Solving Risk-Sensitive Stochastic Orienteering Problems using Optimization and Local Search*

Pradeep Varakantham PRADEEPV@SMU.EDU.SG
School of Information Systems
Singapore Management University

Akshat Kumar AKSHATKUMAR@SMU.EDU.SG
School of Information Systems
Singapore Management University

Hoong Chuin Lau HCLAU@SMU.EDU.SG
School of Information Systems
Singapore Management University

William Yeoh WYEOH@CS.NMSU.EDU
Department of Computer Science
New Mexico State University

Abstract

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, Gendreau, and Thomas (2011) extended OPs to Stochastic OPs (SOPs) to represent uncertain travel times. In this paper, we make the following key contributions: (1) we extend SOPs to Dynamic SOPs (DSOPs), which allow for dynamic 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.

1. Introduction

In competitive orienteering sports, each individual starts at a specified control point, tries to visit as many check points 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*

*. This paper combines our work in previous conference papers (Varakantham & Kumar, 2013; Lau, Yeoh, Varakantham, Nguyen, & Chen, 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 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.

(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 a number of applications. Tsiligrides (1984) provided an example application where a traveling salesperson does not have enough time to visit all possible cities. The salesperson knows the number of expected sales in each city and wants to maximize total sales in the time available. A second application was provided by Golden, Levy, and Vohra (1987), where a fleet of trucks has to deliver home fuel to a large number of customers on a daily basis. The primary goal in these problems is to select a subset of customers who urgently require a delivery and the path needs to be calculated so that the customers who need it the most are covered by the end of the day. *Tourist Trip Design Problems* (TTDPs) (Vansteenwegen & Oudheusden, 2007; Archetti, Feillet, Hertz, & Speranza, 2008) refer to the planning problem faced by tourists visiting a city or region. Typically, it is often impossible for tourists to visit everything of interest and, hence, they have to select the most valuable attractions within the time available.

This research is motivated by the problem of providing route guidance at a theme park, similar to the TTDPs described by Vansteenwegen and Oudheusden (2007) and Archetti et al. (2008). Similar route guidance problems can be observed in large facilities such as museums, world expos, and trade shows. OPs assume static and deterministic travel times between vertices and, hence, cannot accurately represent stochastic (due to the presence of queues at attractions in a theme park) and dynamic (due to varying demand for attractions through the day) travel times observed at theme parks. Towards that end, we consider stochastic and dynamic extensions of OPs, namely *Stochastic OPs* (SOPs) (Campbell et al., 2011) and *Dynamic SOPs* (DSOPs). The travel times between vertices in SOPs and DSOPs are random variables that follow a given input distribution. Due to the stochasticity present in SOPs and DSOPs, we cannot guarantee that the deadline is not violated. Therefore, we consider an objective that is sensitive to the risk of violating the deadline. More specifically, we compute the maximum reward path where the probability of violating the deadline is less than a given risk parameter α.

Our key contributions are in providing optimization and local search based approaches for solving risk-sensitive SOPs and DSOPs. With respect to optimization, we provide a principled approximation technique that builds on *Sample Average Approximation* (SAA) (Pagnoncelli, Ahmed, & Shapiro, 2009) to solve risk-sensitive SOPs and DSOPs. With respect to local search, we provide an iterative technique, where an initial solution that does not violate the deadline is incrementally improved until convergence

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 data set of travel times at a popular theme park in Singapore. We observe the following: (1) Our optimization approaches for solving SOP and DSOP 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) our local search approach is able to solve large-scale DSOPs quickly, while our optimization-based approaches can only solve smaller problem instances.

We now provide the outline for the rest of this paper. In Section 2, we provide a background on the formal description of OPs, SOPs, and SAA. We then describe the formal

models for DSOPs along with the risk-sensitive criterion in Section 3. Optimization and local search based approaches for solving SOPs and DSOPs are provided 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 Orienteering Problems (OPs) and Stochastic OPs (SOPs) and then briefly describe the Sample Average Approximation (SAA) method typically employed to solve stochastic optimization problems.

2.1 Orienteering Problems (OPs)

An *Orienteering Problem* (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.

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, 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. Researchers have shown that solving OPs optimally is NP-hard (Golden et al., 1987).

The source and sink vertices in OPs are typically distinct vertices. In the special case where they are the same vertex, the problem is called a *Orienteering Tour Problem* (OTP) (Ramesh, Yoon, & Karwan, 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 exact branch-and-bound methods to solve OPs (Laporte & Martello, 1990) including optimizations with cutting plane methods (Leifer & Rosenwein, 1994; Fischetti, Gonzlez, & Toth, 1998). However, since OPs are NP-hard, exact algorithms often suffer from scalability issues. Thus, constant-factor approximation algorithms (Blum, Chawla, Karger, Lane, Meyerson, & Minkoff, 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, Golden, & Wasil, 1996), neural network-based algorithms (Wang, Sun, Golden, & Jia, 1995) and genetic algorithms (Tasgetiren, 2001). More recently, Schilde, Doerner, Hartl, and Kiechle (2009) developed an ant colony optimization algorithm to solve a bi-objective variant of OPs.

2.2 Stochastic OPs (SOPs)

The assumption of deterministic travel times is not a valid one in many real-world settings. Using our motivating theme park navigation problem as an example, the travel time of patrons depend on numerous factors like fatigue, natural speed of walking, distractions such as food places or travelling with children or senior citizens. Representing such factors accurately and obtaining deterministic travel times is not feasible. Hence, researchers have extended OPs to *Stochastic OPs* (SOPs) (Campbell et al., 2011), where travel times 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. The expected utility of a vertex is the difference between the expected reward and 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 $U(v_i)$ of a vertex v_i is

$$U(v_i) = Pr(a_i \leq H)\, R(v_i) - Pr(a_i > H)\, C(v_i) \tag{1}$$

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

2.3 Sample Average Approximation (SAA)

The *Sample Average Approximation* (SAA) technique is typically used to solve stochastic optimization problems (Pagnoncelli et al., 2009). SOPs are an instance of such stochastic optimization problems, where the risk-sensitive behavior is often encoded in the form of chance constraints. An example of such an optimization problem is given below:

$$\min_{x \in X} \mathbb{E}_P\big[G(x, W)\big] \qquad \text{such that} \tag{2}$$

$$Pr\big(F(x, W) \leq 0\big) \geq 1 - \alpha \tag{3}$$

where X is the feasible parameter space, W is a random vector with probability distribution P and $\alpha \in (0, 1)$. The above stochastic optimization problem is called a chance constrained problem (Pagnoncelli et al., 2009). Notice that the objective function is an expectation due to the unobserved random variable W. Similarly, the constraint function $F(\,)$ is also a random variable due to its dependence on W. The parameter α can be interpreted as the parameter to tune the risk-seeking or risk-averse behavior.

It may seem that such an optimization problem is too unwieldy to solve. Fortunately, a number of techniques do exist that can transform such stochastic optimization problem into a deterministic problem in a principled manner. One such technique is the Sample Average Approximation (SAA) method. Interestingly, the SAA technique can also provide stochastic bounds on the solution quality and, thus, provides a principled approximation.

