2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Randomized Algorithm for Informative Path Planning with Budget Constraints

Sankalp Arora[1] and Sebastian Scherer[1]

Abstract— **Maximizing information gathered within a budget is a relevant problem for information gathering tasks for robots with cost or operating time constraints. This problem is also known as the informative path planning (IPP) problem or correlated orienteering. It can be formalized as that of finding budgeted routes in a graph such that the reward collected by the route is maximized, where the reward at nodes can be dependent. Unfortunately, the problem is NP-Hard and the state of the art methods are too slow to even present an approximate solution online. Here we present Randomized Anytime Orienteering (RAOr) algorithm that provides near optimal solutions while demonstrably converging to an efficient solution in runtimes that allows the solver to be run online. The key idea of our approach is to pose orienteering as a combination of a Constraint Satisfaction Problem and a Traveling Salesman Problem. This formulation allows us to restrict the search space to routes that incur minimum distance to visit a set of selected nodes, and rapidly search this space using random sampling. The paper provides the analysis of asymptotic near-optimality, convergence rates for RAOr algorithms, and present strategies to improve anytime performance of the algorithm. Our experimental results suggest an improvement by an order of magnitude over the state of the art methods in relevant simulation and in real world scenarios.**

I. Introduction

Data acquisition is relatively easy in the virtual world, where the cost of accessing data can be equivalent to accessing a memory block. But data gathering in physical spaces, where it is impractical to have sensor networks is not as trivial. Currently we rely on humans to carry or drive sensors around to digitize the physical world to collect data.

Some examples of gathering data in the physical realm are - examining oil pipelines for leaks, inspecting infrastructure for faults, exploring water bodies to monitor the ecosystem and surveying disaster scenarios for search and rescue. Using humans to do such meticulous, tedious and often risky tasks is far from ideal. Therefore, there is an urgent need to develop autonomous robots capable of performing these tasks.

Consider a situation where a region is affected by floods. There is an urgent need to locate survivors, provide supplies and establish communication with them. We have at our disposal a UAV with a camera that can fly up to a limited distance of 1 km. Then the problem becomes that of locating and counting as many survivors as possible while being restricted to fly 1 km. The robot needs to reason about the fact that it can gain a lot of information about the area by gaining height and subsequently flying close to survivors to get a more detailed picture.

[1]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA asankalp, basti@cmu.edu

Thus the two salient characteristics of the problem are

1) *Constraint on the total travel distance*. Due to limitations of fuel / battery, the length of the route taken by the robot is limited.
2) *Correlated nature of information*. Equipped with cameras, UAVs can view large areas from a distance to gain information. This leads to reward of visiting different locations being correlated.

The problem of planning routes to gain information is an NP-hard problem [1]. The current state of the art systems attempt to solve this problem using two lines of approach. One approach is to apply a myopic strategy [2], [3] where a set of sensing locations is identified and the system travels to the most promising one. While such strategies are computationally efficient, they fail to effectively account for the constraint on traveling distance. As a result the computed routes can lead to oscillatory behavior.

Another approach is to invoke a long horizon planner [4]–[7]. However, these approaches are far from real-time.

The key contribution of this paper is an anytime, provably near-optimal algorithm, Randomized Anytime Orienteering(RAOr), that can efficiently solve for routes that maximize correlated reward functions subject to constraints on route length. The key ideas behind RAOr is to restrict the search space to routes that incur minimum distance to visit selected nodes, and rapidly search this space using random sampling.

We provide empirical and theoretical analysis of RAOr. We prove it to be near-optimal while providing convergence rate analysis. Empirically the algorithm outperforms the state of the art by more than an order of magnitude in terms of run time required to solve benchmark problems near optimally.

In the next Section we formulate the problem and describe the essential related work. In Section III we elucidate on transforming Correlated Orienteering to a Constraint Satisfaction Problem and a Traveling Salesman Problem. The resultant algorithm (RAOr, Alg. 1) obtained by leveraging the new formulation is described in Section IV with its properties and drawbacks. These drawbacks are addressed in Section V. The evaluation results and conclusions along with scope of future work is presented in Section VI.

II. Problem Formulation

Let us formally define the problem of maximizing the reward in a given experiment while respecting a traveling budget B. Let $V = [v_1, v_2, v_3, ..., v_n]$ be the set of all sensing location in the workspace. Let the robot start from node $v_s \in V$ and end at node $v_e \in V$. Let $r = [v_s, v_1^r, ..., v_m^r, v_e]$

978-1-5090-4632-4/17/$31.00 ©2017 IEEE

be an ordered set of the sensing locations where, $r : r \subseteq V$. For each r, let $I(r)$ be the reward gained by visiting each location in r. Let the cost of traversal be given by $C(r) = \sum_{i=1:|r|-1} C(v_i^r, v_{i+1}^r)$, where v_i is the i^{th} element in r , $\forall i \in [1, |r|]$. The problem then is defined by equation 1.

$$\arg\max_{r \subseteq V} I(r) \text{ subject to } C(r) \leq B \tag{1}$$

The Orienteering Problem (OP) [8] is closely related to the exploration problem. In the Orienteering Problem reward at nodes are defined to be independent of each other or in other words the the reward function is modular. Algorithms that provides a solution of constant approximation ratio of (2 + ϵ), where ϵ is a small number, were given by [9], but such guarantee is unsatisfactory in practice. On the other hand Mixed Integer Programing(MIP) based solutions exist for OP and related problems [10]. But these solutions fail to capture the reward relationship amongst nodes, leading to sub-optimal paths.

