Noname manuscript No.
(will be inserted by the editor)

On the Adaptivity Gap of Stochastic Orienteering

Nikhil Bansal · Viswanath Nagarajan

Received: date / Accepted: date

Abstract The input to the *stochastic orienteering* problem [14] consists of a budget B and metric (V, d) where each vertex $v \in V$ has a job with a deterministic reward and a *random* processing time (drawn from a known distribution). The processing times are independent across vertices. The goal is to obtain a non-anticipatory policy (originating from a given root vertex) to run jobs at different vertices, that maximizes expected reward, subject to the total distance traveled plus processing times being at most B. An *adaptive* policy can choose the next vertex to visit based on observed random instantiations. Whereas, a *non-adaptive* policy is just given by a fixed ordering of vertices. The *adaptivity gap* is the worst-case ratio of the optimal adaptive and non-adaptive rewards. We prove an $\Omega\left((\log\log B)^{1/2}\right)$ lower bound on the adaptivity gap of stochastic orienteering. This provides a negative answer to the $O(1)$-adaptivity gap conjectured by Gupta et al. [14], and comes close to the $O(\log\log B)$ upper bound. This result holds even on a line metric.

We also show an $O(\log\log B)$ upper bound on the adaptivity gap for the *correlated* stochastic orienteering problem, where the reward of each job is random and possibly correlated to its processing time. Using this, we obtain an improved quasi-polynomial time $\min\{\log n, \log B\} \cdot \tilde{O}(\log^2 \log B)$-approximation algorithm for correlated stochastic orienteering.

Keywords approximation algorithms · stochastic · vehicle routing

N. Bansal's research was supported by the NWO Grant 639.022.211 and ERC consolidator grant 617951.

N. Bansal
Department of Mathematics and Computer Science, Eindhoven University of Technology
E-mail: n.bansal@tue.nl

V. Nagarajan
Industrial and Operations Engineering Department, University of Michigan
E-mail: viswa@umich.edu

Mathematics Subject Classification (2010) 68W25 · 90B36 · 90B15 · 90B06

1 Introduction

In the *orienteering* problem [11], we are given a metric (V, d) with a starting vertex $\rho \in V$ and a budget B on length. The objective is to compute a path originating from ρ having length at most B, that maximizes the number of vertices visited. This is a basic vehicle routing problem (VRP) that arises as a subroutine in algorithms for a number of more complex variants, such as VRP with time-windows, discounted reward TSP and distance constrained VRP.

The stochastic variants of orienteering and related problems such as traveling salesperson and capacitated vehicle routing have also been extensively studied. In particular, several dozen variants have been considered depending on which parameters are stochastic, the choice of the objective function, the probability distributions, and optimization models such as *a priori* optimization, stochastic optimization with recourse, probabilistic settings and so on. For more details we refer to a recent survey [19] and references therein.

Here, we consider the following stochastic version of the orienteering problem defined by Gupta et al. [14]. Each vertex contains a job with a deterministic reward and random processing time (also referred to as size); these processing times are independent across vertices. The processing times model the random delays encountered at the node, say due to long queues or activities such as filling out a form, before the reward can be collected. The distances in the metric correspond to travel times between vertices, which are deterministic. The goal is to compute a *policy*, which describes a path originating from the root ρ that visits vertices and runs the respective jobs, so as to maximize the total expected reward subject to the total time (for travel plus processing) being at most B. Stochastic orienteering also generalizes the well-studied stochastic knapsack problem [9,5,4] (when all distances are zero). We also consider a further generalization, where the reward at each vertex is also random and possibly *correlated* to its processing time.

A feasible solution (policy) for the stochastic orienteering problem is represented by a decision tree, where nodes encode the "state" of the solution (previously visited vertices and the residual budget), and branches denote random instantiations. Such solutions are called *adaptive* policies, to emphasize the fact that their actions depend on previously observed random outcomes. Adaptive policies can be very complex and hard to reason about. For example, even for the stochastic knapsack problem an optimal adaptive strategy may have exponential size (and several related problems are PSPACE-hard) [9].

Thus a natural approach for designing algorithms in the stochastic setting is to: (i) restrict the solution space to the simpler class of *non adaptive* policies (eg. in our stochastic orienteering setting, such a policy is described by a fixed permutation to visit vertices in, until the budget B is exhausted), and (ii) design an efficient algorithm to find a (close to) optimum non-adaptive policy.

While non-adaptive policies are often easier to optimize over, the drawback is that they could be much worse than the optimum adaptive policy. Thus, a key issue is to bound the *adaptivity gap*, introduced by Dean et al. [9] in their seminal paper, which is the worst-case ratio (over all problem instances) of the optimal adaptive value to the optimal non-adaptive value.

In recent years, increasingly sophisticated techniques have been developed for designing good non-adaptive policies and for proving small adaptivity gaps [9, 12, 8, 2, 13, 14]. For stochastic orienteering, Gupta et al. [14] gave an $O(\log\log B)$ bound on the adaptivity gap, using an elegant probabilistic argument (previous approaches only gave a $\Theta(\log B)$ bound). More precisely, they considered certain $O(\log B)$ correlated probabilistic events and used martingale tail bounds on suitably defined stopping times to bound the probability that none of these events happen. In fact, Gupta et al. [14] conjectured that the adaptivity gap for stochastic orienteering was $O(1)$, suggesting that the $O(\log\log B)$ factor was an artifact of their analysis.

1.1 Our Results and Techniques

Adaptivity gap for stochastic orienteering: Our main result is the following lower bound.

Theorem 1 *The adaptivity gap of stochastic orienteering is* $\Omega\left((\log\log B)^{1/2}\right)$, *even on a line metric.*

This answers negatively the conjecture of $O(1)$-adaptivity gap [14], and comes close to the $O(\log\log B)$ upper bound proved there. To the best of our knowledge, this gives the first non-trivial $\omega(1)$ adaptivity gap for a natural problem. We note that the stochastic knapsack special case is known to have adaptivity gap at most four [9]. Moreover, if we consider a variant of stochastic orienteering with two *distinct* budgets for travel and processing, it also has an $O(1)$ adaptivity gap [12, 14]. Thus the above $\omega(1)$ adaptivity gap result is somewhat surprising. The high-level reason for this result is the combination of the travel and processing times into a single budget.

The lower bound proceeds in two steps and is based on a somewhat intricate construction. We begin with a basic instance described by a directed binary tree of height $\log\log B$ that essentially represents the optimal adaptive policy. Each processing time is a Bernoulli random variable: it is either zero, in which case the optimal policy goes to its left child, or a carefully set positive value, in which case the optimal policy goes to its right child. The edge distances and processing times are chosen so that when a non-zero size instantiates, it is always possible to take a right edge, while the left edges can only be taken a few times. On the other hand, if the non-adaptive policy chooses a path with mostly right edges, then it cannot collect too much reward.

First, we show that the adaptivity gap of this directed tree instance is $\Omega((\log\log B)^{1/2})$. The main technical difficulty here is to show that every fixed path (which may possibly skip vertices, and gain advantage over the adaptive policy) either runs out of budget B or collects low expected reward.

In the second step, we embed the directed tree onto an undirected line in the obvious way. The issue here is that pairs of nodes that are far apart (or not even reachable from one to the other) on the directed tree may be very close on the line. To get around this, we exploit the asymmetry of the tree distances and some other structural properties to show that the adaptivity gap continues to hold (up to constant factors).

Correlated Stochastic Orienteering: Next, we consider the correlated stochastic orienteering problem, where the reward at each vertex is also random and possibly correlated with its processing time (the distributions are still independent across vertices). In this setting, we prove the following.

Theorem 2 *The adaptivity gap of correlated stochastic orienteering is* $O(\log\log B)$.

This improves upon the $O(\log B)$-factor adaptivity gap from Gupta et al. [14], and matches the adaptivity gap upper bound known for uncorrelated stochastic orienteering. The proof makes use of a martingale concentration inequality [17], as for the uncorrelated problem [14], but dealing with the reward-size correlations requires a different definition of the stopping time. For the uncorrelated case [14] the stopping time used a single "truncation threshold" (equal to B minus the travel time) to compare the instantiated sizes and their expectation. In the correlated setting, we use $\log B$ different truncation thresholds (all powers of 2), irrespective of the travel time, to determine the stopping criteria.

Using some structural properties in the proof of the adaptivity gap upper bound above, we obtain an improved *quasi-polynomial*[1] time algorithm for correlated stochastic orienteering.

Theorem 3 *There is an* $O\left(\alpha \cdot \log^2 \log B / \log\log\log B\right)$*-approximation algorithm for correlated stochastic orienteering, running in time* $(n+\log B)^{O(\log B)}$. *Here* $\alpha \leq \min\{O(\log n),\, O(\log B)\}$ *denotes the best approximation ratio for the orienteering with deadlines problem.*

The *orienteering with deadlines* problem is defined formally in Section 1.3. Previously, a polynomial time $O(\alpha \cdot \log B)$-approximation algorithm was known for correlated stochastic orienteering [14]. They also showed that this problem is at least as hard to approximate as the deadline orienteering problem, i.e. an $\Omega(\alpha)$-hardness of approximation (this result also holds for quasi-polynomial time algorithms). Our algorithm improves the approximation ratio to $O(\alpha \cdot \log^2 \log B)$, but at the expense of quasi-polynomial running time. We note that the running time in Theorem 3 is quasi-polynomial for general inputs where probability distributions are described *explicitly*, since the input size is $n \cdot B$. If probability distributions are specified implicitly, the runtime is quasi-polynomial only for $B \leq 2^{poly(\log n)}$.

[1] A quasi-polynomial time algorithm runs in $2^{\log^c N}$ time on inputs of size N, where c is some fixed constant.

The algorithm in Theorem 3 is based on finding an approximate non-adaptive policy, and losing an $O(\log\log B)$-factor on top by Theorem 2. There are three main steps in the algorithm: (i) we enumerate over $\log B$ many "portal" vertices (suitably defined) on the optimal policy; (ii) using these portal vertices, we solve (approximately) a *configuration LP relaxation* for paths between portal vertices; (iii) we randomly round the LP solution. The quasi-polynomial running time is only due to the enumeration. In formulating and solving the configuration LP relaxation, we also use some ideas from the earlier $O(\alpha\cdot\log B)$-approximation algorithm [14]. Solving the configuration LP requires an algorithm for deadline orienteering (as the dual separation oracle), and incurs an α-factor loss in the approximation ratio. This configuration LP is a "packing linear program", for which we can use fast combinatorial algorithms [16,10]. The final rounding step involves randomized rounding with alteration, and loses an extra $O(\frac{\log\log B}{\log\log\log B})$ factor.

1.2 Related Work

The deterministic orienteering problem was introduced by Golden et al. [11]. It has several applications, and many exact approaches and heuristics have been applied to this problem, see eg. the survey [18]. The first constant-factor approximation algorithm was due to Blum et al. [6]. The approximation ratio has been improved [1,7] to the current best $2+\epsilon$.

