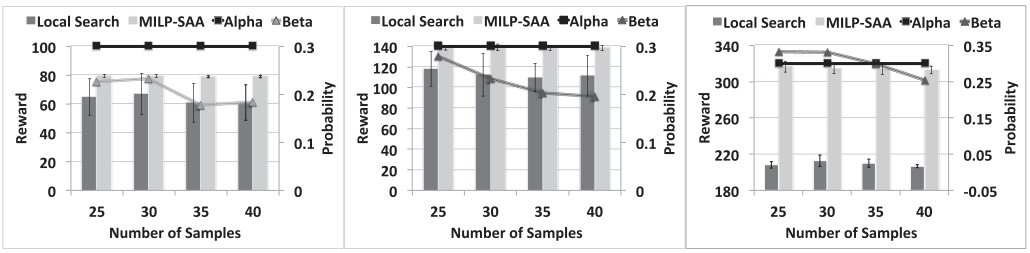
While we obtained results for all combinations of parameters, we only show a representative set of results where we varied only one parameter and set the other parameters to their default values:

$$\theta = 1;\ \alpha = 0.3;\ \alpha' = 0.2;\ H = 25\% \cdot \text{total time};\ |Q| = 40. \tag{39}$$

The local search algorithm always provides a solution with the specified limit α. For the MILP-SAA algorithm, we empirically determine the actual probability of constraint violation for a particular solution π, say β, by generating 1,000 complete samples for edge duration and computing the fraction of samples for which the solution violated the deadline H. Ideally, the probability β should be less than α for the solution to be valid, which is indeed the case in most problem instances.

Runtime. In this article, we do not provide detailed results on runtime because both approaches were able to solve all the problems very quickly. The local search algorithm was able to obtain solutions on the most difficult of problems (i.e., 63 vertices, $H = 20\%, |Q| = 70, \alpha' = 0.01, \theta = 3$) within a few seconds. On the other hand, the MILP-SAA algorithm was able to solve the most difficult problems within 10 minutes.

Deadline H. Figure 1 shows the effect of varying the deadline H on the overall reward for the three graph configurations. The x-axis shows the deadline as a percentage of the total time required to visit all vertices. The primary y-axis (left side) indicates the reward obtained and the secondary y-axis (right side) indicates the probability of violating the deadline. The bars indicate the reward obtained by the local search and MILP-SAA algorithms. In addition, the two lines represent the probability of violating the deadline. The legend "Alpha" denotes the α parameter and "Beta" denotes the empirically computed probability of constraint violation for the MILP-SAA solution using 1,000 samples. We make the following observations:



(a) $|V| = 20$ (b) $|V| = 32$ (c) $|V| = 63$

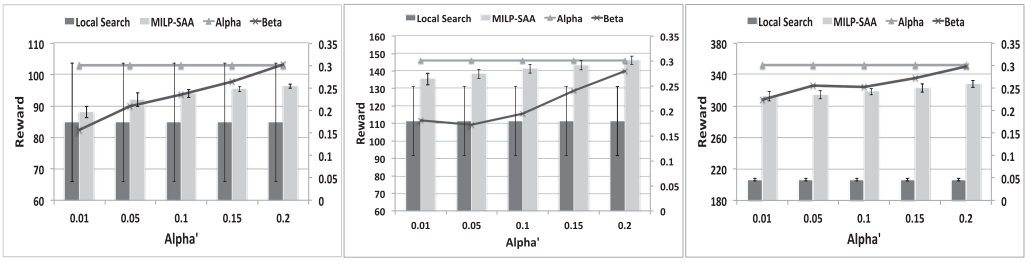
Fig. 2. Plots showing accumulated reward by our approaches as number of samples $|Q|$ is varied.

(1) In general, MILP-SAA finds paths with larger rewards compared to local search. In addition, this improvement in reward is significant in the 63-vertex case (see Figure 1(c)). For example, when the deadline percentage is 25%, the improvement in reward is approximately 100, indicating approximately a 50% improvement over local search. This improvement also implies that MILP-SAA finds paths that traverse an additional 10 vertices in the worst case and 20 vertices in the average case (since rewards are uniformly drawn from the range [1,10]) compared to the paths found by local search.
(2) In most of the cases, the variance in reward of paths found by local search is much higher than those found by MILP-SAA. This observation is important, especially for the few cases where local search finds better paths (on the average) than MILP-SAA. Thus, MILP-SAA is more consistent in finding paths with good quality.
(3) As the deadline percentage increases, the problem becomes less constrained and the difference in the reward of the paths found by the two approaches reduces, which is to be expected.
(4) As the deadline percentage decreases, the problem is more constrained and, hence, the actual probability of the paths found by MILP-SAA violating the deadline (β) increases. Specifically, when the deadline percentage is 20%, the 32- and 63-vertex problems are difficult to solve when MILP-SAA employs 40 samples only. This difficulty is reflected in the β values, which are greater than the $\alpha = 0.3$ threshold. As we show later in this section, this can be addressed by increasing the number of samples (>40) or reducing the α' value employed (<0.1).

