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

RESPONSES TO REVIEWER COMMENTS

*We thank all the reviewers for their very constructive feedback. It has helped us improve the paper considerably. We have addressed all the comments to the best of our understanding of the comments. In order for quick referencing, here are the **10 major** changes we have made to the paper:*

1. ***Motivating Domains:** We have modified introduction (Section 1) to provide three categories of motivating domains*
2. ***Rewards**: Based on field test data at the real theme park, we have now generated reward functions for two categories of visitors. We have used survey data from before the field test and movement trajectory data during the field test of 50 visitors to generate reward values for attractions in a principled manner. We have also performed experiments with our approaches on these reward values for 2 major categories of visitors. Details are present in experimental results section (Sections 6.1.2 and 6.2).*
3. ***Merging of intervals in DSOP:** In Section 6.2, We now provide a detailed performance comparison as consecutive intervals are merged in DSOP.*
4. ***Time windows and precedence constraints:** Appendix B provides description of how time windows and precedence constraints can be trivially accounted for in our DSOP formulation.*
5. ***2-OPT and 3-OPT Experiments:** New results for updated local search on DSOP problems are described in Section 6.*
6. ***Comparison of MILP-SAA and MILP-Percentile on SOP Problems:** Table 4 provides these results.*
7. ***Related Work**: Section 7 contains the revamped related work section. We address all of the comments with respect to existing work in this section and also add comparisons to related work in the last year.*
8. ***Correctness of DSOP formulation:** We have included Proposition 1 to address validity of the linear constraints employed in DSOP.*
9. ***MTZ vs Separation Cuts:** We provide a detailed response here and also provide discussion about the same in the paper.*
10. ***Improving efficiency of solving DSOP formulation:** We have described in detail all the ways in which we have tried to improve the efficiency of solving our linear optimization formulations.*

The text of the original reviews are in regular unitalicized text and our responses are in italicized and indented text.

Reviewer 1

The paper proposes approximate algorithms for risk-sensitive expected utility optimization in the context of the stochastic orienteering problem (SOP) and its variant, dynamic SOP (DSOP). The use of this risk-sensitive criterion for SOPs, and the DSOP class itself, are also introduced in this paper. The algorithms the paper proposes are based on MILP and local search, and are suboptimal. Their performance is tested on a variety of synthetic scenarios and a real-world problem based on the statistics of theme park visits.

Technically, the paper is OK. A few glitches, mostly minor, are listed below. The algorithms don't have theoretical properties to speak of. This would be fine but for the paper's main drawback in my opinion ---

a weak motivation of the models that these algorithms target.

Specifically, the main issue with the paper is that, as it stands, it proposes quite complicated algorithms that make many approximations and yet aren't demonstrated to have a convincing use case. T**he paper gives very few examples of real problems for which SOP would be necessary, and only one for which DSOP would be necessary, the theme park problem**. Moreover, it doesn't use the algorithms it proposes to convincingly solve even a single instance of these real problems, testing them primarily on synthetic data. The theme park problem, the only realistic one that the paper takes a stab at, isn't modeled in the paper very well. **Namely, the rewards, i.e., the visitors' utilities, are generated randomly in the experiments.** Thus, preference elicitation, a key component of applying the paper's algorithms to the actual theme park problem, has been entirely sidestepped. In the meantime, this component seems very difficult to implement in a theme-park setting, and its implementation critically influences the proposed algorithms' ability to provide a satisfactory solution. To sum up, the paper's lack of motivation makes its contributions look uninteresting in my mind, and makes me doubt whether we really need such involved techniques or whether the problems at which these techniques are aimed can be solved in simpler ways.

We understand the reviewer's concerns and have significantly improved the motivation component in the introduction. Specifically, the first few paragraphs and mainly the second paragraph have been modified to provide concrete and compelling use cases for SOPs and DSOPs. The key argument for "dynamic" travel times in DSOPs is that traffic is usually time dependent (e.g., road congestion is high during peak hours and low at other times, traffic at large roller coaster rides is low immediately after lunch, etc.).

In summary, we highlight three broad categories of applications that provide ideal use cases for SOPs and DSOPs [see Section 1: Introduction]:

(a) Delivery (food, equipment, clothing, home fuel, etc.) and service (repairs, plumbing, television, utilities, etc.) businesses that need to identify a subset of requests given the limited amount of time available. Travel time is uncertain and dynamic due to traffic.
(b) Trip design problems, where patrons require guidance in visiting points of interest in large cities, theme parks, large expos, museums, etc. Once again, travel time is uncertain because of uncertain congestion/waiting at points of interest.
(c) Traveling salesperson problems, where there is a limited amount of time to maximize sales. Travel time is once again uncertain and dynamic due to traffic.

With regards to the reward function, for (a) and (c) categories of applications above, it is concretely determined. In the case of (a), reward is based on the delivered item or service and in case of (c), reward is based on the expected number of sales in a city.

As for case (b), note that user preferences are subjective. Contrary to the reviewer's claims, it should be noted that neither our model nor our approaches are tailored to a specific set of preferences/rewards. We have randomly generated our instances and thoroughly tested our approaches to eliminate any bias for specific sets of preferences/rewards. In this sense, our approaches for solving DSOP/SOPs are complementary to any existing preference elicitation approaches for reward functions.