Dean et al. [9] were the first to consider stochastic packing problems in this adaptive optimization framework: they introduced the *stochastic knapsack* problem (where items have random sizes), and obtained a constant-factor approximation algorithm and adaptivity gap. The approximation ratio has subsequently been improved to $2+\epsilon$, due to Bhalgat et al. [5,4]. The stochastic orienteering problem [14] is a common generalization of both deterministic orienteering and stochastic knapsack.

Gupta et al. [13] studied a generalization of the stochastic knapsack problem, to the setting where the reward and size of each item may be correlated, and gave an $O(1)$-approximation algorithm and adaptivity gap for this problem. Recently, Ma [15] improved the approximation ratio to $2+\epsilon$.

The correlated stochastic orienteering problem was studied in [14], where the authors obtained an $O(\log n\cdot\log B)$-approximation algorithm and an $O(\log B)$ adaptivity gap. They also showed the problem to be at least as hard to approximate as the deadline orienteering problem, for which the best approximation ratio known is $O(\log n)$ [1].

A problem related to stochastic orienteering was considered by Guha and Munagala [12] in the context of the *multi-armed bandit* problem. As observed in [14], the approach in [12] yields an $O(1)$-approximation algorithm and adaptivity gap for the variant of stochastic orienteering with two *separate* budgets for travel and processing times. In contrast, our result shows that stochastic orienteering (with a single budget) has super-constant adaptivity gap.

1.3 Problem Definition

An instance of stochastic orienteering (StocOrient) consists of a metric space (V, d) with vertex-set $|V| = n$ and symmetric integer distances $d : V \times V \to \mathbb{Z}^+$ (satisfying the triangle inequality) that represent travel times. Each vertex $v \in V$ is associated with a stochastic job, with a deterministic reward $r_v \geq 0$ and a random processing time (also called size) $S_v \in \mathbb{Z}^+$ distributed according to a known probability distribution. The processing times are independent across vertices. We are also given a starting "root" vertex $\rho \in V$, and a budget $B \in \mathbb{Z}^+$ on the total time available. A solution (policy) must start from ρ, and visit a sequence of vertices (possibly adaptively). Each job is executed non-preemptively, and the solution knows the precise processing time only upon completion of the job. The objective is to maximize the expected reward from jobs completed before the horizon B; note that there is no reward for partially completing a job. The approximation ratio of an algorithm is the worst-case ratio of the expected rewards of an optimal policy to that of the algorithm's policy.

We assume that all times (travel and processing) are integer valued and lie in $\{0, 1, \cdots, B\}$. In the *correlated* stochastic orienteering problem (CorrOrient), the job sizes and rewards are both random, and correlated with each other. The distributions across different vertices are still independent. For each vertex $v \in V$, we use S_v and R_v to denote its random size and reward, respectively. We assume an explicit representation of the distribution of each job $v \in V$: for each $s \in \{0, 1, \cdots, B\}$, job v has size $S_v = s$ and reward $r_v(s)$ with probability $\Pr[S_v = s] = \pi_v(s)$. Note that the input size is nB.

An *adaptive policy* is a decision tree where each node is labeled by a job/vertex of V, with the outgoing arcs from a node labeled by u corresponding to the possible sizes in the support of S_u. A *non-adaptive policy* is simply given by a path P starting at ρ: we just traverse this path, processing the jobs that we encounter, until the total (random) size of the jobs plus the distance traveled reaches B. A randomized non-adaptive policy picks a path P at random from some distribution before it knows any of the size instantiations, and then follows this path as above. Note that in a non-adaptive policy, the order in which jobs are processed is independent of their processing time instantiations.

In our algorithm for CorrOrient, we use the *deadline orienteering* problem as a subroutine. The input to this problem is a metric (U, d) denoting travel times, a reward and deadline at each vertex, start (s) and end (t) vertices, and length bound D. The objective is to compute an s-t path of length at most D that maximizes reward from vertices visited before their deadlines. The best approximation ratio for this problem is $\alpha = \min\{O(\log n), O(\log B)\}$ [1,7].

1.4 Organization

The adaptivity gap lower bound appears in Section 2, where we prove Theorem 1. In Section 3, we consider the correlated stochastic orienteering problem and prove the upper bound on its adaptivity gap (Theorem 2). Finally, the improved quasi-polynomial time algorithm (Theorem 3) for correlated stochastic orienteering appears in Section 4

2 Lower Bound on the Adaptivity Gap

Here we describe our lower bound instance which shows that the adaptivity gap is $\Omega(\sqrt{\log\log B})$ even for an undirected line metric. The proof and the description of the instance is divided into two steps. First we describe an instance where the underlying graph is a directed complete binary tree, and prove the lower bound for it. The directedness ensures that all policies follow a path from the root to a leaf (possibly with some nodes skipped) without any backtracking. Next, we "embed" the directed tree into an undirected line metric, and show that the adaptivity gap stays the same up to a constant factor.

2.1 Directed Binary Tree

Let $L \geq 2$ be an integer and $p := 1/\sqrt{L}$. We define a complete binary tree $\mathcal{T}$ of height L with root ρ. All the edges are directed from the root towards the leaves. The *level* $\ell(v)$ of any node v is the number of nodes on the shortest path from v to any leaf. So all the leaves are at *level* one and the root ρ is at level L. We refer to the two children of each internal node as the left and right child, respectively. Each node v of the tree has a job with some deterministic reward r_v and a random size S_v. Each random variable S_v is Bernoulli, taking value zero with probability $1-p$ and some positive value s_v with the remaining probability p. The budget for the instance is $B = 2^{2^{L+1}}$.

To complete the description of the instance, we need to define the values of the rewards r_v, the job sizes S_v, and the distances $d(u, v)$ on edges $e = (u, v)$.

Defining rewards. For any node v, let $\tau(v)$ denote the number of right-branches taken on the path from the root to v. We define the reward of each node v to be $r_v := (1-p)^{\tau(v)}$.

Defining sizes. Let $e(x) := 2^x$ for any $x \in \mathbb{R}$. The size at the root, $s_\rho := e\left(2^L\right) = 2^{2^L}$. The rest of the sizes are defined recursively. For any non-root node v at level $\ell(v)$ with u denoting its parent, the size is:

$$s_v := \begin{cases} s_u \cdot e\left(2^{\ell(v)}\right) & \text{if } v \text{ is the right child of } u \\ s_u \cdot e\left(-2^{\ell(v)}\right) & \text{if } v \text{ is the left child of } u \end{cases}$$

In other words, for a node v at level ℓ, consider the path $P = (\rho = u_L, u_{L-1}, \ldots, u_{\ell+1}, u_\ell = v)$ from ρ to v. Let $k = \sum_{j=L}^{\ell}(-1)^{i(u_j)}2^j$ where

$i(u_j) = 1$ if u_j is the left child of its parent u_{j+1}, and 0 otherwise (we assume $i(\rho) = 0$). Then $s_v = e\,(k)$. See Figure 1 for an example.

Observe that for a node v, each node u in its left (resp. right) subtree has $s_u < s_v$ (resp. $s_u > s_v$).

It remains to define distances on the edges. This will be done in an indirect way, and it is instructive to first consider the adaptive policy that we will work with. In particular, the distances will be defined in such a way that the adaptive policy can always continue till it reaches a leaf node.

Adaptive policy $\mathcal{A}$. Consider the policy $\mathcal{A}$ that goes left at node u whenever it observes size zero at u, and goes right otherwise.

Clearly, the *residual budget* $b(v)$ at node v under $\mathcal{A}$ will satisfy the following: $b(\rho) = B = e\left(2^{L+1}\right) = 2^{2^{L+1}}$, and

$$b(v) := \begin{cases} b(u) - s_u - d(u, v) & \text{if } v \text{ is the right child of } u \\ b(u) - d(u, v) & \text{if } v \text{ is the left child of } u \end{cases}$$

Defining distances. We will define the distances so that the residual budgets $b(\cdot)$ under $\mathcal{A}$ satisfy the following: $b(\rho) = B$, and for any node v with parent u,

$$b(v) := \begin{cases} b(u) - s_u & \text{if } v \text{ is the right child of } u \\ s_u & \text{if } v \text{ is the left child of } u \end{cases}$$

In particular, this implies the following lengths on edges. For any node v with parent u,

$$d(u, v) := \begin{cases} 0 & \text{if } v \text{ is the right child of } u \\ b(u) - s_u = b(u) - b(v) & \text{if } v \text{ is the left child of } u \end{cases}$$

In Claim 3 below we will show that the distances are non-negative, and hence well-defined. Figure 1 gives a pictorial view of the instance.
Basic properties of the instance. Let $A_d(v)$ denote the distance traveled by the adaptive strategy A to reach v, and let $A_s(v)$ denote the total size instantiation before reaching v. By the definition of the budgets, and as A takes the right branch at u iff the size instantiation at u is positive, we have the following.

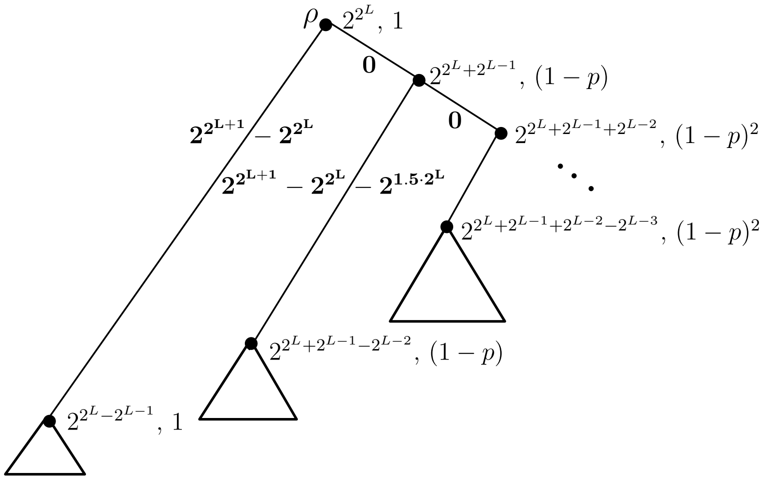
Claim 1 *For any node v, the budget $b(v)$ satisfies $b(v) = B - A_d(v) - A_s(v)$.*

Claim 2 *If a node w is a left child of its parent, then $b(w) = s_w \cdot e\left(2^{\ell(w)}\right)$.*

Proof Let u be the parent of w. By definition of sizes, $s_w = s_u \cdot e\left(-2^{\ell(w)}\right)$. As $b(w) = s_u$ by the definition of residual budgets, the claim follows. □

Claim 3 *For any node u, we have $3 \cdot s_u \leq b(u)$. This implies that all the residual budgets and distances are non-negative.*

Proof Let w denote the lowest level node on the path from ρ to u that is the left child of its parent (if u is the left child of its parent then $w = u$; if there is no such node, set $w = \rho$). Note that by Claim 2 and the definition of s_ρ and $b(\rho)$, in either case it holds that $b(w) = s_w \cdot e\left(2^{\ell(w)}\right)$.



The distances are shown (bold font) on the edges.