Number of Samples $|Q|$. Figure 2 shows the effect of varying the number of SAA samples $|Q|$ on the overall reward for the three graph configurations. We make the following observations:

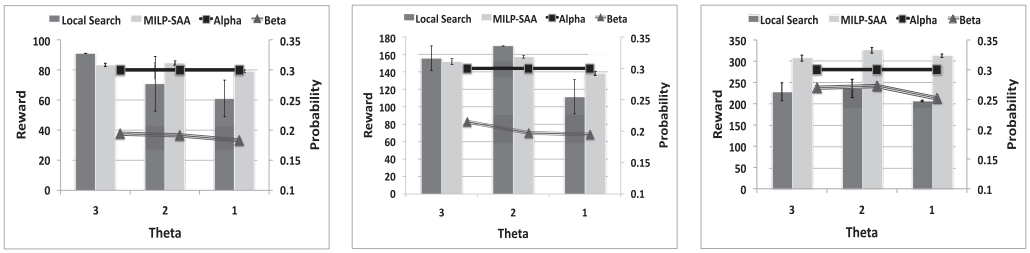
(1) As the number of SAA samples increases, the β value decreases. This behavior is expected as with the increasing number of samples, the SAA approximation becomes tighter.
(2) The reward of the paths found by MILP-SAA remains similar independent of the number of samples. This behavior shows that MILP-SAA can find good paths that minimize the probability of violating the deadline even with increased problem complexity with the higher number of samples.

Deadline Violation Probability α' and Scale Parameter θ. Figures 3 and 4 show the effect of varying the violation probability α' and the scale parameter θ of the gamma distribution, respectively, on the overall reward for the three graph configurations. As expected, as α' increases, the empirical deadline violation probability β increases. However, the increase in reward is minimal for



(a) $|V| = 20$ (b) $|V| = 32$ (c) $|V| = 63$

Fig. 3. Plots showing accumulated reward by our approaches as α' is varied.



(a) $|V| = 20$ (b) $|V| = 32$ (c) $|V| = 63$

Fig. 4. Plots showing accumulated reward by our approaches as scale parameter θ is varied.

increasing α' values. This behavior shows that a smaller value of α' is preferable to limit the probability of violating the deadline.

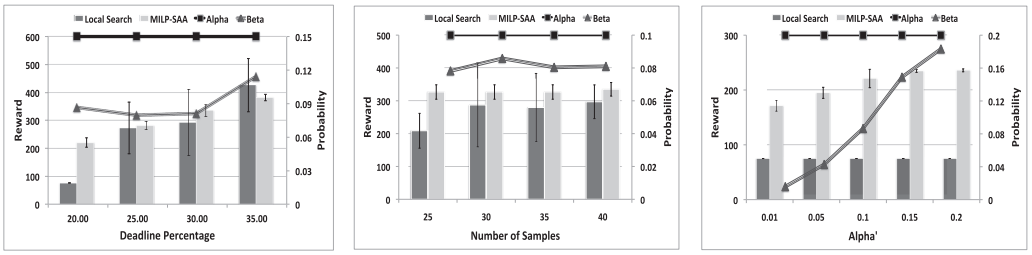
As value of the θ parameter increases, local search on average performs better (albeit with higher standard deviations) than MILP-SAA in smaller problems (20- and 32-vertex problems). However, on the 63-vertex problems, we see that MILP-SAA is significantly better over all values of θ.

6.1.2 Real-World Theme Park Problem. Our real-world example is based on a major theme park in Singapore. This theme park has 21 attractions and, hence, there are 21 vertices in our SOP and DSOP models. Travel time distributions on edges for SOP and DSOP are computed based on real data of two components: (a) 1 year of waiting time data for all attractions (provided at intervals of 15 minutes); and (b) actual observations of time taken to travel between attractions of people at the park. Rewards based on approximate preferences of users for attractions are normalized to be between 0 and 100.