[1] established that information gain is sub-modular and monotonic. Nemhauser in 1978, [11] provided an efficient method to optimize submodular functions. Unfortunately adding the traveling budget constraint makes the problem non-submodular and non-monotonic. [5], [12] suggested using a *recursive-greedy* to find approximate solutions for the orienteering problem if the reward function is submodular. Due to large run-times none of these solutions scale well to real world problems. The runtimes of most of the algorithms exceed 3 minutes on a standard desktop PC for a graph of more than 100 nodes.

The MIP based methods solve for linearly relaxed versions of the problem and then impose integer constraints, [4]. This results in the method spending most of its time finding partial solutions that do not meet the budget constraints. Also, the state of the art algorithms optimize for the nodes to visit and the sequence in which to travel those nodes together. This leads to the algorithms searching a huge space of solutions for which the sequence of nodes traversed is sub-optimal.

The large run times and the failure to model the reward relationships amongst nodes, necessitates the development of better exploration planning algorithms for efficient autonomous information gathering systems. In the following section we propose reformulating the problem as a combination of Constraint Satisfaction and Traveling Salesman Problem to overcome these limitations.

III. From Orienteering to Set Selection and TSP

The solution to the correlated orienteering problem is a route in a graph, such that the reward attained by the route is maximized while the path cost stays within a specified value. We break the problem of finding the route, into finding the set of locations to visit and then finding the optimal order in which to visit those locations. Since, we assume the reward function is independent of the order in which the set is visited. This allows the set selection and set order optimization to run independently, without affecting the reward attained by a set. The set selection problem can be formulated as a Constraint Satisfaction Problem (CSP) and finding the optimal order for the selected set is studied as the Traveling Salesman Problem (TSP). In the following we pose the correlated orienteering as a combination of CSP and TSP. Treating set selection and order optimization as independent problems allows us to exploit efficient TSP solvers to search an exponential search space efficiently in polynomial time. The resulting algorithm (RAOr) is presented in 1 and described in IV

Input: $G = [V, E]$, v_s, v_e, B, T_r
Output: The best route found in run-time T_r

$S = SampleSet(V)$ `// Random set is picked such that it contains` v_s `and` v_e
$r = TSP(S, v_s, v_e)$
$r_{best} = \phi$
if $RouteLength(r) <= B \wedge Reward(r) > Reward(r_{best})$ **then**
  $r_{best} = r$
end
for $i = 1 : 3|V|$ **do**
  $v_{new} = Sample(V)$ `// sample a node`
  if $IsInRoute(r, v_{new})$ **then**
    $r = DeleteFromRoute(r, v_{new})$
  end
  else
    $r = AddToRoute(r, v_{new})$
  end
  if $RouteLength(r) <= B \wedge Reward(r) > Reward(r_{best})$ **then**
    $r_{best} = r$
  end
end
return r

Algorithm 1: Randomized Anytime Orienteering(RAOr)

A. Constraint Satisfaction Problem

In this section we describe how the Correlated Orienteering problem can be viewed as a Constraint Satisfaction Problem. Let r^* be the solution to equation 1. Let, V^* be the set of nodes that are present in r^*. Let, $a_r \in \{0, 1\}^{|V|}$ signify the presence of the nodes in a route r, such that $a_r^i = 1$ if $v_i \in r$ and $a_r^i = 0$, otherwise. In order to solve the correlated orienteering problem we want to find the assignment a_{r^*} and then the optimal order in which nodes belonging to r^*, should be visited.

Randomized technique for efficient search for satisfying assignment of a binary tuple was presented in 1999 in [13] as a solution to the CSP problem. RAOr employs the same technique to search for the optimal assignment of $a = a_{r^*}$, i.e. randomly flipping the assignment of one of $|V|$ bits of a.

B. Traveling Salesman Problem

Once the correct set of nodes are found, then finding the optimal route just requires finding the order in which they need to be visited. Traveling Salesman Problem solvers can provide us with a near-optimal order in polynomial time, [14]. Hence combining the TSP and CSP solvers allows us to develop an algorithm to solve the correlated orienteering problem near optimally and efficiently.

IV. Randomized Anytime Orienteering

In this section we describe our algorithm (Alg.1) in detail. We highlight its properties and drawbacks with examples. In