The sizes s_v and rewards r_v are shown (normal font) next to the nodes.

Fig. 1 The binary tree $\mathcal{T}$.

Let π denote the path from w to u (including w but not u; so $\pi = \emptyset$ if $w = u$). Since π contains only right-branches, $b(u) = b(w) - \sum_{y\in\pi} s_y$ and hence $b(u) \geq b(w) - 3\sum_{y\in\pi} s_y$. Thus to prove $3 \cdot s_u \leq b(u)$ it suffices to show $3(s_u + \sum_{y\in\pi} s_y) \leq b(w)$. For brevity, let $s := s_w$ and $\ell = \ell(w)$. Using the definition of sizes,

$$\begin{aligned} s_u + \sum_{y\in\pi} s_y &\leq \sum_{i=1}^{\ell} s \cdot e\left(2^{\ell-1} + 2^{\ell-2} + \cdots 2^i\right) \quad = \quad s \cdot \sum_{i=1}^{\ell} e\left(2^{\ell} - 2^i\right) \\ &= s \cdot \sum_{i=1}^{\ell} e\left(2^{\ell}\right) \cdot e\left(-2^i\right) \quad \leq \quad s \cdot e\left(2^{\ell}\right) \cdot \sum_{i\geq 1} 4^{-i} \\ &\leq \frac{1}{3} \cdot s \cdot e\left(2^{\ell}\right) \quad = \quad \frac{b(w)}{3}, \end{aligned}$$

as desired. Here the right hand side of the first inequality is simply the total size of nodes in the w to leaf path using all right branches. The inequality in the second line follows as $e\left(-2^i\right) = 2^{-2^i} \leq 2^{-2i} = 4^{-i}$ for all $i \geq 1$.

Thus we always have $3 \cdot s_u \leq b(u)$.

As $b(v) = b(u) - s_u$ if v is the right child of u, or $b(v) = s_u$ otherwise, this implies that all the residual-budgets are non-negative.

Similarly, as $d(u, v)$ is either 0 or $b(u) - s_u$ (and hence at least $\frac{2}{3}b(u)$), this implies that all edge lengths are non-negative. $\square$

This claim shows that the above instance is well defined, and that $\mathcal{A}$ is a feasible adaptive policy that always continues for L steps until it reaches a leaf. Next, we show that $\mathcal{A}$ obtains large expected reward.

Lemma 1 *The expected reward of policy $\mathcal{A}$ is $\Omega(L)$.*

Proof Notice that $\mathcal{A}$ accrues reward as follows: it keeps getting reward 1 (and going left) until the first positive size instantiation, then it goes right for a single step and keeps going left and getting reward $(1-p)$ till the next positive size instantiation and so on. This continues for a total of L steps. In particular, at any time t it collects reward $(1-p)^i$, if exactly i nodes have positive sizes among the t nodes seen.

Let X_i denote the Bernoulli random variable that is 1 if the i^{th} node in $\mathcal{A}$ has a positive size instantiation, and 0 otherwise. So $E[X_i] = p$, and $E[X_1 + \ldots + X_L] = Lp = \sqrt{L}$. By Markov's inequality, the probability that more than $2\sqrt{L}$ nodes in $\mathcal{A}$ have positive sizes is at most half. Hence, with probability at least $\frac{1}{2}$ the reward collected in the last node of $\mathcal{A}$ is at least $(1-p)^{2\sqrt{L}}$. That is, the total expected reward of $\mathcal{A}$ is at least $\frac{1}{2} \cdot L \cdot (1-p)^{2\sqrt{L}} \approx L/2 \cdot e^{-2} = \Omega(L)$. □

Remarks on the lower bound instance: For an adaptivity gap on directed metrics, it suffices to consider instances where the metric is a tree representing the optimal adaptive policy. The sizes and distances also have to be coupled based on those observed in the adaptive optimum. Moreover, in any lower bound instance, most vertices must be located at a distance almost B from the root: otherwise one can obtain an $O(1)$-approximate non-adaptive policy using the algorithm for the two budget version of stochastic orienteering. The main potential advantage that a non-adaptive policy has over the adaptive one is that it may skip some nodes: this captures the metric property of distances. (As an aside, if non-adaptive policies are not allowed to skip nodes then it is straightforward to construct large adaptivity gaps.) So any lower bound construction must limit the budget gain accrued by skipping nodes. For our construction, we will show that the total sizes seen in the adaptive policy before any node v is at most the size at v itself (Lemma 2). Finally, our analysis of the above instance is tight, and improving on this lower bound would require a new construction and additional ideas.

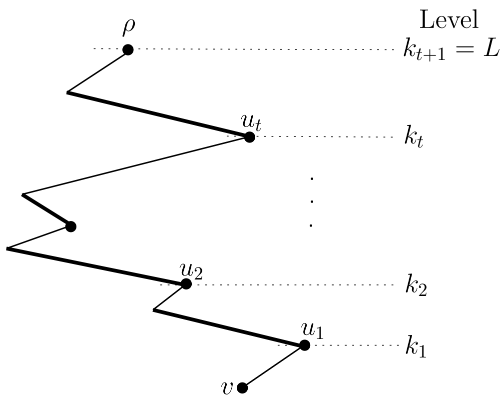
2.2 Bounding Directed Non-adaptive Policies

We will first show that any non-adaptive policy $\mathcal{N}$ that is constrained to visit vertices according to the partial order given by the tree $\mathcal{T}$ gets reward at most $O(\sqrt{L})$. Notice that these correspond precisely to non-adaptive policies on the directed tree $\mathcal{T}$.

The key property we need from the size construction is the following.

Lemma 2 *For any node v, the total size instantiation observed under the adaptive policy $\mathcal{A}$ before v is strictly less than s_v.*

Proof Consider the path π from the root to v, and let $k_1 < k_2 < \cdots < k_t$ denote the levels at which π "turns left". That is, for each i, the node u_i at level k_i in path π satisfies (a) u_i is the right child of its parent, and (b) π contains the left child of u_i if it goes below level k_i. (If v is the right child of its parent then $u_1 = v$ and $k_1 = \ell(v)$.) Let s_i denote the size of u_i, the level k_i node in π. Also, set $k_{t+1} = L$ corresponding to the root. Below we use $[t] := \{1, 2, \cdots, t\}$. See Figure 2 for an illustration.



Path π from ρ to v, with "left turning" nodes $u_1, \cdots, u_t$.

The thick lines denote right-branches (where positive sizes are seen).

Fig. 2 The path π in proof of Lemma 2.

We first bound the size instantiation between levels k_{i+1} and k_i in terms of s_i. Observe that a positive size instantiation is seen in $\mathcal{A}$ only along right branches. So for any $i \in [t]$, the total size instantiation seen in π between levels k_i and k_{i+1} is at most:

$$s_i \cdot \left[e\left(-2^{k_i}\right) + e\left(-2^{k_i} - 2^{k_i+1}\right) + e\left(-2^{k_i} - 2^{k_i+1} - 2^{k_i+2}\right) \cdots \right] \\ \le s_i \cdot e\left(-2^{k_i}\right) \cdot (1 + 1/2 + 1/4 + \cdots) \quad \le \quad 2\, s_i \cdot e\left(-2^{k_i}\right) \tag{1}$$

Now, note that for any $i \in [t]$, the sizes s_{i+1} and s_i are related as follows:

$$s_{i+1} \le s_i \cdot e\left(-2^{k_i} + 2^{k_i+1} + 2^{k_i+2} + \cdots + 2^{k_{i+1}-1}\right) \\ = s_i \cdot e\left(-2^{k_i} - 2^{k_i+1} + 2^{k_{i+1}}\right) \quad \le \quad \frac{s_i}{4} \cdot e\left(-2^{k_i} + 2^{k_{i+1}}\right) \tag{2}$$

The first inequality uses the fact that the path from u_{i+1} to u_i is a sequence of (at least one) left-branches followed by a sequence of (at least one) right-branches. As the size decreases along left-branches and increases along right branches, it follows that conditional on the values of k_i and k_{i+1}, the ratio s_{i+1}/s_i is maximized for the path with a sequence of left branches followed by a single right branch (at level k_i). Using (2), we obtain inductively that:

$$s_{i+1} \cdot e\left(-2^{k_{i+1}}\right) \quad \le \quad \frac{1}{4} \cdot s_i \cdot e\left(-2^{k_i}\right) \quad \le \quad \frac{1}{4^i} \cdot s_1 \cdot e\left(-2^{k_1}\right), \qquad \forall i \in [t]. \tag{3}$$

Using (1) and (3), the total size instantiation seen in π (this does not include the size at v) is at most:

$$\sum_{i=1}^{t} 2\, s_i \cdot e\left(-2^{k_i}\right) \quad \leq \quad 2 \sum_{i=1}^{t} \frac{1}{4^{i-1}} \cdot s_1 \cdot e\left(-2^{k_1}\right) \quad < \quad 4\, s_1 \cdot e\left(-2^{k_1}\right). \quad (4)$$

Finally, observe that the size at the level k_1 node is

$$s_1 \;\leq\; s_v \cdot e\left(2^{k_1-1} + 2^{k_1-2} + \cdots + 2^1\right) \;=\; s_v \cdot e\left(2^{k_1} - 2\right),$$

since k_1 is the lowest level at which π turns left (i.e. π keeps going left below level k_1 until v). Together with (4), it follows that the total size instantiation seen before v is strictly less than

$$4\, s_1 \cdot e\left(-2^{k_1}\right) \leq 4\, e\left(-2^{k_1}\right) \cdot s_v \cdot e\left(2^{k_1} - 2\right) = 4\, e\,(-2)\; s_v = s_v.$$

This completes the proof of Lemma 2. □

We now show that any non-adaptive policy on the directed tree $\mathcal{T}$ achieves reward $O(\sqrt{L})$. Note that any such solution $\mathcal{N}$ is just a root-leaf path in $\mathcal{T}$ that skips some subset of vertices. A node v in $\mathcal{N}$ is an *L-branching* node if the path $\mathcal{N}$ goes left after v. *R-branching* nodes are defined similarly.

Claim 4 *The total reward from R-branching nodes is at most* $\sqrt{L}$.

Proof As rewards decrease by a $(1-p)$ factor after each right branch, the total reward of such nodes is at most $\sum_{i=0}^{L}(1-p)^i \leq \frac{1}{p} = \sqrt{L}$. □

Claim 5 $\mathcal{N}$ *can not get any reward after two L-branching nodes instantiate to positive sizes.*

Proof For any node v in tree $\mathcal{T}$, let $A_d(v)$ (resp. $A_s(v)$) denote the distance traveled (resp. size instantiated) in the adaptive policy $\mathcal{A}$ until v; here $A_s(v)$ does not include the size of v. Observe that Lemma 2 implies that $A_s(v) < s_v$ for all nodes v.