For travel time distribution on edges of SOP, we fit a gamma distribution with scale parameter θ and shape parameter k such that $\mu \approx k\theta$ and $\sigma^2 \approx k\theta^2$, where μ and σ^2 are the mean and variance, respectively, of the data points on sum of travel time and queueing time.

Figure 5 shows the effect of varying the deadline H, number of SAA samples $|Q|$, and deadline violation probability α' on the overall reward for this real-world theme park dataset. We make the following observations:

(1) In all cases except one, MILP-SAA finds paths with larger rewards compared to local search. The exception is when the deadline percentage is 35% or when the problem is only weakly constrained. In some cases, the improvement in reward of MILP-SAA over local search is more than 100%. For example, when the deadline percentage is 20%, the



(a) Deadline H (b) Number of Samples $|Q|$ (c) Violation Probability α'

Fig. 5. Solution quality comparisons on real-world theme park dataset.

improvement in reward is more than 125 and the empirical probability of violating the deadline β is well below the required probability α.

(2) Similar to the synthetic benchmark, the standard deviation in the rewards of paths found by local search is significantly larger than that found by MILP-SAA.

(3) MILP-SAA requires only a small number of samples to obtain sufficiently stable solutions where the empirical probability of violating the deadline β is less than the required probability α.

(4) As the violation probability α' employed by MILP-SAA increases, the overall reward accumulated and probability of violating the deadline increases, which is to be expected.

In summary, in both the synthetic and the real-world dataset, it is clear that the MILP-SAA algorithm outperforms the local search algorithm across a large parameter space. Thus, it should be the preferred algorithm for solving risk-sensitive SOPs.

6.2 DSOP Results

We now describe our results when both the synthetic and the real-world theme park datasets are modeled as risk-sensitive DSOPs. While local search was able to generate results for both synthetic and real-world problems, MILP-Percentile was only able to generate solutions for the real-world theme park problem within the set time limit of 1,000 seconds. MILP-Percentile either ran out of memory or was unable to find solutions within the 1,000 seconds for the synthetic dataset. The key reason is the large number of intervals (= 100) considered for each edge.

6.2.1 Synthetic Dataset Results. We use the same settings as described in Section 6.1.1 except for the following:

—We have a gamma distribution for each time interval for each edge instead of only a single distribution for each edge.

—We bound the possible values of k in the gamma distribution such that the shape of the distributions across time ranges do not vary significantly.

—We vary the deadline $H = \langle 20, 40, 60, 80, 100 \rangle$ instead of the percentage-based settings because computing the maximum completion time was computationally too expensive.

Since MILP-SAA and MILP-Percentile both failed to find feasible solutions within the time limit, we focus on the results of the local search algorithm only. We report only the results for the 32-vertex graph as the trends are similar across all graphs. Table 4 shows our results for the construction heuristic algorithm (labeled CH) and local search algorithm (labeled LS), where we calculate the completion probability of a path (see Equation (2)) using both the matrix-based approach and the sampling-based approach. We report the completion probability of the best path found by the

Table 4. Experimental Results for Synthetic Datasets

(a) Results averaged across all deadlines H and risk parameters α

	Matrix-based Approach				Sampling-based Approach			
	Rewards		Runtimes (s)		Rewards		Runtimes (s)	
	CH	LS	CH	LS	CH	LS	CH	LS
$\theta = 1$	87	88 (0.50)	0.5	568	876	1,033 (18.75)	5.3	2,443
$\theta = 2$	129	134 (1.65)	0.8	987	695	792 (17.03)	2.7	1,477
$\theta = 3$	123	133 (3.97)	0.8	904	533	569 (6.63)	1.3	716

(b) Results averaged across all scale parameters θ and risk parameters α

	Matrix-based Approach				Sampling-based Approach			
	Rewards		Runtimes (s)		Rewards		Runtimes (s)	
	CH	LS	CH	LS	CH	LS	CH	LS
$H = 20$	28	28 (0.00)	0.2	238	193	220 (12.71)	0.2	221
$H = 40$	94	94 (0.11)	0.5	520	432	498 (15.00)	0.8	690
$H = 60$	138	141 (1.43)	0.8	862	657	732 (11.10)	2.0	1,303
$H = 80$	155	160 (2.00)	0.9	1,050	847	952 (11.35)	3.7	1,858
$H = 100$	185	196 (4.12)	1.3	1,485	1,008	1,126 (10.84)	5.7	2,208

