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

Pradeep Varakantham, Singapore Management University
Akshat Kumar, Singapore Management University
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 paper, 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.

Additional Key Words and Phrases: Orienteering Problems, Risk-Sensitive Optimization, Sample Average Approximation

ACM Reference Format:
Pradeep Varakantham, Akshat Kumar, Hoong Chuin Lau and William Yeoh, 2016. Risk-Sensitive Stochastic Orienteering Problems for Trip Optimization in Urban Environments. *ACM TIST* 9, 4, Article 39 (March 2010), 16 pages.
DOI: 0000001.0000001

1. BACKGROUND

In this section, we briefly describe the *Sample Average Approximation* (SAA) method often employed to solve stochastic optimization problems.

1.1. Sample Average Approximation (SAA)

The *Sample Average Approximation* (SAA) technique is often 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:

$$\min_{x \in X} \mathbb{E}_P\big[G(x,W)\big] \qquad \text{such that} \tag{1}$$

$$Pr\big(F(x,W) \leq 0\big) \geq 1-\alpha \tag{2}$$

Author's addresses: P. Varakantham and A. Kumar and H.C. Lau, School of Information Systems, Singapore Management University; W. Yeoh, Department of Computer Science, New Mexico State University;

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 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 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:

$$\hat{\Pr}_N(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\big(F(x, W^i)\big) \tag{3}$$

where

$$\mathbb{I}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t \le 0 \end{cases} \tag{4}$$

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

$$\Pr\big(F(x, W) \le 0\big) = 1 - \Pr\big(F(x, W) > 0\big) \tag{5}$$

$$\approx 1 - \hat{\Pr}_N(x) \tag{6}$$

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

$$\min_{x\in X} \frac{1}{N}\sum_{i=1}^{N} G(x, W^i) \qquad \text{such that} \tag{7}$$

$$\hat{\Pr}_N(x) \le \alpha' \tag{8}$$

The parameter α' plays the role of α in the above optimization problem. Usually, 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 any available MIP solver . 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].

2. SOLVING RISK-SENSITIVE SOPS AND DSOPS USING LOCAL SEARCH

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

2.1. Solving Risk-Sensitive SOPs

2.1.1. 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:

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

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.

2.2. 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 $\Pr(a_n \leq H)$. Given the order $\pi = \langle v_1, v_2, \dots, v_k, v_n \rangle$, we can use the following expression to compute $\Pr(a_n \leq H)$:

$$\begin{aligned}\Pr(a_n \leq H) = &\\ &\int_{a_n=0}^{H} \int_{a_k=0}^{a_n} \int_{a_{k-1}=0}^{a_k} \cdots \int_{a_1=0}^{a_2} \\ &T_{k,n}^{a_k}(a_n - a_k)\, T_{k-1,k}^{a_{k-1}}(a_k - a_{k-1}) \cdots T_{1,2}^{a_1}(a_2 - a_1) \\ &d(a_1)\, d(a_2) \dots d(a_k)\, d(a_n) \end{aligned} \tag{9}$$

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.

2.2.1. 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 $\pi = \langle v_1, v_2, \dots, v_k, v_n \rangle$. 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

$$\Pr(a_n \leq H) \approx \hat{\Pr}(a_n \leq H) = \frac{N^+}{N} \tag{10}$$

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 completion probability. However, as we increase the number of samples, the approximation for the actual distribution becomes tighter.

2.2.2. 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 9. We also make conservative estimates of the probability such that we can provide theoretical guarantees.

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 $\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ of the probability $\Pr(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ 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 9 to an expression that exploits the Markovian property along with ranges of arrival times:

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

and conservatively approximate it by

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

It is clear that $\hat{\Pr}(a_n \leq H) \leq \Pr(a_n \leq H)$ is a conservative estimate if $\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p}) \leq \Pr(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ 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 $\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$. If the range $\mathbf{r}_{i,p}$ contains only a single point a_i, then

$$\begin{aligned}\Pr(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p}) &= \Pr(a_j \in \mathbf{r}_{j,q} \mid a_i) \\ &= \int_{a_j \in \mathbf{r}_{j,q}} T^{a_i}_{i,j}(a_j - a_i)\ d(a_j) \end{aligned} \tag{11}$$

However, the realization of the random variable a_i only occurs at runtime, and computing the integral in Equation 11 at runtime is expensive. Thus, we would like to compute a conservative estimate $\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ of probability $\Pr(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ for all possible realizations of $a_i \in \mathbf{r}_{i,p}$. We thus compute them as follows:

$$\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p}) = \min_{a_i \in \mathbf{r}_{i,p}} \int_{a_j \in \mathbf{r}_{j,q}} T^{a_i}_{i,j}(a_j - a_i)\ d(a_j) \tag{12}$$

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 $\hat{\Pr}(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p}) \leq \Pr(a_j \in \mathbf{r}_{j,q} \mid a_i \in \mathbf{r}_{i,p})$ is a conservative estimate of the true probability.