That said, we take the reviewer's comments well. While doing preference elicitation is out of the scope of this paper, we have designed the rewards based on a field test that we did on the same theme park for which we have wait time data for 1 year. As part of this field test, we got an opportunity to not only survey 50 participants about their preferences but also their movement trajectories in the theme park. This is described in detail in Section 6.1.2 and under "Reward Functions based on Field Test" paragraph of Section 6.2.2.

Here are some detailed technical comments:

- I'm a bit confused by the presentation of SOPs in Section 2.2. Equation 1 involves what you call "the probability that the travel time along the path thus far is no larger (or larger) than H". Over what sample space is that probability defined? That is, how are you sampling paths via which you get to v? Do you sample a random path to get to it? In that case these probabilities seem to be extremely difficult to compute. Or do you always try to get to v via the same fixed path? In that case, your definition of utility must explicitly condition on the path via which you get to v. In short, as it stands, the utility of a node appears to be ill-defined.

  Reviewer has rightly pointed the mistake in equation (1). It was a minor typo with the strategy (sequence of nodes to visit) missing from the equation. We have now rectified the mistake and fixed the notation and text to appropriately reflect the same. It should be noted that calculating the probability of arrival at a node exceeding horizon can be easily done either through sampling (for the general case) or analytically (for standard probability distributions).

  We have also indicated the overall objective so as to give a clear sense of the objective in SOPs as was defined in existing work [see Section 2.2 after equation (1)].

- Note also that an optimal solution to a (D)SOP generally isn't a linear plan, but rather a time-dependent policy. This is easy to see if an SOP is viewed as an MDP (time is part of the state space and implicitly defines a transition function for your actions). In this MDP, a globally optimal policy is dependent on the entire state, and hence on the time component of the state. This means that the algorithms presented in the paper, which look for linear plans, wouldn't find an optimal solution to (D)SOPs even if these algorithms were optimal. Judging by the discussion in the Related Work section at the end of the paper, the authors may be realizing all of this. I would argue that this realization implies that MDP solution approaches should be considered as well when solving (D)SOPs. Although few of them handle continuous state spaces indeed, the algorithms proposed in the paper discretize the edge traversal time distributions as well, and thus don't handle continuous time either.

  Indeed, the reviewer has rightly pointed out the connection between MDPs and (D)SOPs that we have also previously highlighted in our related work section (see Section 7). We have updated the related work section (Section 7.4) to highlight one other major issue (complexity) in using MDPs.

However, there are multiple issues associated with considering an MDP for (D)SOP problems:

A. *For an MDP corresponding to (D)SOP, the state space would have to contain all the nodes that have been visited previously along with time elapsed. In order to make a decision about the next node to visit, we have to know all the nodes we have 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 nodes), the scalability will be severely limited.*
B. *To be the best of our knowledge there is no other work out there which considers all of the following issues at the same time:*
    1. *Semi-continuous time distributions*
    2. *Risk sensitive objectives*
    3. *Time budget*
    4. *Closed-loop policies (node and time dependent policy rather than a sequence of nodes)*

This paper is a concrete first step where we consider the 3 issues of semi continuous distributions, risk sensitivity and time budget. We intend to pursue closed loop policies in the future.

- The local search algorithm (section 5.1) appears to be unsound, unless one assumes the graph G to be fully connected. Otherwise, I don't see how you can insert an arbitrary vertex at a chosen position in the current path; there needs to be an edge from any vertex to any other for this to be possible.

We disagree with the reviewer on this point. One can always convert a non-fully connected graph to a fully connected one by adding edges with infinite cost.

- In line 5 of Algorithm 1, instead of writing "random metric" it would be good to clarify in pseudocode from which set of metrics one is randomly chosen by the algorithm. This is explained in one place in the text, but someone who misses that one place may be puzzled as to what the "random metric" means.

We have introduced a new table [see Table 3] to list down the five metrics in one place. Even in the Algorithm 1 we refer to these metrics as M1, M2, …, etc.

- I found the plots in Figure 1 very confusing, largely due to the cryptic labels that aren't explained in the figure caption. One has to dig around multiple places in the text to find explanations of what they mean. Even when this info is found the plots take effort to understand because of the "double" y-axes that merge two different kinds of information into the same figure.

We have now included a detailed caption to explain all the important aspects represented in Figures 1 - 4.

- Crucially, the paper provides very few details on the real-world theme park dataset. Specifically, it doesn't say how many vertices it has. Considering that in the DSOP experiments the presented methods (especially the MILP-based ones) have various difficulties when solving the synthetic problems but don't have difficulties with the real-world one, this suggests to me that the real-world problem is fairly small (< 30 vertices). This makes this experimental

setting not very convincing, especially because the rewards for it are entirely made up, and eliciting them in the real world would be difficult.