(c) Results averaged across all deadlines H and scale parameters θ

	Matrix-based Approach						Sampling-based Approach					
	Rewards		Runtimes (s)				Rewards		Runtimes (s)			
	CH	LS	CH	LS	P_M	P_S	CH	LS	CH	LS	P_M	P_S
$\alpha = 0.1$	1	1 (0.00)	0.1	168	1.00	1.00	507	605 (18.48)	1.7	1,077	0.17	0.90
$\alpha = 0.2$	46	46 (0.00)	0.2	332	0.90	0.99	585	669 (13.98)	2.1	1,186	0.15	0.81
$\alpha = 0.3$	113	119 (3.38)	0.6	768	0.79	0.99	643	711 (10.03)	2.6	1,270	0.13	0.73
$\alpha = 0.4$	194	197 (1.23)	1.1	1,248	0.66	0.97	679	757 (10.81)	2.8	1,376	0.09	0.63
$\alpha = 0.5$	246	256 (3.05)	1.6	1,640	0.54	0.95	725	785 (7.71)	3.3	1,371	0.07	0.55

local search algorithm using the matrix-based approach (labeled P_M) and the sampling-based approach (labeled P_S). We also report the percentage of improvement in the reward of the path found by the local search algorithm compared to the path found by the construction heuristic algorithm (denoted in parentheses beside the local search rewards). We make the following observations:

(1) Table 4(a) shows that for the matrix-based approach, the solution rewards increase between $\theta = 1$ and $\theta = 2$, and remain relatively unchanged for $\theta = 3$. As θ increases, the variance of the gamma distributions increases as well. When $\theta = 1$, only very few ranges have non-zero transition probabilities. As a result, adding an additional edge to a solution can result in a significant decrease in completion probability. With larger values of θ, more ranges have non-zero transition probabilities, but the number of ranges and transition probabilities do not change much with increasing values of θ. Thus, the path length and, consequently, reward and runtime usually increase as θ increases from 1 to 2, but remains relatively unchanged for $\theta = 3$. The runtime depends on the path length because the number of positions to check to find the best position to insert a vertex, which is

done by the construction heuristic algorithm and phase 3 in the local improvement phase, depends on the path length.

On the other hand, for the sampling-based approach, the solution rewards decrease as θ increases. Since the sampling probabilities are relatively accurate representations of the true probabilities, as the variance increases, adding an additional edge to a solution can result in a significant decrease in completion probability. Thus, the path length and, consequently, reward and runtime typically decreases as θ increases.

(2) Table 4(a) also shows that as θ increases, for the sampling-based approach, the improvement of the local search algorithm over the construction heuristic algorithm decreases. The reason is that as the variance of the gamma distributions increases, there is less distinction between the different gamma distributions. Thus, many of the neighboring solutions are very similar to the solution found by the construction heuristic algorithm. For the matrix-based approach, the improvements are all negligible. The path lengths are short (with one to three vertices excluding the source and sink vertices), and, thus, there is not much room for improvement.

(3) Tables 4(b) and 4(c) show that as H or α increases, the solution reward increases for both matrix- and sampling-based approaches, which is to be expected. Similarly, the runtime also increases since the number of positions to check to find the best position to insert a vertex also increases.

(4) Table 4(c) shows that the completion probabilities P_M and P_S are all no less than $1 - \alpha$ for the matrix- and sampling-based approaches, respectively, which is to be expected.

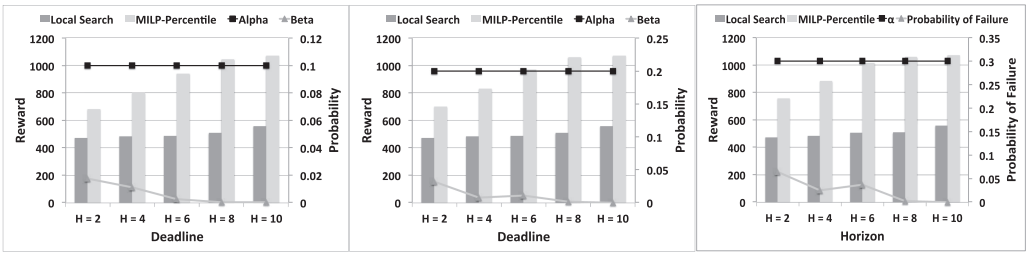
We observe that the problems in this dataset are relatively easy as all gamma distributions have the same scale parameter and their means satisfy the triangle inequality. Thus, we modified the dataset to increase its difficulty in the following ways: (a) we choose the scale parameter θ of the gamma distributions for each edge randomly between 1 and 4 such that not all edges have distributions with the same scale parameter, and (b) we change the shape parameter k of the gamma distributions for some subset of edges such that their means no longer satisfy the triangle inequality. We also performed experiments on this more difficult synthetic datasets. We were able to make the same observations here as with the simpler dataset with the exception that the improvements of the local search algorithm over the construction heuristic algorithm was up to 30% as opposed to 18% earlier.