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

$$\mathbf{P}_{i,j} = \left(\begin{array}{ccc} \hat{\Pr}(a_j \in \mathbf{r}_{j,1}) \mid a_i \in \mathbf{r}_{i,1}) & \hat{\Pr}(a_j \in \mathbf{r}_{j,2}) \mid a_i \in \mathbf{r}_{i,1}) & \cdots \\ \hat{\Pr}(a_j \in \mathbf{r}_{j,1}) \mid a_i \in \mathbf{r}_{i,2}) & \hat{\Pr}(a_j \in \mathbf{r}_{j,2}) \mid a_i \in \mathbf{r}_{i,2}) & \cdots \\ \cdots & \cdots & \cdots \end{array} \right) \tag{13}$$

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.

2.2.3. Optimizing the Local Search by Reusing Matrix Computations. One can optimize the local search algorithm described in Algorithm 1 (of the main paper), 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 reaching the sink 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 sink 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 $\hat{\Pr}(a_2 \leq H)$, and the product of matrices for vertices v_1, v_2 and v_3, which is $\hat{\Pr}(a_3 \leq H)$. While this is not the most efficient approach, it provides a good tradeoff between memory requirements and efficiency.

3. 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 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 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.

3.1. 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.

3.1.1. 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 $\alpha = \langle 0.3, 0.25, 0.2, 0.15, 0.11 \rangle$. Corresponding to each setting of α, we use the parameter $\alpha' = \langle 0.2, 0.15, 0.1, 0.05, 0.01 \rangle$.
- 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

$$f(x; k, \theta) = \frac{1}{\theta^k} \frac{1}{\Gamma(k)} x^{k-1} e^{\frac{x}{\theta}}, \ \ x > 0, \ k, \theta > 0 \tag{14}$$

  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 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:

$$\theta = 1; \ \alpha = 0.3; \ \alpha' = 0.2; \ H = 25\% \cdot \text{total time}; \ |Q| = 40 \tag{15}$$

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.

Comparison against MILP-Percentile:. In this section, we show the performance comparison between Local Search, MILP-SAA and MILP-Percentile as deadline percentage is varied, while keeping $|Q|$ (number of samples) = 40, $\alpha' = 0.2$, and $\alpha = 0.3$. With respect to runtimes, we made the following observations:

— On 20 vertex problems, irrespective of the deadline percentage, local search takes less than 100 milliseconds and MILP-Percentile finished in less than 150 milliseconds. On the other hand, MILP-SAA took anywhere from 315 to 37000 milliseconds as the deadline percentage is decreased from 35% to 20%.

(a) $|V| = 20$

Deadline Percentage	Local Search Quality	Local Search Std. Dev.	MILP-SAA Quality	MILP-SAA Std. Dev.	MILP-SAA β	MILP-Percentile Quality	MILP-Percentile Std. Dev.	MILP-Percentile β	α
0.20	58.90	17.37	89.90	0.83	0.32	66.50	2.62	0.042	0.30
0.25	76.30	10.14	74.90	1.30	0.27	61.10	2.26	0.019	0.30
0.30	68.40	2.20	85.60	2.15	0.17	79.90	2.77	0.016	0.30
0.35	101.00	0.00	97.80	3.49	0.15	92.30	3.55	0.008	0.30

(b) $|V| = 32$

Deadline Percentage	Local Search Quality	Local Search Std. Dev.	MILP-SAA Quality	MILP-SAA Std. Dev.	MILP-SAA β	MILP-Percentile Quality	MILP-Percentile Std. Dev.	MILP-Percentile β	α
0.20	80.10	2.26	142.30	0.78	0.49	111.40	2.73	0.013	0.30
0.25	155.00	0.00	145.20	2.32	0.35	123.60	3.35	0.006	0.30
0.30	118.10	0.30	147.20	2.82	0.27	139.40	2.80	0.005	0.30
0.35	136.20	8.09	173.30	1.56	0.14	166.70	2.97	0.002	0.30