We now briefly describe SAA and refer readers to (Pagnoncelli et al., 2009) for further details. The main idea behind SAA is to generate N number of samples for the random vector W, where W^i denotes the i-th sample. Based on these samples, we define the approximate probability of constraint violation for a particular point x as follows:

$$\hat{Pr}_N(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\big(F(x, W^i)\big) \tag{4}$$

where

$$\mathbb{I}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t \leq 0 \end{cases} \tag{5}$$

is an indicator-like function that returns 1 if the argument is positive and 0 otherwise. Therefore,

$$Pr\big(F(x, W) \leq 0\big) = 1 - Pr\big(F(x, W) > 0\big) \tag{6}$$

$$\approx 1 - \hat{Pr}_N(x) \tag{7}$$

and the stochastic optimization can be reformulated (approximately) as the following deterministic optimization problem:

$$\min_{x \in X} \frac{1}{N}\sum_{i=1}^{N} G(x, W^i) \qquad \text{such that} \tag{8}$$

$$\hat{Pr}_N(x) \leq \alpha' \tag{9}$$

The parameter α' plays the role of α in the above optimization problem. Typically, we set $\alpha' < \alpha$ to get a feasible solution. Often, the above optimization problem can be formulated as a mixed-integer program and, thus, can be solved using CPLEX. Based on the number of samples and the parameter α', several bounds for the solution quality and feasibility can be derived (Pagnoncelli et al., 2009).

3. Models

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

3.1 Dynamic Stochastic OPs

Stocahstic OPs (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.

To capture dependencies between 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 paper, 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 higher 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 between the level of conservativeness (or risk) against the total utility. More precisely, given a value $0 \leq \alpha \leq 1$, we are interested in obtaining a path, where the probability of reaching the destination vertex v_n after the deadline H is less than α. Or, more precisely,

$$Pr(a_n \geq H) \leq \alpha \tag{10}$$

where a_n is the arrival time at the last vertex of the path. The objective value is therefore inversely proportional to the value of α.

4. Solving Risk-Sensitive SOPs and DSOPs using Optimization

In this section, we employ optimization methods to solve risk-sensitive SOPs and DSOPs. Our key contributions are in formulating both SOPs and DSOPs with the risk-sensitive criterion as linear mathematical programs.

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. 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.

$$\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \tag{11}$$

$$\pi_{i,j} \in \{0,1\} \qquad \forall v_i, v_j \in V \tag{12}$$

$$\sum_j \pi_{j,i} \leq 1 \qquad \forall v_i \in V \tag{13}$$

$$\sum_j \pi_{i,j} \leq 1 \qquad \forall v_i \in V \tag{14}$$

$$\sum_j \pi_{1,j} = 1 \tag{15}$$

$$\sum_j \pi_{j,n} = 1 \tag{16}$$

$$\sum_j \pi_{i,j} - \sum_j \pi_{j,i} = \begin{cases} 1 & \text{if } i = 1 \\ -1 & \text{if } i = n \\ 0 & \text{otherwise} \end{cases} \qquad \forall v_i \in V \tag{17}$$

$$r_i \in [1, n] \qquad \forall v_i \in V \tag{18}$$

$$r_1 = 1 \tag{19}$$

$$r_n = n \tag{20}$$

$$r_i \leq r_j - 1 + (1 - \pi_{i,j})\, M \qquad \forall v_i, v_j \in V \tag{21}$$

$$\Pr\Big(\sum_{i,j} \pi_{i,j}\, T_{i,j} > H\Big) \leq \alpha \tag{22}$$

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

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 13-14 specify that there is a single incoming and outgoing active edge for each vertex. Constraints 15-16 ensure that the path starts at v_1 and ends at v_n. Constraint 17 denotes the flow conservation. We refer to constraints 12-17 as flow preservation constraints and hence forth represented as $\mathbf{F}_\pi \leq 0$.

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 21 models this ranking scheme. The parameter M is a large constant used to maintain the consistency of the constraint. We refer to constraints 18-21 as cycle prevention constraints and referred to as $\mathbf{C}_r \leq 0$.

Constraint 22 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 function $\mathbb{I}(\cdot)$ of Equation 5 using the following linear constraints

$$z^q \geq \frac{\sum_{i,j} \pi_{i,j}\, t^q_{i,j} - H}{M} \qquad \forall q \in Q \tag{23}$$

$$z^q \in \{0, 1\} \qquad \forall q \in Q \tag{24}$$

where we have introduced auxiliary integer variables z^q for each sample q. Using these auxiliary variables, Constraint 9 is represented as

$$\frac{\sum_q z^q}{Q} \leq \alpha' \tag{25}$$

where α' is a parameter that is set by the user and is generally smaller than the parameter α as used in constraint 22. 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 22 with linear constraints 23, 24, and 25. The following theoretical results establish the convergence guarantees for the SAA technique.

Theorem 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 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 for 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.

$$\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \tag{26}$$

$$\mathbf{F}_\pi \leq 0 \tag{27}$$

$$\mathbf{C}_r \leq 0 \tag{28}$$

$$a_i \in [0, M] \tag{29}$$

$$a_i = \sum_j \pi_{j,i}\, a_j + \pi_{j,i}\, T_{j,i}^{a_j} \qquad \forall v_i \in V \setminus \{v_1\} \tag{30}$$

$$a_1 = 0 \tag{31}$$

$$Pr(a_n > H) \leq \alpha \tag{32}$$

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

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 i leads to a different travel time distribution, we assume that there exists 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 30 and 32.

We first address the non-linearity present in constraint 30. For ease of explanation, we provide the linearization by assuming that $T_{j,i}^{a_j}$ is a normal variable and not a random variable. We will later relax this assumption by sampling the random variable, $T_{j,i}^{a_j}$. Within constraint 30, 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:

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

This definition of $b_{j,i}$ can be linearized as follows: :

$$b_{j,i} \leq a_j \qquad \forall (v_j, v_i) \in E \tag{33}$$

$$b_{j,i} \leq \pi_{j,i}\, M \qquad \forall (v_j, v_i) \in E \tag{34}$$

$$a_j \leq b_{j,i} + (1 - \pi_{j,i})\, M \qquad \forall (v_j, v_i) \in E \tag{35}$$

where M is a large number. Intuitively, $b_{j,i}$ is equal to a_j when $\pi_{j,i}$ is 1 and 0 otherwise. 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. We exploit the intervals present in travel time distribution at each vertex in finding the linear equivalent. Let

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

We can then linearize this quadratic term with the following linear constraints:

$$\hat{T}_{j,i} = \sum_m T_{j,i}^p \qquad \forall (v_j, v_i) \in E \tag{36}$$

$$T_{j,i}^p \leq \pi_{j,i}\, D_{j,i}^p \qquad \forall p \in P, (v_j, v_i) \in E \tag{37}$$

$$T_{j,i}^p \leq sl_j^p\, D_{j,i}^p \qquad \forall p \in P, (v_j, v_i) \in E \tag{38}$$

$$D_{j,i}^p - T_{j,i}^p \leq (1 - sl_j^p)\, D_{j,i}^p \qquad \forall p \in P, (v_j, v_i) \in E \tag{39}$$

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 j belongs to interval p, which is achieved through the following linear constraints:

$$1 - sl_j^p \geq \frac{\check{sl}_j^p - a_j}{M} \qquad \forall p \in P, v_j \in V \tag{40}$$

$$1 - sl_j^p \geq \frac{a_j - \hat{sl}_j^p}{M} \qquad \forall p \in P, v_j \in V \tag{41}$$

$$\sum_p sl_j^p = 1 \qquad \forall v_j \in V \tag{42}$$

The overall Mixed Integer Linear Program (MILP) without considering the uncertainty distributions for $T_{j,i}^{a_j}$ is provided in Table 5 in the appendix.