In the non-adaptive solution $\mathcal{N}$, let u and v be any two L-branching nodes that instantiate to positive sizes s_u and s_v; say u appears before v. Under this outcome, we will show that $\mathcal{N}$ exhausts its budget after v. Note that the distance traveled to node v in $\mathcal{N}$ is exactly $A_d(v)$, the same as that under $\mathcal{A}$. So the total distance plus size instantiated in $\mathcal{N}$ is at least $A_d(v) + s_v + s_u$, which (as we show next) is more than the budget B.

By Claim 1, $b(v) = B - A_d(v) - A_s(v)$. Moreover, the residual budget $b(u')$ at the left child u' of u equals s_u. Since the residual budgets are non-increasing down the tree $\mathcal{T}$, we have $B - A_d(v) - A_s(v) = b(v) \leq b(u') = s_u$, i.e. $A_d(v) \geq B - A_s(v) - s_u$. Hence, the total distance plus size in $\mathcal{N}$ is at least

$$A_d(v) + s_v + s_u \quad \geq \quad B - A_s(v) + s_v \quad > \quad B,$$

where the last inequality follows from Lemma 2. So $\mathcal{N}$ can not obtain reward from any node after v. □

Combining the above two claims, we obtain:

Claim 6 *The reward of any directed non-adaptive policy is at most* $3\sqrt{L}$*.*

Proof Using Claim 5, the expected reward from L-branching nodes is at most the expected number of L-branching nodes until two positive sizes instantiate, i.e. at most $\frac{2}{p} = 2\sqrt{L}$. Claim 4 implies that the expected reward from R-branching nodes is at most $\sqrt{L}$. Adding the two, we obtain the claim. □

This proves an $\Omega(\sqrt{\log\log B})$ adaptivity gap for stochastic orienteering on directed metrics. We remark that the $O(\log\log B)$ upper bound in [14] also holds for directed metrics.

2.3 Adaptivity Gap on Line Metric

We now show that the previous directed tree instance can be embedded into a line metric such that the adaptivity gap does not change much. This gives an $\Omega\left(\sqrt{\log\log B}\right)$ adaptivity gap for stochastic orienteering even on line metrics.

The line metric $\mathcal{L}$ is defined as follows. Each node v of the tree $\mathcal{T}$ is mapped (on the real line) to the coordinate $d(\rho, v)$ which is the distance in $\mathcal{T}$ from the root ρ to v. Since all distances in our construction are integers, each node lies at a non-negative integer coordinate. Note that multiple nodes may be at the same coordinate (for example, as all right-edges in $\mathcal{T}$ have zero length). Below, $d(\cdot)$ will denote distances in the tree metric $\mathcal{T}$, and $d_{\mathcal{L}}(\cdot)$ denotes distances in the line metric $\mathcal{L}$.

Note that $d_{\mathcal{L}}(\rho, v) = d(\rho, v)$ for all nodes $v \in \mathcal{T}$. Moreover, the distance $d_{\mathcal{L}}(u, v)$ between two nodes u and v in the line metric is $|d(\rho, v) - d(\rho, u)|$, which is at most the distance $d_{\mathcal{T}}(u, v)$ in the tree metric. Thus the adaptive policy $\mathcal{A}$ for the tree $\mathcal{T}$ is also valid for the line, which (by Lemma 1) has expected reward $\Omega(L)$. However, the distances $d_{\mathcal{L}}(u, v)$ on the line could be arbitrarily smaller than $d(u, v)$, and thus the key issue is to show that non-adaptive policies cannot do much better. To this end, we begin by observing some more properties of the distances $d(\rho, v)$ and the embedding on the line.

Lemma 3 *For any internal node* $u \in \mathcal{T}$*, let* L_u *(resp.* R_u*) denote the subtree rooted at the left (resp. right) child of* u*. Then, for any node* $v \in L_u$*,* $d(\rho, v) > B - 2s_u$ *and for any node* $v \in R_u$*,* $d(\rho, v) \le B - 4s_u$*.*

Proof For any node v, recall that its residual budget $b(v) = B - d(\rho, v) - A_s(v)$, where $A_s(v)$ is the total size instantiated in the adaptive policy $\mathcal{A}$ before node v. Suppose $v \in L_u$, and let u' be the left child of u. Then

$$d(\rho, v) \ge d(\rho, u') = B - b(u') - A_s(u') = B - s_u - A_s(u') = B - s_u - A_s(u) > B - 2s_u,$$

where we use that $b(u') = s_u$, $A_s(u') = A_s(u)$ and the last inequality follows from Lemma 2.

Now consider $v \in R_u$. We have $A_s(v) \geq s_u$, as v lies in the right subtree under u and so u must have instantiated to a positive size before reaching v. By Claim 3, $b(v) \geq 3s_v$ which is at least $3s_u$ since $s_v > s_u$ for each $v \in R_u$. Thus $d(\rho, v) = B - b(v) - A_s(v) \leq B - 4s_u$. □

This implies the following useful fact.

Corollary 1 *In the line embedding, for any node $u \in \mathcal{T}$, all nodes in the left-subtree L_u appear after all nodes in the right-subtree R_u.*

We will now show that any non-adaptive policy has reward at most $O(\sqrt{L})$. This requires more work than in the tree metric case, but the high level idea is quite similar: we restrict how any non-adaptive policy can look like by using the properties of distances, and show that such policies cannot obtain too much profit. Observe that a non-adaptive policy $\mathcal{N}''$ is just a walk on $\mathcal{L}$, originating from ρ and visiting a subset of vertices.

Lemma 4 *Any non-adaptive policy on $\mathcal{L}$ must visit vertices ordered by non-decreasing distance from ρ.*

Proof We will show that if vertex v is visited before w and $d(\rho, v) > d(\rho, w)$ then the walk to w has length more than B; this would prove the lemma.

Let u denote the least common ancestor of v and w. There are two cases depending on whether $u = w$ or $u \notin \{v, w\}$; note that the ancestor u cannot be v as $d(\rho, v) > d(\rho, w)$.

If $u \notin \{v, w\}$, since $d(\rho, v) > d(\rho, w)$, it must be that $v \in L_u$ and $w \in R_u$ by Corollary 1. Moreover, the total distance traveled by the path is at least

$$\begin{aligned} d_{\mathcal{L}}(\rho, v) + d_{\mathcal{L}}(v, w) \geq d(\rho, v) + d(\rho, v) - d(\rho, w) &= 2d(\rho, v) - d(\rho, w) \\ > 2(B - 2s_u) - (B - 4s_u) &= B, \end{aligned}$$

where the second inequality is by Lemma 3.

If $u = w$, since $d(\rho, v) > d(\rho, w)$, there must be at least one left edge $e = (x, y)$ on the path from w to v in the tree (as the length of the right edges is 0). Then, the distance traveled by the path is at least $d_{\mathcal{L}}(\rho, v) + d_{\mathcal{L}}(v, w) \geq d(\rho, y) + d(x, y) = d(\rho, x) + 2d(x, y)$. As $d(\rho, x) = B - b(x) - A_s(x) > B - b(x) - s_x$ by Lemma 2, and as $d(x, y) = b(x) - s_x$ (by definition of distances on left edges), we have

$$d(\rho, x) + 2d(x, y) \geq (B - b(x) - s_x) + 2(b(x) - s_x) = B + b(x) - 3s_x > B$$

where the last inequality follows from Claim 3. □

By Lemma 4, an optimal non-adaptive policy $\mathcal{N}''$ visits vertices in non-decreasing coordinate order. For vertices at the same coordinate, we show next that the optimal non-adaptive policy visits such nodes in decreasing order of their level in $\mathcal{T}$.

Claim 7 *If $\mathcal{N}''$ visits two vertices $\{u, v\}$ consecutively that have the same coordinate in $\mathcal{L}$ and have levels $\ell(u) > \ell(v)$, then u must be visited before v.*

Proof Since u and v have the same coordinate in $\mathcal{L}$, by Lemma 3 it must be that one is an ancestor of the other, and the $u-v$ path in $\mathcal{T}$ consists only of right-edges. Since $\ell(u) > \ell(v)$, node u is an ancestor of v in $\mathcal{T}$. Suppose that $\mathcal{N}''$ chooses to visit v before u. We will show that the alternative solution $\overline{\mathcal{N}}$ that visits u before v has at least as much expected reward as $\mathcal{N}''$. This is intuitively clear since u stochastically dominates v in our setting: the probabilities are identical, size of u is less than v, and reward of u is more than v. The formal proof also requires independence of u and v, and is by a case analysis.

Let us *condition* on all instantiations other than u and v: we will show that $\overline{\mathcal{N}}$ has larger conditional expected reward than $\mathcal{N}''$. This would also show that the (unconditional) expected reward of $\overline{\mathcal{N}}$ is more than $\mathcal{N}''$. Let X denote the total distance plus size in $\mathcal{N}''$ (resp. $\overline{\mathcal{N}}$) when it reaches v (resp. u). Irrespective of the outcomes at u and v, the residual budgets in $\mathcal{N}''$ and $\overline{\mathcal{N}}$ before/after visiting $\{u, v\}$ will be identical. So the only difference in (conditional) expected reward is at u and v. The following table lists the different possibilities for rewards from u and v, as X varies (recall that B is the budget).

Cases	Reward ($\mathcal{N}''$)	Reward ($\overline{\mathcal{N}}$)
$X + s_u + s_v \le B$	$r_u + r_v$	$r_u + r_v$
$X + s_v \le B < X + s_u + s_v$	$(1-p^2)r_u + r_v$	$r_u + (1-p^2)r_v$
$X + s_u \le B < X + s_v$	$(1-p)(r_u + r_v)$	$r_u + (1-p)r_v$
$X \le B < X + s_u$	$(1-p)^2 r_u + (1-p)r_v$	$(1-p)r_u + (1-p)^2 r_v$
$B < X$	0	0

In each case, $\overline{\mathcal{N}}$ gets at least as much reward as $\mathcal{N}''$ since $r_u > r_v$. This completes the proof. □

For any node v in $\mathcal{N}''$, let E_v denote the set of nodes u satisfying (i) u appears before v in $\mathcal{N}''$, and (ii) u is *not* an ancestor of v in tree $\mathcal{T}$. We refer to E_v as the "blocking set" for node v.

Claim 8 *For any $v \in \mathcal{N}''$ and $u \in E_v$, we must have $u \in$ right-subtree(a) and $v \in$ left-subtree(a) at the lowest common ancestor a of v and u. Moreover $\mathcal{N}''$ can not get reward from v if any vertex in its blocking set E_v instantiates to a positive size.*

Proof Observe that u and v are incomparable in $\mathcal{T}$ because:

- u is not an ancestor of v by definition of E_v.
- u is not a descendant of v. Suppose (for a contradiction) that u is a descendant of v. Note that u and v are not co-located in $\mathcal{L}$: if it were, then by Claim 7 and the fact that $\mathcal{N}''$ visits u before v, u must be an ancestor of v, which contradicts the definition of E_v. So the only remaining case is that u is located further from ρ than v: but this contradicts Lemma 4 as $\mathcal{N}''$ visits u before v.

So the lowest common ancestor a of v and u is distinct from both v, u. Since $d(\rho, u) \leq d(\rho, v)$, we must have $u \in R_a$ and $v \in L_a$. This proves the first part of the claim.