Overall, using the sampling-based approach, the local search algorithm provides reasonably better solutions compared to the construction heuristic algorithm. However, it is not guaranteed that these solutions are feasible, that is, they satisfy Equation (2). However, the feasibility likelihood increases with the number of samples. Thus, this approach is better suited for users without strict feasibility requirements. On the other hand, solution feasibility is guaranteed for algorithms using the matrix-based approach. Unfortunately, the local search algorithm fails to reasonably improve on the solutions found by the construction heuristic algorithm. Thus, the construction heuristic algorithm using the matrix-based approach is better suited for users with strict feasibility requirements.

6.2.2 Real-World Dataset Results. For the real-world theme park dataset, we also use the same settings as described in Section 6.1.2 except for the following:

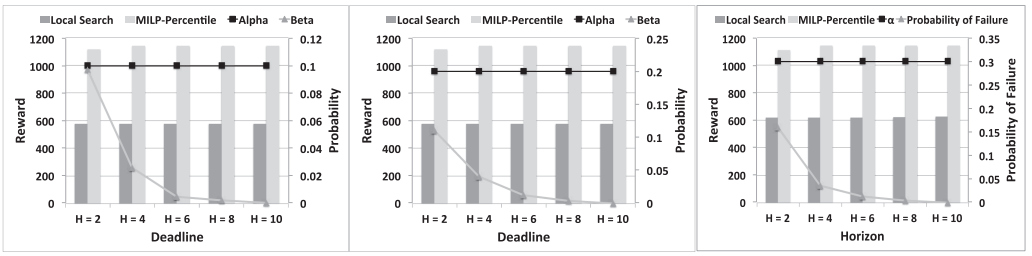
—The total number of intervals in the real-world example is 11, corresponding to the operation hours of the theme park (9:00 AM–8:00 PM).

—We fit gamma distributions to the data collected for each time interval on each edge instead of fitting a single distribution to the data collected for the entire day on each edge.



(a) $\alpha' = 0.1$ (b) $\alpha' = 0.2$ (c) $\alpha' = 0.3$

Fig. 6. Solution quality comparisons on real-world theme park (peak days) dataset.



(a) $\alpha' = 0.1$ (b) $\alpha' = 0.2$ (c) $\alpha' = 0.3$

Fig. 7. Solution quality comparisons on real-world theme park (non-peak days) dataset.

—We vary the deadline $H = \langle 2, 4, 6, 8, 10 \rangle$ (measured in hours) instead of the percentage-based settings because computing the maximum completion time was computationally too expensive.
—We segment our data points into two categories, peak days and non-peak days,[7] and present results for both categories.

Since MILP-SAA failed to find feasible solutions within the time limit, we focus on the results of MILP-Percentile and the local search algorithm only. We consider the following settings for the MILP-Percentile algorithm:

—We vary the number of samples from the following values: $\langle 15, 20, 25, 30, 35 \rangle$.
—We vary α' among the values $\langle 0.1, 0.2, 0.3, 0.4, 0.5 \rangle$.
—The result for each problem is averaged over 15 sample sets, where each sample set contains the "number of samples" mentioned above.
—In computing the probability of failure for a given policy computed by MILP-Percentile, we use 1,000 complete samples.

Figures 6 and 7 show the effect of varying the deadline H and deadline violation probability α' on the overall reward for the peak days and non-peak days datasets, respectively. We make the following observations:

(1) In all cases, MILP-Percentile finds paths with significantly larger rewards than local search.[8] This difference in rewards indicates that the paths found by MILP-Percentile

[7]Peak days are Fridays, Sundays, and Mondays according to our theme park operator.
[8]The results for $\alpha' = 0.4$ and $\alpha' = 0.5$ have similar trends.

