MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0364-765X | EISSN 1526-5471 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000
© 0000 INFORMS

Running Errands in Time: Approximation Algorithms for Stochastic Orienteering*

Anupam Gupta
Department of Computer Science, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA 15213. Email: anupamg@cs.cmu.edu

Ravishankar Krishnaswamy
Computer Science Department, Princeton University, 35 Olden St, Princeton, NJ 08540. Email: ravishan@cs.cmu.edu

Viswanath Nagarajan
IBM T.J. Watson Research Center, 1101 Kitchawan Rd, Yorktown Heights, NY 10598. Email: viswanat@us.ibm.com

R. Ravi
Tepper School of Business, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA 15213. Email: ravi@cmu.edu

In the *Stochastic Orienteering* problem, we are given a finite metric space, where each node contains a job with some deterministic reward and a random processing time. The processing time distributions are known and independent across nodes. However the actual processing time of a job is not known until it is completely processed. The objective is to compute a non-anticipatory policy to visit nodes (and run the corresponding jobs) so as to maximize the total expected reward, subject to the total distance traveled plus the total processing time being at most a given budget of B. This problem combines aspects of the *stochastic knapsack* problem with uncertain item sizes [18], as well as the *deterministic orienteering* problem [8].

In this paper, we consider both non-adaptive and adaptive policies for Stochastic Orienteering. We present a constant-factor approximation algorithm for the non-adaptive version, and an $O(\log \log B)$-approximation algorithm for the adaptive version. We extend both these results to directed metrics and a more general *sequence orienteering* problem.

Finally, we address the Stochastic Orienteering problem when the node rewards are also random and possibly correlated with the processing time, and obtain an $O(\log n \log B)$-approximation algorithm; here n is the number of nodes in the metric. All our results for adaptive policies also bound the corresponding "adaptivity gaps".

Key words: Approximation Algorithms, Adaptivity Gap, Orienteering Problem, Stochastic Optimization
MSC2000 subject classification: 68W25, 90B36, 90B15, 90B06
OR/MS subject classification: Primary: Stochastic algorithms; secondary: Network/Graph algorithms

* A preliminary version of this paper appeared in the ACM-SIAM Symposium on Discrete Algorithms, 2012.

1. Introduction

Consider the following problem: you start your day at home with a set of jobs to run at various locations (e.g., at the bank, the post office, the grocery store), but you only have limited time to run those jobs in (say, you have from 9am until 5pm, when all these shops close). Each successfully completed job j gives you some fixed reward r_j. You know the time it takes you to travel between the various job locations: these distances are deterministic and form a metric (V, d). However, you do not know the amount of time you will spend processing each job (e.g., standing in the queue, filling out forms). Instead, for each job j, you are only given the probability distribution π_j governing the random amount of time you need to spend performing j. That is, once you start performing the job j, the job finishes after S_j time units and you get the reward, where S_j is a random variable denoting the size, and distributed according to π_j. Before you reach the job, all you know about its size is what can be gleaned from the distribution π_j of S_j; and even having worked on j for t units of time, all you know about the actual size of j is what you can infer from the conditional distribution $(S_j \mid S_j > t)$. We consider a non-preemptive setting, where each job must be run to completion once started (we can also handle a variant where job cancellations are allowed). The goal is now a natural one: given the metric (V, d), the starting point ρ, rewards of jobs, the time budget B, and the probability distributions for all the jobs, give a policy for traveling around and processing the jobs that maximizes the expected reward accrued. Due to the hard budget constraint, there might be a partially finished job at the horizon B— such jobs do not contribute to the objective.

The case when all the sizes are zero (i.e., $S_j = 0$ with probability 1) is the deterministic orienteering problem, for which we now know a $(2 + \epsilon)$-approximation algorithm [8, 12]. Another special case, where all the jobs are located at the start node (i.e. the metric is zero), but the sizes are random, is the stochastic knapsack problem, which also admits a $(2 + \epsilon)$-approximation algorithm [18, 6]. However, the stochastic orienteering problem above, which combines aspects of both these problems, seems to have been hitherto unexplored in the approximation algorithms literature.

Furthermore, it is not known even for stochastic knapsack, whether an optimal adaptive policy can always be represented using polynomial space; moreover certain questions on the optimal policy are PSPACE-hard [18]. This raises the issue of how well we can approximate the optimal adaptive policies, by policies of polynomially bounded descriptions? Indeed, a natural class of policies which fit this description are the so-called *non-adaptive* solutions. A non-adaptive solution for stochastic orienteering is simply a permutation P of points in the metric space starting at the root ρ: we visit the points in this fixed order, performing the jobs at the points we reach, until time runs out. The ratio of the expected reward of the best non-adaptive policy to that of the optimal adaptive policy is called the *adaptivity gap* of the problem [18].

1.1. Our Results and Techniques

Our main result is the following:

THEOREM 1. *There is an $O(\log\log B)$-approximation algorithm for adaptive stochastic orienteering.*

The algorithm also gives a *bicriteria* approximation guarantee that for any $\epsilon > 0$ finds a solution that spends time $(1 + \epsilon) \cdot B$ and whose expected reward is $O(\log\log \frac{1}{\epsilon})$ times the expected reward of the optimal policy using time B.

Our proof proceeds by first showing the following structural result that bounds the adaptivity gap: *there exists a value W^* such that the optimal non-adaptive solution which spends at most W^* time in processing jobs and $B - W^*$ time in traveling, gets an $\Omega(1/\log\log B)$ fraction of the optimal reward.* Naïvely we would expect only a logarithmic fraction of the reward by considering $\log_2 B$ possibilities for W^* (all powers of two). However, we do better, and the underlying structure result (Lemma 4) is the technical heart of the paper. The proof is via a martingale argument. We then obtain Theorem 1 by combining Lemma 4 with the following result about non-adaptive stochastic orienteering.

THEOREM 2. *There is an $O(1)$-approximation algorithm for non-adaptive stochastic orienteering.*

It turns out that the dependence on $O(\log\log B)$ for the adaptivity gap is not just a byproduct of our analysis. Indeed, very recently, Bansal and Nagarajan [4] have established an $\Omega(\sqrt{\log\log B})$ lower bound on the adaptivity gap of the stochastic orienteering problem!

Most previous adaptivity gaps in the literature are proved using linear programming relaxations that capture optimal adaptive policies, and then rounding the fractional LP solutions to get non-adaptive policies. However, we do not know a good relaxation for even the deterministic orienteering problem, so taking this approach seems difficult. Thus we argue directly about the optimal adaptive policy to prove our adaptivity gap results. In particular, we use a martingale argument to show the existence of a "path" (i.e., a non-adaptive policy) with large reward within the optimal "tree" (i.e., the optimal adaptive policy).

Next, we extend our results to a generalization of the basic orienteering problem called *sequence orienteering*. In this problem we are given a sequence of k "portal vertices", and a solution to sequence orienteering must visit the portals in the given order while not exceeding the budget. (A formal definition appears in Section 2.) The basic orienteering problem corresponds to having a single portal, namely the starting vertex ρ. Our results for sequence orienteering also extend to the case of *directed* metrics. The performance of our algorithms for this problem is as follows:

THEOREM 3. *The stochastic sequence orienteering problem admits the following guarantees.*
- *An $O(\alpha)$-approximation algorithm for the optimal non-adaptive policy*
- *An $O(\alpha\cdot\log\log B)$-approximation algorithm for the optimal adaptive policy.*

Here, the quantity α denotes the best approximation ratio for the point-to-point orienteering problem.

The *point-to-point orienteering* problem [2] is the special case of sequence orienteering with $k=2$: namely, given a metric with rewards at vertices, a length bound B, starting and ending vertices s and t respectively, find an s-t path of length at most B that maximizes the reward on its vertices. The best approximation ratio known for point-to-point orienteering is $\alpha = 2+\epsilon$ for symmetric metrics [12], and $\alpha = O\left(\min\{\frac{\log^2 n}{\log\log n}, \log^2 \mathsf{Opt}\}\right)$ in directed metrics [28, 12]. As far as we know, even the deterministic version of sequence orienteering has not been studied before, and a central step in proving Theorem 3 is to give an $O(\alpha)$-approximation algorithm for deterministic sequence orienteering.

A second generalization is to the setting where *both the rewards and job sizes* are random and not necessarily independent of each other. In this setting we show the following result.

THEOREM 4. *There is a polynomial-time algorithm that outputs a non-adaptive policy for correlated stochastic orienteering, achieving an $O(\log n\log B)$-approximation to the best adaptive policy. Moreover, this problem is at least as hard as the orienteering-with-deadlines problem.*

The *orienteering-with-deadlines* problem [2] is one where we are given a metric with deadlines at vertices and a starting vertex ρ, and want to compute a path starting at ρ (at time zero) that maximizes the number of vertices visited before their respective deadlines. The currently best approximation algorithm for the orienteering-with-deadlines problem achieves an $O(\log n)$ ratio [2].

1.2. Related Work

The (deterministic) orienteering problem is known to be APX-hard, and the first constant-factor approximation algorithm was due to Blum et al. [8]. Their factor of 4 was improved by [2] and ultimately by [12] to $(2+\epsilon)$ for every $\epsilon>0$. There is a PTAS known for the orienteering problem on low-dimensional Euclidean space [16]. The orienteering problem has also been useful as a subroutine for obtaining approximation algorithms for other vehicle routing problems such as TSP with deadlines and time-windows [2, 13, 14].