Since $u \in R_a$, its size $s_u \geq s_a \cdot e\left(2^{\ell(a)-1} - 2^{\ell(a)-2} - \cdots - 2^1\right) = 4 \cdot s_a$. As $v \in L_a$, by Lemma 3 we have $d(\rho, v) > B - 2 \cdot s_a$ and hence if u has non-zero size, the total distance plus size until v is more than $4s_a + B - 2s_a > B$, i.e. $\mathcal{N}''$ can not get reward from v. □

The next key claim shows that the sets E_v are increasing along $\mathcal{N}''$.

Claim 9 *If node v appears before w in $\mathcal{N}''$ then $E_v \subseteq E_w$.*

Proof Consider nodes v and w as in the claim, and suppose (for contradiction) that there is some $u \in E_v \setminus E_w$. Since $u \in E_v$, by Claim 8, $u \in$ right-subtree(a) and $v \in$ left-subtree(a), where a is the lowest common ancestor of u and v. Clearly u appears before w in $\mathcal{N}''$ (u is before v which is before w). And since $u \notin E_w$, u must be an ancestor of w. Hence w is also in the right-subtree(a), and $d(\rho, w) < d(\rho, v)$; recall that $v \in$ left-subtree(a). This contradicts with Lemma 4 since v is visited before w. Thus $E_v \subseteq E_w$. □

Based on Claim 9, the blocking sets in $\mathcal{N}''$ form an increasing sequence. So we can partition $\mathcal{N}''$ into contiguous segments $\{N_i\}_{i=1}^k$ with u_i (resp. v_i) denoting the first (resp. last) vertex of N_i, so that the following hold for each $i \in [k] := \{1, 2, \cdots, k\}$.

- The first vertex u_i of N_i has $|E_{u_i}| \geq (i-1)\sqrt{L}$, and
- the increase in the blocking set $|E_{v_i} \setminus E_{u_i}| = |E_{v_i}| - |E_{u_i}| < \sqrt{L}$.

Defining directed non-adaptive policies from $\mathcal{N}''$. For each $i \in [k]$ consider the non-adaptive policy $\mathcal{N}_i$ that traverses segment N_i and visits only vertices in $N_i \setminus E_{v_i}$; note that $N_i \setminus E_{v_i} = N_i \setminus (E_{v_i} \setminus E_{u_i})$ since $E_{u_i} \cap N_i = \emptyset$. Notice that the blocking set is always empty in $\mathcal{N}_i$: this means that nodes in $\mathcal{N}_i$ are visited in the order of some root-leaf path in tree $\mathcal{T}$, i.e. $\mathcal{N}_i$ is a directed non-adaptive policy (as considered in Section 2.2). So by Claim 6, the expected reward in each $\mathcal{N}_i$ is at most $3\sqrt{L}$. That is,

$$\max_{i=1}^{k} \mathbb{E}\left[\text{reward } \mathcal{N}_i\right] \quad \leq \quad 3\sqrt{L} \tag{5}$$

Now we upper bound the reward in the original non-adaptive policy $\mathcal{N}''$.

$$\begin{aligned}
\mathbb{E}\left[\text{reward } \mathcal{N}''\right] &= \sum_{i=1}^{k} \Pr\left[\mathcal{N}'' \text{ reaches } u_i\right] \cdot \mathbb{E}\left[\text{reward in } N_i \,|\, \mathcal{N}'' \text{ reaches } u_i\right] \\
&\leq \sum_{i=1}^{k} e^{-i+1} \cdot \mathbb{E}\left[\text{reward in } N_i \,|\, \mathcal{N}'' \text{ reaches } u_i\right] && (6) \\
&\leq \sum_{i=1}^{k} e^{-i+1} \cdot \left(\mathbb{E}\left[\text{reward } \mathcal{N}_i\right] + |E_{v_i} \setminus E_{u_i}|\right) && (7) \\
&\leq \sum_{i=1}^{k} e^{-i+1} \cdot 4\sqrt{L} \quad \leq \quad \frac{4e}{e-1}\sqrt{L}. && (8)
\end{aligned}$$

Inequality (6) uses $|E_{u_i}| \geq (i-1)\sqrt{L}$ and the second part of Claim 8 which implies $\Pr[\mathcal{N}'' \text{ reaches } u_i] \leq (1-p)^{|E_{u_i}|} \leq e^{-i+1}$. Inequality (7) uses the definition of $\mathcal{N}_i$ which is obtained by skipping nodes $E_{v_i} \setminus E_{u_i}$ in N_i, and also the independence of the sizes (which allows us to drop the conditioning). Finally, (8) uses the property $|E_{v_i} \setminus E_{u_i}| < \sqrt{L}$ and (5). This completes the proof of Theorem 1.

3 Adaptivity Gap Upper Bound for Correlated Orienteering

Here we show that the adaptivity gap of stochastic orienteering is $O(\log \log B)$ even in the setting of *correlated* sizes and rewards (Theorem 2). This is an extension of the result proved in [14] for the uncorrelated case.

The correlated stochastic orienteering problem again consists of a budget B and metric (V, d) with each vertex $v \in V$ denoting a random job. Each vertex has a joint distribution over its reward and processing time (size); so the reward and size at any vertex are correlated. These distributions are still independent across vertices. The basic stochastic orienteering problem is the special case when vertex rewards are deterministic.

Notation. Recall that for each vertex $v \in V$, S_v and R_v denote its (random) size and reward. As before, all sizes are integers in the range $[B] := \{0, 1, \cdots, B\}$. We assume an explicit representation of each job's distribution: the job at vertex $v \in V$ has size $s \in [B]$ and reward $r_v(s)$ with probability $\Pr[S_v = s] = \pi_v(s)$.

We represent the optimal adaptive policy naturally as a decision tree $\mathcal{T}$. Nodes in $\mathcal{T}$ are labeled by vertices in V and branches correspond to size (and reward) instantiations. Note that the same vertex of V may appear at multiple nodes of $\mathcal{T}$ (note the distinction between nodes and vertices). However, any root-leaf path in $\mathcal{T}$ contains each vertex at most once. For nodes u, u', we use $u \prec u'$ to denote u being an ancestor of u' in $\mathcal{T}$, where $u \neq u'$. For any node $u \in \mathcal{T}$, let d_u (resp. i_u) denote the total distance traveled (resp. size observed) in $\mathcal{T}$ before u. For node $u \in \mathcal{T}$, we overload notation and use S_u, R_u etc to denote the respective term for the vertex labeling u.

Note that at any node $u \in \mathcal{T}$, only size instantiations $S_u \leq B - d_u - i_u$ contribute reward (any larger size violates the budget). Define the expected reward at node u as $\overline{r}_u := \sum_{s=0}^{B-d_u-i_u} \Pr[S_u = s] \cdot r_u(s)$. Observe that the optimal adaptive reward is $\mathsf{Opt} = \sum_{u \in \mathcal{T}} \Pr[\mathcal{T} \text{ reaches } u] \cdot \overline{r}_u$.

For each $j = 0, 1, \cdots, \lceil \log_2 B \rceil$ and node $u \in \mathcal{T}$, define $X_u^j := \min\{S_u, 2^j\}$. Let $K := \Theta(\log \log B)$ be a parameter that will be fixed later.

We are now ready to prove Theorem 2. It is along similar lines as the proof for uncorrelated stochastic orienteering in [14], and also makes use of the following concentration inequality.

Theorem 4 (Theorem 1 in [17]) *Let $I_1, I_2, \ldots$ be a sequence of possibly dependent random variables; for each $k \geq 1$ variable I_k depends only on $I_{k-1}, \ldots, I_1$. Consider also a sequence of random functionals $\xi_k(I_1, \ldots, I_k)$*

that lie in $[0,1]$. *Let* $\mathbb{E}_{I_k}[\xi_k(I_1,\ldots,I_k)]$ *denote the expectation of* ξ_k *with respect to* I_k*, conditional on* $I_1,\ldots,I_{k-1}$. *Furthermore, let* τ *denote any stopping time. Then,*

$$\Pr\left[\sum_{k=1}^{\tau}\mathbb{E}_{I_k}[\xi_k(I_1,\ldots,I_k)] \;\geq\; \frac{e}{e-1}\cdot\left(\sum_{k=1}^{\tau}\xi_k+\delta\right)\right] \;\leq\; \exp(-\delta), \quad \forall\delta\geq 0.$$

This result is used in proving the following important property.

Lemma 5 *Assume* $K \geq 12$*, and fix any* $j \in \{0,1,\cdots,\lceil\log B\rceil\}$. *Then, the probability of reaching a node* $u \in \mathcal{T}$ *with (a)* $\sum_{v\preceq u} X_v^j \leq 2\cdot 2^j$*, and (b)* $\sum_{v\preceq u}\mathbb{E}_v\left[X_v^j\right] > K\cdot 2^j$*, is at most* $e^{-K/3}$.

Proof For each $k = 1,2,\cdots$, set I_k to be the k^{th} node seen in $\mathcal{T}$, and

$$\xi_k(I_1,\ldots,I_k) \quad := \quad \frac{X_{I_k}^j}{2^j} \quad = \quad \min\left\{\frac{S_{I_k}}{2^j},1\right\}.$$

Observe that this sequence satisfies the condition in Theorem 4, since the identity of the k^{th} node in $\mathcal{T}$ depends only on the outcomes of the previous $k-1$ nodes. Moreover, each $\xi_k(\cdot)$ has a value in the range $[0,1]$. Note that the conditional expectation $\mathbb{E}_{I_k}[\xi_k(I_1,\ldots,I_k)] = \mathbb{E}_{I_k}\left[X_{I_k}^j\right]/2^j$. Define stopping time τ to be the first node $I_k = u$ (if one exists) at which the following two conditions hold:

$$\sum_{h=1}^{k}\xi_h(I_1,\ldots,I_h) \leq 2, \qquad \text{i.e. } \sum_{v\preceq u} X_v^j \leq 2\cdot 2^j; \qquad \text{and}$$

$$\sum_{h=1}^{k}\mathbb{E}_{I_h}\left[\xi_h(I_1,\ldots,I_h)\right] > K, \qquad \text{i.e. } \sum_{v\preceq u}\mathbb{E}_v\left[X_v^j\right] > K\cdot 2^j.$$

If there is no such node, then τ stops when $\mathcal{T}$ ends (i.e. after a leaf-node of $\mathcal{T}$). Clearly, if τ stops before $\mathcal{T}$ ends, then

$$\sum_{h=1}^{\tau}\mathbb{E}_{I_h}[\xi_k(I_1,\ldots,I_h)] \;>\; K \;>\; \frac{e}{e-1}\cdot\left(\sum_{h=1}^{\tau}\xi_h(I_1,\ldots,I_h)+\frac{K}{2}-2\right)$$

Now, setting $\delta = \frac{K}{2}-2$ in Theorem 4, the probability that τ stops before $\mathcal{T}$ ends (i.e. we reach a node u satisfying the two conditions stated in the lemma) is at most $e^{-K/2+2} \leq e^{-K/3}$ using $K \geq 12$. □

Lemma 6 *Assume* $K \geq 3\cdot\log(6\log B)+12$. *There is some node* $s \in \mathcal{T}$ *such that the path* σ *from the root to* s *satisfies:*

– *Total reward:* $\sum_{v\in\sigma}\overline{r}_v = \sum_{v\preceq s}\overline{r}_v \geq \mathsf{Opt}/2$.