Table 5. Solution Quality as Number of Intervals is Reduced (Peak Days)

Number of Intervals	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Hoirizon = 10
11	757.2	882.8	1,014.4	1,058.4	1,072
6	616.4	819.4	1,018.2	1,072	1,072
4	592.4	808.2	1,014.2	1,072	1,072
3	592	806.3	1,025	1,072	1,072
Local Search	474	485	507	549	558

Table 6. Solution Quality as Number of Intervals is Reduced (Non-Peak Days)

Number of Intervals	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Horizon = 10
11	1,109.8	1,141	1,141.0	1,141.0	1,141.0
6	806.8	1,047	1,141	1,141	1,141.0
4	753.8	1,090.4	1,141	1,141	1,141.0
3	727.6	1,044.4	1,131.2	1,141	1,141.0
Local Search	620	620	620	621	624

allow theme park visitors to visit at least four more attractions than the paths found by local search.

(2) The empirical probability of violating the deadline β is less than the required probability α in all cases.

(3) As we increase the required probability α and deadline H, as expected, the reward obtained by local search and MILP-Percentile increases until the maximum possible reward. Additionally, as the deadline H increases, the empirical probability of failure β decreases. The reason is that MILP-Percentile found the same path after a certain deadline. Thus, increasing the deadline only reduces the probability of failure.

(4) Reward obtained on non-peak days is higher than the corresponding case on peak days. This is because non-peak days have smaller waiting times at the attractions, which allows for more attractions to be visited before the deadline.

7 RELATED WORK

There are four threads of research that are of relevance to the research presented in this article:[9]

—**Deterministic, Stochastic, and Dynamic Orienteering Problems:** The difference between our work and existing work in this space (Laporte and Martello 1990; Arkin et al. 1998; Vansteenwegen et al. 2011; Campbell et al. 2011; Li 2012; Ilhan et al. 2008) is that they seek to maximize the expected reward without considering risk sensitivity and also they assume that the traveling time between vertices is time independent.

—**Stochastic and Dynamic Traveling Salesman and Purchaser Problems:** Researchers have not considered stochastic (Seungmo and Ouyang 2011) and dynamic (Angelelli et al. 2011a, 2011b) variants of TPP together. These differences coupled with the lack of a budget in TPP provide distinguishing factors for our contributions.

—**Risk-Sensitive Decision Making:** Our approach of defining a risk-sensitive measure that allows the user to specify a level of risk (failure tolerance) is along the lines of using chance constraints to model and account for different risk preferences. While it has been applied to

[9]Please refer to the detailed related work in the online appendix.

solve planning and scheduling problems (Lehmann 1955; Hanoch and Levy 1969; Markowitz 1952; Miller and Wagner 1965; Prekopa 2003; Beck and Wilson 2007; Chen et al. 2008; Fu et al. 2012), to the best of our knowledge, it has not been applied to solve OPs.

— **Graphical Models and Markov Decision Processes:** SOPs also bear some similarity with Markov Random Fields (MRFs) (Wainwright and Jordan 2008) and Bayesian networks (Russell and Norvig 1995). The goal in these two models is to compute the Maximum A Posteriori (MAP) assignment, which is the most probable assignment to all the random variables of the underlying graph in MRFs (Wainwright and Jordan 2008; Sontag et al. 2011) and Bayesian networks (Park and Darwiche 2003; Huang et al. 2006; Yuan and Hansen 2009). The main difference between MAP assignment problems and SOPs is that MAP assignment problems are *inference* problems, while SOPs are *planning* problems.

While there exists research in Markov Decision Processes (MDPs) (Puterman 1994) that individually addresses continuous state spaces (Marecki and Tambe 2008; Boyan and Littman 2001; Li and Littman 2005), open-loop policies (Weinstein and Littman 2013; Yeoh et al. 2013), constrained MDPs (Altman 1999), and risk-sensitive objectives (Yu et al. 1998; Liu and Koenig 2008; Hou et al. 2014), we are not aware of research that considers all four aspects at the same time.

8 SUMMARY AND FUTURE WORK

OPs and SOPs are rich models that have been shown to be useful in modeling various applications such as a modified traveling salesman problem (Tsiligrides 1984) and logistic applications (Golden et al. 1987). However, they are unable to accurately capture characteristics of our problem of interest, namely, the theme park navigation problem, where patrons need to plan their path in a theme park to visit as many attractions as possible before a given deadline.