To the best of our knowledge, the stochastic version of the orienteering problem has not been studied before from the perspective of approximation algorithms. Heuristics and empirical guarantees for a similar problem were given by Campbell et al. [10].

The stochastic knapsack problem [18] is a special case of stochastic orienteering, where all the jobs are located at the root ρ itself. Dean et al. [18] gave the first constant factor approximation algorithm for this basic problem. Recently, Gupta et al. [22] considered an extension with *correlated* rewards and sizes, and obtained a different $O(1)$-approximation algorithm.

Another very related body of work is on budgeted learning with metric costs. Specifically, in the work of Guha and Munagala [21], there is a collection of Markov chains located in a metric, each state of each chain having an associated reward. When at a Markov chain at location j, the policy can advance that chain one step every unit of time. Given a bound of L time units for traveling, and a bound of C time units for advancing Markov chains, the goal is to maximize some function (say the sum or the max) of rewards of the final states in expectation. [21] gave an elegant constant factor approximation algorithm for this problem (under some mild conditions on the rewards) via a reduction to classical orienteering using Lagrangean multipliers. Our algorithm/analysis for the "knapsack orienteering" problem (defined in Section 2) is inspired by theirs; the analysis of our algorithm though is simpler, due to the problem itself being deterministic. This can be used to obtain a constant-factor approximation algorithm for the variant of stochastic orienteering with two *separate* budgets for travel time and processing time. However, it is unclear how to use the approach from [21] to obtain an approximation ratio better than $O(\log B)$ for the (single budget) stochastic orienteering problem that we consider.

Approximation algorithms have been studied for adaptive versions of a number of combinatorial optimization problems. Many of these results, machine scheduling [27], knapsack [18], budgeted learning [20], matchings [3] etc. are based on LP relaxations that capture certain expected values of the optimal adaptive policy. Such an LP-based approach was also used in earlier optimality proofs for some stochastic queuing problems [25] and the multi-armed bandit problem [5]. An LP-based approach is not directly useful for stochastic orienteering since we do not know good LP relaxations even for deterministic orienteering.

On the other hand, there are also other papers on stochastic matchings [17], stochastic knapsack [7, 6] and optimal decision trees [26, 1, 23] that have had to reason about the optimal adaptive policies directly. We hope that our martingale-based analysis for stochastic orienteering will add to the set of tools used for adaptive optimization problems.

1.3. Outline

We begin with some definitions in Section 2, and then give an algorithm for the deterministic *knapsack orienteering problem* in Section 3, which will be a crucial sub-routine in the subsequent algorithms. We then present a constant-factor approximation algorithm for non-adaptive stochastic orienteering (Theorem 2) in Section 4. This naturally leads us to our main result in Section 5, the $O(\log\log B)$-adaptivity gap for stochastic orienteering (Theorem 1). In Section 6 we consider the stochastic sequence orienteering problem and extend our results to this general setting (Theorem 3). Then in Section 7, we obtain a poly-logarithmic approximation algorithm for the variant of stochastic orienteering where rewards and sizes are correlated (Theorem 4). Finally, as mentioned earlier, our model is non-preemptive, i.e. each job is run to completion once started. In Section 8 we show that the same results can be obtained in the setting where jobs can be prematurely canceled.

2. Definitions and Notation

Stochastic Orienteering. An instance of stochastic orienteering ($\mathsf{StocOrient}$) is defined on an underlying metric space (V,d) with ground 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, which is also referred to as v. For most of the paper (with the exception of Section 7), each job v has a fixed reward $r_v \in \mathbb{Z}^+$; and a random processing time (also called size) S_v, which is distributed according to a known but arbitrary probability distribution $\pi_v : \mathbb{R}^+ \to [0,1]$. We are also given a starting "root" vertex ρ, and a budget B on the total time available.

The only actions allowed to an algorithm are to travel to a vertex v and begin processing the job there: when the job finishes after its random length S_v of time, we get the reward r_v (so long as the total time elapsed, i.e., travel time plus processing time, is at most B), and we can then move to the next job. Recall that this is a non-preemptive model. We show in Section 8 that all our results extend to a related model that allows cancelations: here we can cancel any job at any time without receiving its reward, but we are not allowed to attempt this job again in the future. Furthermore, once we complete a job, we are not allowed to revisit it and process it again. If the application requires that a job be allowed to run multiple times, then we can place many identical copies of the job at the vertex where it is located, and use our algorithms.

Note that any solution (policy) corresponds to a decision tree where each "state" depends on which previous jobs were processed, and what information we obtained about their sizes. Now the goal is to devise a policy which, starting at the root ρ, decides for each possible state the next job to visit and process. Such a policy is called "non-anticipatory" due to the fact that its action at any point in time can only depend on already observed information. The objective is to obtain a policy that maximizes the expected sum of rewards of jobs successfully completed before the total time (travel and processing) reaches the threshold of B. The approximation ratio of an algorithm is defined to be the ratio of the expected reward of an optimal policy to that of the algorithm's policy.

Stochastic Sequence Orienteering. We also consider (in Section 6) a substantial generalization of the stochastic orienteering problem. In the stochastic sequence orienteering problem, the input is a *directed* metric (V,d), sequence $\langle s_1,\dots,s_k\rangle$ of portal vertices, bound B, and at each vertex $v\in V$: reward r_v and random size $S_v\sim\pi_v$. A solution here is an adaptive path that visits vertices (and processes the respective jobs) such that the portals $s_1,\dots,s_k$ are necessarily visited and in that order. The objective is to maximizes the expected reward obtained such that the total time taken is at most B. Since any policy must visit all the portals, if it is running some job v when the residual budget equals the distance from v to the remaining portals, then job v is canceled and the policy terminates by directly visiting the remaining portals. Note that the basic stochastic orienteering problem is the special case of $k=1$ and a symmetric metric.

Stochastic Orienteering with Correlated Rewards. Another extension that we consider (in Section 7) is the setting of *correlated rewards and sizes*. In correlated stochastic orienteering (CorrOrient), the job sizes and rewards are both random, and correlated with each other. The distributions across different vertices are still independent. (Recall that the stochastic knapsack version of this problem also admits a constant factor approximation algorithm [22]).

Adaptive and Non-Adaptive Policies. We are interested in both adaptive and non-adaptive policies, and in particular, want to bound the ratio between the optimal adaptive and non-adaptive policies. 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 j corresponding to the possible sizes in the support of π_j. A *non-adaptive policy*, on the other hand, 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* may pick 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.

Finally, for any integer $m\geq 0$ we use $[m]$ to denote the set $\{0,1,\dots,m\}$.

3. The (Deterministic) Knapsack Orienteering Problem

We now define a variant of the orienteering problem which will be crucially used in the rest of the paper. Recall that in the basic orienteering problem, the input consists of a metric (V,d), the root vertex ρ, rewards r_v for each job v, and total budget B. The goal is to find a path P of length at most B starting at ρ that maximizes the total reward $\sum_{v\in P} r_v$ of vertices in P.

In the *knapsack orienteering* problem (KnapOrient), we are given a metric (V,d), root vertex ρ, and two budgets: L which is the "travel" budget, and W which is the "knapsack" budget. Each job v has a reward $\widehat{r}_v$, and also a "size" $\widehat{s}_v$. A feasible solution is a path P originating at ρ having length at most L, such that the total size $\widehat{s}(P):=\sum_{v\in P}\widehat{s}_v$ is at most W. The goal is to find a solution P of maximum reward $\sum_{v\in P}\widehat{r}_v$.

THEOREM 5. *There is an $O(1)$-approximation algorithm* AlgKO *for the* KnapOrient *problem.*

Proof. The idea of the proof is to consider the *Lagrangian relaxation* of the knapsack constraint; we remark that such an approach was also taken in [21] for a related problem. This way we alter the rewards of items while still optimizing over the set of feasible orienteering solutions. For a suitable choice of the Lagrange parameter, we will show that we can recover a solution with large (unaltered) reward while meeting both the knapsack (W) and length (L) constraints.

For a value $\lambda \geq 0$, define an orienteering instance $\mathcal{I}(\lambda)$ on metric (V, d) with root ρ, travel budget L, and profits $r_v^\lambda := \widehat{r}_v - \lambda \cdot \widehat{s}_v$ at each $v \in V$. Note that the optimal solution to this orienteering instance has value at least $\mathsf{Opt} - \lambda \cdot W$, where Opt is the optimal value of the original $\mathsf{KnapOrient}$ instance.

Let $\mathsf{Alg}_o(\lambda)$ denote an α-approximate solution to $\mathcal{I}(\lambda)$ as well as its profit; we have $\alpha = 2 + \delta$ via the algorithm from [12]. By exhaustive search, let us find:

$$\lambda^* := \max\left\{\lambda \geq 0 : \mathsf{Alg}_o(\lambda) \geq \frac{\lambda \cdot W}{\alpha}\right\} \tag{3.1}$$

Observe that by setting $\lambda = \frac{\mathsf{Opt}}{2W}$, we have $\mathsf{Alg}_o(\lambda) \geq (\mathsf{Opt} - \lambda W)/\alpha = \frac{\mathsf{Opt}}{2\alpha} = \frac{\lambda \cdot W}{\alpha}$. Thus $\lambda^* \geq \frac{\mathsf{Opt}}{2W}$.