We now relax the assumption that $T_{j,i}^{a_j}$ is a variable and not a random variable. For this, we draw a set of Q samples, $\xi = \{\xi_1, \cdots, \xi_q, \cdots, \xi_Q\}$, where $\xi_q = \{T_{j,i}^{p,q}\}_{v_j \in V, v_i \in V, p \in P}$ 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, $b_{j,i}$, $T_{j,i}^p$ will now have an index associated with the sample q.

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

$$z^q \geq \frac{a_n^q - H}{M} \qquad \forall q \in Q \tag{43}$$

$$z^q \in \{0, 1\} \qquad \forall q \in Q \tag{44}$$

where we have introduced auxiliary integer variables z^q for each sample q. Using these auxiliary variables, the constraint 32 is represented as:

$$\frac{\sum_q z^q}{Q} \leq \alpha' \tag{45}$$

where α' is the parameter that is set by the user and is smaller than the parameter α used in constraint 32. The updated MILP is provided in Table 6 in the 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 paper [1]. 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 etc. 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 time on edges are obtained by computing $(1-\alpha')$ percentile duration over all Q samples. MILP-Percentile is equivalent to the one provided in Table 5 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:

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

The key intuition for considering "Percentile" as the summarisation 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

In this section, we describe a local search algorithm that solves SOPs and DSOPs.

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 exit 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

1. We were unable to generate a solution within the threshold time limit of 1000 seconds.

$\pi = \langle v_1, v_2, \ldots, v_k \rangle$, 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 Equation 10 is not satisfied if one appends the exit vertex to the end of the path. The algorithm returns the vertex with the largest value and the branch of that vertex with the exit 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 greedily 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 exit vertex, and it terminates when it can no longer insert any attraction at any position without violating the condition in Equation 10.

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 is 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 $Pr(a_n \leq H)$ is the probability of arriving at the exit 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) $\frac{(\Delta R)^2}{1+\Delta Pr}$, (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.

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 1 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):

Phase 1: If the path contains at least two vertices (not including the source and sink vertices), then the algorithm performs a 2-Opt 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 Equation 10, 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 exit vertex.) Once the path

Algorithm 1: Local Search Algorithm

```
   /*Generate Initial Solution                                          */
1  currentPath = ConstructionHeuristic()
   /*Make Local Improvements                                            */
2  bestPath = currentPath
3  numIterNoImprove = 0
4  currentMetric = random metric
5  T = starting temperature
6  for iterations = 1 to maxIterations do
7      T = T · ΔT
8      Z = numIterNoImprove / (2 · maxIterNoImprove)
       /* Perform 2-Opt Operation on currentPath                        */
9      currentPath = 2-Opt(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 = new random metric
25             numIterNoImprove = 0
26         end
27     end
28 end
29 return bestPath
```

is feasible, the algorithm repeatedly removes the second last vertex probabilistically (lines 10-12).[2]

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

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

randomly (line 4). If there are no improvements in *maxIterNoImprove* iterations, the algorithm chooses a new different metric randomly (lines 25-26). 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 paper, 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:

$$\sum_i N(\mu_i, \sigma_i^2) = N\Big(\sum_i \mu_i, \sum_i \sigma_i^2\Big)$$

Similarly, for a gamma distribution:

$$\sum_i Gamma(k_i, \theta) = Gamma\Big(\sum_i k_i, \theta\Big)$$

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 Equation 10 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 $Pr(a_n \leq H)$:

$$\begin{aligned} Pr(a_n \leq H) = & \\ & \int_{a_n=0}^{H} \int_{a_k=0}^{a_n} \int_{a_{k-1}=0}^{a_k} \cdots \int_{a_1=0}^{a_2} \\ & T_{k,n}^{a_k}(a_n - a_k)\, T_{k-1,k}^{a_{k-1}}(a_k - a_{k-1}) \cdots T_{1,2}^{a_1}(a_2 - a_1) \\ & d(a_1)\, d(a_2) \ldots d(a_k)\, d(a_n) \end{aligned} \tag{46}$$

where a_n is the arrival time at the exit 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 take around 30 minutes with an exponential distribution (most scalable of all distributions with integration) on our machine using the Matlab software. To address this issue of scalability, we introduce two approximation approaches – a sampling-based approach and a matrix-based approach.

5.2.1 Sampling-based Approximation of the Completion Probability

One can 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. For example, assume that we want to compute $Pr(a_n \leq H)$ for the path $\pi = \langle v_1, v_2, \ldots, v_k, v_n \rangle$. Using the starting time a_1, we generate a travel time sample from the distribution $T_{1,2}^{a_1}$ to represent the travel time from vertex v_1 to vertex v_2, which is also the arrival time a_2 at vertex v_2. We then generate a travel time sample from the distribution $T_{2,3}^{a_2}$ to represent the travel time from vertex v_2 to vertex v_3. The arrival time a_3 at vertex v_3 is thus the sum of both travel times. We continue this process until we generate a travel time sample to represent the travel time from vertex v_k to vertex v_n, and the arrival time a_n is thus the sum of all travel times. We count this entire process as a single sample. We can then approximate

$$Pr(a_n \leq H) \approx \hat{Pr}(a_n \leq H) = \frac{N^+}{N} \tag{47}$$

where N^+ is the number of samples whose arrival time $a_n \leq H$ is no larger than the deadline H and N is the total number of samples. Unfortunately, this approach does not provide any theoretical guarantees on whether Equation 10 is truly satisfied. However, as we increase the number of samples, the approximation for the actual distribution becomes tighter.

5.2.2 Matrix-based Approximation of the Completion Probability

Alternatively, one can 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 46. We also make conservative estimates of the probability such that we can provide theoretical guarantees on whether Equation 10 is truly satisfied.

The key ideas here are (1) to divide the possible arrival times a_i at vertex v_i into a finite number of ranges $\mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,k}$, where $\mathbf{r}_{i,j}$ is the j-th range of arrival time at vertex v_i and (2) to pre-compute for all pairs of vertices v_i and v_j a conservative estimate $\hat{P}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of the probability $P(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of transitioning between ranges of arrival times $\mathbf{r}_{i,p}$ and $\mathbf{r}_{j,q}$. Thus, we can now decompose the expression of Equation 46

to an expression that exploits the Markovian property along with ranges of arrival times:

$$\begin{aligned} Pr(a_n \leq H) = & \\ & \sum_i Pr(a_1 \in \mathbf{r}_{1,i}) \cdot \sum_j Pr(a_2 \in \mathbf{r}_{2,j} | a_1 \in \mathbf{r}_{1,i}) \\ & \cdots \sum_y Pr(a_k \in \mathbf{r}_{k,y} | a_{k-1} \in \mathbf{r}_{k-1,x}) \\ & \cdot \sum_z Pr(a_n \in \mathbf{r}_{n,z} | a_k \in \mathbf{r}_{k,y}) \end{aligned}$$

and conservatively approximate it by

$$\begin{aligned} \hat{Pr}(a_n \leq H) = & \\ & \sum_i \hat{Pr}(a_1 \in \mathbf{r}_{1,i}) \cdot \sum_j \hat{Pr}(a_2 \in \mathbf{r}_{2,j} | a_1 \in \mathbf{r}_{1,i}) \\ & \cdots \sum_y \hat{Pr}(a_k \in \mathbf{r}_{k,y} | a_{k-1} \in \mathbf{r}_{k-1,x}) \\ & \cdot \sum_z \hat{Pr}(a_n \in \mathbf{r}_{n,z} | a_k \in \mathbf{r}_{k,y}) \end{aligned}$$