In this article, we extend SOPs to DSOPs, where traveling times between attractions differ based on the time of the day. Additionally, we introduce a *risk-sensitive criterion* for SOPs and DSOPs, where the goal is now to find a path that can be completed before the deadline with at least a probability α. We also provide two solution approaches to solve these problems: (1) an optimization-based approach that uses non-linear chance constraints as well as its linearized version via the SAA approach; and (2) a local search-based approach that is based on variable neighborhood search. Experimental results on our synthetic and real-world theme park datasets show that the optimization-based approach consistently finds better solutions than the local search algorithm across a large space of problem parameters for risk-sensitive SOPs. However, the optimization-based approach could not scale to the more complex risk-sensitive DSOPs unlike the local search algorithm.

Future work includes investigating the use of Constrained MDPs (Altman 1999; Dolgov and Durfee 2005). Constrained MDPs cannot be used off the shelf as they enforce all constraints (e.g., the total traveling time of a path is no larger than a threshold) as hard constraints that cannot be violated. We would like to investigate if one can relax that hard constraint in a manner similar to SAA to model and solve risk-sensitive SOPs and DSOPs.

REFERENCES

Eitan Altman. 1999. *Constrained Markov Decision Processes*. Chapman and Hall/CRC.

Enrico Angelelli, Renata Mansini, and Michele Vindigni. 2011a. Look-ahead heuristics for the dynamic traveling purchaser problem. *Computers and Operations Research* 38 (2011), 1867–1876.

Enrico Angelelli, Renata Mansini, and Michele Vindignii. 2011b. Exploring greedy criteria for the dynamic traveling purchaser problem. *Central European Journal of Operations Research* 17 (2011), 141–158.

C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza. 2008. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society* 13, 1 (2008), 49–76.

Esther Arkin, Joseph Mitchell, and Giri Narasimhan. 1998. Resource-constrained geometric network optimization. In *Proceedings of the ACM Symposium on Computational Geometry*. 307–316.

J. Christopher Beck and Nic Wilson. 2007. Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research* 28, 1 (2007), 183–232.

Avrim Blum, Shuchi Chawla, David Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. 2007. Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing* 37, 2 (2007), 653–670.

Justin Boyan and Michael Littman. 2001. Exact solutions to time dependent MDPs. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. 1026–1032.

Ann Campbell, Michel Gendreau, and Barrett Thomas. 2011. The orienteering problem with stochastic travel and service times. *Annals of Operations Research* 186, 1 (2011), 61–81.

I-Ming Chao, Bruce Golden, and Edward Wasil. 1996. Theory and methodology – The team orienteering problem. *European Journal of Operational Research* 88 (1996), 464–474.

Xin Chen, Melvyn Sim, Peng Sun, and Jiawei Zhang. 2008. A linear decision-based approximation approach to stochastic programming. *Operations Research* 56, 2 (2008), 344–357.

Dmitri A. Dolgov and Edmund H. Durfee. 2005. Stationary deterministic policies for constrained MDPs with multiple rewards, costs, and discount factors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1326–1331.

Matteo Fischetti, Juan Jos Salazar Gonzlez, and Paolo Toth. 1998. Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing* 10 (1998), 133–148.

M. Fischetti and P. Toth. 1997. A polyhedral approach to the asymmetric traveling salesman problem. *Management Science* 43 (1997), 1520–1536.

Na Fu, Hoong Chuin Lau, Pradeep Varakantham, and Fei Xiao. 2012. Robust local search for solving RCPSP/max with durational uncertainty. *Journal of Artificial Intelligence Research* 28, 1 (2012), 43–86.

Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. 2014. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics* 20, 3 (2014), 291–328.

Bruce Golden, Larry Levy, and Rakesh Vohra. 1987. The orienteering problem. *Naval Research Logistics* 34, 3 (1987), 307–318.

Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255, 2 (2016), 315–332.

Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. 2012. Approximation algorithms for stochastic orienteering. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1522–1538.

Giora Hanoch and Haim Levy. 1969. The efficiency analysis of choices involving risk. *Review of Economic Studies* 36, 3 (1969), 335–346.

Ping Hou, William Yeoh, and Pradeep Varakantham. 2014. Revisiting risk-sensitive MDPs: New algorithms and results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. 136–144.

Jinbo Huang, Mark Chavira, and Adnan Darwiche. 2006. Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. 143–148.