Let σ denote the path in solution $\mathsf{Alg}_o(\lambda^*)$, and let $\sum_{v\in\sigma} \widehat{s}_v = y \cdot W$ for some $y \geq 0$. Partition the vertices of σ into $c = \max\{1, \lfloor 2y \rfloor\}$ parts $\sigma_1, \ldots, \sigma_c$ with $\sum_{v\in\sigma_j} \widehat{s}_v \leq W$ for all $j \in \{1, \ldots, c\}$. This partition can be obtained by greedy aggregation since $\max_{v\in V} \widehat{s}_v \leq W$ (all vertices with larger size can be safely excluded by the algorithm). Set $\sigma' \leftarrow \sigma_k$ for $k = \arg\max_{j=1}^{c} \widehat{r}(\sigma_j)$. We then output σ' (which follows path σ but only visits vertices in σ_k) as our approximate solution to the $\mathsf{KnapOrient}$ instance. Clearly σ' satisfies both the length and knapsack constraints. It remains to bound the reward we obtain.

$$\begin{aligned} \widehat{r}(\sigma') \geq \frac{\widehat{r}(\sigma)}{c} \quad &\geq \quad \frac{\lambda^* yW + \lambda^* W/\alpha}{c} \quad = \quad \lambda^* W \cdot \left(\frac{y + 1/\alpha}{c}\right) \\ &\geq \lambda^* W \cdot \min\left\{y + \frac{1}{\alpha}, \frac{1}{2} + \frac{1}{2\alpha y}\right\} \quad \geq \quad \frac{\lambda^* W}{\alpha} \end{aligned}$$

The second inequality is by $\widehat{r}(\sigma) - \lambda^* \cdot \widehat{s}(\sigma) = \mathsf{Alg}_o(\lambda^*) \geq \frac{\lambda^* W}{\alpha}$ due to the choice (3.1), which implies that $\widehat{r}(\sigma) \geq \lambda^* \cdot \widehat{s}(\sigma) + \frac{\lambda^* \cdot W}{\alpha} = \lambda^* yW + \frac{\lambda^* W}{\alpha}$ by the definition of y. The third inequality is by $c \leq \max\{1, 2y\}$. The last inequality uses $\alpha \geq 2$. It follows that $\widehat{r}(\sigma') \geq \frac{\mathsf{Opt}}{2\alpha}$, giving us the desired approximation ratio. ■

As an aside, this Lagrangian approach can be used to obtain a constant-factor approximation algorithm for a two-budget version of stochastic orienteering (with separate bounds on travel and processing times). But it is unclear if this can be extended to the single-budget version. In particular, we are not able to show that the Lagrangian relaxation (of processing times) has objective value $\Omega(\mathsf{Opt})$. This is because different decision paths in the Opt tree might vary a lot in their processing times, implying that there is no reasonable candidate for a Lagrange multiplier.

In the next subsection we discuss some simple reductions from $\mathsf{StocOrient}$ to deterministic orienteering that fail to achieve a good approximation ratio. This serves as a warm up for our algorithm which reduces $\mathsf{StocOrient}$ to $\mathsf{KnapOrient}$; we outline this in Subsection 3.2.

3.1. A Strawman Approach: Reduction to Deterministic Orienteering

A natural approach for $\mathsf{StocOrient}$ is to replace stochastic jobs by deterministic ones with size equal to the expected size $E[S_v]$, and find a near-optimal orienteering solution P to the deterministic instance which gets reward R. One can then use this path P to get a non-adaptive policy for the original $\mathsf{StocOrient}$ instance with expected reward $\Omega(R)$. Indeed, suppose the path P spends time L traveling and W processing the deterministic jobs such that $L + W \leq B$. Then, picking a random half of the jobs and visiting them results in a non-adaptive solution for $\mathsf{StocOrient}$ which travels at most L and processes jobs for time at most $W/2$ in expectation. Hence, Markov's inequality says that with probability at least $1/2$, all jobs finish processing within W time units and we get the entire reward of this sub-path, which is $\Omega(R)$.

However, the problem is in showing that $R = \Omega(\mathsf{Opt})$—i.e., that the deterministic instance has a solution with reward that is comparable to the $\mathsf{StocOrient}$ optimum.

The above simplistic reduction of replacing random jobs by deterministic ones with mean size fails even for stochastic knapsack: suppose the knapsack budget is B, and each of the n jobs has size Bn with probability $1/n$, and size 0 otherwise. Note that the expected size of every job is now B. Therefore, a deterministic solution can pick only one job, whereas the optimal solution would finish $\Omega(n)$ jobs with high probability. However, observe that this problem disappears if we truncate all sizes at the budget, i.e., set the deterministic size to be the expected "truncated" size $\mathbb{E}[\min(S_j, B)]$ where S_j is the random size of job j. We also have to set the reward to be $r_j \Pr[S_j \le B]$ to discount the reward from impossible size realizations. Now $\mathbb{E}[\min(W_j, B)]$ reduces to B/n and so the deterministic instance can now get $\Omega(n)$ reward. Indeed, this is the approach used by [18] to get an $O(1)$-approximation algorithm and adaptivity gap.

But for StocOrient, is there a good truncation threshold?

Considering $\mathbb{E}[\min(S_j, B)]$ fails on the example where all jobs are co-located at a point at distance $B-1$ from the root. Each job v has size B with probability $1/B$, and 0 otherwise. Truncation by B gives an expected size $\mathbb{E}_{S_v \sim \pi_v}[\min(S_v, B)] = 1$ for every job, and so the deterministic instance gets reward from only one job, while the StocOrient optimum can collect $\Omega(B)$ jobs. Now noticing that any algorithm *has* to spend $B-1$ time traveling to reach any vertex that has some job, we can instead truncate each job j's size at $B - d(\rho, j)$, which is the maximum amount of time we can possibly spend at j (since we must reach vertex j from ρ). However, while this fix works for the aforementioned example, the following example shows that such a deterministic instance might only get an $O(\frac{\log\log B}{\log B})$ fraction of the optimal stochastic reward.

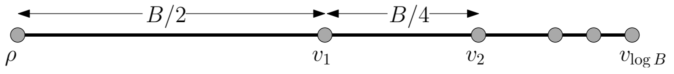

FIGURE 3.1. Bad example for replacing by expectations.

Consider $n = \log B$ jobs on a line as in Figure 3.1. For $i = 1, 2, \ldots, \log B$, the i^{th} job is at distance $B(1 - 1/2^i)$ from the root ρ; job i takes on size $B/2^i$ with probability $p := 1/\log B$ and size 0 otherwise. Each job has unit reward. The optimal (adaptive and non-adaptive) solution to this instance is to try all the jobs in order $1, 2, \ldots, \log B$: with probability $(1-p)^{\log B} \approx 1/e$, all the jobs instantiate to size 0 and we will accrue reward $\Omega(\log B)$.

In the deterministic orienteering instance, each job i has its expected truncated size $\mu_i = \mathbb{E}[\min\{S_i, B - d(\rho, i)\}] = B/(2^i \log B)$. A feasible solution consists of a subset of jobs where the total travel plus expected sizes is at most B. Suppose j is the first job we pick along the line. Then, because of its size being μ_j we cannot reach any jobs in the last μ_j length of the path. The number of these lost jobs is $\log \mu_j = \log B - j - \log\log B$ because of the geometrically decreasing gaps between jobs. Hence we can reach only jobs $j, j+1, j + \log\log B - 1$, giving us a maximum profit of $\log\log B$ even if we ignore the space these jobs would take. (Since their sizes decrease geometrically, we can indeed get all but a constant number of these jobs.)

This shows that replacing jobs in a StocOrient instance by their expected truncated sizes gives a deterministic instance whose optimal reward is smaller by an $\Omega(\frac{\log B}{\log\log B})$ factor.

3.2. Our Approach: Reduction to Knapsack Orienteering

The reason why the deterministic techniques described above worked for stochastic knapsack, but failed for stochastic orienteering is the following: the total sizes of jobs is always roughly B in knapsack (so truncating at B was the right thing to do). But in orienteering, it depends on the total time spent traveling, *which in itself is a random quantity, even for a non-adaptive solution.* One way around this is to guess the amount of time W spent processing jobs (up to a factor of 2) which gets the largest profit, and use that as the truncation threshold, to define a knapsack orienteering instance. It seems that such an approach should lose an $\Omega(\log B)$ fraction of the optimal reward, since there are $\log_2 B$ choices for the truncation parameter W. Somewhat surprisingly, we show that this algorithm actually gives a much better reward: it achieves a constant factor approximation relative to a non-adaptive optimum, and an $O(\log\log B)$-approximation when compared to the adaptive optimum!

Given an instance $\mathcal{I}_{so}$ of StocOrient with optimal (non-adaptive or adaptive) solution having expected reward Opt, our algorithm is outlined in Figure 3.2. However, there are many details to be addressed, and we flesh out the details of this algorithm over the next two sections. We will prove that $\alpha = O(1)$ for non-adaptive StocOrient, and $\alpha = O(\log\log B)$ in the adaptive case.

Step 1: Enumerate over all choices for the truncation threshold W. Construct a suitable instance $\mathcal{I}_{ko}(W)$ of *Knapsack Orienteering* (KnapOrient), with the guarantee that the optimal reward from this KnapOrient instance $\mathcal{I}_{ko}(W)$ is at least Opt/α.
Step 2: Use Theorem 5 on $\mathcal{I}_{ko}$ to find a path P with reward $\Omega(\mathsf{Opt}/\alpha)$.
Step 3: Convert this KnapOrient solution P into a non-adaptive policy for StocOrient (Lemma 1).