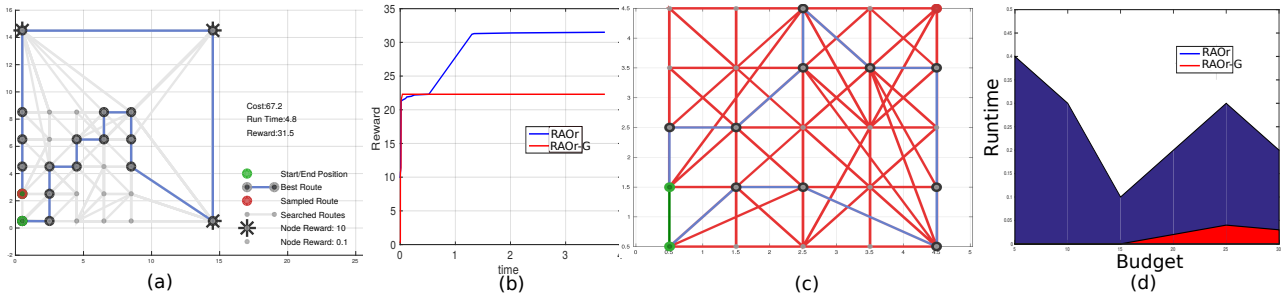

Fig. 1: Illustration of drawbacks of RAOr algorithm. a) The problem shown in here consists of three high valued nodes and a budget just sufficient to visit all three. The optimal solution is to visit all three nodes but very few routes exist that have a reward close to optimal. The solution shown is found by running RAOr-G algorithm, Alg. 2. b) Shows the run time vs reward plot for the problem shown in (a) for both RAOr-G and RAOr.RAOr can only attain approximately 66% of the optimal value, as it has a hard time selecting the correct set of nodes. c) Illustrates the problem RAOr faces while running on problems with a small budget. Here a 5X5 grid of nodes, distributed uniformly at a resolution of 1 with a budget of 15 is shown. The routes in red are routes sampled by RAOr that exceed the budget, whereas routes in blue are routes that were sampled that do not exceed the budget. Visibly, red routes outnumber the blue routes. d) The same problem leads to poor run times of RAOr for relatively low budget problems. Here we show the run times of RAOr and RAOr-G on the 5X5 grid shown in (c) while varying the available budget. A runtime of 0 signifies that the algorithm did not converge. Clearly RAOr did not converge for low budget problems in the allotted time. Highlighting the limitation of RAOr in dealing with low budget problems.

the next section we will discuss methods to overcome these drawbacks.

The algorithm starts with uniformly sampling a set of nodes $S \subset V | v_s, v_e \in S$ (Alg. 1 Line 1). The order in which S should be visited is computed using a TSP solver (Alg. 1 Line 2). Running a TSP solver searches an exponential space of routes in polynomial time, leading to run time reduction. The generated route is then checked for satisfying the budget constraint and saved as the best available route if it is admissible (Alg.1 Line 2-3).

The algorithm then uniformly samples a node $v_{new} \in V$ and changes it's status of being in route r (Alg. 1 Line 6-12). i.e if the node v_{new} was in route r, it is removed from route r or if it was not in route r, it is added to it. The new route obtained is checked for satisfying the budget and saved as best available route, if it exceeds the value of r_{best}. This process is repeated $3|V|$ times.

The algorithm described above is the standard probabilistic algorithm for constraint satisfaction problem as suggested in [13] changed to return the best route available in the budget. We now list some of the properties and drawbacks of the algorithm.

Theorem 1 (Optimality of Randomized Anytime Orienteering)**.** *Randomized Anytime Orienteering algorithm almost surely finds the optimal route from start to end nodes, within budget B, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions.*

The proof of the theorem is provided in the appendix.

Finding the optimal route for the TSP is an NP-Hard in a space that is exponential in number of nodes. But, polynomial time α approximation algorithms, where $\alpha \geq 1$, for a TSP exist in literature [14]. We leverage the polynomial time TSP solver to make Alg. 1 tractable.

Theorem 2 (α-Optimality of Randomized Anytime Orienteering)**.** *Randomized Anytime Orienteering algorithm almost surely finds the optimal route from start to end nodes, within budget B/α, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions. Given that the TSP solver in the inner loop is α-approximate.*

A. Drawbacks

1) Uniform Sampling of Sets: RAOr has low run-time for the case where a large number of routes lie within the budget and perform near optimally. But the problem scenarios where the probability measure of near optimal paths is small the runtime of the algorithm is unacceptable. Fig. 1 (a) and (b) presents a problem where the route has to pass through the three high value nodes to achieve close to optimal reward, while the budget is just sufficient to do so. RAOr needs to be able to sample from a highly restricted set of feasible near optimal sets. This leads to large run times.

2) Sampling Inadmissible Sets: RAOr can potentially spend a lot of time considering sets that are out of budget if budget is small as compared to graph size. As is demonstrated in Fig. 1 (c) and (d).

To alleviate the drawbacks of the problem we improve the algorithm such that it still keeps its global solution finding properties but improve its convergence properties in these pathological cases. These improvements are defined in the next section and the improved algorithm is described in Alg. 2.

V. Randomized Anytime Orienteering - Greedy (RAOr-G)

The RAOr algorithm (Alg. 1) uniformly samples in the space of sets to find the optimal set of nodes that should be in route. This provides with global solution optimality guarantees while sacrificing on run-times for some pathological cases.

We augment the algorithm to improve its anytime properties by leveraging the problem structure in the following ways -

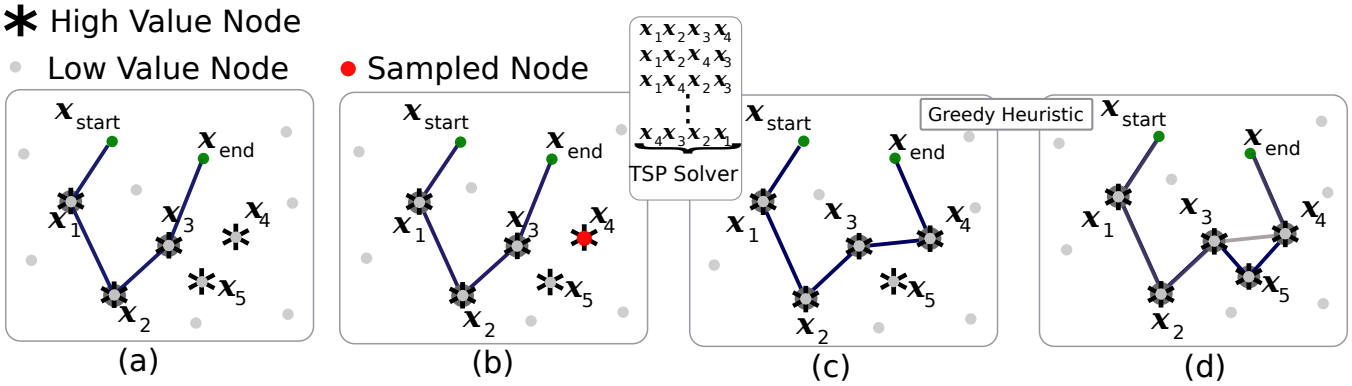

Fig. 2: An illustration how RAOr-G combines CSP and TSP solvers in combination with local greedy heuristics to explore the space of routes rapidly, resulting in improvement of run times for finding near optimal solutions for the correlated orienteering problem. a) Using the CSP algorithm the current admissible route is given by $r = x_{start}, x_1, x_2, x_3, x_{end}$. b) At the next step x_4 is sampled, and is to be added to the route. c) The exponential number of ways in which x_4 can be added to r is reduced to a near optimal order in polynomial time by the TSP solver. This step reduces an exponential search space with in polynomial computation costs. d) The new route obtained is then improved by conducting a local search using greedy heuristics.