> *Our real world example is based on a major theme park in Singapore. This theme park has 21 attractions and hence we have 21 nodes. Please note that even the existing work on dynamic OPs (without stochasticity) also experimented with 21 nodes. It is not to indicate that 21 nodes is a big problem, but stochasticity adds significant complexity. This has also been emphasized in the context of TSPs as well (which typically take smaller runtimes than solving equivalent OPs) by [Cheong and White, 2012; A. Toriello and Poremba, 2014]. There, they solve a maximum of 44 node problems considering only stochasticity and no time dependence.* ***We thus believe that the size of the problem in the real-world theme park dataset is of reasonable size considering that it includes both time dependence (dynamism) and stochasticity.***
>
> *The travel time samples (which include wait times as a major component) are obtained from real data provided by the theme park operator for a 1 year period. It should be noted that performance of the approach is independent of the actual reward values employed. Thus, our approaches are complementary to any preference elicitation mechanism employed to get rewards.*
>
> *In addition, we have also considered the reviewer's point and thought about a principled method of designing the rewards. While doing preference elicitation is out of the scope of this paper, we have designed the rewards based on a field test that we did on the same theme park. As part of this field test, we got an opportunity to not only survey 50 participants about their preferences but also their movement trajectories in the theme park. This is described in detail in Section 6.1.2 and under "Reward Functions based on Field Test" paragraph of Section 6.2.2.*

Writing-wise, the paper is mostly clear and well-structured. Here are some suggestions for rephrasing a few places in the text:

Towards this end --> To this end

Typically, it is often impossible --> omit "often"

N number of samples --> N samples

Stocahstic --> Stochastic

The objective value is therefore inversely proportional to the value of \alpha. --> I didn't understand this sentence.

remains normal and gamma distributions respectively --> remains a normal and a gamma distribution, respectively

For other complex distributions, including the case for gamma distribution, where \theta for individual edges is different --> For other complex distributions, including the case where gamma's \theta parameter varies across different edges,

computing triple integrals take -> computing triple integrals takes

most scalable of all distributions with integration --> the most scalable of all distributions for the purposes of integration

exiting the exit vertex --> reaching the exit vertex(?)

tradeoff between memory requirement and efficiency --> tradeoff between memory constraints and efficiency(?)

We varied a large number of combinations of parameters --> We tried a large number of combinations of parameters

tradeoff between runtime and solution quality --> tradeoff between speed and solution quality

we vary the deadlines H by setting it --> we vary the deadline H by setting it

for all combination of parameters --> for all combinations of parameters

we varied only one parameter and setting the other --> we varied only one parameter and set the other

good paths that minimizes --> good paths that minimize

for each time interval in each edge --> for each time interval for each edge

no much room --> not much room

to increase its difficulties --> to increase its difficulty

We have fixed the above typos.

In summary, my main suggestion to the authors, in addition to fixing the above bugs, is to come up with a more convincing, significantly-sized application for the proposed DSOP framework, invest in modeling that application, and evaluate the proposed algorithms on that application.

Reviewer 2

First, a comment on the terminology: "dynamic" travel times (as other dynamic aspects in general) usually refer to estimated travel times that may change on-line during the execution of the route. Here, I think you are solving a "static" problem that is: computing off-line a maximum reward path. So I would rather speak of "time-dependent" travel times which is the classical terminology used in TSP/VRP. I will stick to this terminology in my review.

The article deals with a very challenging problem which is a stochastic version of a time-dependent orienteering problem.

The description of the problem is very clear and, more generally the paper is well organized and easily accessible.

The problem is motivated by a real application for sequencing the activities in a theme park. A weakness of the article in its current state is that the adequacy of the problem definition with the

application should be better justified. That the theme park activity sequencing should be considered as an orienteering problem is clear, and it is a very good and original application of OPs by the way. The extensions to stochastic aspects make the problem much more complex and harder to solve. As a result, these problems must be somehow simplified (scenario sampling like for MILP-SAA or even reduction to a unique scenario, so to a deterministic problem for the proposed MILP-Percentile approach). For these stochastic aspects, the experimental results provide enough arguments to justify the model extension. But it is less clear for the time-dependent aspects:

- What if you would compute the paths assuming time-independent travel times (for instance by considering average travel times or by integrating the time-dependency into the stochastic aspects). That way, the MILP approaches would probably allow solving the (simpler and less accurate) problem to optimality for the synthetic dataset. It would be very interesting to show in the experiments how it would compare with the local search. Or even, you could just decrease the number of time steps (100 intervals more or less correspond to 5mn time-windows, is such a precision really required?). In general, my point here is that solving a complex and accurate problem with a relatively poor method (like the Local search seems to be here) is not necessarily better than optimally solving a less accurate but simpler problem so the question to answer with a some additional experiments is "does it worth complexifying the problem?".

- A possible direction to simplify the problem while exploiting the particular features of the application is that it seems that the time-dependency (and the uncertainties) depends more on the activity itself (as mentioned in the article: queues at a particular attraction) than on the transition between attractions so that most of the variation is on the duration of the visit of attractions rather than on the travel time between attractions. Don't you think that considering time-dependent and stochastic durations of visits and simpler time-independent or/and deterministic travel times between visits would be a possible compromise?

*In other words, **both the above questions** pertain to whether reducing the number of intervals (by merging of intervals) helps. The reviewer is right in that we had not provided a comparison of MILP-Percentile for DSOP with reduced interval space against local search. To address this, we have now provided the results for multiple reductions in number of intervals for the same original DSOP on both real and synthetic problems. These results are provided in Tables 7, 8 and 9 with descriptions provided under the title "Approximating DSOP by merging consecutive intervals" in Section 6.2.2. Please note that if we have one interval, DSOP is equivalent to SOP (i.e. time independent travel times).*

Conclusion from these results is that MILP-Percentile is significantly better than local search if reduction in intervals is not significant (i.e. in real world problem). However, local search performs better in most cases if reduction in intervals is significant (i.e., in the synthetic problem).