FIGURE 3.2. High Level Overview

4. Non-Adaptive Stochastic Orienteering

Here we consider the *non-adaptive* StocOrient problem, and present an $O(1)$-approximation algorithm (Theorem 2). This also contains many ideas used in the more involved analysis of the adaptive setting.

Recall that the input consists of metric (V,d) with each vertex $v \in V$ representing a stochastic job having a deterministic reward $r_v \in \mathbb{Z}^+$ and a random processing time/size S_v distributed according to $\pi_v : \mathbb{R}^+ \to [0,1]$; we are also given a root ρ and budget B. A non-adaptive policy is an ordering σ of the vertices (starting with ρ), which corresponds to visiting vertices (and processing the respective jobs) in the order σ. The goal in the non-adaptive StocOrient problem is to compute an ordering that maximizes the expected reward, i.e., the total reward of all items which are completed within the budget of B (travel + processing times). We first perform some preprocessing on the input instance. Throughout, Opt will denote the optimal non-adaptive solution to the given StocOrient instance, as well as its expected reward.

ASSUMPTION 1. *We may assume that:*
- *No single-vertex solution has expected reward more than* $\mathsf{Opt}/8$.
- *For each vertex* $u \in V$, $\Pr_{S_u \sim \pi_u}[\, S_u > B - d(\rho,u)\,] \leq 1/2$.

The resulting optimal value remains at least $\frac{3}{4} \cdot \mathsf{Opt}$.

Proof.(1) Note that we can enumerate over all single vertex solutions (there are only n of them) and output the best one– if any such solution has value greater than $\mathsf{Opt}/8$ then we already have an 8-approximate solution. So the first assumption follows.

(2) For the second assumption, call a vertex u *bad* if $\Pr_{S_u \sim \pi_u}[\, S_u > B - d(\rho,u)\,] > 1/2$. Notice that if Opt visits a bad vertex then the probability that it continues further decreases geometrically by a factor $1/2$, because the total budget is exceeded with probability at least $1/2$. Therefore, the total expected reward that Opt collects from all bad jobs is at most twice the maximum expected reward from any single bad vertex. By the first assumption, the maximum expected reward from any single vertex is at most $\mathsf{Opt}/8$. So the expected reward obtained by ignoring bad vertices is at least $\frac{3}{4} \cdot \mathsf{Opt}$. ■

DEFINITION 1 (TRUNCATED MEANS). For any vertex $u \in V$ and any positive value $Z \geq 0$, let $\mu_u(Z) := \mathbb{E}_{S_u \sim \pi_u}[\min(S_u, Z)]$ denote the expected size truncated at Z. Note that for all $Z_2 \geq Z_1 \geq 0$, $\mu_u(Z_1) \leq \mu_u(Z_2)$ and $\mu_u(Z_1 + Z_2) \leq \mu_u(Z_1) + \mu_u(Z_2)$.

DEFINITION 2 (VALID KnapOrient INSTANCES). Given an instance $\mathcal{I}_{so}$ of StocOrient and value $W \leq B$, define KnapOrient instance $\mathcal{I}_{ko}(W) := \mathsf{KnapOrient}(V, d, \{(\widehat{s}_u, r_u) : \forall u \in V\}, L, W, \rho)$ where:
(i) The travel budget $L = B - W$ and size budget is W.
(ii) For all $u \in V$, its deterministic size $\widehat{s}_u = \mu_u(W)$.

Recall that AlgKO is an $O(1)$-approximation algorithm for KnapOrient. Algorithm 1 for non-adaptive StocOrient proceeds in the following manner: (i) it enumerates over all possible powers-of-two for the choice of size budget W (see the definition of valid KnapOrient instances), (ii) uses AlgKO to find a near-optimal solution for each of the valid KnapOrient instances, and finally (iii) converts the best of them into a non-adaptive StocOrient solution. The final part of this procedure is characterized by the following Lemma 1. The proof is similar to that used in earlier works on stochastic knapsack [18].

LEMMA 1. *Given any solution P to* KnapOrient *instance $\mathcal{I}_{ko}(W)$ for any $W \leq B$, having reward R, we can obtain in polynomial time a non-adaptive policy for* StocOrient *of expected reward $R/12$.*

Proof. To reduce notation, let P also denote the set of vertices visited in the solution to $\mathcal{I}_{ko}(W)$.

$$L := \left\{u \in P : \mu_u(W) > \frac{W}{4}\right\} \quad \text{and} \quad S := \left\{u \in P : \mu_u(W) \le \frac{W}{4}\right\}$$

Notice that $|L| < 4$ since $\sum_{u\in P} \mu_u(W) \le W$ by the size budget in $\mathcal{I}_{ko}(W)$. By averaging $\max\{r_u : u \in L\} \ge r(L)/3$. Moreover, by Assumption 1 the best single vertex solution (to StocOrient) among L has expected reward at least $\frac{1}{2} \cdot \max\{r_u : u \in L\} \ge r(L)/6$.

Since each $v \in S$ has $\mu_v(W) \le W/4$ and $\sum_{u\in S} \mu_u(W) \le W$, we can partition S into 3 parts such that each part has total size at most $W/2$. Again by averaging, one of these parts $S' \subseteq S$ satisfies $\sum_{u\in S'} \mu_u(W) \le W/2$ and $r(S') \ge r(S)/3$. Consider the following non-adaptive policy for StocOrient: visit (and process) vertices in S' in the order of P. By triangle inequality, the travel time is at most that of P, namely $B - W$. By Markov's inequality, with probability at least $1/2$, the total processing time of S' is at most W. Hence the expected reward of this policy to StocOrient is at least $\frac{1}{2} \cdot r(S') \ge r(S)/6$.

The better of the two policies above (from L and S) has reward at least $R/12$. ■

Algorithm 1 Algorithm AlgSO for StocOrient on input $\mathcal{I}_{so} = (V, d, \{(\pi_u, r_u) : \forall u \in V\}, B, \rho)$

```
1:  for all $v \in V$ do
2:  let $R_v := r_v \cdot \Pr_{S_v\sim\pi_v}[S_v \le (B - d(\rho, v))]$ be the expected reward of the single-vertex solution to $v$.
3:  end for
4:  with probability $1/2$, just visit the vertex $v$ with the highest $R_v$ and exit.
5:  delete all vertices $u \in V$ with $\Pr_{S_u\sim\pi_u}[S_u > B - d(\rho, u)] > 1/2$.
6:  for $i = 0, 1, \ldots, \lceil \log B \rceil$ do
7:  set $W = B/2^i$
8:  let $P_i$ be the path returned by AlgKO on the valid KnapOrient instance $\mathcal{I}_{ko}(W)$.
9:  let $R_i$ be the reward of this KnapOrient solution $P_i$.
10: end for
11: let $P_{i^*}$ be the solution among $\{P_i\}_{i\in[\log B]}$ with maximum reward $R_i$.
12: output the non-adaptive StocOrient policy corresponding to $P_{i^*}$, using Lemma 1.
```

Therefore, in order to prove a constant approximation ratio, it suffices to show the existence of some $W = B/2^i$ for which the optimal value of $\mathcal{I}_{ko}(W)$ is $\Omega(\mathsf{Opt})$. Formally,

LEMMA 2. *Given any instance $\mathcal{I}_{so}$ of non-adaptive* StocOrient *satisfying Assumption 1, there exists $W = B/2^i$ for some $i \in \{0, 1, \ldots, \lceil \log B \rceil\}$ such that $\mathcal{I}_{ko}(W)$ has optimal value at least* $\mathsf{Opt}/50$.

The rest of this section proves this result. We restrict attention to vertices satisfying the condition in Assumption 1; let $\mathsf{Opt}' \ge \frac{3}{4} \cdot \mathsf{Opt}$ denote the resulting optimal value.

Without loss of generality, let the optimal non-adaptive ordering be $\{\rho = v_0, v_1, v_2, \ldots, v_n\}$. For any $v_j \in V$ let $D_j = \sum_{i=1}^{j} d(v_{i-1}, v_i)$ denote the *total distance* spent before visiting vertex v_j. Note that while the total time (travel plus processing) spent before visiting any vertex is a random quantity, the distance (i.e. travel time) is deterministic, since we deal with non-adaptive policies. Let j^* be the first index j such that

$$\sum_{i<j} \mu_{v_i}(B - D_j) \quad \ge \quad K \cdot (B - D_j) \tag{4.2}$$

Here K is some constant which we will fix later. Observe that this condition is trivially satisfied when $D_j = B$; so we may assume, without loss of generality, that $D_{j^*-1} \le B - 1$.

LEMMA 3. *For index j^* as in (4.2), we have $\sum_{i\le j^*-1} r_{v_i} \ \ge\ \mathsf{Opt}'/2$.*

Proof. We first deal with the corner case that $D_{j^*}=B$. In this case, v_{j^*} is the last possible vertex visited by the optimal solution. By Assumption 1, the expected reward from vertex v_{j^*} even if it is visited directly from the root, is at most $\mathsf{Opt}/8$. So the expected reward from the first j^*-1 vertices is at least $\mathsf{Opt}'-\frac{1}{8}\cdot\mathsf{Opt}\geq \mathsf{Opt}/2$, which implies the lemma. In the following, we assume that $D_{j^*}\leq B-1$.

CLAIM 1. *The optimal solution visits a vertex indexed j^* or higher with probability* $\leq e^{1-\frac{K}{2}-\frac{1}{2K}}$.