Input: $G = [V, E]$, v_s, v_e, B, T_r
Output: The best route found in run-time T_r

1 $r_g = TSP(v_s, v_e)$ // Seed local search if no global solution is found
2 $R = r_g$
3 **if** $RouteLength(r_g) > B$ **then**
4 | **return** $\emptyset$
5 **end**
6 $S = SampleSet(V, v_s, v_e)$
7 $r = TSP(S, v_s, v_e)$
8 **if** $RouteLength(r) <= B$ **then**
9 | $R = R \cup r$
10 **end**
11 **for** $i = 1 : 3|V| \wedge runtime < T_r$ **do**
12 | $v_{new} = WeightedSample(V)$
13 | **if** $IsInRoute(r, v_{new})$ **then**
14 | | $r = DeleteFromRoute(r, v_{new})$
15 | **end**
16 | **else**
17 | | $r = AddToRoute(r, v_{new})$
18 | **end**
19 | **if** $RouteLength(r) <= B$ **then**
20 | | $R = R \cup r$
21 | **end**
22 | $R = GreedyLocalSearch(G, v_{new}, R, B)$
23 **end**
24 **return** $\arg\max_{r \in R} I(R)$

Algorithm 2: RAOr-Greedy(RAOr-G)

1) Conducting local searches in the space of routes.
2) Restricting the local search to admissible sets.
3) Informed sampling to improve the likelihood of sampling high value nodes.

The resulting algorithm is called Randomized Anytime Orienteering - Greedy (RAOr-G) and is presented in Alg. 2, Fig. 2.

Since, the RAOr-G (Alg. 2)is very similar to RAOr (Alg. 1) we highlight the differences here. RAOr-G is seeded with a route which consists of only v_s, v_e, Alg. 2, Line 1-2. The set of nodes is sampled as in RAOr (Alg. 1) but if the resulting route generated by running TSP on selected nodes lies within budget it is added to the set of feasible routes R, Alg. 2, Line 8-10. R is then used by the local search algorithm as the candidate set for running greedy local search, Alg. 2, Line 22. The $GreedyLocalSearch$ function is presented in Alg 3 and described in Section V-A. Another major difference is that instead of uniformly sampling for the node to add or delete from the route, the sampling is weighted in order to sample more valuable nodes often. The intuition behind weighted sampling is described in Section V-B.

A. Local Greedy Search Heuristic

Input: $G = [V, E]$, v_{new}, R, B
Output: Updated set R after conducting local greedy search

1 $r_c = FindBestRouteInBudget(v_{new}, R)$
2 **if** $r_c == \emptyset$ **then**
3 | $r_{cn} = Route(v_e, v_{new}, v_{new})$ // TSP
4 **end**
5 **else**
6 | $r_{cn} = AddToBestRoute(v_{new}, r_c)$ // Run a TSP with v_{new} added to r_c.
7 **end**
8 $R = R - r_c$
9 $r_{cn} = AddNearByNodes(G, r_{cn}, B, d, c)$
10 $R = R \cup r_{cn}$
11 **return** R

Algorithm 3: GreedyLocalSearch

Local greedy search heuristic is used to improve the runtime of RAOr algorithm and also to improve the anytime performance of the algorithm. The greedy search takes as input a set of feasible routes found so far, R and the sampled node v_{new}. It finds the route $r_c \in R$ according to equation 2.

$$r_c = \arg\max_{r \in R} \frac{I(r \cup v_{new}) - I(r)}{RouteLength(r \cup v_{new})} \quad (2)$$

$$\text{subject to } RouteLength(r \cup v_{new}) \leq B$$

The reward of the updated route is further improved by adding nodes that are within distance d and increased the reward gained by the route by at least c, while keeping it in budget (Alg. 4).

The total computation cost of a single iteration of local greedy search heuristic is $O(|R||V|)$. In practice we have found that the speedup achieved far outweighs the cost, Fig. 1.

Input: $G = [V, E]$, r_{cn}, B, d, c
Output: Updated set R after conducting local greedy search

forall the $v \in V | DistanceFromRoute(v, r_{cn}) <= d$ **do**
  if $(I(r_{cn} \cup v) - I(r_{cn})) \geq c$ **then**
    if $DistanceFromRoute(v, r_{cn}) * 2 \leq B - RouteLength(r_{cn})$ **then**
      $r_{cn} = AddToRoute(v, r_{cn})$
      // Run a TSP with v added to r_{cn}.
    end
  end
end
return r_{cn}

Algorithm 4: AddNearByNodes

In the next Section we describe the intuition behind using weighted sampling instead of uniform sampling and how it affects the theoretical guarantees provided by the RAOr-G algorithm (Alg 2).

B. Weighted Sampling

Weighted sampling, samples nodes with the probability directly proportional to the reward they offer independently. The intuition behind weighted sampling is that nodes with high values tend to be the part of optimal routes. Unfortunately weighted sampling adversely affects the theoretical guarantees of the algorithm, but empirically it leads to alleviating the problems caused by uniform sampling IV-A.1.