It is clear that $\hat{Pr}(a_n \leq H) \leq Pr(a_n \leq H)$ is a conservative estimate if $\hat{Pr}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) \leq Pr(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ are all conservative estimates. $\hat{Pr}(a_1 \in \mathbf{r}_{1,i})$ depends on the starting time at vertex v_1, which is provided as an input. We now describe how to compute the other probabilities $\hat{Pr}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$. If the range $\mathbf{r}_{i,p}$ contains only a single point a_i, then

$$\begin{aligned} Pr(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) = & Pr(a_j \in \mathbf{r}_{j,q} | a_i) \\ & = \int_{a_j \in \mathbf{r}_{j,q}} T_{i,j}^{a_i}(a_j - a_i)\ d(a_j) \end{aligned} \tag{48}$$

However, the realization of the random variable a_i only occurs at runtime, and computing the integral in Equation 48 at runtime is expensive. Thus, we would like to compute a conservative estimate $\hat{Pr}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ of probability $Pr(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ for all possible realizations of $a_i \in \mathbf{r}_{i,p}$. We thus compute them as follows:

$$\begin{aligned} \hat{Pr}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) = & \\ & \min_{a_i \in \mathbf{r}_{i,p}} \int_{a_j \in \mathbf{r}_{j,q}} T_{i,j}^{a_i}(a_j - a_i)\ d(a_j) \end{aligned} \tag{49}$$

The value in the integral is the probability of the arrival time a_j to be in the range $\mathbf{r}_{j,q}$ for a given value of a_i. Thus, by taking the minimum of these probabilities over all possible values of a_i in the range $\mathbf{r}_{i,p}$, the conditional probability $\hat{Pr}(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p}) \leq Pr(a_j \in \mathbf{r}_{j,q} | a_i \in \mathbf{r}_{i,p})$ is a conservative estimate of the true probability.

Once all the probabilities are pre-computed, they form transition matrices

$$\mathbf{P}_{i,j} = \begin{pmatrix} \hat{Pr}(a_j \in \mathbf{r}_{j,1}) | a_i \in \mathbf{r}_{i,1}) & \hat{Pr}(a_j \in \mathbf{r}_{j,2}) | a_i \in \mathbf{r}_{i,1}) & \cdots \\ \hat{Pr}(a_j \in \mathbf{r}_{j,1}) | a_i \in \mathbf{r}_{i,2}) & \hat{Pr}(a_j \in \mathbf{r}_{j,2}) | a_i \in \mathbf{r}_{i,2}) & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix} \tag{50}$$

which represent the transition probabilities from vertices v_i to v_j. Finally, to compute $\hat{Pr}(a_n \leq H)$, we compute the multiplication of matrices $\mathbf{P}_1 \cdot \mathbf{P}_{1,2} \cdot \mathbf{P}_{2,3} \cdots \mathbf{P}_{k-1,k} \cdot \mathbf{P}_{k,n}$ and in the resultant matrix, sum up all the probabilities for ranges of arrival times a_n that are less than or equal to the deadline H.

5.2.3 Optimizing the Local Search by Reusing Matrix Computations

One can optimize the local search algorithm described in Algorithm 1 when it is used to solve risk-sensitive DSOPs by reusing matrix computations from previous iterations. Specifically, we re-compute the probability $\hat{Pr}(a_n \leq H)$ of exiting the exit vertex whenever we make a local move during the search, that is, when (a) two vertices are swapped, (b) a vertex is removed, or (c) a vertex is inserted. To compute these probabilities efficiently, we store the results of the products of transition matrices for subsets of vertices. For example, in a path $\pi = \langle v_1, v_2, \ldots, v_i, \ldots, v_j, \ldots \rangle$, if we swap vertices v_i and v_j, then the product of transition matrices for the vertices before v_i, the product of matrices for the vertices between v_{i+1} and v_{j-1}, and the product of matrices for the vertices after v_{j+1} remain unchanged. By storing all of these intermediate results, it is possible to make the computation of probabilities very efficient. However, it requires a significant amount memory for larger problems. In this paper, we store only the products of matrices for the vertices between the source vertex and every subsequent vertex in the path except for the exit vertex. For example, for a path $\pi = \langle v_1, v_2, v_3, v_n \rangle$, we store the product of matrices for vertices v_1 and v_2, which is $\hat{P}(a_2 \leq H)$, and the product of matrices for vertices v_1, v_2 and v_3, which is $\hat{P}(a_3 \leq H)$. While this is not the most efficient approach, it provides a good tradeoff between memory requirement and efficiency.

6. Experimental Results

We now show empirical comparisons between our optimization-based approaches and our local search algorithm for both risk-sensitive SOPs and DSOPs on a synthetic benchmark as well as a real-world theme park data set. 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* = 1500, $T = 0.1$ and $\Delta T = 0.99$. We divided each travel time distribution to 100 ranges for the matrix-based computations and used 1000 samples for the sampling-based computations. We varied 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 22). 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 25).
- 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

$$f(x; k, \theta) = \frac{1}{\theta^k} \frac{1}{\Gamma(k)} x^{k-1} e^{\frac{x}{\theta}}, \;\; x > 0, \; k, \theta > 0 \tag{51}$$

  We randomly set k for each edge and varied $\theta = \langle 1, 2, 3 \rangle$.
- 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: $\langle 20\%, 25\%, 30\%, 35\% \rangle$.

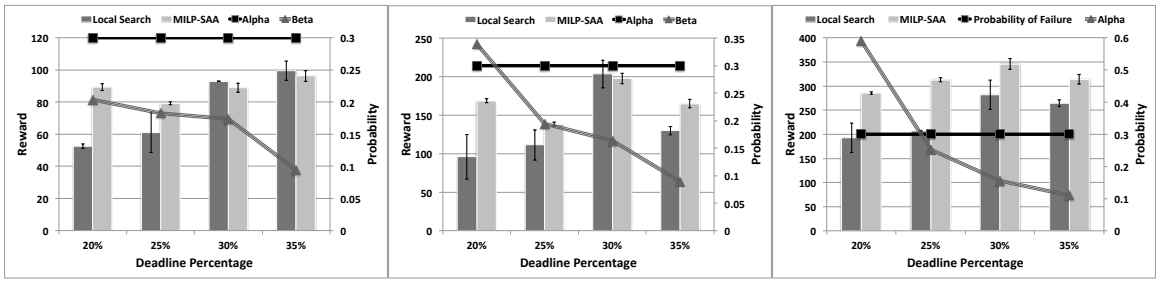
While we obtained results for all combination of parameters, we only show a representative set of results where we varied only one parameter and setting the other parameters to their default values:

$$\theta = 1; \;\; \alpha = 0.3; \;\; \alpha' = 0.1; \;\; H = 25\% \cdot \text{total time}; \;\; Q = 40 \tag{52}$$