Proof. If the optimal solution visits vertex v_{j^*} then we have $\sum_{i<j^*} S_{v_i}\leq B-D_{j^*}$. This also implies that $\sum_{i<j^*}\min(S_{v_i},B-D_{j^*})\leq B-D_{j^*}$. Now, for each $i<j^*$ let us define a random variable $X_i:=\frac{\min(S_{v_i},B-D_{j^*})}{B-D_{j^*}}$. Note that the X_i's are independent $[0,1]$ random variables, and that $\mathbb{E}[X_i]=\mu_{v_i}(B-D_{j^*})/(B-D_{j^*})$. From this definition, it is also clear that the probability that the optimal solution visits v_{j^*} is upper bounded by the probability that $\sum_{i<j^*}X_i\leq 1$. To this end, we have from Inequality (4.2) that $\sum_{i<j^*}\mathbb{E}[X_i]\geq K$. Therefore we can apply a standard Chernoff bound to conclude that

$$\Pr\left[\text{Optimal solution visits vertex } v_{j^*}\right] \quad\leq\quad \Pr\left[\sum_{i<j^*}X_i\leq 1\right] \quad\leq\quad e^{1-\frac{K}{2}-\frac{1}{2K}}$$

This completes the proof. ■

CLAIM 2. *Conditional on reaching v_{j^*}, the expected reward obtained by the optimal policy from vertices $\{v_{j^*},v_{j^*+1},\ldots\}$ is at most* Opt'.

Proof. Consider the alternate policy $\{\rho=v_0,v_{j^*},v_{j^*+1},\ldots,v_n\}$ that skips all vertices before v_{j^*}. By triangle inequality, the distance $d(\rho,v_{j^*})\leq D_{j^*}$. So the expected reward from this policy is at least the conditional reward of the optimal policy obtained beyond vertex v_{j^*}. The claim now follows by optimality. ■

Combining these two claims and setting $K=3.5$, the expected reward from the first j^*-1 vertices is at least $\mathsf{Opt}'/2$, which implies the lemma. ■

Recall that $D_{j^*-1}\leq B-1$; let $\ell\in\mathbb{Z}_+$ be such that $B/2^\ell<B-D_{j^*-1}\leq B/2^{\ell-1}$. Set $W^*=B/2^\ell$. We will show that the $\mathsf{KnapOrient}$ instance $\mathcal{I}_{ko}(W^*)$ has optimal value at least $\mathsf{Opt}'/(8K+8)$. Consider path $P^*=\langle\rho=v_0,v_1,\ldots,v_{j^*-1}\rangle$. The reward on this path is at least $\mathsf{Opt}'/2$ and it satisfies the travel budget $B-W^*$ in $\mathcal{I}_{ko}(W^*)$. The total size on this path is:

$$\begin{aligned}\sum_{i\leq j^*-1}\mu_{v_i}(W^*)\leq \sum_{i\leq j^*-1}\mu_{v_i}(B-D_{j^*-1}) \quad&=\quad \sum_{i<j^*-1}\mu_{v_i}(B-D_{j^*-1})+\mu_{v_{j^*-1}}(B-D_{j^*-1})\\ \leq (K+1)(B-D_{j^*-1}) \quad&\leq\quad 2(K+1)W^*\end{aligned}$$

The second inequality is by choice of j^* in equation (4.2). Although P^* may not satisfy the size budget of W^*, we obtain a subset $P'\subseteq P^*$ that does. Since each vertex has size at most W^* and the total size of P^* is at most $2(K+1)W^*$, there is a partition of P^* into at most $4(K+1)$ parts such that each part has size at most W^*. (Such a partition can be obtained greedily: starting with the trivial partition with each vertex of P^* in a single part, repeatedly merge any two parts that have combined size at most W^*. In the final partition, every pair of parts has combined size more than W^*; since the total size of P^* is at most $2(K+1)W^*$, the final number of parts is at most $4K+4$.) Choosing the maximum reward part amongst these yields a *feasible* solution to $\mathcal{I}_{ko}(W^*)$ of value at least $\frac{\mathsf{Opt}'}{8(K+1)}\geq\frac{\mathsf{Opt}}{50}$, setting $K=3.5$. This completes the proof of Lemma 2.

Combining Lemmas 1 and 2, we obtain Theorem 2. The approximation ratio obtained by this approach (after optimizing parameters) is around 500. We chose not to present the calculations here, so as to focus only on the main ideas. We note however that obtaining a significantly smaller constant factor seems to require additional techniques.

5. Adaptive Stochastic Orienteering

In this section we consider the adaptive $\mathsf{StocOrient}$ problem. We will show the same algorithm (Algorithm AlgSO) is an $O(\log\lceil\log B\rceil)$-approximation algorithm to the best adaptive solution, thus proving Theorem 1. Note that this also establishes an adaptivity gap of $O(\log\log B)$.

Assumption 1 holds in this adaptive setting as well; the proof is almost identical and not repeated here. This decreases the optimal value by a constant factor: we refer to the resulting optimal adaptive policy (and its expected reward) by Opt.

Recall the definition of valid $\mathsf{KnapOrient}$ instances and Lemma 1. The main result that we need is an analog of Lemma 2, namely:

LEMMA 4. *Given any instance $\mathcal{I}_{so}$ of adaptive* $\mathsf{StocOrient}$ *satisfying Assumption 1, there exists $W = B/2^i$ for some $i \in \{0, 1, \ldots, \lceil\log B\rceil\}$ such that $\mathcal{I}_{ko}(W)$ has optimal value $\Omega(\mathsf{Opt}/\log\log B)$.*

Before we begin, recall the typical instance $\mathcal{I}_{so} := \mathsf{StocOrient}(V, d, \{(\pi_u, r_u) : \forall u \in V\}, B, \rho)$ of the stochastic orienteering problem.

Roadmap. We begin by giving a roadmap of the proof. Let us view the optimal adaptive policy Opt as a decision tree where each node is labeled with a vertex/job, and the children correspond to different size instantiations of the job. For any sample path P in this decision tree, consider the first node x_P where the sum of expected sizes of the jobs processed until x_P exceeds the "budget remaining" by some small factor—here, if $L_{x,P}$ is the total distance traveled from the root ρ to this node x_P by visiting vertices along P, then the remaining budget is $B - L_{x,P}$. Call such a node a *frontier node*, and the *frontier* is the union of all such frontier nodes. To make sense of this definition, note that if the orienteering instance was non-stochastic (and all the sizes were equal to their expectations), then we would not get any reward from portions of the decision tree on or below the frontier nodes. Unfortunately, since job sizes are random for us, this is not necessarily the case. The main idea in the proof is to show that we do not lose too much reward by truncation: i.e., even if we truncate Opt along this frontier, we still obtain an expected reward of $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$ from the truncated tree. Then, an averaging argument can be used to show the existence of some path P^* of length L where (i) the total rewards of jobs is $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$, and (ii) the sum of expected sizes of the jobs is $O(B - L)$. This gives us the candidate $\mathsf{KnapOrient}$ solution.

Viewing Opt **as a Discrete Time Stochastic Process.** Note that the transitions of the decision tree Opt represent travel between vertices: if the parent node is labeled with vertex u, and its child is labeled with v, the transition takes $d(u, v)$ time. To simplify notation, we take every such transition, and subdivide it into $d(u, v)$ unit length transitions. The intermediate nodes added in are labeled with new dummy vertices, with dummy jobs of deterministic size 0 and reward 0. We denote this tree as Opt'. Note that the amount of time spent traveling to any node is exactly the number of edges from the root to this node. Now, if we start a particle at the root, and let it evolve down the tree based on the random outcomes of job sizes, then the node reached at timestep t corresponds to some job with a random size and reward. This naturally gives us a discrete-time stochastic process $\mathcal{T}$, which at *every* timestep picks a job of size $\mathbf{S}_t \sim \mathcal{D}_t$ and reward $\mathbf{R}_t$. Note that $\mathbf{S}_t, \mathbf{R}_t$ and the probability distribution $\mathcal{D}_t$ all are random variables that depend on the outcomes of the previous timesteps $0, 1, \ldots, t-1$ (since the actual job that the particle sees depends on past outcomes). We stop the process $\mathcal{T}$ at the first (random) timestep t_{end} such that $\sum_{t=0}^{t_{end}} \mathbf{S}_t \geq (B - t_{end})$—this is the natural point to stop, since it is precisely the time step when the total processing plus the total distance traveled exceeds the budget B.

Some notation: *Nodes* will correspond to states of the decision tree Opt', whereas vertices are points in the metric (V, d). The *level* of a node x in Opt' is the number of hops in the decision tree from the root to reach x—this is the timestep when the stochastic process would reach x, or equivalently the travel time to reach the corresponding vertex in the metric. We denote this by $\mathsf{level}(x)$. Let $\mathsf{label}(x)$ be the vertex labeling x. We abuse notation and use S_x, r_x, π_x and $\mu_x(\cdot)$ to denote the size, reward, size distribution and truncated mean for node x—hence $S_x = S_{\mathsf{label}(x)}$, $r_x = r_{\mathsf{label}(x)}$, $\pi_x = \pi_{\mathsf{label}(x)}$ and $\mu_x(\cdot) = \mu_{\mathsf{label}(x)}(\cdot)$. We use $x' \preceq x$ to denote that x' is an ancestor of x.

Now to begin the proof of Lemma 4. We assume that there are no co-located stochastic jobs, i.e., there is only one job at every vertex. Note that this also implies that we have to travel for a non-zero integral distance between jobs. This is only to simplify the exposition of the proof: we explain how to discharge this assumption at the end of this section.

