

# Searching k-Optimal Goals for a Orienteering Problem on a Specialized Graph with Budget Constraints

Abhinav Sharma, Advait Deshpande, Yanming Wang, Xinyi Xu

<sup>1</sup>University of Melbourne  
Parkville Campus  
Victoria, Australia 3010

## Abstract

In this paper, we are proposing a novel non-randomized anytime orienteering algorithm for finding k-optimal goals that maximizes reward on a specialized graph with budget constraints. This specialized graph represents a real-world scenario of finding k-most optimal goal states which is analogous to an orienteering problem (OP).

## Introduction

Orienteering Problem (OP) is a special case of the Informative Path Planning (IPP) problem where rewards at different nodes are calculated independently of each other. However, the OP is considered to be NP-hard and mostly solved with heuristic-based search strategies and customized algorithms (Wei and Zheng 2020). In this research work, we are trying to solve a domain-related orienteering problem which can be formalized for a specialized directed weighted graph. We created a specialized graph for mapping our Parkville campus of the University. This specialized graph can be used to formalize our problem of finding the most optimal nearest building from a starting building such that reward can be maximized within the provided travelling budget constraint. Using our proposed non-randomized algorithm, we are going to find the k-most optimal nearest buildings inside our campus from a given starting building as shown in the Experimental Results. We are also going to show how we can easily include COVID lockdown situation in our algorithm to solve our defined orienteering problem.

## Problem Formulation

In this section, we are going to formulate our domain-related optimal building finding problem into a generic orienteering problem (OP) for a specialized graph.

Let us assume a weighted directed specialized graph  $G_s = (V, E)$  for  $n$  number of nodes where  $v_s \in V$  is the pre-defined start node such that  $V = \{v_1, v_2, v_3, \dots, v_n\}$  and  $E = \{(v_s, v_i) \setminus (v_s, v_s) \mid \forall i \in [1, n]\}$ . Here,  $v_s$  is having  $n$  out-degree with 0 in-degree i.e. connected to every other node in  $V$  and  $v_i \forall i \in [1, n] \setminus v_s$  is connected to only  $v_s$  with 1 in-degree and 0 out-degree.

Let  $v_g$  be the set of  $k$ -optimal goal nodes s.t.  $v_g \in V$  and  $k \leq n$ . These goals are attained in the decreasing order of their gained rewards after respecting budget constraints i.e.  $v_{g1} > v_{g2} > \dots > v_{gk}$ . Let  $r$  be the set of nodes which we can visit such that  $r \subseteq V \setminus v_s$ . Let  $B$  be the travelling budget which will help us to enable budget constraints. Let  $O$  be the generic objectives and  $F$  be the generic factors which can be used to tweak the reward function of our problem.

For each  $r$ , let  $R(r, o, f)$  be the reward function where  $R : (r, o, f) \rightarrow \mathbb{R}_0^+ \cup \{\infty\}$  calculates reward based on the provided factors  $f \subseteq F$  and objective  $o \in O$ . Let  $I(r) = R(r, o, f)$  be the reward gained by visiting each node in  $r$ . Let the cost of traversal be given by  $C(r) = C(v_s, v_i^r)$ , where  $v_i$  is the  $i^{th}$  element in  $r$ ,  $\forall i \in [1, |r|]$ . Let  $L \in \mathbb{R}_0^+ \cup \{\infty\}$  be the constraint limit. Using above notations, the hard-constraint problem can then be defined by the equation 1.

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

We can relax the above hard-constraint by introducing a hyperparameter  $\delta$  to formulate a soft-constraint problem as shown in the equation 2.

$$\arg \max_{r \subseteq V} I(r) \text{ subject to } C(r) \leq B + \delta \leq L \quad (2)$$

where  $\delta \in \mathbb{R}_0^+ \cup \{\infty\}$ .

The solution to our stated problem is a set of ordered  $k$ -optimal goal nodes, such that the reward attained by visiting the node is maximized while the path cost stays within a specified travelling budget  $B$ .

## Non-randomized Anytime Orienteering

In this section, we propose a novel way of solving the problem formulation shown in the equation 2 which is inspired by the general randomized algorithm for IPP problems (Arora and Scherer 2017).

The algorithm starts with a priority queue and creates  $r$  subset s.t.  $r \subseteq V \setminus v_s$ . Then, for each node in  $r$ , path cost  $C(r)$  and node reward  $I(r)$  is calculated. It is then ensured that the budget constraint is satisfied and selected node is pushed into the priority queue with negative reward as priority. As a result, we can easily pop the queue item with minimum priority  $k$ -times to find the  $k$ -most optimal goal nodes. This process is described in Algorithm 1.