(c) $|V| = 63$

Deadline Percentage	Local Search Quality	Local Search Std. Dev.	MILP-SAA Quality	MILP-SAA Std. Dev.	MILP-SAA β	MILP-Percentile Quality	MILP-Percentile Std. Dev.	MILP-Percentile β	α
0.20	171.40	18.94	308.60	2.50	0.56	248.90	8.12	0.003	0.30
0.25	207.20	11.57	311.50	5.52	0.35	282.20	11.17	0.001	0.30
0.30	257.90	23.08	355.50	14.82	0.23	343.70	9.58	0.001	0.30
0.35	252.60	7.14	306.60	1.96	0.20	301.90	8.30	0.000	0.30

Table I: Comparison between Local Search, MILP-SAA and MILP-Percentile, where $(1 - \alpha')$ percentile durations are considered for MILP-Percentile.

— On 32 vertex problems, irrespective of the deadline percentage, local search takes less than 150 milliseconds and MILP-Percentile finished in less than 450 milliseconds. On the other hand, MILP-SAA took anywhere from 800 to 39000 milliseconds as the deadline percentage is decreased from 35% to 20%.
— On 63 vertex problems, irrespective of the deadline percentage, local search takes less than 200 milliseconds and MILP-Percentile finished in less than 1200 milliseconds. On the other hand, MILP-SAA took anywhere from 2800 to 82000 milliseconds as the deadline percentage is decreased from 35% to 20%.

Comparison results with respect to solution quality among all three approaches are provided in Table I. Each subtable shows the average and standard deviation of the solution quality for each of the three approaches, the β value the MILP-based approaches, and the α value. Overall, the key observations from the three tables are as follows:

— In most of the cases, MILP-Percentile provides solution qualities that are higher than local search and lower than MILP-SAA.
— MILP-Percentile has much smaller standard deviation than local search and has standard deviation values that are on par with that of MILP-SAA.
— As expected, the probability of failure obtained β by using MILP-Percentile is much lower than α value in all cases.

Overall, MILP-Percentile provides a good tradeoff approach with respect to the combination of runtime and solution quality. In terms of runtime, it is better than MILP-SAA and in terms of solution quality, it is (much) better than local search.

Local Search	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Hoirizon = 10
50, 1500	474	485	507	549	558
500, 3500	490	515	521	568	575

Table II: Solution Quality as Local Search Parameters are Changed on Real-World Theme Park (Peak Days).

Local Search	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Horizon = 10
50, 1500	620	620	620	621	624
500, 3500	641	645	647	647	651

Table III: Solution Quality as Local Search Parameters are Changed on Real-World Theme Park (Non Peak Days).

Local Search	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Hoirizon = 10
2-Exchange	474	485	507	549	558
3-Exchange	474	515	524	562	571

Table IV: Solution Quality as Local Search Parameters are Changed on Real-World Theme Park (Peak Days).

3.2. DSOP Results

Improving Local Search: Unlike in SOPs, local search gets trapped in bad local optima in DSOPs particularly for the real-world problems. In order to address this issue, we have tried the following two options:

- Increased the values of $numIterNoImprove$ and $maxIterations$: For small values of increase, there was little or no improvement. However, when we increased these values to 500 and 3500, there was a small yet noticeable difference. We did not increase it further as the time taken now was on-par with that taken by linear formulations. We show the results in Table II and Table III.

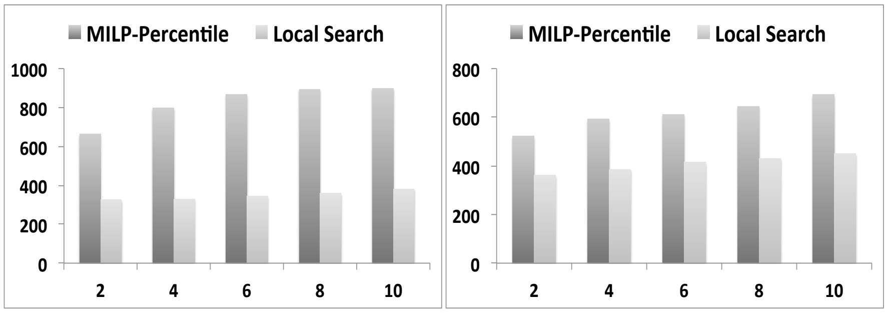
  As can be noted here, the maximum improvement even with such significant changes in parameter values is less than 30 in all cases of peak and non-peak days.
- Considered 3-Exchange operations instead of 2-Exchange operations: We also considered a 3-way exchange of vertices to consider an increased size neighborhood. This, however, resulted in a very similar performance as the increase in $numIterNoImprove$. We show the results in Table IV and Table V.