Theorem 3 (Optimality of Randomized Anytime Orienteering - Greedy)**.** *Randomized Anytime Orienteering - Greedy algorithm almost surely finds the optimal route from start to end nodes, within budget B/α, if there exists one in polynomial factor of $\left(\frac{2}{1+\zeta}\right)^{|V|}$ repetitions, where $\zeta = \left(\frac{((|V|-I_{min})I_{min})^{\beta}}{(|V|-1)^{(\beta+1)}}\right)$, $I_{min} = \min_{v \in V \text{ and } I(v) \neq 0} I(v)$, $\beta = \frac{1}{|V|-2}$.*

VI. Results

We evaluate the computational performance of RAOr on various benchmark examples against the state of the art methods. Then, we apply the algorithm on a realistic coverage scenario in simulation. All simulation computations are performed on a computer equipped with Intel Core i7-4870HQ using Matlab. We also deploy the algorithm on a UAV and demonstrate its application for exploration planning.

A. Computational Performance

The benchmark problem, Fig 3 consists of a graph with nodes located in a grid at uniform resolution. The reward for visiting each cell in the grid is 1. There is no reward for visiting a cell twice. Table I shows the comparative run times of state of the art algorithms vs RAOr-G for different problem sizes and varying budgets. Both RAOr-G and eSIP are anytime in nature. Each algorithm was stopped when it reached 95% of the optimal value. RAOr-G outperforms the state of the art in all the benchmark problems. RAOr-G solves the problem with 400 nodes approximately 100 times faster than the eSIP.

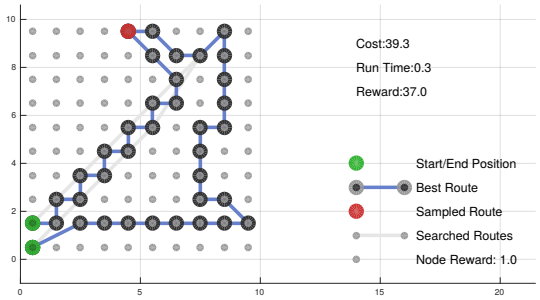

Fig. 3: Illustrated here is $10X10$ grid size benchmark problem with RRO-G run. For all the benchmark problems the nodes are situated in a uniform grid with 1 resolution. For this particular problem RAOr-G is able to find a near-optimal solution in 6.9 seconds while the state of the art takes 143.6 seconds for finding same quality of solution.

Grid Size	Budget		eSIP	MIQP	RAOr-G
5X5	30	cost	29.8	29.6	29.4
		utility	24	25	24
		time(s)	6.8	15.3	0.5
7X7	60	cost	56.6	57.3	56.1
		utility	48	49	48
		time(s)	26.7	1358.5	2.4
10X10	100	cost	99.0	99.8	98.9
		utility	87	92	87
		time(s)	143.6	15330	6.9
20X20	100	cost	99.7	-	99.2
		utility	87	-	87
		time(s)	763.7	-	7.2

TABLE I: Comparative run time analysis of state of the art v/s RAOr-G. RAOr-G consistently outperforms the state of the art in terms of run times for achieving near-optimal solutions. All the solutions obtained are atleast 95% of the optimal.

B. Correlated Rewards

In the first scenario we compared the computational performance against algorithms that promise optimality but on a simple case, where rewards of the nodes were independent. Here we evaluate RAOr-G's performance against algorithms that can potentially work on-board robots given their relatively low runtimes.

The algorithm has to find a route given start and end locations such that the reward collected is maximized. Reward at a node is defined by the region visible from a 25^o field of view, downward facing camera. Viewpoints exist on grid of $4m$, at a height of $10m$ and at a uniform distance of $20m$ at a height of $55m$. Resulting a total of 650 nodes/viewpoints. The area is randomly strewn with 5 high value objects, covering those from lower viewpoints results in a 10 times higher reward than other locations. Two distance metrics are implemented, Case 1 Euclidean distance and Case 2 Euclidean distance with travel along z-direction costing 3 times more vs. traveling in x-y plane, see figure 4. Since nodes/viewpoints are distributed in a uniform fashion, greedy solutions are able to perform near optimally in Case 1. RAOr-G performs comparably to the greedy solutions, while incurring a bit more runtime. In Case 2, greedy solutions are clearly stuck in a local maxima, leading to a much

worst performance than RAOr-G. In both cases RAOr-G outperforms RIG algorithm both in runtimes and solution quality, see figure 4.

C. Robotic Exploration

The algorithm was deployed on an autonomous UAV system described in this link-`https://goo.gl/bNr6Du`. It ran at $3Hz$ on an Odroid-C2. Figure 5 describes the vehicle mission. The vehicle is deployed to scout for cars and collect high resolution data if a car is found. The exploration reward function is weighted probability distance between current robot's belief and expected updated belief and a fixed reward of 20000 is allocated to scanning a car, for more details see following link-`https://goo.gl/bNr6Du`.The algorithm is run adaptively as the vehicle's representation is updated and is able to guide the vehicle to explore the environment and locate and map both the cars, see figure 5.

VII. Conclusion

The paper presents RAOr-G, an anytime, near-optimal, randomized algorithm for solving correlated orienteering problems. Preliminary results presented in the paper demonstrate the algorithm to be an order of magnitude faster than state of the art. Infact, the run-times of the RAOr-G algorithm allowed us to deploy the algorithm on-board a UAV for the purpose of budgeted data gathering.

Future work will involve rigorous bench marking of the algorithm against the state of the art methods. We will also test the algorithm as the information gathering planner on more complex scenarios on the UAV. Since, RAOr-G is the global exploration planning algorithm, we will implement a local data gathering algorithm similar to [15] and ensure vehicles safety as described in [16].