Defining the Frontiers. Henceforth, we will focus on the decision tree Opt' and the induced stochastic process $\mathcal{T}$. Consider any intermediate node x and the sample path from the root to x in Opt'. We call x a *star node* if x is the first node along this sample path for which the following condition is satisfied:

$$\textstyle\sum_{x' \prec x} \mu_{x'}\left(B - \mathsf{level}(x)\right) \quad \geq \quad 8K\left(B - \mathsf{level}(x)\right) \tag{5.3}$$

Above, K is a parameter that will later be set to $\Theta(\log\log B)$. Observe that this condition obviously holds when $\mathsf{level}(x) = B$, and that no star node is an ancestor of another star node. To get a sense of this definition of star nodes, ignore the truncation for a moment: then x is a star node if the expected sizes of all the $\mathsf{level}(x)$ jobs on the sample path until x sum to at least $8K(B - \mathsf{level}(x))$. But since we have spent $\mathsf{level}(x)$ time traveling to reach x, the process only continues beyond vertex x if the actual sizes of the jobs is at most $B - \mathsf{level}(x)$; i.e., if the sizes of the jobs are a factor $8K$ smaller than their expectations. If this were an unlikely event, then pruning Opt' at the star nodes would result in little loss of reward. And that is precisely what we show.

Let Opt'' denote the subtree of Opt' obtained by pruning it at star nodes. Opt'' does not include rewards at star nodes. Note that leaf-nodes in Opt'' are either leaves of Opt' or *parents* of star nodes. In particular, $\mathsf{level}(s) \leq B-1$ for each leaf-node $s \in \mathsf{Opt}''$. We will show that:

LEMMA 5. *The expected reward in* Opt'' *is at least* $\mathsf{Opt}/2$.

Remark: The difference from the analysis of the non-adaptive case is that we set parameter $K = O(\log\log B)$ instead of a constant in the definition of the truncated tree Opt'' (5.3). The main reason for the larger factor is the difficulty in directly analyzing the truncated decision tree when the threshold $B - \mathsf{level}(x)$ is changing. Instead, we prove Lemma 5 by grouping star nodes into $\log B$ "bands" according to geometrically decreasing threshold values, and analyze each band separately as a martingale process. For a single band we then use a concentration inequality to upper bound the loss by factor that is exponentially small in K. Finally, adding the loss over the $\log B$ bands yields Lemma 5. The details now follow.

Before proving Lemma 5, we show how this implies Lemma 4.

Proof of Lemma 4: We start with the following claim that uses the definition of star nodes.

CLAIM 3. *Every leaf node* $s \in \mathsf{Opt}''$ *satisfies* $\sum_{x \preceq s} \mu_x\left(B - \mathsf{level}(s)\right) \leq 9K\left(B - \mathsf{level}(s)\right)$.

Proof. By definition of Opt'', leaf-node s is not a star node (nor a descendant of one). So:

$$\sum_{x \preceq s} \mu_x\left(B - \mathsf{level}(s)\right) = \sum_{x \prec s} \mu_x\left(B - \mathsf{level}(s)\right) + \mu_s\left(B - \mathsf{level}(s)\right) \quad < \quad (8K+1)\cdot\left(B - \mathsf{level}(s)\right).$$

The inequality is by (5.3). This proves the claim. ■

For each root-leaf path P in Opt'' let $\Pr[P]$ denote the probability that this path is traced, and let $r(P)$ be the sum of rewards on P. Then, Lemma 5 implies $\sum_P \Pr[P]\cdot r(P) \geq \mathsf{Opt}/2$. So there exists a sample path P^* in Opt'' to some leaf node s^* with total reward at least $\mathsf{Opt}/2$. Moreover, Claim 3 implies that the sum of means (truncated at $B - \mathsf{level}(s^*)$) of jobs in P^* is at most $9K\left(B - \mathsf{level}(s^*)\right)$.

Recall that every leaf in Opt'' has level at most $B-1$, so $\mathsf{level}(s^*) \leq B-1$. Choose $\ell \in \{0, 1, \ldots, \lceil \log B \rceil\}$ so that $B/2^\ell \leq B - \mathsf{level}(s^*) \leq 2B/2^\ell$, and set $W^* = B/2^\ell$. Then we have:

$$\sum_{x \preceq s^*} \mu_x\left(W^*\right) \quad \leq \quad \sum_{x \preceq s^*} \mu_x\left(B - \mathsf{level}(s^*)\right) \quad \leq \quad 9K\cdot\left(B - \mathsf{level}(s^*)\right) \quad \leq \quad 18K \cdot W^*$$

Consider the $\mathsf{KnapOrient}$ instance $\mathcal{I}_{ko}(W^*)$; we will show that it has optimal value at least $\Omega(\mathsf{Opt}/K)$, which would prove Lemma 4. Note that path P^* has length $\mathsf{level}(s^*) \le B - W^*$. The above calculation shows that the total size of P^* is at most $18K \cdot W^*$. Using the bin-packing-type argument as in the previous section, we obtain a subset $P' \subseteq P^*$ that has total size at most W^* and reward at least $r(P^*)/(36K) \ge \frac{\mathsf{Opt}}{72K}$. Thus we obtain Lemma 4. ■

We now prove Lemma 5. Group the star nodes into $\lceil \log B \rceil + 1$ *bands* based on the value of $B - \mathsf{level}(x)$. Star node x is in band i if $B - \mathsf{level}(x) \in (B/2^{i+1}, B/2^i]$ for $0 \le i \le \lceil \log B \rceil$, and in band $\lceil \log B \rceil + 1$ if $\mathsf{level}(x) = B$.

First consider star nodes of band $\lceil \log B \rceil + 1$. Note that the policy terminates after these nodes (since B time units have already been spent traveling). By Assumption 1, the loss in reward by ignoring star nodes of band $\lceil \log B \rceil + 1$ is at most $\mathsf{Opt}/8$.

Next we consider bands $\{0, \ldots, \lceil \log B \rceil\}$. We use the following key lemma that upper bounds the probability of reaching star nodes in any particular band i.

LEMMA 6. *For any $i \in \{0, \ldots, \lceil \log B \rceil\}$, the probability of reaching band i is at most $\frac{1}{10\lceil \log B \rceil}$.*

Taking a union bound, the probability of reaching some band $\{0, \ldots, \lceil \log B \rceil\}$ is at most $\frac{1}{10}$. Then we have the following claim (similar to Claim 2 in the non-adaptive case).

CLAIM 4. *Conditional on reaching any node $x \in \mathsf{Opt}'$, the expected reward obtained by the optimal policy from nodes below x is at most* Opt.

Proof. Consider the alternate adaptive policy that visits node x directly from the root. Using triangle inequality, the expected reward from this policy is at least the conditional reward of Opt' obtained below vertex x. The claim now follows by optimality. ■
Thus we obtain that the loss in reward by truncating at star nodes in bands $\{0, \ldots, \lceil \log B \rceil\}$ is at most $\mathsf{Opt}/10$. Combined with the loss due to band $\lceil \log B \rceil + 1$, it follows that Opt'' has reward at least $\mathsf{Opt}/2$.

It only remains to prove Lemma 6, which we do in the rest of this section.
Proof of Lemma 6: Fix any i. In order to bound the probability of reaching band-i, consider the following altered stochastic process $\mathcal{T}_i$: follow $\mathcal{T}$ as long as it could lead to a star node in band i. If we reach a node y such that there is no band-i star node as a descendant of y, then we stop the process $\mathcal{T}_i$ at y. Else we stop when we reach a star node in band i. An illustration of the optimal decision tree, the different bands and altered processes is given in Figure 5.3. By a straightforward coupling argument, the probabilities of reaching a band-i star node in $\mathcal{T}$ and in $\mathcal{T}_i$ are identical, and hence it suffices to bound the probability of continuing beyond a band-i star node in $\mathcal{T}_i$.

CLAIM 5. *For each $i \in \{0, 1, \ldots, \lceil \log B \rceil\}$, and any star node x in band i,*

$$2K\frac{B}{2^i} \quad \le \quad \sum_{x' \prec x} \mu_{x'}\left(B/2^{i+1}\right) \quad \le \quad 17K\frac{B}{2^i}$$

Proof. By definition of a star node (5.3), and since node x is in band-i, $B/2^{i+1} \le B - \mathsf{level}(x) \le B/2^i$,

$$\begin{aligned}\sum_{x' \prec x} \mu_{x'}\left(B/2^{i+1}\right) \ge \sum_{x' \prec x} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x))\right) \quad &\ge \quad \frac{1}{2}\sum_{x' \prec x} \mu_{x'}\left(B - \mathsf{level}(x)\right) \\ \ge 4K(B - \mathsf{level}(x)) \quad &\ge \quad 2K\frac{B}{2^i}.\end{aligned}$$

The first two inequalities used the monotonicity and subadditivity of $\mu_{x'}(\cdot)$.