  The reason for this result is that 3-Exchange can be represented as two 2-Exchange operators. Given sufficient number of chances to find the right sequence of 2-Exchange operations, 3-Exchange based local search is simulated using 2-Exchange based local search.

Reward Functions based on Field Test: Apart from the data provided by the theme park operator, we also did a field test to understand people's movement trajectories at the same theme park. We surveyed and studied trajectories of 50 groups of people including (a) individuals/families without children; and (b) families with children. Each of these two main categories of people have different set of preferences for attractions. For instance, families with children prefer kid and wet rides more than thrill (large roller coasters) rides. Through the use of trajectories and surveys of the groups, we

Local Search	Horizon = 2	Horizon = 4	Horizon = 6	Horizon = 8	Horizon = 10
2-Exchange	620	620	620	621	624
3-Exchange	621	645	645	645	651

Table V: Solution Quality as Local Search Parameters are Changed on Real-World Theme Park (Non Peak Days).



(a) Reward 1 (Families without Children) (b) Reward 2 (Families with Children)

Fig. 1: Solution Quality: Comparisons of Local Search and MILP-Percentile on Two Real Reward Configurations. X-axis represents the time horizon and Y-axis represents the solution quality or reward. We$\alpha = 0.3$ and $\alpha' = 0.2$.

created our reward values that reflect the preferences for attractions.[1] These rewards values are normalized to be between 0 - 100.

We show the results of the performance on the two reward configurations in Figure 1. We considered the default $\alpha = 0.3$ and $\alpha' = 0.2$ setting. MILP-Percentile was cutoff after 1000 seconds and local search was able to finish within 200 seconds on both configurations. As earlier, MILP-Percentile completely outperforms Local Search in both cases in terms of the overall solution reward. This further strengthens our argument that our results are not biased based on reward configurations and hence any preference elicitation approach to eliciting rewards is complimentary.

Approximating DSOP by Merging Consecutive Intervals: Here we evaluate the performance if multiple intervals are merged together on both real and synthetic data sets. In terms of the method, we essentially take the $(1 - \alpha')$ percentile on the expected durations of all the intervals that get merged.[2] Table VI provides the solution quality, as the number of intervals used for the same problem are reduced on synthetic data set. In all cases, the probability of failure was well below the given α when intervals were merged. Here are some of the important observations:

— Since we consider a limit of 1000 seconds to compute the solution, it should be noted that reduction in number of intervals does not imply lower solution quality. In fact, with fewer intervals, CPLEX can potentially explore more of the search space and get

[1]More specifically, reward values are considered as a weighted linear combination of preferences for kid, thrill, dark and wet factors of rides, which are obtained from the initial survey. We then tuned the importance for the four factors (kid, thrill, dark and wet) for the two main categories of people based on the observed trajectories for them on the ground.
[2]It should be noted that when we have one interval for every vertex, DSOP is equivalent to SOP

Number of Intervals	Horizon = 20	Horizon = 40	Horizon = 60	Horizon = 80	Horizon = 100
10	32	65	85	140	164
7	36	79	115	139	156
4	35	96	121	176	155
1	-	-	-	-	-
Local Search	29	83	132	193	240

Table VI: Solution Quality as Number of Intervals is Reduced on Synthetic Problem.

better solutions within the time limit. This is observed with both real and synthetic data sets.

— Even after reducing the number of intervals from 11 to 3, the solution quality obtained by using MILP-Percentile is well above the local search method for the real data sets for all horizon values.
— On the synthetic data, MILP-Percentile was able to outperform local search with number of intervals = 4, a time limit of 1000 seconds and horizon $\leq$ 60. In other cases, local search was able to outperform MILP-Percentile. Local search could be performing better in most cases because the reduction is more significant in the synthetic case (from 100 to 10, 7, 4 and 1).
— On the synthetic data, MILP-Percentile could not find a feasible solution when we merged all intervals into one interval.