We agree with the reviewer that main component of travel time is wait time. However, that does not imply that travel time is time independent. In addition to the results above the following visitation patterns in user trajectories observed during the field test at the theme park provide a strong case for time dependence:

(a) People do not prefer to go to thrill rides (large roller coasters) immediately after lunch.
(b) There is a preference to go to dark and wet rides in the afternoon when the temperatures are the highest.
(c) People tend to visit the most popular attractions early in the day or late in the evening.

- On the other hand, there are some aspects of a theme park activity sequencing that seem to be important and are not mentioned in the article like precedence constraints between visits or, even more important, specific time-windows (for instance the shows that run according to a predefined schedule).

Indeed, time windows and precedence constraints are interesting and important. They can be accounted for in our MILP formulation with minimal changes. We have described these changes in Appendix B and also here.

For a precedence constraint to be placed between node i and node j, all we need to have as constraint is that arrival time at node i is lower than arrival time at node j. It is represented as follows in both the MILP formulations:
a_i \leq a_j (for MILP-Percentile) and
*a_i^q \leq a_j^q + z^q * M (for MILP-SAA)*

Similarly for a time window [t^{min}, t^{max}] for an attraction i, we need to enforce the following
For MILP-Percentile
a_i \leq t^{max}
a_i \geq t^{min}
For MILP-SAA
*a_i^q \leq t^{max} + z^q * M*
*a_i^q \geq t^{min} - z^q * M*

Note the use of z^q in MILP-SAA. These z variables ensure that if the precedence or time window constraints are not satisfied, overall set of violations is increased.

For time windows, it should be noted that travel times derived from past wait time data would help in capturing such windows with our current approach. In times when there is no show, there would be no edge from any other node to the node where 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 travel time data available from other nodes.

The proposed local search approach does not perform well compared to the MILP models (when they can be solved). Do you have an idea why ? You mention that the LS is very fast. What if you let it run longer (maxIterNoImprove=50 seem quite small) ? Is it really trapped in local optima? In fact the local search does not seem very sophisticated.

Indeed, the solution seems to be trapped in local optima. For small increases in numIterNoImprove, there was no improvement in value. When we increased numIterNoImprove to 500 and total number of iterations to 3500, the solution improved. However, this increase was not substantial (maximum difference of 30 on peak and non-peak data sets) and came at a significant improvement in runtime (more than 400 seconds).

This could be because of the time-dependent travel times that require significant amount of backtracking (many nodes in the current best solution to be changed) to identify a better solution.

Results provided in the section on "Improving Local Search" in 6.2.

Did you try more complex moves like 3-opt?

We did try 3-Exchange operators. However, it had the same impact as increasing the numIterNoImprove. That is to say, there was a small improvement in solution quality. For the fixed 1500 iterations and 50 numIterNoImprove, 3-Exchange resulted in a maximum improvement of 20 on peak and non-peak data sets. More specifically, the performance of 3-Exchange with 1500 iterations and 50 maxIterNoImprove was very similar to 2-Exchange with 3500 iterations and 500 maxIterNoImprove.

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.

Results provided in the section on "Improving Local Search" in 6.2.

By the way, I do not get the description of the 2-opt move described as "randomly swapping two vertices".

The reviewer is right in that we have not used the term "2-opt" according to standard definitions. We have renamed it as "2-Exchange" to more accurately reflect the task accomplished by this neighbourhood operator. The primary reason for considering 2-Exchange in our local search is efficiency. Each evaluation of a complete strategy is computationally non-trivial, and hence we choose to go with a 2-Exchange operator.

Usually, the 2-opt move consists in reordering routes that
crosses so it is not just about swapping two vertices in a path.

We have not used the standard method of reordering routes that crosses because of the following reasons:

1. *Travel times between nodes are time dependent, so one could construct an example where un-crossed routes (in the sense of travel time) does not always produce a better solution than than crossed routes.*

2. *Our problem is not restricted to the 2-dimensional Euclidean space, i.e., DSOPs can also represent more abstract problems where routes and crossing of routes are not purely based on distance. Travel times in our theme park problem is one such example, where waiting times at attractions are merged together with the travel times. As the reviewer points in the next comment, oversubscription planning is another such example.*

Some more detailed comments

In the introduction, when mentioning the applications of OPs, you
could also mention its application in planning, more precisely in
over-subscription planning problems. See for instance: D. Smith.
Choosing objectives in over-subscription planning. In Proceedings of
the 14th International Conference on Automated Planning and Scheduling

(ICAPS-2004), p 393-401, 2004. Here an OP is used as a relaxation to solve the original planning problem.

We agree with the reviewer that oversubscription planning problems provide a good set of applications for OPs. We have included the reference provided by the reviewer in the first paragraph of introduction.

About the MILP-Percentile approach:

- In the section 4.2.2 on MILP-Percentile, you say that the percentile is computed over all the Q samples. Why don't you compute it from the original probability distributions? Wouldn't it be more accurate?

Indeed, if we compute from the original probability distribution, then it would be more accurate. However, except for a few well-known distributions, the method employed to compute percentile is through numerical methods.

On the other hand, computing from samples is a generic method that is applicable for any distribution and is the reason we have used it.

- In general, in the experimental section, you never directly compare MILP-SAA and MILP-Percentile. You just say that MILP-Percentile scales better than MILP-SAA and use it instead for the time-dependent problems. What about a direct comparison between MILP-SAA and MILP-Percentile on the SOP instances?