VIII. Appendix

Proposition 1 (Returning optimal route on optimal set selection)**.** *If RAOr finds the optimal set of nodes to visit during it's run-time, it is guaranteed to return the corresponding optimal route, if an optimal TSP solver is used.*

Proof. In every iteration, RAOr (Alg. 1) checks if the set of nodes selected in that iteration can be visited within the budget constraint. The optimal order to visit the selected set is given by a TSP solver. If the selected set can be visited within the budget, it is stored as the best set encountered if it is more rewarding than the previous best set encountered. The most rewarding set encountered in the budget is returned. Therefore, if set V^* is encountered it is guaranteed that r^* will be found and returned, given we have an optimal TSP solver. Hence, if RAOr finds the optimal set of nodes to visit during it's run-time, it is guaranteed to return the corresponding optimal route, if an optimal TSP solver is used. □

We next compute the probability of finding the optimal assignment or equivalently the optimal path using $RAOr$.

Proposition 2 (Probability of finding the correct path)**.** *Randomized Anytime Orienteering algorithm finds the optimal route from start to end nodes, within budget B, if there exists one, with a probability of at least $\left(\frac{1}{2}\left(1+\frac{1}{|V|-1}\right)\right)^{|V|}$ in one run.*

Proof. The proof of this theorem is taken from the work on probabilistic algorithm for constraint satisfaction problem [13].

Given a_{r^*} is the optimal solution, we want estimate the lower bound on the probability that Alg. 1 finds a_{r^*}. Once we have found this success probability p, the expected number of independent repetitions of the procedure until we find the optimal solution is $1/p$.

Now, we calculate p. It is clear that the random variable X that counts the number of bits in which the random assignment a and the fixed assignment a_{r^*} disagree (i. e. the Hamming distance between a and a_{r^*}) is binomially distributed. That is, $Pr(X=j)=\binom{|V|}{j}2^{-|V|}$. If the system is in state 0, this means, an optimal assignment has been found.

At any given point in the algorithm, if a is not the optimal assignment then there must be atleast one vertex out of $|V|$ that needs to be flipped (included or excluded from the set selection) to reduce the hamming distance of a to a_{r^*} reduces by 1. Selecting the correct vertex would mean that the current state transfers j to transfers to state $j-1$ with probability at least $1/|V|$, and transfers to state $j+1$ with probability at most $\frac{|V|-1}{|V|}$. This markov chain is the same as described in [13]. We present the proof of value of p here for completeness.

Given that the process has initially transferred into state j, we calculate the probability q_j that the process reaches the absorbing state 0. For this to happen the process needs at least j steps. We consider the case that the random walk takes $i \leq j$ steps in the "wrong" direction and $i+j$ steps are required toward the "right" direction so that the process stops in state 0 after $j+2i$ steps. To calculate this probability requires us to calculate the number of paths on a rectangular grid (which represents the possible movements over the Markov chain over the time scale) which transfers the process from state j to state 0 while exactly i steps in the "wrong" direction. Using the ballot theorem from [17], page 73, it can be seen that this number is $\binom{j+2i}{i}.\frac{j}{j+2i}$. Therefore, the probability can be estimated as follows.

$$q_j \geq \sum_{i=0}^{j}\binom{j}{j+2i}.\frac{j}{j+2i}.\left(\frac{|V|-1}{|V|}\right)^{i}.\left(\frac{1}{|V|}\right)^{i+j}$$
$$q_j \geq \frac{1}{3}\sum_{i=0}^{j}\binom{j}{j+2i}.\left(\frac{|V|-1}{|V|}\right)^{i}.\left(\frac{1}{|V|}\right)^{i+j} \quad (3)$$

Further we can lower bound the above sum by its largest term as follows. We use the following fact [18]. $\binom{n}{\beta n} \sim 2^{h(\beta)n} = \left(\frac{1}{\beta}\right)^{\beta n}\left(\frac{1}{1-\beta}\right)^{(1-\beta)n}$ where $h(\beta) = -\beta\log_2\beta - (1-\beta)\log_2(1-\beta)$ is the binary entropy function. In particular, the two functions

$$\binom{(1+2\beta)j}{\beta j} \text{ and } \left[\left(\frac{1+2\beta}{\beta}\right)^{\beta}.\left(\frac{1+2\beta}{1+\beta}\right)^{1+\beta}\right]^{j} \quad (4)$$