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 $\boldsymbol{\pi}$, say β, by generating 1000 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 paper, 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



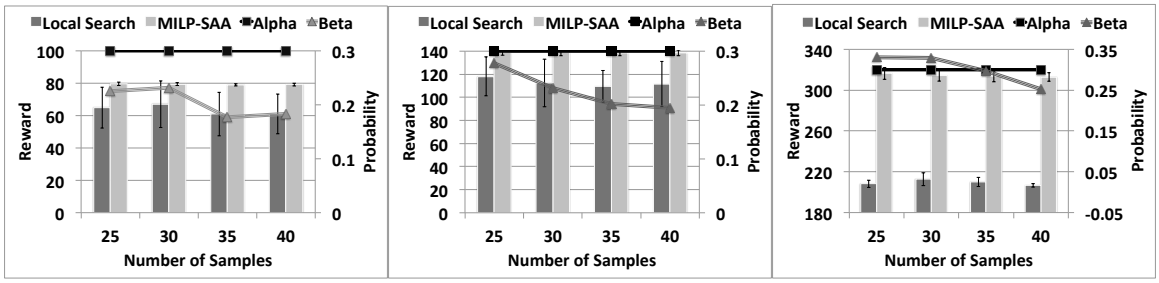
(a) $|V|$=20 (b) $|V|$=32 (c) $|V|$=63

Figure 1: Effect on Reward with Varying Deadlines H

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 1000 samples. We make the following observations:

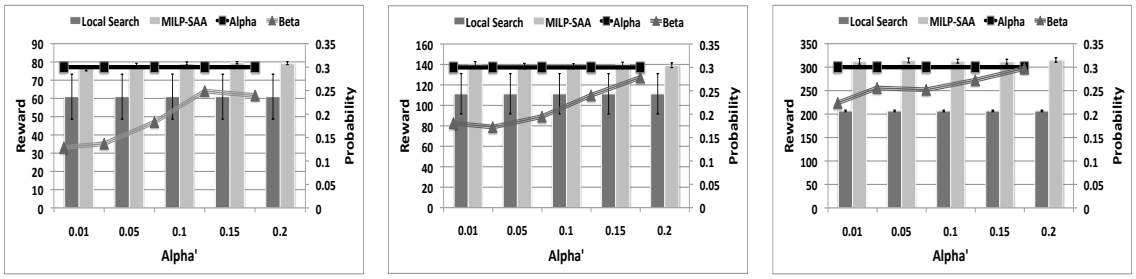
- In general, MILP-SAA finds paths with larger rewards compared to local search. In addition, this improvement in reward is significant in the 63-vertex case (see Figure 1(c)). For example, when the deadline percentage is 25%, the improvement in reward is approximately 100, indicating approximately a 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.
- 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.
- 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.
- 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:



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

Figure 2: Effect on Reward with Varying Number of SAA Samples Q



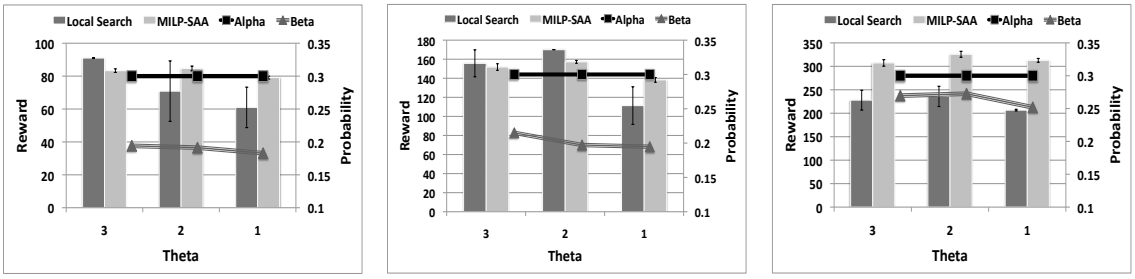
(a) $|V| = 20$ (b) $|V| = 32$ (c) $|V| = 63$

Figure 3: Effect on Reward with Varying Deadline Violation Probability α'

- 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.
- 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 minimizes the probability of violating the deadline even with increased problem complexity with the higher number of samples.

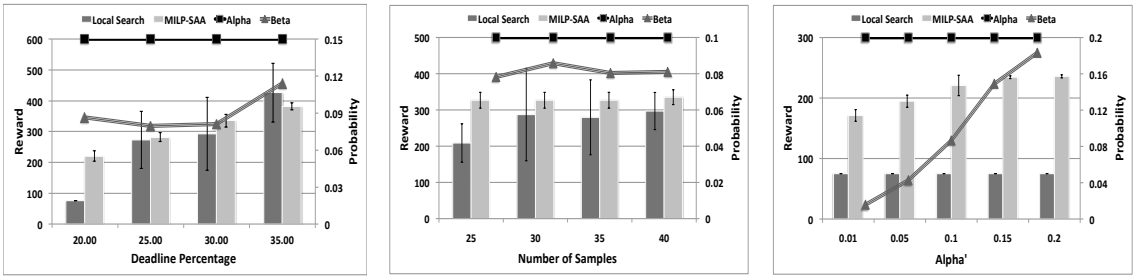
Deadline Violation Probability α' and Scale Parameter θ: Figures 3 and 4 show the effect of varying the violation probability α' and the scale parameter θ of the gamma distribution, respectively, on the overall reward for the three graph configurations.

- As expected, as α' increases, the empirical deadline violation probability β increases. However, the increase in reward is minimal for 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.



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

Figure 4: Effect on Reward with Varying Gamma Distribution Scale Parameter θ



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

Figure 5: Solution Quality Comparisons on Real-World Theme Park Dataset

6.1.2 Real-World Theme Park Problem

For our real-world theme park dataset, we collected the travel times and queuing times at a theme park in Singapore over an entire year. We then chose the 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 is the mean and variance, respectively, of our data points for the sum of travel time and queueing time. We set the reward R_i obtained from visiting a vertex v_i to a random integer between 1 and 100.

Figure 5 show 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:

- 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 improvement in reward is more than 125 and the empirical probability of violating the deadline β is well below the required probability α.
- Similar to the synthetic benchmark, the standard deviation in the rewards of paths found by local search is significantly larger that that found by MILP-SAA.