The reviewer is right in that we did not compare MILP-SAA and MILP-Percentile. To address this, we have now provided an exhaustive comparison in Table 4 between Local Search, MILP-SAA and MILP-Percentile on the SOP instances. Overall conclusion from the results is that MILP-Percentile is better than MILP-SAA with respect to run-time and is better than Local Search with respect to solution quality.

- It seems to me it should be possible to provide more theoretically grounded results related with the fact the MIP-Percentile approach is a good approximation. The total duration of the path is just the sum of all the individual travel times (the situation would be more complex if you would have waiting times due to time-windows on visits for instance) so one should probably be able to show some results for particular probability distributions or when the path involves many visits...

Unfortunately, we have not been able to derive the theory behind MILP-Percentile yet. Here, we explain through an example, the difficulty in getting theoretical bounds in the worst case.

Instead of computing a policy that on ***alpha fraction of samples*** *has total duration less than horizon,*
MILP Percentile *computes a policy for* ***one sample, which has alpha percentile durations****, and has a total duration less than horizon.*

For purposes of exposition, let us consider the following simple example where policy is given by n1, n2, n3, n4, n5. Travel time for the policy is therefore sum of travel times on n1-n2, n2-n3, n3-n4, n4-n5. Let us say alpha is 0.9.

Essentially for the above approximation to be theoretically guaranteed in terms of equivalence to the original SAA formulation, we should have

90th Percentile (n1-n2) + 90th Percentile (n2-n3) + …+ 90th Percentile (n4-n5) <= Horizon
implies that
Pr(\sum_{q} [q_{n1-n2} + q_{n2-n3} + …+ q_{n4-n5}] <= Horizon) > 0.9

Given we have found a policy where sum of 90th percentile durations is less than horizon, we have:

Pr((n1-n2) + (n2-n3) + …+(n4-n5) <= Horizon)

>= Pr(n1-n2 <= 90th Percentile(n1-n2)). Pr(n2-n3 <= 90th Percentile(n2-n3)). Pr(n3-n4 <= 90th Percentile(n3-n4)). Pr (n4-n5 <= 90th Percentile(n4-n5))

= 0.9^4

= 0.6561

Given that sum of 90th percentile durations is less than Horizon, we have used ">=" because the only case where we can be sure that (n1-n2) + (n2-n3) + (n3-n4) + (n4-n5) is less than Horizon is when each of n1-n2, n2-n3, n3-n4 and n4-n5 are less than their 90th percentile duration. In any other case (i.e., one, two, three or four of n1-n2, n2-n3, n3-n4, n4-n5 are greater than their 90th percentile duration) , there is a chance that sum can be greater than Horizon.

Therefore, if we use 90th percentile duration for each travel time and get a solution, we can only guarantee that sum <= Horizon is with a probability 0.6561 and not with a probability of 0.9.

However, in practice and in the average case, we can additionally have the probability that two (or more) out of the four edges (say the ones with the least means) can exceed the 90th percentile.

Hence, MILP-Percentile is a heuristic that works well in practice and dependent heavily on the underlying problem, so it may or may not have theoretical bounds.

In the section about the optimization of LS by using matrix computations, it seems that what you can get as a speed-up in average is more or less a factor 2 as you short-cut the computation for the steps before the first change (which can be considered as uniformly randomly distributed). In this case, does it really worth the additional memory consumption?

It depends on the problem domain. If memory is not a major consideration and decision support is time sensitive (like at theme parks), then a 2 fold speedup can be quite useful.

In the experimental section:

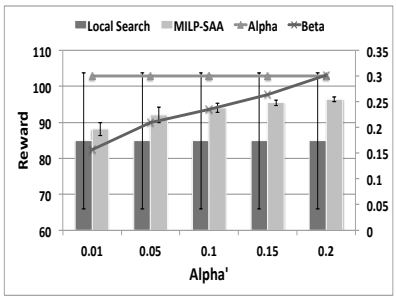
- Is there a theoretical justification for the choice of different values for \alpha and \alpha' or were these values chosen empirically and, if yes, how?

As indicated in the seminal paper by [Pagnoncelli, Ahmed & Shapiro], the relation between \alpha' and \alpha varies from domain to domain. We obtained it in this paper based on few trial experiments on problems with tight budget constraints (63 nodes with 20% budget)

- The default values for \alpha (0.3) and \alpha' (0.1) do not map to each other according to the values described in the second item in the list before. Is it expected?

It was a typo and is now fixed. \alpha' = 0.2 is used corresponding to \alpha = 0.3.

- On p20 when discussing the effect of varying the violation probability \alpha', I'm very surprised that increasing \alpha' does not increase the reward. This is very counter-intuitive as for instance for an extreme case (\alpha'=1 !) one should be able to perform all the visits and achieve a very high reward (while of course being sure to violate the deadline). Do you have some explanation?



In the synthetic examples and the settings we had chosen for those results, the reviewer is right that the improvement is not visible. One reason for this small gap is because we were computing 90% optimal solutions in cplex. Once we increased it to 99% optimal solutions and reduced the number of values shown on Y-axis, the difference is more prominent. A sample of the results are shown in the graph above for 20 node problems. In the paper, we have updated all three figures and, as can be seen there, there is a noticeable difference.

Furthermore, this minor difference does not happen always. In fact, real world results of Figure 6(c) with respect to changing $\alpha' show a significant difference in solution quality.