– *Prefix size: For each $v \in \sigma$ and j, one of the following conditions hold:*

$$\sum_{w \preceq v} X_w^j > 2 \cdot 2^j \quad \text{or} \quad \sum_{w \preceq v} \mathbb{E}_w \left[X_w^j\right] \leq K \cdot 2^j.$$

Proof For each $j = 0, 1, \cdots, \lceil \log_2 B \rceil$, define *band j "star nodes"* to be those $u \in \mathcal{T}$ that satisfy the two conditions in Lemma 5. Using Lemma 5 and a union bound over $1 + \lceil \log_2 B \rceil$ values of j, the probability of reaching any star node is at most $\frac{3 \log_2 B}{e^{K/3}} \leq \frac{1}{2}$ since $K \geq 3 \cdot \log(6 \log B)$.

Observe that for any node $u \in \mathcal{T}$, the conditional expected reward from the subtree of $\mathcal{T}$ under u is at most Opt: otherwise, the alternate policy that visits u directly from the root and follows the subtree below u would be a feasible policy of value more than Opt, contradicting the optimality of $\mathcal{T}$.

Consider tree $\mathcal{T}$ truncated just before all the star nodes. By the above two properties, the expected reward lost is at most $\mathsf{Opt} \cdot \Pr[\text{reach a star node}] \leq \mathsf{Opt}/2$. So the remaining reward is at least $\mathsf{Opt}/2$. By averaging, there is some leaf node s in the truncated $\mathcal{T}$, at which $\sum_{v \preceq s} \overline{r}_v \geq \mathsf{Opt}/2$; this proves the "total reward" property. Let σ denote the path from root to s in $\mathcal{T}$. Since σ does not contain a star node (of any band), every $v \in \sigma$ violates one of the conditions in Lemma 5 for each value of j. This proves the "prefix size" property. □

The non-adaptive policy. We now implement the path σ from Lemma 6 as a non-adaptive solution $\mathcal{N}$. The policy $\mathcal{N}$ simply involves visiting the vertices on σ (in that order) and attempting each job independently with probability $\frac{1}{4K}$. By an analysis as in [14] it can be shown (using Markov's inequality and independence across vertices) that $\mathcal{N}$ has expected reward at least $\frac{1}{6K} \sum_{v \in \sigma} \overline{r}_v \geq \frac{\mathsf{Opt}}{12K}$, which proves Theorem 2. We omit the formal proof here since a similar argument also appears in the analysis of our algorithm for correlated orienteering (after Claim 13). The interested reader is referred to [3].

3.1 Additional Structure of the Non-Adaptive Policy

Here we show the existence of a non-adaptive policy with some additional structure, by extending Lemma 6. This will be useful in the approximation algorithm given in the next section. First, some notation.

Definition 1 (Capped sizes and rewards) For any vertex v and integer $j \geq 0$, let $\mu_v^j := \mathbb{E}[X_v^j] = \mathbb{E}\left[\min\{S_v, 2^j\}\right]$ be the mean size of v capped at 2^j. For any vertex v and integer $d \geq 0$, let $\eta_v(d) := \sum_{s=0}^{d} \pi_v(s) \cdot r_v(s)$ be the expected reward from v under size instantiation at most d.

Recall the path σ from Lemma 6. Note that the expected reward of any $v \in \sigma$ in $\mathcal{T}$ is $\overline{r}_v = \eta_v(B - d_v - i_v)$.

Also, capped sizes satisfy the following property which will play a crucial role in our algorithm:

$$\frac{\mu_v^{j+1}}{2^{j+1}} \quad \leq \quad \frac{\mu_v^j}{2^j}, \qquad \forall v \in V \text{ and } j \geq 0. \tag{9}$$

This inequality follows from the fact that $\mathbb{E}\left[\min\{S_v, 2^{j+1}\}\right] \leq 2 \cdot \mathbb{E}\left[\min\{S_v, 2^j\}\right]$.

We now define some special nodes on the path σ. Let $L := \lceil \log_2 B \rceil$, and $[L] := \{0, 1, \cdots, L\}$. For each $j \in [L]$:

$$\text{set } \textit{portal } v_j \text{ to be the first vertex } u \in \sigma \text{ with } i_u \geq 2^{j+1} - 1. \tag{10}$$

Recall that σ starts at the root ρ; for notational convenience set $v_{-1} = \rho$. Clearly, $v_0 \preceq v_1 \preceq \cdots \preceq v_L$. For any $j \in [L]$, define *segment* O_j to consist of the vertices $v_{j-1} \preceq u \prec v_j$ in path σ. The following lemma shows that we can round size instantiations in σ to powers of two, and still retain the two properties in Lemma 6.

Lemma 7 *Given any instance of correlated stochastic orienteering, there exist "portal vertices" $\{v_j\}_{j=0}^L$ and path σ originating from ρ and visiting the portals in that order, such that:*

- *Reward: $\sum_{j \in [L]} \sum_{u \in O_j} \eta_u(B - d_u - 2^j + 1) \geq \mathsf{Opt}/2$. Here, for each $j \in [L]$, O_j consists of the vertices in σ between v_{j-1} and v_j. For any $u \in \sigma$, d_u is the distance from ρ to vertex u along σ.*
- *Prefix mean size: $\sum_{\ell=0}^{j} \sum_{u \in O_\ell} \mu_u^j \leq (K+1) \cdot 2^j$, for all $j \in [L]$. Here $K = \Theta(\log \log B)$.*

Proof The path σ is from Lemma 6, and portals $\{v_j\}_{j=0}^L$ are as defined in (10). For the first property, consider any segment O_j and vertex $u \in O_j$. By the definition of portals, we have $i_u \geq i_{v_{j-1}} \geq 2^j - 1$; so $\overline{r}_u = \eta_u(B - d_u - i_u) \leq \eta_u(B - d_u - 2^j + 1)$. Using the "total reward" property in Lemma 6, we have:

$$\sum_{j \in [L]} \sum_{u \in O_j} \eta_u(B - d_u - 2^j + 1) \quad \geq \quad \sum_{v \in \sigma} \overline{r}_v \quad \geq \quad \mathsf{Opt}/2.$$

We used the fact that for any vertex $w \in \sigma$ with $v_L \prec w$, $\overline{r}_w = 0$ since the total size observed before w is at least $i_{v_L} > B$.

To see the second property consider any $j \in [L]$. Let $w \in \sigma$ be the vertex immediately preceding v_j, and let w' be the immediate predecessor of w. By definition of portal v_j, we have $i_w < 2^{j+1}$; i.e. $\sum_{u \preceq w'} X_u^j = i_w < 2^{j+1}$. Using the "prefix size" property in Lemma 6 with w' and j, we obtain that $\sum_{u \prec w} \mu_u^j \leq K \cdot 2^j$. So $\sum_{\ell=0}^{j} \sum_{u \in O_\ell} \mu_u^j = \sum_{u \prec w} \mu_u^j + \mu_w^j \leq (K+1) \cdot 2^j$. $\square$

4 New Approximation Algorithm for Correlated Orienteering

In this section, we present an improved quasi-polynomial time approximation algorithm for correlated stochastic orienteering, and prove Theorem 3.

An important subroutine in our algorithm is the *deadline orienteering* problem [1]. The input to deadline-orienteering is a metric (U, d) denoting travel times, rewards $\{r_v\}_{v\in U}$ and deadlines $\{\Delta_v\}_{v\in U}$ at all vertices, start (s) and end (t) vertices, and length bound D. The objective is to compute an $s-t$ path of length at most D that maximizes the reward from vertices visited before their deadlines. The best approximation ratio for this problem is $\alpha = \min\{O(\log n), O(\log B)\}$ due to Bansal et al. [1]; see also Chekuri et al. [7]. (Strictly speaking, this definition is slightly more general than the usual one where there is no end-vertex or length bound; but all known approximation algorithms also work for the version we use here.) We actually need an algorithm for a generalization of this problem, in the presence of an additional knapsack constraint. The input to *knapsack deadline orienteering* (KDO) is the same as deadline-orienteering, along with a knapsack constraint given by sizes $\{a_v : v \in U\}$ and capacity A. The objective is an $s-t$ path of length at most D, having total knapsack size at most A, that maximizes the reward from vertices visited before their deadlines. We will use the following known result for KDO.

Theorem 5 ([14]) *There is an $O(\alpha)$-approximation algorithm for knapsack deadline orienteering, where α denotes the best approximation ratio for the deadline orienteering problem.*

Outline: The algorithm involves three main steps. First, it guesses (by enumeration) $\log B$ many "portal vertices" corresponding to the near-optimal structure in Lemma 7, as well as the distances traveled between consecutive portal vertices. (This is the only step that requires quasi-polynomial time.) Next, based on this information, the algorithm solves a configuration LP-relaxation for paths between the portal vertices. This step also makes use of some results/ideas from the previous algorithm [14]. Finally, the algorithm uses randomized rounding with alterations to compute the non-adaptive policy from the LP solution.

Portals. The enumeration algorithm guesses the $L+1$ portal vertices $\{v_j\}_{j=0}^{L}$ as in Lemma 7; recall that $L = \lceil \log_2 B \rceil$. It also guesses the lengths $\{D_j\}_{j=0}^{L}$ of the segments O_js in Lemma 7. This requires running time $(nB)^{L+1}$. A more careful enumeration can be used to reduce the running time to $(n \log B)^{O(L)}$; see [3] for details.

Knapsack Deadline Orienteering Instances. Our goal is to find a $v_{j-1} - v_j$ path P_j for each $j \in [L]$, that have properties similar to the segments in Lemma 7. To this end, we define a knapsack-deadline-orienteering instance $\mathcal{I}_j$ for each $j \in [L]$. Instance $\mathcal{I}_j$ is defined on metric (V, d) with start vertex v_{j-1}, end vertex v_j and length bound D_j. The rewards, sizes and deadlines will be defined shortly. As in [14], we introduce a suitable set of *copies* of each vertex $u \in V$, with deadlines that correspond to visiting u at different possible times.

$$\text{For each } v \in V \text{ and } d \in [B],\ f_v^j(d) := \eta_v(B - \textstyle\sum_{i=0}^{j-1} D_i - d - 2^j + 1), \quad (11)$$

the expected reward from v under instantiation at most $B - \sum_{i=0}^{j-1} D_i - d - 2^j + 1$. Intuitively, $f_v^j(d)$ is the expected reward obtained by visiting vertex v in segment j at distance d along P_j (so the total distance to v is $\sum_{i=0}^{j-1} D_i + d$) and having observed total size $2^j - 1$ until v.