- 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.
- 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. It should thus 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 1000 seconds. MILP-Percentile either ran out of memory or was unable to find solutions within the 1000 seconds for the synthetic data set. 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 in 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 3 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 10) using both the matrix-based approach (see Equations 49 and 50) and the sampling-based approach (see Equation 47). We report the completion probability of the best path found by the local search algorithm using the matrix-based approach (labeled P_M) and the sampling-based approach (labeled P_S). We also report the 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:

- Table 3(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 computed with Equation 49. As a

(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	1033	(18.75)	5.3	2443
$\theta = 2$	129	134	(1.65)	0.8	987	695	792	(17.03)	2.7	1477
$\theta = 3$	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	1303
$H = 80$	155	160	(2.00)	0.9	1050	847	952	(11.35)	3.7	1858
$H = 100$	185	196	(4.12)	1.3	1485	1008	1126	(10.84)	5.7	2208

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

	Matrix-based Approach							Sampling-based Approach						
	Rewards			Runtimes (s)		P_M	P_S	Rewards			Runtimes (s)		P_M	P_S
	CH	LS		CH	LS			CH	LS		CH	LS		
$\alpha = 0.1$	1	1	(0.00)	0.1	168	1.00	1.00	507	605	(18.48)	1.7	1077	0.17	0.90
$\alpha = 0.2$	46	46	(0.00)	0.2	332	0.90	0.99	585	669	(13.98)	2.1	1186	0.15	0.81
$\alpha = 0.3$	113	119	(3.38)	0.6	768	0.79	0.99	643	711	(10.03)	2.6	1270	0.13	0.73
$\alpha = 0.4$	194	197	(1.23)	1.1	1248	0.66	0.97	679	757	(10.81)	2.8	1376	0.09	0.63
$\alpha = 0.5$	246	256	(3.05)	1.6	1640	0.54	0.95	725	785	(7.71)	3.3	1371	0.07	0.55

Table 3: Experimental Results for Synthetic Datasets

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, typically increases 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.

(a) Results averaged across all risk parameters α

	Matrix-based Approach				Sampling-based Approach			
	Rewards		Runtimes (s)		Rewards		Runtimes (s)	
	CH	LS	CH	LS	CH	LS	CH	LS
$H = 20$	29	29 (0.00)	0.2	262	430	537 (30.35)	0.9	1115
$H = 40$	102	103 (0.43)	0.5	571	864	1052 (22.27)	3.8	4526
$H = 60$	147	160 (6.77)	0.7	915	1156	1394 (20.92)	7.5	8255
$H = 80$	181	183 (0.44)	1.1	1326	1503	1588 (5.73)	13.6	9339
$H = 100$	247	263 (5.11)	1.8	2004	1579	1644 (4.14)	15.3	7716

(b) Results averaged across all deadlines H

	Matrix-based Approach						Sampling-based Approach					
	Rewards		Runtimes (s)		P_M	P_S	Rewards		Runtimes (s)		P_M	P_S
	CH	LS	CH	LS			CH	LS	CH	LS		
$\alpha = 0.1$	16	16 (0.00)	0.1	215	0.98	1.00	1004	1162 (24.68)	6.9	6165	0.00	0.90
$\alpha = 0.2$	83	83 (0.00)	0.4	500	0.87	0.97	1071	1222 (19.87)	8.2	6237	0.00	0.81
$\alpha = 0.3$	141	144 (1.37)	0.7	975	0.78	0.97	1109	1255 (18.38)	8.1	6561	0.00	0.73
$\alpha = 0.4$	204	224 (9.17)	1.2	1448	0.61	0.95	1144	1264 (11.25)	8.5	6054	0.00	0.64
$\alpha = 0.5$	264	272 (2.23)	1.8	1940	0.55	0.87	1205	1311 (9.24)	9.6	5934	0.00	0.56

Table 4: Experimental Results for More Difficult Synthetic Datasets

- Table 3(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 1-3 vertices excluding the source and sink vertices), and thus there is no much room for improvement.

- Tables 3(b) and 3(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.

- Table 3(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 difficulties 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. Table 4 shows our results for this more difficult synthetic dataset. We make the same observations here as in the simpler dataset with the exception that the improvements of the local search algorithm over the construction heuristic algorithm is now 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 10. 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:

- 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.
- 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,[3] and present results for both categories.

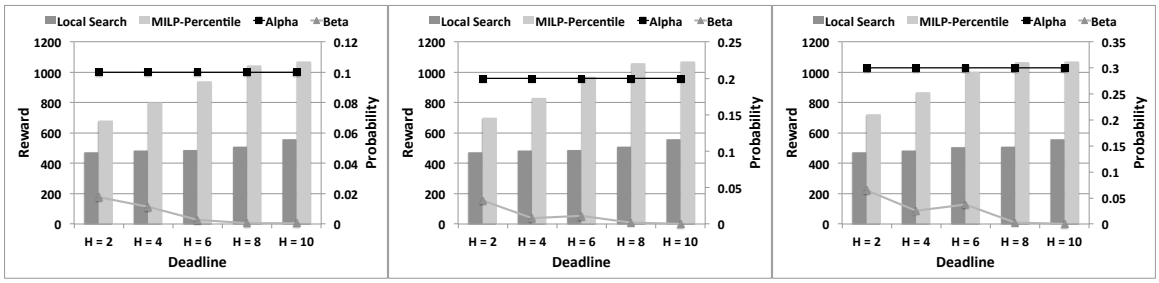
Since MILP-SAA failed to find feasible solutions within the time limit, we focus on the results of MILP-Percentile and the local search algorithm only. We consider the following settings for the MILP-Percentile algorithm:

- 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 1000 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:

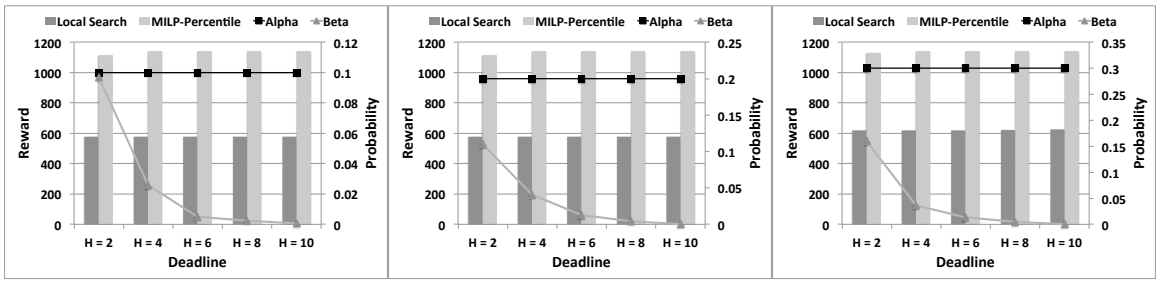
- In all cases, MILP-Percentile finds paths with significantly larger rewards than local search.[4] This difference in rewards indicate that the paths found by MILP-Percentile

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



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

Figure 6: Solution Quality Comparisons on Real-World Theme Park (Peak Days) Dataset



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

Figure 7: Solution Quality Comparisons on Real-World Theme Park (Non-Peak Days) Dataset

allows theme park visitors to visit at least 4 more attractions than the paths found by local search.

- The empirical probability of violating the deadline β is less than the required probability α in all cases.
- 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.
- Reward obtained on non-peak days is typically 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

OPs have a long history and has been known by a variety of other names including the selective TSP (Laporte & Martello, 1990), the maximum collection problem (Kataoka & Morito, 1988) and the bank robber problem (Arkin, Mitchell, & Narasimhan, 1998). Vansteenwegen, Souffriau, and Oudheusden (2011) recently presented a broad overview of the problem, its

variants and associated solution methods. In contrast, to the best of our knowledge, there has not been much work in stochastic variants of OP thus far. Aside from the work on SOPs (Campbell et al., 2011), two closely related problems with stochastic travel times are the time-constrained TSP with stochastic travel and service times (Teng, Ong, & Huang, 2004) and the stochastic selective TSP (Tang & Miller-Hooks, 2005). Lastly, aside from travel time, researchers have also investigated stochasticity in the reward values of vertices (Ilhan, Iravani, & Daskin, 2008). The difference between our work and theirs is that they seek to either maximize the expected reward without considering risk sensitivity or they assume that the traveling time between vertices is time independent.