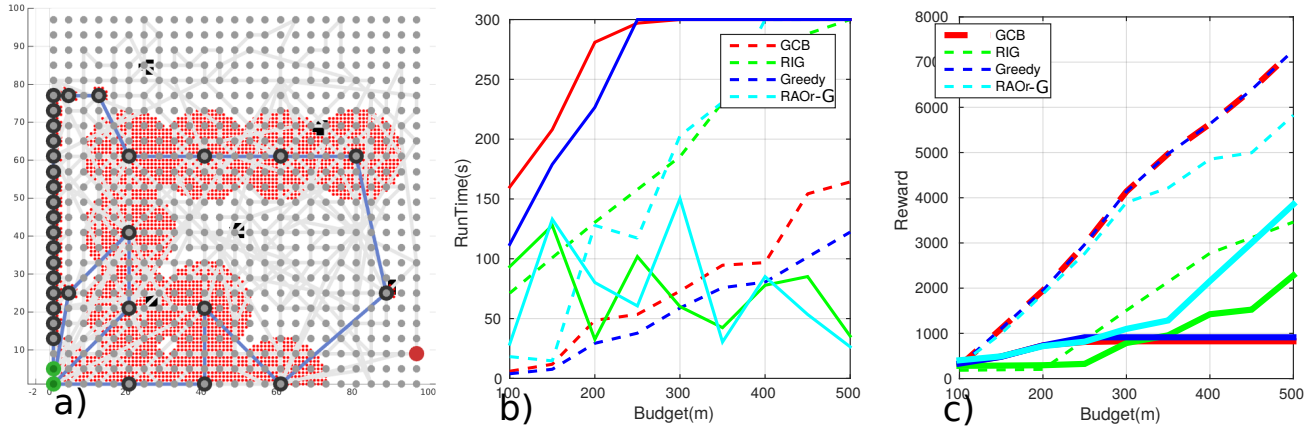

Fig. 4: a)RAOr-G planned path with a $500m$ budget on the $100X100$ area for $Case2$. Red marks the sensor footprint, if the vehicle flies low, the sensor footprint is small. Notice how the path visits high value regions, displayed in black. Grey paths show all the paths searched by RAOr-G. b) and c) Case 1 is displayed in dashed line, Case 2 in solid lines. RAOr-G is competitive with greedy algorithms in Case 1 and dramatically outperforms greedy and RIG for Case 2, where greedy algorithms are stuck in local maxima.

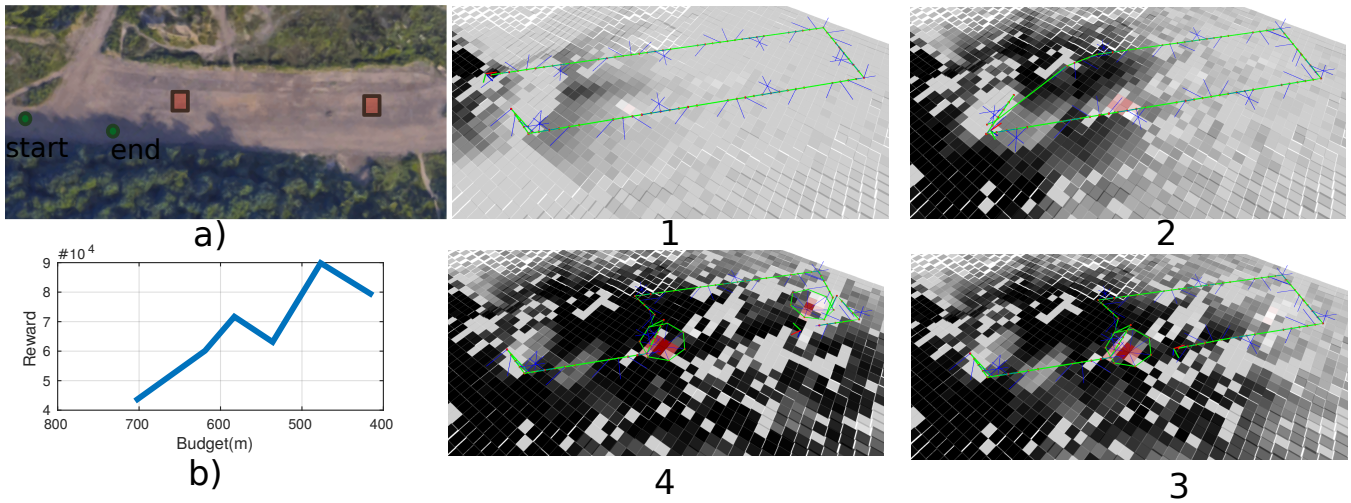

Fig. 5: a) Testing site, start and end are marked by green nodes and car locations are shown in orange. b) Vehicle starts with a budget of $700m$, the reward increases as likelihood of finding the car increases, the crest in reward marks the time at which the global planner found and decided to map the car. Figures 1,2,3 and 4 show the series of plans at various stages of the exploration mission, Dark squares indicate absence of cars and red squares presence of cars. Shades of grey and red signify certainty. Once the car is recognized, a 360 view of the car is associated with an extra reward.

are within polynomial factors of each other. We lower bound the above estimation for q_j by setting $\beta = \frac{1}{|V|-2}$.

$$q_j \geq \frac{1}{3}\sum_{i=0}^{j}\binom{j}{j+2i}\cdot\left(\frac{|V|-1}{|V|}\right)^{i}\cdot\left(\frac{1}{|V|}\right)^{i+j}$$

$$\geq \left[\left(\frac{1+2\beta}{\beta}\right)^{\beta}\cdot\left(\frac{1+2\beta}{1+\beta}\right)^{1+\beta}\left(\frac{|V|-1}{|V|}\right)^{\beta}\cdot\left(\frac{1}{|V|}\right)^{1+\beta}\right]^{j}$$

$$\text{where } \beta = \frac{1}{|V|-2}$$

$$= \left(\frac{1}{|V|-1}\right)^{j} \tag{5}$$

where the last inequality holds up to some polynomial factor. Therefore, up to some polynomial factor, using the binomial theorem, we obtain the following estimate for success probability p

$$p \geq (\frac{1}{2})^{|V|}\sum_{j=0}^{|V|}\binom{|V|}{j}\left(\frac{1}{|V|-1}\right)^{j}$$

$$= \left(\frac{1}{2}\left(1+\frac{1}{|V|-1}\right)\right)^{|V|} \tag{6}$$

□

Theorem 1 (Optimality of Randomized Anytime Orienteering)**.** *Randomized Anytime Orienteering algorithm almost surely finds the optimal route from start to end nodes, within budget B, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions.*