- In the generated problems described in the end of page 24 by randomly generating some theta parameters, you end up with mean travel times that do not satisfy the triangle inequality. But it does not really make sense in practice, right? (if for traveling from A to B it is shorter to go through C in between, then you will go through C and you can decrease the travel time between A and B). So you probably should rework these values, for instance by running a Floyd-Warshall algorithm…

Orienteering can also be used to represent (as the reviewer suggested) scheduling problems, where link between nodes is just the temporal lags and not travel time. In such cases, triangle inequality is not satisfied.

In the section 7 about related work, you do not cite any work in time-dependent travel times in deterministic versions of the OP problem. Is it because there is no work in this domain? You could mention some works on time-dependent travel times in TSP/VRP.

We have done a major revamp of the related work section and we do include many references associated with time dependent travel times.

Typos
---

p5, section 3.1: Stocahstic

p22, close to the end of page: We also report the the percentage …

Fixed the above typos

Summary

The topic of the article as well as the current material show that there is clearly the potential for a very good JAIR paper. I would recommend rejecting the current version of the paper with strong encouragement to resubmit. The two main points to be improved are:

- a better justification of the compromise that should be made between the accuracy/complexity of the model and the capacity to solve it (in particular as far as time-dependent travel times are involved), and

- a more detailed analysis of why the proposed Local search performs so poorly (and ideally, some improvements of the local search)

Reviewer 3

This paper describes an event-based formulation and local search method for heuristically solving dynamic, stochastic orienteering problems with risk aversion. The contributions in this version of the paper differ from previously published work by the provision of an explicit event-based formulation in mixed-integer programming format.

The highlight of the paper is the event-based formulation, where the authors relax the state-dependent traversal time variable T^{a_j}_{j,i} to be a semi-continuous variable. This allows them to find an approximate solution to what is effectively an automated planning problem, by directly applying mixed-integer programming techniques. The relaxation is not as severe as, say, supposing the traversal time is constant, and therefore it should be possible to obtain better approximations.

I have the following suggested revisions. Since this list is long and substantial, I will recommend this paper for a major revision.

+ literature. The TSP and its variants are heavily studied problems. While the OP does have some differences, the core structures (in a mixed-integer programming sense) are very nearly identical. See Laporte (1997), to see what I mean. The OP and Travelling Purchaser Problem have identical core structures, with (inconsequential) side constraints being the only difference. I therefore had a lot of trouble with statements such as "there has not been much work in stochastic variants of OP thus far", or suggestions that the dynamic OP has not been well studied. A good starting place is Kang et al (2011), and the citations contained therein. Please extend your literature to include Angelelli et al's work (citations below), and describe very clearly how your work differs from theirs.

We thank the reviewer for pointers to existing work and we have referred to them in our related work section. We have restated the characterization of existing work, particularly the stochastic and dynamic variants (7.1 and 7.2).

Unfortunately, we were unable to find any mixed integer formulation in Laporte (1997). The contribution of that paper is about transforming arc routing problems into traveling salesman problems. However, the reviewer point is well taken and we were able to find another paper (Assad, A.A. and Golden, B.L. Arc Routing methods and applications. In Network Routing, Handbooks in Operations Research, 1995.) which provides the MILP formulation and other relevant details.

We agree with the reviewer that core structure of the MILPs corresponding to OP, TSP and arc routing are similar. However, there are multiple key differences with respect to the pointers provided by the reviewer as highlighted in the new related work section (Section 7).

+ feasibility. Since there are constraints on the time horizon, it is conceivable that the relaxation of T^{a_j}_{j,i} (from state-dependent variable to semi-continuous variable) could lead to infeasible solutions. Please provide a rigorous discussion, with proof if necessary, of whether this is true or false.

Our intention there was to indicate that if the travel time distribution is semi continuous then we can exploit it in the constraints. Please note that this is not a relaxation. Reviewer is right in that we have not shown the equivalence of the non-linear and linear constraints.

In order to address this, we have introduced Proposition 1 to demonstrate equivalence of the non-linear constraint of (34) to the linear constraints of (35)-(42).

Because of the equivalence when travel time distribution is semi continuous, everything that is feasible with the earlier constraint is feasible with the new constraints and vice versa.

+ complexity vs. solvability. The statement '...solving OPs optimally is NP-hard' seems out of place where it is -- consider moving it down to the discussion of solvability. The computational complexity of the OP is used to lever an argument that local search techniques are necessary to obtain scalable solution approaches. What size of OP are you intending to solve? Your real-world instance is pretty small compared to the 85,900 city problems solved by Concorde for the TSP. This justification is too weak in the context of the examples you are providing. Does an exact approach (in the context of the general OP) scale for these examples? I would expect so. If not, please provide more rigorous empirical or other evidence. Please consider my comments on the formulation when you do this.

We have moved the sentence to the discussion on solvability. As for the complexity comparison with TSPs, please note the following:

(a) We focus on DSOPs and they are much more complex than OPs and whose formulation is significantly different (as shown in the paper) to OPs, which are already a step above in terms of complexity than TSPs because of the budget constraint.
(b) When it is mentioned that 85,900 city problems are solved with TSP, typically the time taken is a few months and this is mentioned on their website (http://www.math.uwaterloo.ca/tsp/pla85900/compute/compute.htm). Our results are generated within a few minutes as our emphasis is on route guidance.
(c) Furthermore, the latest work on Stochastic TSPs [Cheong, T., & White, C. (2012), A. Toriello, W. H., & Poremba, M. (2014)] also indicate the significant complexity involved

in solving even small TSP problems when stochasticity is involved. They show experimental results on a maximum of 44 node problems.