With respect to modeling and accounting for different risk preferences, there are generally the following three approaches: (1) Stochastic dominance, whose theory was developed in statistics and economics (Lehmann, 1955; Hanoch & Levy, 1969). Stochastic dominance defines partial orders on the space of random variables and allow for pairwise comparison of different random variables. (2) Mean-risk analysis, whose models originate from finance. They include the well known mean-variance optimization model in portfolio optimization, where the variance of the return is used as the risk functional (Markowitz, 1952). (3) Chance constraints or percentile optimization, whose models were initiated and developed in operations research (Miller & Wagner, 1965; Prekopa, 1973). Recently, researchers have provided a thorough overview of the state-of-the-art of the optimization theory with chance constraints (Prekopa, 2003). 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 solve planning and scheduling problems (Beck & Wilson, 2007; Chen, Sim, Sun, & Zhang, 2008; Fu, Lau, Varakantham, & Xiao, 2012), to the best of our knowledge, it has yet to be applied to solve OPs.

SOPs also bear some similarity with Markov random fields (MRFs) (Wainwright & Jordan, 2008) and Bayesian networks (Russell & Norvig, 1995). They are both graphical models, where vertices in a graph correspond to random variables and edges in a graph correspond to potential functions between pairs of random variables. While MRF graphs can be cyclic, Bayesian network graphs are strictly acyclic. 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 & Jordan, 2008; Kumar & Zilberstein, 2010; Sontag, Globerson, & Jaakkola, 2011) and Bayesian networks (Park & Darwiche, 2003; Huang, Chavira, & Darwiche, 2006; Yuan & Hansen, 2009). Thus, the main difference between MAP assignment problems and SOPs is that MAP assignment problems are *inference* problems while SOPs are *planning* problems.

Markov decision processes (MDPs) (Puterman, 1994) are commonly used to model planning problems under transition uncertainty. In fact, MDPs can potentially be used to represent SOPs and DSOPs. However, this mapping is non-trivial. We now provide three key attributes that make it difficult to use MDP solvers to solve risk-sensitive SOPs and DSOPs:

1. **Continuous-Time Distributions:** The presence of continuous time distributions for travel times between vertices in SOPs and DSOPs requires the state space in MDPs to be continuous as well.

2. **Open-Loop Solutions:** Approaches for solving MDPs typically compute closed-loop policies, that is, policies that are dependent on the current state. In a DSOP, this would imply that a solution needs to provide a different destination vertex based on the time of arrival at the current vertex.
3. **Risk-Sensitive Solutions:** The objective in MDPs is typically to maximize the expected reward of a policy and it does not consider risk sensitivity.

While there exists research in MDPs that individually addresses continuous state spaces (Marecki & Tambe, 2008; Boyan & Littman, 2001; Li & Littman, 2005), open-loop policies (Weinstein & Littman, 2013), and risk-sensitive objectives (Yu, Lin, & Yan, 1998; Liu & Koenig, 2008; Hou, Yeoh, & Varakantham, 2014), we are not aware of research that considers all the three aspects at the same time.

8. Summary and Future Work

Orienteering Problems (OPs) and Stochastic OPs (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 paper, we extend SOPs to Dynamic SOPs (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 Sample Average Approximation (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 shows 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 & 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.

Appendix

The overall optimization problems for solving risk-sensitive DSOPs assuming that $T_{j,i}^{a_j}$ is a constant number and a random variable are provided in Tables 5 and 6, respectively.

$$
\begin{aligned}
&\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \\
&\quad \mathbf{F}_\pi \leq 0 \\
&\quad \mathbf{C}_r \leq 0 \\
&\quad a_i \in [0, M] && \forall v_i \in V \\
&\quad a_i = \sum_j \left[b_{j,i} + \hat{T}_{j,i} \right] && \forall v_i \in V \setminus \{v_1\} \\
&\quad a_1 = 0 \\
&\quad a_n \leq H \\
&\quad b_{j,i} \in [0, M] && \forall (v_j, v_i) \in E \\
&\quad b_{j,i} \leq a_j && \forall (v_j, v_i) \in E \\
&\quad b_{j,i} \leq \pi_{j,i} M && \forall (v_j, v_i) \in E \\
&\quad a_j \leq b_{j,i} + \left(1 - \pi_{j,i}\right) M && \forall (v_j, v_i) \in E \\
&\quad \hat{T}_{j,i} \in [0, \max_p D^p_{j,i}] && \forall (v_j, v_i) \in E \\
&\quad T^p_{j,i} \in [0, D^p_{j,i}] && \forall p \in P, (v_j, v_i) \in E \\
&\quad sl^p_i \in \{0, 1\} && \forall p \in P, v_i \in V \\
&\quad \hat{T}_{j,i} = \sum_p T^p_{j,i} && \forall (v_j, v_i) \in E \\
&\quad T^p_{j,i} \leq \pi_{j,i} D^p_{j,i} && \forall p \in P, (v_j, v_i) \in E \\
&\quad T^p_{j,i} \leq sl^p_j D^p_{j,i} && \forall p \in P, (v_j, v_i) \in E \\
&\quad D^p_{j,i} - T^p_{j,i} \leq (1 - sl^p_j) D^p_{j,i} && \forall p \in P, (v_j, v_i) \in E \\
&\quad 1 - sl^p_j \geq \frac{\check{sl}^p_j - a_j}{M} && \forall p \in P, v_j \in V \\
&\quad 1 - sl^p_j \geq \frac{a_j - \hat{sl}^p_j}{M} && \forall p \in P, v_j \in V \\
&\quad \sum_p sl^p_j = 1 && \forall v_j \in V
\end{aligned}
$$

Table 5: Risk-Sensitive DSOP Formulated as a Mixed Integer Linear Program, with $T^{a_j}_{j,i}$ as a Variable

References

Altman, E. (1999). *Constrained Markov Decision Processes*. Stochastic Modeling Series. Chapman and Hall/CRC.