Proof. According to Proposition 2 the RAOr algorithm

finds the optimal set with probability atleast $p \geq (\frac{1}{2})^{|V|} \sum_{j=0}^{|V|} \binom{|V|}{j} \left(\frac{1}{|V|-1}\right)^j = \left(\frac{1}{2}\left(1+\frac{1}{|V|-1}\right)\right)^{|V|}$. The expected number of independent repetitions of the $RAOr$ algorithm are $1/p$. The probability that $RAOr$ will not find a satisfying assignment after h repetitions is at most $(1-p)^h \leq e^{-ph}$. Therefore, to achieve acceptable chance of missing the optimal path, say e^{-50}, we can choose $h = 50/p$. So the complexity is within a polynomial factor of $1/p$. Therefore, Randomized Anytime Orienteering algorithm almost surely finds the optimal route from start to end nodes, within budget B, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions. □

Theorem 2 (α-Optimality of Randomized Anytime Orienteering Algorithm)**.** *Randomized Anytime Orienteering algorithm almost surely finds the optimal route from start to end nodes, within budget B/α, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions. If the TSP solver in the inner loop is α-approximate.*

Proof. If the TSP solver is α optimal, if V^* is encountered during set selection, the length of the route covering the set may not fit inside the budget constraint.

Although, if V_b^* is the most rewarding set that can be visited in $\frac{B}{\alpha}$, then it can be guaranteed that if V_b^* set is selected, it will be returned as the most rewarding set by $RAOr$. The probability of finding V_b^* is presented in equation 6 and hence the $RAOr$ algorithm almost surely finds the optimal route from start to end nodes, within budget B/α, if there exists one, within a polynomial factor of $(2(1-1/|V|))^{|V|}$ repetitions. If the TSP solver in the inner loop is α-approximate. □

Theorem 3 (Optimality of Randomized Anytime Orienteering - Greedy)**.** *Randomized Anytime Orienteering - Greedy algorithm almost surely finds the optimal route from start to end nodes, within budget B/α, if there exists one in polynomial factor of $\left(\frac{2}{1+\zeta}\right)^{|V|}$ repetitions, where $\zeta = \left(\frac{((|V|-I_{min})I_{min})^\beta}{(|V|-1)^{(\beta+1)}}\right)$, $I_{min} = \min_{v \in V \text{ and } I(v)\neq 0} I(v)$, $\beta = \frac{1}{|V|-2}$.*

Proof. $RAOr-G$ relies on the same process as $RAOr$ for its global convergence characteristics. The random variable X that counts the number of bits in which the random assignment a_r and the fixed assignment a_{r^*} disagree (i. e. the Hamming distance between a and a_{r^*}) is binomially distributed. That is, $Pr(X=j) = \binom{|V|}{j} 2^{-|V|}$. If the system is in state 0, this means, an optimal assignment has been found.

At any given point in the algorithm, if a is not the optimal assignment then there must be atleast one vertex out of $|V|$ that needs to be flipped to reduce the hamming distance of a to a_{r^*} reduces by 1. Selecting the correct vertex would mean that the current state transfers j to transfers to state $j-1$ with probability at least $\frac{I_{min}}{|V|}$ and transfers to state $j+1$ with probability at most $1-\frac{I_{min}}{|V|}$. The change in probabilities is due to the weighted sampling of nodes instead of uniform sampling.

Substituting these values in equation 5 and following the algebra we get the following.

$$q_j \geq \left(\frac{((|V|-I_{min})I_{min})^\beta}{(|V|-1)^{(\beta+1)}}\right)^j = \zeta^j \tag{7}$$

where the inequality holds up to some polynomial factor. Therefore, up to some polynomial factor, using the binomial theorem , we obtain the following estimate for success probability p

$$p \geq (\frac{1}{2})^{|V|} \sum_{j=0}^{|V|} \binom{|V|}{j} \zeta^j = \left(\frac{1}{2}(1+\zeta)\right)^{|V|} \tag{8}$$

Hence the number of repetitions required for $RAOr-G$ to find the optimal route is given by a polynomial factor of $\left(\frac{2}{1+\zeta}\right)^{|V|}$. Where ζ is given by $\left(\frac{((|V|-I_{min})I_{min})^\beta}{(|V|-1)^{(\beta+1)}}\right)$. □

References

[1] A. Krause, "Optimizing sensing: Theory and applications," Ph.D. dissertation, Carnegie Mellon University, December 2008.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. IEEE, 1997, pp. 146–151.

[3] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4791–4798.

[4] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 342–349.

[5] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, pp. 707–755, 2009.

[6] C. CHEN, S.-F. Cheng, and H. C. Lau, "Multi-agent orienteering problem with time-dependent capacity constraints," *Web Intelligence and Agent Systems*, 2014.

[7] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," 2016.

[8] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987.

[9] C. Chekuri, N. Korula, and M. Pál, "Improved algorithms for orienteering and related problems," *ACM Transactions on Algorithms (TALG)*, vol. 8, no. 3, p. 23, 2012.

[10] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[11] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[12] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *Foundations of Computer Science*. IEEE, 2005.

[13] U. Schöning, "A probabilistic algorithm for k-sat and constraint satisfaction problems," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 1999, pp. 410–414.

[14] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," DTIC Document, Tech. Rep., 1976.

[15] S. Arora and S. Scherer, "Pasp: Policy based approach for sensor planning," in *2015 ICRA*. IEEE, 2015, pp. 3479–3486.

[16] S. Arora, S. Choudhury, D. Althoff, and S. Scherer, "Emergency maneuver library-ensuring safe navigation in partially known environments," in *2015 ICRA*. IEEE, 2015, pp. 6431–6438.

[17] W. Feller, "An introduction to probability theory and its applications. vol. i." 1950.

[18] R. Motwani and P. Raghavan, *Randomized algorithms*. Chapman & Hall/CRC, 2010.