Dynamic Orienteering Problems (without stochasticity) have also previously been shown on examples with only 20 nodes in the following paper:

"Jin Li. Research on Team Orienteering Problem with Dynamic Travel Times. Journal of Software. 2012. Volume 7, Number 2, Pages 249-255."

Please note that we have now provided results on both 21 and 32 node problems with our Dynamic and Stochastic OPs

+ notation. Not all notation has been defined. Notably:
- p.5, eq.5, define t
- p.10, eq.36, define m

Please note that we are defining the "l" function using a free variable "t". There is no inherent definition to "t".

M is a large positive number that is an upper bound on the maximum possible value for any arrival time a_i.

+ Elaborate what you mean by 'The objective value is therefore inversely proportional to the value of \alpha'.

It was a mistake and this sentence is removed.

+ formulation (Table 1). Constraints (15) and (16) are duplicated in constraint set (17). Remove them. Constraints (13) and (14) are valid inequalities, since they are not required for correctness of the formulation. This is guaranteed by the single-commodity balancing equations in constraint set (17). It could be that (13) and (14) don't contribute positively to the solve time--this should be investigated. Please move all variables to the bottom line, so that they can easily be identified and differentiated from parameters in the model.

Indeed, (15) and (16) were duplicated and we have removed them.
As for (13) and (14), while their presence is not required, they have a significant impact (specifically in the DSOP case) on the run-time, as they tighten the relaxation. We have verified this empirically.

+You have used the MTZ path sequencing formulation for subtour elimination, but you have not cited it. See Miller et al (1960) below. You did not justify the use of these particular constraints, rather than, say, formulations that have tighter LP relaxations and better solve times. The fact that you have used MTZ means you also require commodity-balancing (or flow) constraints. This is a doubling up of subtour elimination constraints, and could be contributing a lot to the poor solve times. Why not use the tightest formulation in the literature? For example, Laporte (1997), where the subtour elimination constraints can be implemented using a simple separation algorithm.

We have now included the citation. Indeed, the justification was missing and the reason is that our formulation for solving (D)SOPs would work irrespective of the type of subtour elimination method (either ranking or separation algorithm).

However, the reviewer's point is well taken and we have conducted experiments to compare the method of using ranking variables and one where separation constraints are introduced incrementally. It should be noted that there is a difference in how separation constraints are introduced in TSPs and OPs. In TSPs since all nodes have to be part of the final solution, if we have a sub-tour given by vertices "V", then the separation constraint has a tight relaxation with the following constraint:

\sum_{i \in V'} \sum_{j \in V} x_{ij} >= 1

On the other hand, for OPs, because of the budget constraint, not all nodes need to be part of the final solution. Because of this, we cannot use the tighter separation constraint mentioned earlier. Instead, the separation constraint is given by

\sum_{i \in V} \sum_{j \in V} x_{ij} \leq |V| - 1

Due to this difference, running TSPs on problems with same distance matrix (as long as budget <= time taken by TSP optimal solution) takes lower time than that of OPs on similar problems.

On Deterministic OPs and SOP formulations, ranking based method took two times longer than an algorithm that incrementally introduces cuts based on the set of sub-tours. The following timing results (in milli seconds) are on a 50 node problem, with varying time budget for OP (20%-90% of optimal solution). Each timing result is averaged over 500 randomly generated settings.

Budget	TSP	OP Separation	OP with Ranking
20%	524.08	780.72	2640.61
30%	560.29	1176.44	3281.16
40%	440.07	786.12	2799.88
50%	502.29	1075.56	3511.47
60%	419.28	10914.41	13873.74
70%	480.23	670.23	1502.32
80%	490.25	540.25	1400.32
90%	470.23	495.2	1300.12

On SOPs we observed a similar kind of improvement (usually 2-3 fold improvement) with 20 node problems. However, for 32 node problems the improvement was around 1.5 fold and for 63 node problem, there was little or no improvement and in some cases (parameter settings) ranking based method even performed better by a small margin.

As for DSOPs, when using a separation algorithm for MILP-Percentile, the time taken to complete 1 iteration is non-trivial. Because of this, once we set a fixed time interval of 1000 seconds, we are only able to finish few iterations (actual number of iterations depends on the problem parameters such as \alpha', horizon etc.). In all the cases of parameters, after these few iterations, we obtain a solution that still contains sub-tours.

Because of this uneven performance, generality of our formulation to work with both methods of sub-tour elimination and to focus on the stochastic and dynamic aspects of the formulation, we have not included any results in the paper.

+ formulation (Table 2, Table 5, Table 6). Include the bounds on all variables, including \pi_{j,i} and T^{a_j}_{j,i}, at the bottom of the formulation.

Included all the bounds at the bottom of the formulation.

+ formulation (interval constraints). Intervals are defined, for example, by (p1,p2], (p2,p3], and so on. In your formulation, you have (p1,p2), (p2,p3). Therefore, if a solution lands precisely on p2, the solver can choose the 'cheapest' interval to be in. This is a cheat and leads to discretisation error. Please comment on the types of instances that will suffer from this formulation.

We were unable to find references to the above overlapping intervals anywhere in the paper.