Moreover, since y, the parent node of x, is not a star node, it satisfies

$$\sum_{x' \prec y} \mu_{x'}\left(B - \mathsf{level}(y)\right) \quad < \quad 8K(B - \mathsf{level}(y)) \quad = \quad 8K(B - \mathsf{level}(x) + 1).$$

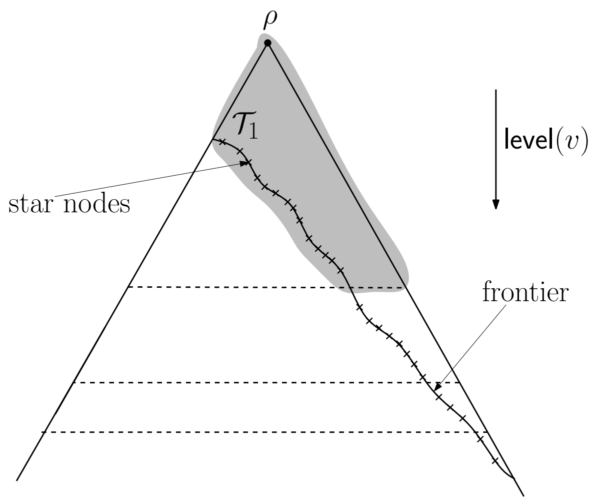

FIGURE 5.3. Optimal decision tree example: dashed lines indicate the bands, $\times$ indicates star nodes.

But since we are not considering band number $\lceil \log B \rceil + 1$ and all distances are at least 1, $\mathsf{level}(x) \le B-1$, and hence $B - \mathsf{level}(x) + 1 \le 2(B - \mathsf{level}(x)) \le 2B/2^i$. Thus we have $\sum_{x' \prec y} \mu_{x'}\,(B - \mathsf{level}(y)) < 16K \cdot B/2^i$. Now,

$$\begin{aligned}\sum_{x' \prec x} \mu_{x'}\left(B/2^{i+1}\right) = \sum_{x' \prec y} \mu_{x'}\left(B/2^{i+1}\right) + \mu_y\left(B/2^{i+1}\right) \quad &\le \quad \sum_{x' \prec y} \mu_{x'}\,(B - \mathsf{level}(y)) + \frac{B}{2^{i+1}} \\ \le 16K \cdot \frac{B}{2^i} + \frac{B}{2^{i+1}} \quad &\le \quad 17K \cdot \frac{B}{2^i}\end{aligned}$$

The first inequality uses $B - \mathsf{level}(y) \ge B - \mathsf{level}(x) \ge B/2^{i+1}$. This completes the proof. ■

CLAIM 6. *For any $i \in \{0, 1, \ldots, \lceil \log B \rceil\}$ and any star node x in band i, if process $\mathcal{T}_i$ reaches x then:*

$$\sum_{x' \prec x} \min\left(S_{x'}, \frac{B}{2^{i+1}}\right) \quad \le \quad \frac{B}{2^i}$$

Proof. Clearly, if process $\mathcal{T}_i$ reaches node x, it must mean that $\sum_{x' \prec x} S_{x'} \le (B - \mathsf{level}(x)) \le B/2^i$, else we would have run out of budget earlier. And, the truncation can only decrease the left hand side. ■

We now finish upper bounding the probability of reaching a star node in band i using a Martingale analysis. Define a sequence of random variables $\{Z_t,\ t = 0, 1, \ldots\}$ where

$$Z_t = \sum_{t'=0}^{t} \left(\min\left\{ \mathbf{S}_{t'}, \frac{B}{2^{i+1}} \right\} - \mu_{t'}\left(B/2^{i+1}\right) \right). \tag{5.4}$$

Above, $\mu_{t'}(\cdot)$ denotes the truncated mean (Definition 1) of random variable $\mathbf{S}_{t'}$. Since the subtracted term is precisely the expectation of the first term, the one-term expected change is zero and the sequence $\{Z_t\}$ forms a martingale. In turn, $\mathbb{E}[Z_\tau] = 0$ for any stopping time τ. We will define τ to be the time when the process $\mathcal{T}_i$ ends—recall that this is the first time when (a) either the process reaches a band-i star node, or (b) there is no way to get to a band-i star node in the future.

Claim 6 says that when $\mathcal{T}_i$ reaches any star node x, the sum over the first terms in (5.4) is at most $B/2^i$, whereas Claim 5 says the sums of the means is at least $2K\frac{B}{2^i}$. Because $K \ge 1$, we can infer that the $Z_t \le -K(B/2^i)$ for any star node (at level t). To bound the probability of reaching a star node in $\mathcal{T}_i$, we appeal to Freedman's concentration inequality for martingales.

THEOREM 6 **(Freedman [19] (Theorem 1.6)).** *Consider a real-valued martingale sequence $\{X_k\}_{k\geq 0}$ such that $X_0 = 0$, and $\mathbb{E}[X_{k+1} \mid X_k, X_{k-1}, \ldots, X_0] = 0$ for all k. Assume that the sequence is uniformly bounded, i.e., $|X_k| \leq M$ almost surely for all k. Now define the predictable quadratic variation process of the martingale to be*

$$W_k = \textstyle\sum_{j=0}^{k} \mathbb{E}\left[X_j^2 \mid X_{j-1}, X_{j-2}, \ldots X_0\right]$$

for all $k \geq 1$. Then for all $l \geq 0$ and $\sigma^2 > 0$, and any stopping time τ we have

$$\Pr\left[|\sum_{j=0}^{\tau} X_j| \geq l \text{ and } W_\tau \leq \sigma^2\right] \quad \leq \quad 2\exp\left(-\frac{l^2/2}{\sigma^2 + Ml/3}\right)$$

We apply the above theorem to the Martingale difference sequence $\{X_t = Z_t - Z_{t-1}\}$. Now, since each term X_t is just $\min(S_t, \frac{B}{2^{i+1}}) - \mu_t(B/2^{i+1})$, we get that $\mathbb{E}[X_t \mid X_{t-1}, \ldots] = 0$ by definition of $\mu_t(B/2^{i+1}) = \mathbb{E}\left[\min(S_t, \frac{B}{2^{i+1}})\right]$. Moreover, since the sizes and means are both truncated at $B/2^{i+1}$, we have $|X_t| \leq B/2^{i+1}$ with probability 1; hence we can set $M = B/2^{i+1}$. Finally in order to bound the variance term W_t we appeal to Claim 5. Indeed, consider a single random variable $X_t = \min(S_t, \frac{B}{2^{i+1}}) - \mu_t(B/2^{i+1})$, and abbreviate $\min(S_t, \frac{B}{2^{i+1}})$ by Y. Then:

$$\mathbb{E}\left[X_t^2 \mid X_{t-1}, \ldots\right] = \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right] \quad = \quad \mathbb{E}\left[Y^2\right] - \mathbb{E}[Y]^2 \leq Y_{max} \cdot \mathbb{E}[Y] \quad \leq \quad \frac{B}{2^{i+1}} \cdot \mu_t\left(B/2^{i+1}\right)$$

Here, the first inequality uses $Y \geq 0$ and Y_{max} as the maximum value of Y. The last inequality uses the definition of Y. Hence the term W_t is at most $(B/2^{i+1}) \sum_{t' \leq t} \mu_{t'}(B/2^{i+1})$ for the process at time t. Now, from Claim 5 we have that for any star node (say at level t) in band i, we have $\sum_{t' \leq t} \mu_{t'}(B/2^{i+1}) \leq 17K(B/2^i)$. Therefore we have $W_t \leq 9K \cdot (B/2^i)^2$ for star nodes, and we set σ^2 to be this quantity.

So by setting $\ell = K(B/2^i)$, $\sigma^2 = 9K(B/2^i)^2$, and $M = B/2^{i+1}$, we get that

$$\Pr[\text{ reaching star node in } \mathcal{T}_i] \quad \leq \quad \Pr\left[|Z_\tau| \geq K(B/2^i) \text{ and } W_\tau \leq 9K(B/2^i)^2\right] \quad \leq \quad 2\,e^{-K/20}.$$

Setting $K = \Omega(\log\lceil\log B\rceil)$ and performing a simple union bound calculation over the $\lceil\log B\rceil$ bands completes the proof of Lemma 6. ■

Handling Co-Located Jobs. To help with the presentation in the above analysis, we assumed that a node x which is at depth l in the decision tree for Opt is actually processed after the adaptive policy has traveled a distance of l. In particular, this meant that there is at most one stochastic job per node. However, if we define the truncations of any node (in equation (5.3)) by its *actual length along the path*, instead of simply its depth/level in the tree, then we can handle co-located jobs in exactly the same manner as above. In this situation, there could be several nodes in a sample path which have the same truncation threshold but it is not difficult to see that the rest of the analysis would proceed in an identical fashion. We do not present additional details here– the analysis in the next section handles this issue in a more general setting.

6. Stochastic Sequence Orienteering

In this section we consider a general *stochastic sequence orienteering* problem. The input is a *directed* metric (V, d) with integer distances[1] where each vertex $v \in V$ contains a job having reward r_v and a random processing time (or size) $S_v \sim \pi_v$. We are also given a specified sequence $\langle s_1, \ldots, s_k\rangle$ of *portal vertices* and a bound $B \in \mathbb{Z}_+$. A solution (policy) is an adaptive path originating from s_1 that visits vertices (and processes the respective jobs) such that the total time taken (travel plus processing) is at most B. An additional constraint here is that the path must visit all the portals $s_1, \ldots, s_k$, and in that order; the policy terminates after visiting vertex s_k. The objective is to maximize the expected reward. Since the portal vertices are always visited, we can assume without loss of generality, that they have zero rewards. One can view the

[1] The distance function $d : V \times V \to \mathbb{Z}_+$ satisfies the triangle inequality, but is not necessarily symmetric.

portals as essential jobs that any policy must complete (in the prescribed order), and the remaining vertices as optional, from which a policy seeks to maximize reward.