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

Arkin, E., Mitchell, J., & Narasimhan, G. (1998). Resource-constrained geometric network optimization. In *Proceedings of the ACM Symposium on Computational Geometry*, pp. 307–316.

$$
\begin{aligned}
&\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} \\
&\mathbf{F}_\pi \leq 0 \\
&\mathbf{C}_r \leq 0 \\
&a_i^q \in [0, M] && \forall v_i \in V, q \in Q \\
&a_i^q = \sum_j \left[b_{j,i}^q + \hat{T}_{j,i}^q \right] && \forall v_i \in V \setminus \{v_1\}, q \in Q \\
&a_1^q = 0 && \forall q \in Q \\
&z^q \in \{0, 1\} && \forall q \in Q \\
&z^q \geq \frac{a_n^q - H}{H} && \forall q \in Q \\
&\frac{\sum_q z^q}{Q} \leq \alpha' \\
&b_{j,i}^q \in [0, M] && \forall (v_j, v_i) \in E, q \in Q \\
&b_{j,i}^q \leq a_j^q && \forall (v_j, v_i) \in E, q \in Q \\
&b_{j,i}^q \leq \pi_{j,i} M && \forall (v_j, v_i) \in E, q \in Q \\
&a_j^q \leq b_{j,i}^q + \left(1 - \pi_{j,i}\right) M && \forall (v_j, v_i) \in E, q \in Q \\
&\hat{T}_{j,i}^q \in [0, \max_p D_{j,i}^{p,q}] && \forall (v_j, v_i) \in E, q \in Q \\
&T_{j,i}^{p,q} \in [0, D_{j,i}^{p,q}] && \forall p \in P, (v_j, v_i) \in E, q \in Q \\
&sl_i^{p,q} \in \{0, 1\} && \forall p \in P, v_i \in V, q \in Q \\
&\hat{T}_{j,i}^q = \sum_p T_{j,i}^{p,q} && \forall (v_j, v_i) \in E, q \in Q \\
&T_{j,i}^{p,q} \leq \pi_{j,i} D_{j,i}^{p,q} && \forall p \in P, (v_j, v_i) \in E, q \in Q \\
&T_{j,i}^{p,q} \leq sl_j^{p,q} D_{j,i}^{p,q} && \forall p \in P, (v_j, v_i) \in E, q \in Q \\
&D_{j,i}^{p,q} - T_{j,i}^{p,q} \leq (1 - sl_j^{p,q}) D_{j,i}^{p,q} && \forall p \in P, (v_j, v_i) \in E, q \in Q \\
&1 - sl_j^{p,q} \geq \frac{\check{sl}_j^p - a_j^q}{M} && \forall p \in P, v_j \in V, q \in Q \\
&1 - sl_j^{p,q} \geq \frac{a_j^q - \hat{sl}_j^p}{M} && \forall p \in P, v_j \in V, q \in Q \\
&\sum_p sl_j^{p,q} = 1 && \forall v_j \in V, q \in Q
\end{aligned}
$$

Table 6: Risk-Sensitive DSOP with $T_{j,i}^{a_j}$ as a Random Variable Formulated as a Mixed Integer Linear Program

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

Blum, A., Chawla, S., Karger, D., Lane, T., Meyerson, A., & Minkoff, M. (2007). Approximation algorithms for orienteering and discounted-reward TSP. *SIAM Journal on Computing*, *37*(2), 653–670.

Boyan, J., & Littman, M. (2001). Exact solutions to time dependent MDPs. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1026–1032.

Campbell, A., Gendreau, M., & Thomas, B. (2011). The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, *186*(1), 61–81.

Chao, I.-M., Golden, B., & Wasil, E. (1996). Theory and methodology – the team orienteering problem. *European Journal of Operational Research*, *88*, 464–474.

Chen, X., Sim, M., Sun, P., & Zhang, J. (2008). A linear decision-based approximation approach to stochastic programming. *Operations Research*, *56*(2), 344–357.

Dolgov, D. A., & Durfee, E. H. (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)*, pp. 1326–1331.

Fischetti, M., Gonzlez, J. J. S., & Toth, P. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, *10*, 133–148.

Fu, N., Lau, H. C., Varakantham, P., & Xiao, F. (2012). Robust local search for solving RCPSP/max with durational uncertainty. *Journal of Artificial Intelligence Research*, *28*(1), 43–86.

Golden, B., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, *34*(3), 307–318.

Gupta, A., Krishnaswamy, R., Nagarajan, V., & Ravi, R. (2012). Approximation algorithms for stochastic orienteering. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

Hanoch, G., & Levy, H. (1969). The efficiency analysis of choices involving risk. *Review of Economic Studies*, *36*(3), 335–346.

Hou, P., Yeoh, W., & Varakantham, P. (2014). Revisiting risk-sensitive MDPs: New algorithms and results. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Huang, J., Chavira, M., & Darwiche, A. (2006). Solving MAP exactly by searching on compiled arithmetic circuits. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 143–148.

Ilhan, T., Iravani, S., & Daskin, M. (2008). The orienteering problem with stochastic profits. *IIE Transactions*, *40*, 406–421.

Kataoka, S., & Morito, S. (1988). An algorithm for the single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, *31*(4), 515–530.

Kumar, A., & Zilberstein, S. (2010). MAP estimation for graphical models by likelihood maximization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1180–1188.

Laporte, G., & Martello, S. (1990). The selective traveling salesman problem. *Discrete Applied Mathematics*, *26*, 193–207.

Lau, H. C., Yeoh, W., Varakantham, P., Nguyen, D. T., & Chen, H. (2012). Dynamic stochastic orienteering problems for risk-aware applications. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 448–458.

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

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

Li, L., & Littman, M. (2005). Lazy approximation: A new approach for solving continuous finite-horizon mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Liu, Y., & Koenig, S. (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)*, pp. 453–460.

Marecki, J., & Tambe, M. (2008). Towards faster planning with continuous resources in stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

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

Miller, B., & Wagner, H. (1965). Chance constrained programming with joint constraints. *Operations Research*, *13*(6), 930–945.

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

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

Park, J., & Darwiche, A. (2003). Solving MAP exactly using systematic search. In *Proceedings of the Conference on Uncertainty in Articial Intelligence (UAI)*, pp. 459–468.

Prekopa, A. (1973). Contributions to the theory of stochastic programming. *Mathematical Programming*, *4*(1), 202–221.

Prekopa, A. (2003). Probabilistic programming. In Ruszczynski, A., & Shapiro, A. (Eds.), *Stochastic Programming*. Elsevier.

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

Ramesh, R., Yoon, Y.-S., & Karwan, M. (1992). An optimal algorithm for the orienteering tour problem. *INFORMS Journal on Computing*, *4*(2), 155–165.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Schilde, M., Doerner, K., Hartl, R., & Kiechle, G. (2009). Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, *3*(3), 179–201.

Sontag, D., Globerson, A., & Jaakkola, T. (2011). Introduction to dual decomposition for inference. In Sra, S., Nowozin, S., & Wright, S. (Eds.), *Optimization for Machine Learning*. MIT Press.

Tang, H., & Miller-Hooks, E. (2005). Algorithms for a stochastic selective travelling salesperson problem. *Journal of the Operational Research Society*, *56*, 439–452.

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

Teng, S., Ong, H. L., & Huang, H. C. (2004). An integer l-shaped algorithm for the time-constrained traveling salesman problem with stochastic travel times and service times. *Asia-Pacific Journal of Operational Research*, *21*, 241–257.

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

Vansteenwegen, P., & Oudheusden, D. V. (2007). The mobile tourist guide: An OR opportunity. *OR Insights*, *20*(3), 21–27.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*, 1–10.

Varakantham, P., & Kumar, A. (2013). Optimization approaches for solving chance constrained stochastic orienteering problems. In *Proceedings of the International Conference on Algorithmic Decision Theory (ADT)*.

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

Wang, Q., Sun, X., Golden, B. L., & Jia, J. (1995). Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research*, *61*, 111–120.

Weinstein, A., & Littman, M. (2013). Open-loop planning in large-scale stochastic domains. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

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

Yuan, C., & Hansen, E. (2009). Efcient computation of jointree bounds for systematic MAP search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1982–1989.