In the KDO instance $\mathcal{I}_j$, we would like to introduce a copy of vertex v for each distance $d \in [B]$, in order to permit all possibilities for visiting v in segment j. However, solutions to such an instance may obtain a large reward just by visiting multiple copies of the same vertex. In order to control the total reward obtainable from multiple copies of a vertex, we introduce copies of each vertex corresponding only to a suitable subset of $[B]$. This relies on the following construction.

Claim 10 ([14]) *For each $j \in [L]$ and $v \in V$, we can compute in polynomial time, a subset $I_v^j \subseteq [B]$ s.t.*

$$\frac{1}{3} \cdot \sum_{y \in I_v^j : y \geq d} f_v^j(y) \quad \leq \quad f_v^j(d) \quad \leq \quad 2 \cdot \max_{y \in I_v^j : y \geq d} f_v^j(y), \qquad \forall d \in [B].$$

Roughly speaking, the copies I_v^j of vertex v are those times t at which its expected reward $f_v^j(t)$ doubles. We now formally define the entire instance $\mathcal{I}_j$.

Definition 2 (KDO instance $\mathcal{I}_j$) The metric is (V, d), start-vertex v_{j-1}, end-vertex v_j, length bound D_j and knapsack capacity $(K+1) \cdot 2^j$. For each $v \in V$ and $t \in I_v^j$, a job $\langle v, j, t \rangle$ with reward $f_v^j(t)$, deadline t and knapsack-size μ_v^j is located at vertex v.

To reduce notation, when it is clear from context, we will use u, v etc. to refer to jobs as well. A feasible solution τ to $\mathcal{I}_j$ is a $v_{j-1} - v_j$ path of length $d(\tau) \leq D_j$ and total size $\sum_{u \in \tau} \mu_u^j \leq (K+1) \cdot 2^j$; the objective value is the total reward of jobs visited by τ before their deadlines. We will also use $\mathcal{I}_j$ to denote the set of all feasible solutions to this KDO instance.

Configuration LP relaxation. We now present an LP relaxation to the near-optimal structure in Lemma 7. We make use of the guessed portals $\{v_j\}_{j=0}^L$ and lengths $\{D_j\}_{j=0}^L$. The segments O_js in Lemma 7 will correspond to solutions of KDO instances $\mathcal{I}_j$s. The LP relaxation is given as (LP) below. Here, $\{x_\tau^j : \tau \in \mathcal{I}_j\}$ is intended to be the indicator variable that chooses a solution path for the KDO instance $\mathcal{I}_j$. Variables $y_{v,t}^j$ indicate whether/not job $\langle v, j, t \rangle$ is selected in $\mathcal{I}_j$. These are enforced by constraints (12) and (13). Constraint (14) requires at most one copy of each vertex to be chosen, over all segments. And constraint (15) bounds the "prefix mean size" as in Lemma 7. The objective corresponds to the reward in Lemma 7.

$$\max \sum_{j=0}^{L} \sum_{v\in V} \sum_{t\in I_v^j} f_v^j(t)\cdot y_{v,t}^j \tag{LP}$$

$$\text{s.t.} \sum_{\tau\in\mathcal{I}_j} x_\tau^j \le 1 \qquad \forall\, j\in[L] \tag{12}$$

$$y_{v,t}^j \le \sum_{\tau\ni\langle v,j,t\rangle} x_\tau^j \qquad \forall\, j\in[L],\, v\in V,\, t\in I_v^j \tag{13}$$

$$\sum_{j=0}^{L}\sum_{t\in I_v^j} y_{v,t}^j \le 1 \qquad \forall\, v\in V \tag{14}$$

$$\sum_{v\in V}\mu_v^j\cdot\sum_{\ell=0}^{j}\sum_{t\in I_v^\ell} y_{v,t}^\ell \le (K+1)\cdot 2^j \qquad \forall\, j\in[L] \tag{15}$$

$$\mathbf{x},\,\mathbf{y}\ge 0. \tag{16}$$

First, observe that we can ensure equality in (13).

Claim 11 *Any* LP *solution* (x, y) *can be modified to another solution that satisfies* (13) *with equality and has the same objective value.*

Proof This follows immediately from the fact that each solution-set $\mathcal{I}_j$ is "down monotone", i.e. $I\in\mathcal{I}_j$ and $I'\subseteq I$ implies $I'\in\mathcal{I}_j$. Formally, consider the constraints (13) in any order. For each $\langle v,j,t\rangle$, order arbitrarily the x_τ^j variables with $\tau\ni\langle v,j,t\rangle$ as $x_{\tau(1)}^j, x_{\tau(2)}^j,\cdots,x_{\tau(a)}^j$. Let b denote the unique index with $\sum_{k=1}^{b-1} x_{\tau(k)}^j < y_{v,t}^j \le \sum_{k=1}^{b} x_{\tau(k)}^j$. Now perform the following modification:

– For each index $k>b$, drop $\langle v,j,t\rangle$ from the solution $\tau(k)$.
– For index b, consider the solutions $\sigma=\tau(b)$ and $\sigma'=\tau(b)\setminus\langle v,j,t\rangle$. Set $x_{\sigma'}^j=\sum_{k=1}^{b} x_{\tau(k)}^j - y_{v,t}^j$ and $x_\sigma^j = y_{v,t}^j-\sum_{k=1}^{b-1} x_{\tau(k)}^j$.

After the change, it is clear that we have equality for $\langle v,j,t\rangle$ in constraint (13). Also, (12) remains feasible. Finally, since the y variables remain unchanged, (14)-(15) remain feasible and the objective value stays the same. □

Claim 12 *The optimal value of* LP *is at least* Opt/4.

Proof We will show that the subpaths $\{O_j\}_{j=0}^{L}$ in Lemma 7 correspond to a feasible integral solution to LP. Let σ denote the combined path $O_0, O_1,\cdots,O_L$. Based on our guess of the portals and distances, we have $d(O_j)\le D_j$. For any vertex $u\in O_j$ let t_u denote the distance to u along O_j; note that the distance to u along σ is then $d_u=\sum_{i=0}^{j-1} D_i + t_u$ (this quantity also appears in Lemma 7). For each $u\in O_j$ let $t'_u := \min\{t\in I_u^j : t\ge t_u\}$ be the deadline of the earliest copy of u that path O_j can visit in the KDO instance $\mathcal{I}_j$. Consider O_j as a solution to $\mathcal{I}_j$ which visits jobs $\{\langle u,j,t'_u\rangle : u\in O_j\}$. In order to verify the feasibility of O_j, we only need to check that $\sum_{u\in O_j} \mu_u^j \leq (K+1)\cdot 2^j$, which follows from the second property in Lemma 7.

Consider now the solution to LP with $x^j_{O_j}=1$ for all $j\in[L]$; $y^j_{u,t'_u}=1$ for each $u\in O_j$, $j\in[L]$; and all other variables set to zero. Constraints (12), (13), (14) and (16) are clearly satisfied. Observe that the left-hand-side of (15) is exactly $\sum_{\ell=0}^{j}\sum_{u\in O_\ell}\mu_u^j$ which is at most $(K+1)\cdot 2^j$ by the second property in Lemma 7. So $(\mathbf{x},\mathbf{y})$ is a feasible (integral) solution to LP. We now bound the objective value:

$$\sum_{j=0}^{L}\sum_{u\in O_j} f_u^j(t'_u) \geq \frac{1}{2}\cdot\sum_{j=0}^{L}\sum_{u\in O_j} f_u^j(t_u) \;=\; \frac{1}{2}\cdot\sum_{j=0}^{L}\sum_{u\in O_j}\eta_u\big(B-\sum_{i=0}^{j-1}D_i - t_u - 2^j+1\big)$$

$$=\frac{1}{2}\cdot\sum_{j=0}^{L}\sum_{u\in O_j}\eta_u\left(B-d_u-2^j+1\right) \quad\geq\quad \frac{\mathsf{Opt}}{4}.$$

The first inequality is by definition of t'_u and Claim 10; the next two equalities use the definitions $f_u^j(\cdot)$ and d_u; and the final inequality is by the first property in Lemma 7. □

Solving the configuration LP. Note that LP has an exponential number of variables but only a polynomial number of constraints. We show that LP can be solved approximately using an approximation algorithm for KDO. This is based on applying the ellipsoid algorithm to the dual of LP, which is given below:

$$\min \;\sum_{j=0}^{L}\beta_j \;+\; \sum_{v\in V}\delta_v \;+\;(K+1)\cdot\sum_{j=0}^{L}2^j\cdot z_j \qquad (\mathsf{DLP})$$

$$\text{s.t.}\quad \gamma^j_{v,t}+\delta_v+\sum_{\ell\geq j}\mu_v^\ell\cdot z_\ell \;\geq\; f_v^j(t) \qquad \forall\, j\in[L],\, v\in V,\, t\in I_v^j \qquad (17)$$

$$-\sum_{\langle v,j,t\rangle\in\tau}\gamma^j_{v,t}\;+\;\beta_j\;\geq\;0 \qquad \forall\, j\in[L],\,\tau\in\mathcal{I}_j \qquad (18)$$

$$\beta,\,\gamma,\,\delta,\,z\geq 0. \qquad (19)$$

In order to solve DLP using the ellipsoid method, we need to provide a separation oracle that tests feasibility. Observe that constraints (17) are polynomial in number, and can be checked explicitly. Constraint (18) for any $j\in[L]$ is equivalent to asking whether the optimal value of KDO instance $\mathcal{I}_j$ with rewards $\{\gamma^j_{v,t} : v\in V,\, t\in I_v^j\}$ is at most β_j. Using an $O(\alpha)$-approximate separation oracle (the knapsack deadline orienteering algorithm from Theorem 5) within the Ellipsoid algorithm, we obtain an $O(\alpha)$-approximation algorithm for DLP and hence LP.

Remark: Alternatively, we can solve LP using a faster combinatorial algorithm that is based on multiplicative weight updates. Note that we can eliminate y-variables in LP by setting constraints (13) to equality (Claim 11). This results in a *packing LP*, consisting of non-negative variables with each constraint of the form $\mathbf{a}^T x \leq b$ where all entries in $\mathbf{a}$ and b are non-negative. So we can solve LP using faster approximation algorithms [16,10] for packing LPs, that also require only an approximate dual separation oracle (which is KDO in our setting).

Rounding the LP solution. Let (x, y) denote the $O(\alpha)$-approximate solution to LP. By Claim 12 the objective value is $\Omega(\mathsf{Opt}/\alpha)$. The rounding algorithm below describes a (randomized) non-adaptive policy. In the following we use the shorthand $\zeta := \frac{\log\log L}{\log L} = \Theta(\frac{\log\log\log B}{\log\log B})$.