---

**Algorithm 1:** Non-Randomized Anytime Orienteering to find  $k$ -optimal goals for a specialized graph

---

**Inputs:**  $G_s = (V, E), v_s, B, L, k, \delta, o \in O, f \subseteq F$

**Output:**  $v_g = \{v_{g1}, \dots, v_{gk}\}$  s.t.  $v_{g1} > \dots > v_{gk}$

queue := new priority queue

$v_g = \emptyset$

$r := r \subseteq V \setminus v_s$

**for**  $v_i$  **in**  $r$  **do**

$I(r) = R(r, o, f)$  //node reward

$C(r) = C(v_s, v_i)$  //path cost

**if**  $C(r) \leq B + \delta \leq L$  **then**

$priority = -1 * I(r)$

        queue.insert( $v_i, priority$ )

**end**

**end**

**while** not queue.empty() **do**

$\rho := \text{queue.pop-min()}$  //best node

**if**  $\text{len}(v_g) < k$  **then**

$v_g := v_g \cup \rho$

**end**

**end**

---

**Time Complexity.** If we assume standard binary heap implementation of the priority queue then the insertion and deletion time complexity is  $O(\log n)$ , where  $n$  is the size of the input (Atkinson et al. 1986). This can be further optimized by several customizations (Edelkamp, Elmasry, and Katajainen 2017). Hence, the time complexity of our proposed algorithm for the best and the worst case can be stated as  $O(n - 1 * \log n) + O(k * \log n) \leq O(n \log n)$ .

**Space Complexity.** If we again assume heap data structure implementation of the priority queue, then the space complexity of storing  $n$  elements in the priority queue is  $O(n)$  (Atkinson et al. 1986). Hence, the best and worst case space complexity of our proposed algorithm is  $O(n)$ .

**Limitations.** Our algorithm relies on the assumption that the graph is a specialized weighted directed graph with 1 central node (0 in-degree and  $n$  out-degree) and  $n$  isolated nodes connected with only 1 central node. Due to this assumption, the algorithm is efficient and applicable only for such versions of the specialized graph and cannot be extended implicitly to any general weighted directed graph.

## Experimental Results

In this section, we are going to show experimental results for a domain-specific orienteering problem solved using our proposed algorithm. In our problem, we need to find the  $k$ -most optimal nearest building inside our Parkville campus of the University. These buildings should be within a specific radius ( $B$ ) that maximises the chances (reward) of either booking a meeting room or using a toilet facility based on supply, demand and other preferences or factors. A specific scenario is shown in Figure 1 where  $R(\cdot)$  are the rewards given by the buildings with no factors and  $R(\text{COVID})$  are the rewards based on COVID lockdown situation.

Using our proposed algorithm, we were able to solve the stated scenario with the results as shown in the Table 1. These results are based on the fact that there is no

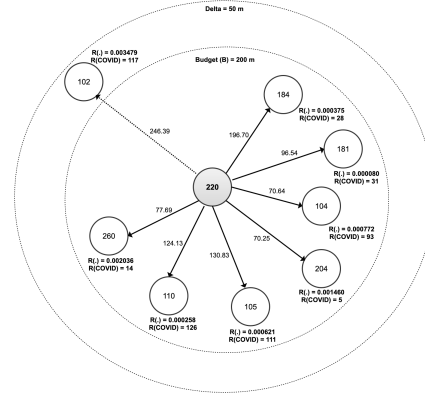


Figure 1: Finding  $k = 3$  most optimal nearest building from  $v_s = 220$  that maximises the chances (reward) of booking a meeting room within  $B = 200$  meters and  $\delta = 50$  meters

COVID lockdown situation. We were also able to simulate COVID lockdown scenario by enhancing the reward function  $R(r, o, f)$  in which our algorithm provides results as shown in the Table 2.

Goals	Cost	$R(\cdot)$	Goals	Cost	$R(\cdot)$
$v_{g1} = 260$	77.69	0.002036	$v_{g1} = 102$	246.39	0.003479
$v_{g2} = 204$	70.25	0.001460	$v_{g2} = 260$	77.69	0.002036
$v_{g3} = 104$	70.64	0.000772	$v_{g3} = 204$	70.25	0.001460

Table 1:  $k = 3$  most optimal nearest buildings without any factors with  $B = 200$  m hard-constraint (left) and  $B + \delta = 250$  m soft-constraint (right)

Goals	Cost	$R(\text{COVID})$	Goals	Cost	$R(\text{COVID})$
$v_{g1} = 110$	124.13	126	$v_{g1} = 110$	124.13	126
$v_{g2} = 105$	130.83	111	$v_{g2} = 102$	246.39	117
$v_{g3} = 104$	70.64	93	$v_{g3} = 105$	130.83	111

Table 2:  $k = 3$  most optimal nearest buildings in COVID lockdown situation with  $B = 200$  m hard-constraint (left) and  $B + \delta = 250$  m soft-constraint (right)

## References

- Arora, S.; and Scherer, S. 2017. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 4997–5004.
- Atkinson, M. D.; Sack, J.-R.; Santoro, N.; and Strothotte, T. 1986. Min-Max Heaps and Generalized Priority Queues. *Commun. ACM* 29(10): 996–1000. ISSN 0001-0782. doi:10.1145/6617.6621. URL <https://doi.org/10.1145/6617.6621>.
- Edelkamp, S.; Elmasry, A.; and Katajainen, J. 2017. Optimizing Binary Heaps. *Theory of Computing Systems* 61. doi:10.1007/s00224-017-9760-2.
- Wei, Y.; and Zheng, R. 2020. Informative Path Planning for Mobile Sensing with Reinforcement Learning. doi:10.1109/INFOCOM41043.2020.9155528.