Overall, MILP-Percentile seems to provide better solution than local search if the reduction in intervals is not significant (as observed in the real-world example). On the other hand, when the reduction is significant (as in synthetic case), local search seems to perform better especially for higher horizons.

4. RELATED WORK

There are four threads of research that are of relevance to the research presented in this paper.

4.1. Deterministic, Stochastic, and Dynamic Orienteering Problems

OPs have a long history and have been known by a variety of other names including the selective TSP [Laporte and Martello 1990], the maximum collection problem [Kataoka and Morito 1988] and the bank robber problem [Arkin et al. 1998]. [Vansteenwegen et al. 2011] recently presented a broad overview of the problem, its variants and associated solution methods. Unfortunately, in many problems the assumptions of deterministic and time independent travel times are not reasonable (as motivated in the introduction) and hence the focus of this paper on those issues.

Stochasticity and time dependent travel times have received less attention in OPs. 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 et al. 2004] and the stochastic selective TSP [Tang and Miller-Hooks 2005]. There has also been work on dynamic or time dependent travel times in the context of team orienteering problems by [Li 2012]. Our work extends on this line of existing work in OPs to account for stochastic and time dependent (dynamic) travel times.

Lastly, aside from travel time, researchers have also investigated stochasticity in the reward values of vertices [Ilhan et al. 2008]. The difference between our work and theirs 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.

4.2. Stochastic and Dynamic Traveling Salesman and Purchaser Problems

Recently, there has been work on considering stochasticity in traveling salesman problems [Cheong and White 2012; A. Toriello and Poremba 2014]. While this line of work is similar, the following key characteristics are different from the research presented in this paper:

— Dynamism (time dependence) in travel times is not considered.
— There is no budget on travel time in standard traveling salesman problems.
— Constraints on risk arising due to the stochasticity (and dynamism) are not considered.

An experimental result highlighted in this line of existing work that also resonates with our experimental results is that with stochasticity (not even including dynamism), the size of problems that can be solved are much smaller (10-45 vertices) than their deterministic counterparts.

Given a list of marketplaces, the cost of traveling between different marketplaces, and a list of available goods together with the price of each such good at each marketplace, the Traveling Purchaser Problem (TPP) is to find for a given list of articles, the route with the minimum combined cost of purchasing and traveling. The traveling salesman problem is a special case of TPP. Researchers have not considered stochastic [Seungmo and Ouyang 2011] and dynamic [Angelelli et al. 2011a; Angelelli et al. 2011b] variants of TPP together. These differences coupled with the lack of a budget in TPP provide distinguishing factors for our contributions.

4.3. Risk-Sensitive Decision Making

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 and 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 and 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 and Wilson 2007; Chen et al. 2008; Fu et al. 2012], to the best of our knowledge, it has yet to be applied to solve OPs.

4.4. 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]. 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 and Jordan 2008; Kumar and Zilberstein 2010; Sontag et al. 2011] and Bayesian networks [Park and Darwiche 2003; Huang et al. 2006; Yuan and 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 four key attributes that make it difficult to use MDP solvers to solve risk-sensitive SOPs and DSOPs:

1. **Semi Continuous-Time Distributions:** The presence of semi continuous time distributions for travel times between vertices in SOPs and DSOPs requires the state space in MDPs to be continuous as well.
2. **Budget on Time:** Because of the time budget, we have to consider the constrained MDPs framework and not the basic MDP framework.
3. **Open-Loop Solutions:** Approaches for solving MDPs often 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.
4. **Risk-Sensitive Solutions:** The objective in MDPs is usually 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 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.

In addition, for an MDP corresponding to (D)SOP, the state space would have to contain all the vertices that have been visited previously along with time elapsed. In order to make a decision about the next vertex to visit, we have to know all the vertices that have been visited previously. This is because of time dependence (in DSOP) and time budget (in SOP and DSOP). Because of this exponential complexity of state space (all combinations of vertices) and time budget, the computational complexity will be significant and scalability will be severely limited.

Appendices

A. LINEAR OPTIMIZATION FORMULATIONS FOR DSOP

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 VII and VIII, respectively.