We actually separated the intervals by a small fraction to prevent such cases from happening. That is to say, our intervals are { [p1, p2], [p2+0.000001, p3], [p3 + 0.000001, p4],…} and so on.

+ equivalence. You have made several references to the 'equivalence' between the Risk-sensitive SOP formulation and the SAA version (see p.8, l.4,l.14). These are not equivalent, as the SAA is an approximation of the SOP.

Fixed these references to indicate that SAA is an approximation of the SOP.

+ non-monotonicity. The SAA method is not guaranteed to converge monotonically in the general case. Therefore, it is very important to have an understanding of how many samples are required for this sampling method to be useful.

Indeed, non-monotonicity does not hold and it changes from problem domain to domain. For us, we ran preliminary experiments on the largest problem we had (63 node) with the most restrictive of time horizon. We tried number of samples from 25 to 70 in increments of 5. We found that if we used a minimum of 40 samples, we were able to get solutions with probability of failure less than the input \alpha value. That is how the selection of samples was made.

+ model vs. solution approach. You have provided several interesting formulations for risk-averse, dynamic and stochastic OPs. These are not themselves solution approaches. Please amend the claim that you have provided an optimization solution approach, and the title accordingly.

We have made these changes to the title and other places where this appears.

+ experiments. Please provide some data on the MIP related experiments. For example, what solver did you use? Did you tune the solver? What values did you use for M? What was the

linear-programming gap? Did you try warm-starting the MIP with the solution from the construction heuristic? What other techniques did you try to improve the performance of the model in a generic solver?

We use CPLEX. Here are some of the things we tried to improve the performance of MILP-Percentile formulation:

(a) Initially, we used an M value of 1000.
(b) For the M specifically associated with a_{j}^q and b_{j,i}^q related constraints, we calculated it by using a sum of maximum travel times for all samples. That is to say, for each node j, we find the edge that has the maximum travel time over all samples and intervals. We sum over top "n" (number of nodes) of those values to get M. This provided some improvement over M = 1000.
(c) While M calculated in (b) provided some improvement, the best method was to use the logical constraint constructs provided by CPLEX. Specifically, cplex.ifThen() functions provided by CPLEX. Once the logical constructs are employed, "M" is completely eliminated.
(d) We did try warm starting the MILP with the construction heuristic. Unfortunately, this did not provide any consistent improvement in the policy computed within the 1000 second duration.
(e) We tried prioritising the variables when performing branch and bound. If we initialise $ \pi_{j,i} variables, rest of the MILP becomes easy, so we assigned higher priority, so that CPLEX will initialise π variables first. This did not yield any consistent and significant improvement for all settings. We also tried other combinations with priority, but unfortunately there was no improvement in those cases.

The LP gap was quite high for smaller horizons (Ex: for horizon = 2, our MILP solution was 662, while LP relaxation was 901. In terms of policy, that would imply adding attractions with top 4 utilities). However, for higher horizons (8 and 10), the LP gap was less than 50. These were results with our default alpha value of 0.3.

+ references.
- correct the journal for Tsiligrides (1984) to 'Journal of the Operational Research Society'.
- correct Yuan et al (2009) title to 'Efficient computation of jointree bounds for systematic MAP search'.

Fixed these issues with references

Minor revisions:
[a minus sign indicates counting from the bottom of the page]
+ define 'dynamic travel times'
+ p.2, l.-12, Elaborate what you mean by 'convergence' when you say "..is incrementally improved until convergence", e.g. '...until convergence to a feasible local solution'.
+ p.2, l.-12 add full-stop at the end of the sentence.
+ p.3, l.1, Insert 'We provide' before 'Optimization and local search...' and remove 'are provided' later to put the text into active rather than passive voice.
+ p.3, l.16, insert 'while the' before 'rest of the vertices'
+ p.3, l.-15, insert 'n' after 'is called a*'
+ p.4, l.1, change 'a valid one' to 'valid'.
+ p.4., l.5, you use the word 'feasible' when you mean 'possible with current methods'.
+ p.4, l.-13,l-15, remove 'typically', change 'is often' to 'can be'
+ p.5, l,-11, change 'CPLEX' to 'any available MIP solver'.

+ p.5, l.-7, change 'Stocahstic' to 'Stochastic'
+ p.11, please clarify if the solution you could not obtain was 'feasible' or 'optimal', i.e. you could not find any primal solutions?
+ p.12, l.10, remove 'greedily', as it is implied by the fact it is a greedy heuristic.

Fixed these minor issues

Suggested references:

Laporte, Gilbert. "Modeling and solving several classes of arc routing problems as traveling salesman problems." Computers & operations research 24.11 (1997): 1057-1061.

C.E. Miller, A.W. Tucker, R.A. Zemlin. "Integer programming formulations and traveling salesman problems." J. ACM, 7 (1960), pp. 326-329

Kang, Seungmo, and Yanfeng Ouyang. "The traveling purchaser problem with stochastic prices: Exact and approximate algorithms." European Journal of Operational Research 209.3 (2011): 265-272.

Angelelli, Enrico, Renata Mansini, and Michele Vindigni. "Look-ahead heuristics for the dynamic traveling purchaser problem." Computers & Operations Research 38.12 (2011): 1867-1876.

Angelelli, E., R. Mansini, and M. Vindigni. "Exploring greedy criteria for the dynamic traveling purchaser problem." Central European Journal of Operations Research 17.2 (2009): 141-158.

All these suggested references and citations made in these papers are included in the paper.