Taylan Ilhan, Seyed Iravani, and Mark Daskin. 2008. The orienteering problem with stochastic profits. *IIE Transactions* 40 (2008), 406–421.

Gilbert Laporte and Silvano Martello. 1990. The selective traveling salesman problem. *Discrete Applied Mathematics* 26 (1990), 193–207.

Hoong Chuin Lau, William Yeoh, Pradeep Varakantham, Duc Thien Nguyen, and Huaxing Chen. 2012. Dynamic stochastic orienteering problems for risk-aware applications. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 448–458.

Erich Lehmann. 1955. Ordered families of distributions. *Annals of Mathematical Statistics* 26, 3 (1955), 399–419.

Adrienne Leifer and Moshe Rosenwein. 1994. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research* 73 (1994), 517–523.

Jin Li. 2012. Research on team orienteering problem with dynamic travel times. *Journal of Software* 7 (2012), 249–255.

Lihong Li and Michael Littman. 2005. Lazy approximation for solving continuous finite-horizon MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 1175–1180.

Yaxin Liu and Sven Koenig. 2008. An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 453–460.

Janusz Marecki and Milind Tambe. 2008. Towards faster planning with continuous resources in stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 1049–1055.

Harry Markowitz. 1952. Portfolio selection. *Journal of Finance* 7, 1 (1952), 77–91.

Bruce Miller and Harvey Wagner. 1965. Chance constrained programming with joint constraints. *Operations Research* 13, 6 (1965), 930–945.

C. E. Miller, A. T. Tucker, and R. A. Zemlin. 1960. Integer programming formulations and traveling salesman problems. *Journal of ACM* 7 (1960), 326–329.

Robert Nauss. 1976. An efficient algorithm for the 0-1 Knapsack problem. *Management Science* 23, 1 (1976), 27–31.

B. K. Pagnoncelli, S. Ahmed, and A. Shapiro. 2009. Sample average approximation method for chance constrained programming: Theory and applications. *Journal of Optimization Theory and Applications* 142 (2009), 399–416.

James Park and Adnan Darwiche. 2003. Solving MAP exactly using systematic search. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. 459–468.

Andras Prekopa. 2003. Probabilistic programming. In *Stochastic Programming*, Andrzej Ruszczynski and Alexander Shapiro (Eds.). Elsevier.

Martin Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

R. Ramesh, Yong-Seok Yoon, and Mark Karwan. 1992. An optimal algorithm for the orienteering tour problem. *INFORMS Journal on Computing* 4, 2 (1992), 155–165.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ.

Michael Schilde, Karl Doerner, Richard Hartl, and Guenter Kiechle. 2009. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence* 3, 3 (2009), 179–201.

Kang Seungmo and Yanfeng Ouyang. 2011. The traveling purchaser problem with stochastic prices: Exact and approximate algorithms. *European Journal of Operational Research* 209 (2011), 265–272.

David Smith. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. 393–401.

David Sontag, Amir Globerson, and Tommi Jaakkola. 2011. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, Suvrit Sra, Sebastian Nowozin, and Stephen Wright (Eds.). MIT Press.

M. Faith Tasgetiren. 2001. A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research* 4, 2 (2001), 1–26.

T. Tsiligrides. 1984. Heuristic methods applied to orienteering. *Journal of Operational Research Society* 35, 9 (1984), 797–809.

Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209 (2011), 1–10.

Pradeep Varakantham and Akshat Kumar. 2013. Optimization approaches for solving chance constrained stochastic orienteering problems. In *Proceedings of the International Conference on Algorithmic Decision Theory (ADT)*. 387–398.

Martin Wainwright and Michael Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning* 1 (2008), 1–305.

Qiwen Wang, Xiaoyun Sun, B. L. Golden, and Jiyou Jia. 1995. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research* 61 (1995), 111–120.

Ari Weinstein and Michael Littman. 2013. Open-loop planning in large-scale stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 1436–1442.

William Yeoh, Akshat Kumar, and Shlomo Zilberstein. 2013. Automated generation of interaction graphs for value-factored Dec-POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 411–417.

Stella Yu, Yuanlie Lin, and Pingfan Yan. 1998. Optimization models for the first arrival target distribution function in discrete time. *Journal of Mathematical Analysis and Applications* 225 (1998), 193–223.

Changhe Yuan and Eric Hansen. 2009. Efficient computation of jointree bounds for systematic MAP search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 1982–1989.