1. For each $j \in [L]$, independently select solution $\tau_j \in \mathcal{I}_j$ as:

$$\tau_j \leftarrow \begin{cases} T & \text{with probability } \frac{x_T^j}{2}, \text{ for each } T \in \mathcal{I}_j \\ \text{edge } (v_{j-1}, v_j) & \text{with the remaining probability } 1 - \frac{1}{2}\sum_{T \in \mathcal{I}_j} x_T^j \end{cases}$$

2. If any vertex $v \in V$ appears in more than one solution $\{\tau_j\}_{j=0}^L$, then drop v from all of them.
3. If solution $\{\tau_j\}_{j=0}^L$ exceeds any constraint (15) by a factor more than $\frac{1}{\zeta}$ then return an empty solution.
4. For each $j \in [L]$, if τ_j contains multiple copies of any vertex $v \in V$ then retain only the copy with earliest deadline (i.e. highest reward).
5. Return the non-adaptive policy that traverses the path $\tau_0 \cdot \tau_1 \cdots \tau_L$ and attempts each vertex independently with probability $\frac{\zeta}{4K}$.

Analysis. We now show that the expected reward of this non-adaptive policy is $\Omega(\frac{\zeta}{\alpha K}) \cdot \mathsf{Opt}$, which would prove Theorem 3; recall that $K = \Theta(\log\log B)$ and $L = \Theta(\log B)$.

Lemma 8 *For any $j \in [L], v \in V, t \in I_v^j$,*

$$\Pr[\langle v, j, t\rangle \in \tau_j \textit{ after Step 3}] \geq y_{v,t}^j / 8.$$

Proof Let τ_j^1, τ_j^2 and τ_j^3 denote the solution τ_j after Step 1, 2 and 3 respectively.

$$\Pr\left[\langle v, j, t\rangle \in \tau_j^1\right] \quad = \sum_{T \ni \langle v,j,t\rangle} \frac{x_T^j}{2} \quad = \quad \frac{y_{v,t}^j}{2}. \tag{20}$$

The last equality uses Claim 11. Note that $\langle v, j, t\rangle$ gets dropped in Step 2 exactly when there is some other solution $\{\tau_\ell^1 : \ell \in [L] \setminus j\}$ that contains a copy of v. By union bound, (20) and (14),

$$\Pr\left[\langle v, j, t\rangle \notin \tau_j^2 \mid \langle v, j, t\rangle \in \tau_j^1\right] \quad \leq \sum_{\ell \in [L]\setminus j} \sum_{t \in I_v^\ell} \frac{y_{v,t}^j}{2} \quad \leq \quad \frac{1}{2}. \tag{21}$$

In Step 3, the entire solution is declared empty if any constraint (15) is violated by more than a factor of $\frac{1}{\zeta}$; otherwise, $\tau_j^3 = \tau_j^2$. Consider the constraint (15) with index $h \in [L]$. This reads $\sum_{\ell=0}^h \mu^h(\tau_\ell) \leq (K+1) \cdot 2^h$, where $\mu^h(\tau_\ell) := \sum_{\langle v,\ell,t\rangle \in \tau_\ell} \mu_v^h$. The key observation is the following:

$$\text{For any } \ell \leq h \text{ and } \tau_\ell \in \mathcal{I}_\ell, \text{ we have } \mu^h(\tau_\ell) \leq (K+1) \cdot 2^h. \tag{22}$$

This uses the fact that τ_ℓ satisfies the knapsack constraint $\mu^\ell(\tau_\ell) \leq (K+1)\cdot 2^\ell$ in KDO instance $\mathcal{I}_\ell$; and by the observation (9) on capped sizes, we have $\frac{\mu^h(\tau_\ell)}{2^h} \leq \frac{\mu^\ell(\tau_\ell)}{2^\ell}$ since $h \geq \ell$.

Using (22) it follows that $Z_h := \sum_{\ell=0}^{h} \frac{\mu^h(\tau_\ell)}{(K+1)\cdot 2^h}$ is the sum of independent $[0,1]$ bounded random variables. Using (20), and LP constraint (15) we have:

$$\mathbb{E}[Z_h] \quad \leq \quad \frac{1}{(K+1)\cdot 2^h} \sum_{\ell=0}^{h} \sum_{v \in V} \sum_{t \in I_v^\ell} \mu_v^h \cdot \frac{y_{v,t}^j}{2} \quad \leq \quad \frac{1}{2}.$$

Hence, by a Chernoff bound, $\Pr\left[Z_h > \frac{1}{\zeta}\right] \leq \frac{1}{2L}$.

Taking a union bound over all L constraints (15), it follows that with probability at least half, none of them is violated by a factor more than $\frac{1}{\zeta}$. That is, $\Pr\left[\langle v,j,t\rangle \in \tau_j^3 \mid \langle v,j,t\rangle \in \tau_j^2\right] \geq \frac{1}{2}$.

Combined with (20) and (21), we obtain the lemma. □

Claim 13 *The expected* LP *objective of solution* $\{\tau_j\}_{j=0}^{L}$ *after Step 4 is* $\Omega(\mathsf{Opt}/\alpha)$.

Proof By Lemma 8 it follows that expected LP objective value of solution $\{\tau_j\}_{j=0}^{L}$ after Step 3 is at least:

$$\sum_{j=0}^{L} \sum_{v \in V} \sum_{t \in I_v^j} f_v^j(t) \cdot \frac{y_{v,t}^j}{8} \quad \geq \quad \Omega\left(\frac{\mathsf{Opt}}{\alpha}\right).$$

The last inequality uses Claim 12 and the fact that we have an $O(\alpha)$ approximately optimal LP solution. In Step 4, we retain only one copy of each vertex in each τ_j. Using Claim 10, since we retain the most profitable copy of each vertex, this decreases the total reward of each τ_j by at most a factor of 3. □

Consider now the non-adaptive policy in Step 5 and *condition* on any solution $\{\tau_j\}_{j=0}^{L}$. Fix any $j \in [L]$ and $\langle v,j,t\rangle \in \tau_j$. The distance traveled until v is at most $\sum_{i=0}^{j-1} D_i + t$. By Step 4 and LP constraint (15), the total μ^j-size of vertices in $\{\tau_i\}_{i=0}^{j}$ is $\sum_{i=0}^{j} \sum_{v\in\tau_i} \mu_v^j \leq \frac{K+1}{\zeta}\cdot 2^j$. Since each vertex is attempted only with probability $\frac{\zeta}{4K}$, we have by Markov's inequality:

$$\Pr\left[\sum_{u \prec v} \min\{S_u, 2^j\} \geq 2^j\right] \;\leq\; \frac{1}{2},$$

where the summation ranges over all vertices visited before v. Thus with probability at least half, vertex v is visited by time $\Delta_v := \sum_{i=0}^{j-1} D_i + t + 2^j - 1$. Since job distributions are independent across vertices, the expected reward from vertex v is at least:

$$\frac{1}{2}\cdot \Pr[\text{attempt } v] \cdot \sum_{s=0}^{B-\Delta_v} \pi_v(s)\cdot r_v(s) \;=\; \Omega\left(\frac{\zeta}{K}\right)\cdot \eta_v(B - \sum_{i=0}^{j-1} D_i - t - 2^j + 1),$$

that is $\Omega\left(\frac{\zeta}{K}\right) \cdot f_v^j(t)$. Adding this contribution over all vertices, the total reward is $\Omega\left(\frac{\zeta}{K}\right)$ times the LP objective of solution $\{\tau_j\}_{j=0}^{L}$ after Step 4. Taking expectations over Steps 1-4, and using Claim 13, it follows that our non-adaptive policy has expected reward $\Omega\left(\frac{\zeta}{\alpha K}\right) \cdot \mathsf{Opt}$. This completes the proof of Theorem 3; recall $K = \Theta(\log\log B)$ and $\zeta = \Theta(\frac{\log\log\log B}{\log\log B})$.

5 Conclusion

In this paper, we proved an $\Omega(\sqrt{\log\log B})$ lower bound on the adaptivity gap of stochastic orienteering. The best known upper bound is $O(\log\log B)$ [14]. Closing this gap is an interesting open question. Our lower bound instance has a tight $\Theta(\sqrt{\log\log B})$ adaptivity gap, and an improved result would require additional ideas. Although our $\omega(1)$ adaptivity gap holds even on line-metrics, there may be other classes of metrics for which the adaptivity gap is a constant.

For the *correlated* stochastic orienteering problem, we gave a quasi polynomial time $O(\alpha \cdot \log^2 \log B)$-approximation algorithm, where α denotes the best approximation ratio for the deadline-orienteering problem. It is known that correlated stochastic orienteering can not be approximated to a factor better than $\Omega(\alpha)$ [14]. Finding an $O(\alpha)$ approximation algorithm for correlated stochastic orienteering is another interesting direction.

References

1. Bansal, N., Blum, A., Chawla, S., Meyerson, A.: Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In: STOC, pp. 166–174 (2004)
2. Bansal, N., Gupta, A., Li, J., Mestre, J., Nagarajan, V., Rudra, A.: When LP is the cure for your matching woes: Improved bounds for stochastic matchings. Algorithmica **63**(4), 733–762 (2012)
3. Bansal, N., Nagarajan, V.: On the adaptivity gap of stochastic orienteering. CoRR **abs/1311.3623** (2013)
4. Bhalgat, A.: A $(2+\epsilon)$-approximation algorithm for the stochastic knapsack problem (2011). Unpublished Manuscript
5. Bhalgat, A., Goel, A., Khanna, S.: Improved approximation results for stochastic knapsack problems. In: SODA, pp. 1647–1665 (2011)
6. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward TSP. SIAM J. Comput. **37**(2), 653–670 (2007)
7. Chekuri, C., Korula, N., Pál, M.: Improved algorithms for orienteering and related problems. ACM TALG **8**(3) (2012)
8. Chen, N., Immorlica, N., Karlin, A.R., Mahdian, M., Rudra, A.: Approximating matches made in heaven. In: International Colloquium on Automata, Languages and Programming (ICALP), pp. 266–278 (2009)
9. Dean, B.C., Goemans, M.X., Vondrák, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. Math. Oper. Res. **33**(4), 945–964 (2008)
10. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. SIAM J. Comput. **37**(2), 630–652 (2007)
11. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics **34**(3), 307318 (1987)

12. Guha, S., Munagala, K.: Multi-armed bandits with metric switching costs. In: ICALP, pp. 496–507 (2009)
13. Gupta, A., Krishnaswamy, R., Molinaro, M., Ravi, R.: Approximation algorithms for correlated knapsacks and non-martingale bandits. In: FOCS, pp. 827–836 (2011)
14. Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R.: Approximation algorithms for stochastic orienteering. In: SODA, pp. 1522–1538 (2012)
15. Ma, W.: Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms. In: SODA (2014)
16. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. In: FOCS, pp. 495–504 (1991)
17. Tong, Z.: Data dependent concentration bounds for sequential prediction algorithms. In: COLT, pp. 173–187 (2005)
18. Vansteenwegena, P., Souffriaua, W., Oudheusdena, D.V.: The orienteering problem: A survey. Eur. J. Oper. Res. **209**(1), 1–10 (2011)
19. Weyland, D.: Stochastic vehicle routing - from theory to practice. Ph.D. thesis, University of Lugano, Switzerland (2013)