B. TIME WINDOWS AND PRECEDENCE CONSTRAINTS

Time windows and precedence constraints are interesting and important aspects in the domains of interest in this paper. We can account for both of them in our MILP formulation with minimal changes.

For a precedence constraint – vertex i should be visited before vertex j – to be considered, all we need to have as constraint is that arrival time at vertex i is lower than arrival time at vertex j. It is represented as follows in our MILP formulations:

— MILP-Percentile:

$$a_i \leq a_j$$

$$
\begin{array}{ll}
\max\limits_{\boldsymbol{\pi}} \sum\limits_{i,j} \pi_{i,j} R_i \qquad \text{such that} & \\
\mathbf{F}_\pi \leq 0 & \\
\mathbf{C}_r \leq 0 & \\
a_i = \sum\limits_{j} \left[b_{j,i} + \hat{T}_{j,i} \right] & \forall v_i \in V \setminus \{v_1\} \\
a_1 = 0 & \\
a_n \leq H & \\
b_{j,i} \leq a_j & \forall (v_j, v_i) \in E \\
b_{j,i} \leq \pi_{j,i}\, M & \forall (v_j, v_i) \in E \\
a_j \leq b_{j,i} + \big(1 - \pi_{j,i}\big)\, M & \forall (v_j, v_i) \in E \\
\hat{T}_{j,i} = \sum\limits_{p} T^p_{j,i} & \forall (v_j, v_i) \in E \\
T^p_{j,i} \leq \pi_{j,i}\, D^p_{j,i} & \forall p \in P, (v_j, v_i) \in E \\
T^p_{j,i} \leq sl^p_j\, D^p_{j,i} & \forall p \in P, (v_j, v_i) \in E \\
D^p_{j,i} - T^p_{j,i} \leq (2 - sl^p_j - \pi_{j,i})\, D^p_{j,i} & \forall p \in P, (v_j, v_i) \in E \\
1 - sl^p_j \geq \dfrac{\check{sl}^p_j - a_j}{M} & \forall p \in P, v_j \in V \\
1 - sl^p_j \geq \dfrac{a_j - \hat{sl}^p_j}{M} & \forall p \in P, v_j \in V \\
\sum\limits_{p} sl^p_j = 1 & \forall v_j \in V \\
a_i \in [0, M] & \forall v_i \in V \\
b_{j,i} \in [0, M] & \forall (v_j, v_i) \in E \\
sl^p_i \in \{0, 1\} & \forall p \in P, v_i \in V \\
\hat{T}_{j,i} \in [0, \max\limits_{p} D^p_{j,i}] & \forall (v_j, v_i) \in E \\
T^p_{j,i} \in [0, D^p_{j,i}] & \forall p \in P, (v_j, v_i) \in E \\
\pi_{j,i} = [0, 1] & \forall v_i, v_j
\end{array}
$$

Table VII: Risk-Sensitive DSOP Formulated as a Mixed Integer Linear Program, with $T^{a_j}_{j,i}$ is a normal variable.

— MILP-SAA:

$$a^q_i \leq a^q_j + z^q \cdot M$$

Similarly for a time window $[t^{min}_i, t^{max}_i]$ for an attraction i, we have the following additional constraints:

— MILP-Percentile:

$$a_i \leq t^{max}_i$$

$$a_i \geq t^{min}_i$$

— MILP-SAA:

$$a^q_i \leq t^{max}_i + z^q \cdot M$$

$$
\begin{array}{ll}
\max_{\boldsymbol{\pi}} \sum_{i,j} \pi_{i,j} R_i \qquad \text{such that} & \\
\mathbf{F}_\pi \leq 0 & \\
\mathbf{C}_r \leq 0 & \\
a_i^q = \sum_j \left[b_{j,i}^q + \hat{T}_{j,i}^q \right] & \forall v_i \in V \setminus \{v_1\}, q \in Q \\
a_1^q = 0 & \forall q \in Q \\
z^q \geq \frac{a_n^q - H}{H} & \forall q \in Q \\
\frac{\sum_q z^q}{|Q|} \leq \alpha' & \\
b_{j,i}^q \leq a_j^q & \forall (v_j, v_i) \in E, q \in Q \\
b_{j,i}^q \leq \pi_{j,i} M & \forall (v_j, v_i) \in E, q \in Q \\
a_j^q \leq b_{j,i}^q + \left(1 - \pi_{j,i}\right) M & \forall (v_j, v_i) \in E, q \in Q \\
\hat{T}_{j,i}^q = \sum_p T_{j,i}^{p,q} & \forall (v_j, v_i) \in E, q \in Q \\
T_{j,i}^{p,q} \leq \pi_{j,i} D_{j,i}^{p,q} & \forall p \in P, (v_j, v_i) \in E, q \in Q \\
T_{j,i}^{p,q} \leq sl_j^{p,q} D_{j,i}^{p,q} & \forall p \in P, (v_j, v_i) \in E, q \in Q \\
D_{j,i}^{p,q} - T_{j,i}^{p,q} \leq (2 - sl_j^{p,q} - \pi_{j,i}) D_{j,i}^{p,q} & \forall p \in P, (v_j, v_i) \in E, q \in Q \\
1 - sl_j^{p,q} \geq \frac{\check{sl}_j^p - a_j^q}{M} & \forall p \in P, v_j \in V, q \in Q \\
1 - sl_j^{p,q} \geq \frac{a_j^q - \hat{sl}_j^p}{M} & \forall p \in P, v_j \in V, q \in Q \\
\sum_p sl_j^{p,q} = 1 & \forall v_j \in V, q \in Q \\
a_i^q \in [0, M] & \forall v_i \in V, q \in Q \\
b_{j,i}^q \in [0, M] & \forall (v_j, v_i) \in E, q \in Q \\
\hat{T}_{j,i}^q \in [0, \max_p D_{j,i}^{p,q}] & \forall (v_j, v_i) \in E, q \in Q \\
T_{j,i}^{p,q} \in [0, D_{j,i}^{p,q}] & \forall p \in P, (v_j, v_i) \in E, q \in Q \\
sl_i^{p,q} \in \{0, 1\} & \forall p \in P, v_i \in V, q \in Q \\
z^q \in \{0, 1\} & \forall q \in Q \\
\pi_{j,i} = [0, 1] & \forall v_i, v_j
\end{array}
$$

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

$$a_i^q \geq t^{min} - z^q \cdot M$$

Note the use of z^q in MILP-SAA. If these constraints are not satisfied, they will add to the overall set of violations.

For time windows, since our travel times are derived from past wait time data, our current approach already captures such time windows. In times when there is no show, there would be no edge from any other vertex to the vertex where a show is being held.

When there is no edge, the travel time there is infinite. Only before the show time, there would be an edge and hence travel time data available from other vertices.

Since time windows and precedence constraints are not the main focus of the paper, we do not explore time windows and precedence constraints to great depth in the paper.

REFERENCES

W.B. Haskell A. Toriello and M. Poremba. 2014. A Dynamic Traveling Salesman Problem with Stochastic Arc Costs. *Operations Research* 62 (2014), 1107–1125.

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.

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.

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.

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.

T. Cheong and C.C. White. 2012. Dynamic Traveling Salesman Problem: Value of Real-Time Traffic Information. *IEEE Transactions on Intelligent Transportation Systems* 13 (2012), 619–630.

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.

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.

S. Kataoka and S. Morito. 1988. An Algorithm for the Single Constraint Maximum Collection Problem. *Journal of the Operations Research Society of Japan* 31, 4 (1988), 515–530.

Akshat Kumar and Shlomo Zilberstein. 2010. MAP Estimation for Graphical Models by Likelihood Maximization. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*. 1180–1188.

Gilbert Laporte and Silvano Martello. 1990. The Selective Traveling Salesman Problem. *Discrete Applied Mathematics* 26 (1990), 193–207.

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

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.

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 Articial Intelligence (UAI)*. 459–468.

Andras Prekopa. 1973. Contributions to the Theory of Stochastic Programming. *Mathematical Programming* 4, 1 (1973), 202–221.

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.

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

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 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.

Hao Tang and Elise Miller-Hooks. 2005. Algorithms for a Stochastic Selective Travelling Salesperson Problem. *Journal of the Operational Research Society* 56 (2005), 439–452.

Suyan Teng, Hoon Liong Ong, and Huei Chuen Huang. 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 (2004), 241–257.

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

Martin Wainwright and Michael Jordan. 2008. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning* 1 (2008), 1–305.

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. *J. Math. Anal. Appl.* 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.