A modeling assumption we make is that, since any feasible policy *must* visit each of these portal vertices in that sequence, it must satisfy the following property. If the policy is running the job at some vertex v after having visited portals $\langle s_1, \ldots, s_i \rangle$ for some $i \in \{1, \ldots, k-1\}$ and the remaining time becomes equal to $d(v, s_{i+1}) + \sum_{j=i+1}^{k-1} d(s_j, s_{j+1})$ then this job v is terminated (without accruing reward) and the policy moves directly to vertices $\langle s_{i+1}, \ldots, s_k \rangle$ and ends — it cannot accrue any reward from any vertex along this shortest path $\langle v, s_{i+1}, \ldots, s_k \rangle$.

Notice that the basic stochastic orienteering problem (studied in the previous sections) is a special case of stochastic sequence orienteering when $k = 1$ and the metric is symmetric.

An important subroutine in our algorithm for stochastic sequence orienteering is an approximation algorithm for its *deterministic* version. The deterministic sequence orienteering with $k = 2$ has been studied previously, called *point to point orienteering* [2, 28, 12]. Here, the input consists of a metric (V, d) with rewards on vertices, bound B, and specified source (s_1) and destination (s_2) vertices; the goal is to compute a path from s_1 to s_2 of length at most B that maximizes the total reward. A constant-factor approximation algorithm is known for this problem in symmetric metrics [2], and the directed setting admits approximation ratios of: $O(\log^2 n / \log\log n)$ [28] and $O(\log^2 \mathsf{Opt})$ [12]. Our first result is to show that the deterministic sequence orienteering problem *for arbitrary* k, admits an $O(\alpha)$-approximation algorithm, where α is the best known approximation ratio for point to point orienteering.

Using this we obtain the main result of this section (Theorem 3), i.e. any α-approximation algorithm for directed point to point orienteering can be used to obtain:

- An $O(\alpha)$-approximation algorithm for non-adaptive stochastic sequence orienteering.
- An $O(\alpha \cdot \log\log B)$-approximation algorithm for adaptive stochastic sequence orienteering.

We follow the same framework as for basic stochastic orienteering (i.e. undirected $k = 1$ case). In Subsection 6.1 we obtain an $O(\alpha)$-approximation algorithm for *knapsack sequence orienteering*. Then we use this to obtain an $O(\alpha)$-approximation algorithm for non-adaptive sequence orienteering in Subsection 6.2, and an $O(\alpha \cdot \log\log B)$-approximation algorithm for adaptive sequence orienteering in Subsection 6.3.

6.1. (Deterministic) Knapsack Sequence Orienteering

In the *knapsack sequence orienteering* problem, we are given a directed metric (V, d), a sequence of of portal vertices to visit $\langle s_1, \ldots, s_k \rangle$ and two budgets: L which is the "travel" budget, and W which is the "knapsack" budget. Each job v has a reward $\widehat{r}_v$, and also a "size" $\widehat{s}_v$. A feasible solution is a path P which (i) visits $s_1, \ldots, s_k$ in that order, (ii) has total length at most L, and (iii) total size $\widehat{s}(P) := \sum_{v \in P} \widehat{s}_v$ is at most W. The goal is to find a feasible solution of maximum reward $\sum_{v \in P} \widehat{r}_v$.

In order to devise an algorithm for this problem, we first consider the problem without the knapsack constraint and give an $O(\alpha)$ approximation, where α is the approximation factor for the point-to-point orienteering problem (i.e., $k = 2$). Using this (and the Lagrangian relaxation á la Theorem 5), we show an $O(\alpha)$-approximation for the knapsack sequence orienteering problem.

6.1.1. Approximating Sequence Orienteering

In this problem, there are no sizes at vertices. The goal is to find a path visiting $\langle s_1, \ldots, s_k \rangle$ of length at most B with maximum reward. Our main idea is to view this problem as that of submodular maximization over a (partition) matroid, with an additional knapsack constraint. We first give a high-level description of the algorithm. Let U_i denote the set of all paths from s_i to s_{i+1}. Then, a sequence is simply a set of paths, one from each U_i, i.e., an independent set in the partition matroid $\{U_1, U_2, \ldots, U_{k-1}\}$ (with cardinality bound of one on each part). Furthermore, the total reward of any subset of $\cup_{i=1}^{k-1} U_i$ can be represented by a weighted coverage function, which is submodular. In order to ensure that the path we find has length bounded by B, we define an appropriate knapsack constraint. So the overall problem reduces to submodular maximization over the intersection of a partition matroid and a knapsack constraint. An additional issue is that the groundset $\cup_{i=1}^{k-1} U_i$ is of exponential size: we deal with this using an implicit reduction from knapsack constraints to partition matroids [24, 11]. We now present the details.

THEOREM 7. *There is an $O(\alpha)$-approximation algorithm for sequence orienteering, where α denotes the best approximation ratio for directed point-to-point orienteering.*

Proof. For this reduction, it will be convenient to define the groundset $U = \bigcup_{i=1}^{k-1} U_i$ where

$$U_i \quad = \quad \{\langle P, i\rangle : P \text{ is an } s_i - s_{i+1} \text{ path}\}$$

Notice that this groundset is exponentially large; however our algorithm will not use it explicitly. Define a *partition matroid* $\mathcal{M}$ on U, where subset $S \subseteq U$ is independent if and only if $|S \cap U_i| \leq 1$ for each index $i \in \{1, \ldots, k-1\}$. Note that any base in $\mathcal{M}$ corresponds to a valid $\langle s_1, \ldots, s_k\rangle$ sequence path. Let $\mathcal{I}(\mathcal{M}) \subseteq 2^U$ denote the collection of independent sets in the partition matroid $\mathcal{M}$.

In order to ensure the length bound of B, we define a *knapsack constraint* $\mathcal{K}$. For each $\langle P, i\rangle \in U$ define weight $w_{\langle P,i\rangle} = d(P) - d(s_i, s_{i+1})$; and set the knapsack capacity to $W := B - \sum_{j=1}^{k-1} d(s_j, s_{j+1})$. Let $\mathcal{I}(\mathcal{K}) = \{S \subseteq U : \sum_{e\in S} w_e \leq W\} \subseteq 2^U$ denote the collection of "independent sets" in knapsack $\mathcal{K}$.

CLAIM 7. *There is an exact correspondence between:*
1. *Subsets $S \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K})$, that are independent in both $\mathcal{M}$ and $\mathcal{K}$.*
2. *Paths P of length at most B that contain vertices $s_1, \ldots, s_k$ in that order.*

Proof. In one direction, consider any $S \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K})$. Note that each $i \in \{1, \ldots, k-1\}$ contains a "dummy element" $e_i \in U_i$ corresponding to the shortest path $\langle s_i, s_{i+1}\rangle$ with $w(e_i) = 0$. If S is not a base in the partition matroid $\mathcal{M}$ then augment it to a base, by adding element e_i for each part $i \in \{1, \ldots, k-1\}$ with $S \cap U_i = \emptyset$. Since the dummy elements have zero weight, we still have $S \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K})$. Now, S corresponds to a collection $\mathcal{P}$ of $s_i - s_{i+1}$ paths, exactly one for each $i = 1, \ldots, k-1$. By the definition of weights in the knapsack $\mathcal{K}$, it follows that the total length of $\mathcal{P}$ is at most B. Concatenating the paths in $\mathcal{P}$ yields the desired $\langle s_1, \ldots, s_k\rangle$ sequence path.

In the other direction, consider any $\langle s_1, \ldots, s_k\rangle$ sequence path P. Clearly P is a concatenation of sub-paths $\{P_1, \ldots, P_{k-1}\}$, where P_i is an $s_i - s_{i+1}$ path for each $i = 1, \ldots, k-1$. Consider the set $S' = \{\langle P_i, i\rangle : i = 1, \ldots, k-1\} \subseteq U$. Clearly $S' \in \mathcal{I}(\mathcal{M})$. Also, the total weight of S' in knapsack $\mathcal{K}$ is

$$\sum_{i=1}^{k-1}(d(P_i) - d(s_i, s_{i+1})) \quad \leq \quad B - \sum_{i=1}^{k-1} d(s_i, s_{i+1}) \quad = \quad W,$$

since P has length at most B. Thus $S' \in \mathcal{I}(\mathcal{K})$ as well and hence $S' \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K})$. ■

Now, define the objective function:

$$f(S) \; := \; \sum_{v\in V} r_v \cdot \min\left\{ \sum_{\langle P,i\rangle \in S} \mathbf{1}_{v\in P}, \; 1 \right\}, \qquad \forall S \subseteq U$$

Above $r : V \to \mathbb{R}_+$ denotes the rewards at different vertices, and $\mathbf{1}_{v\in P}$ is the indicator of event "$v \in P$". Note that f is a weighted coverage function, and so it is monotone and submodular on U. Therefore, by Claim 7, the sequence orienteering problem is precisely:

$$\max \quad \{ f(S) \quad : \quad S \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K}) \} \tag{6.5}$$

Submodular maximization over the intersection of matroid and knapsack constraints admits a constant factor approximation algorithm [24, 15]; but we need to take some more care since the groundset is not available explicitly. Below we show that a slight modification of the approach in [24] suffices. Specifically, we show that the knapsack $\mathcal{K}$ can be approximately simulated by another partition matroid.

THEOREM 8 (**[24]**). *Given any knapsack constraint $\sum_{e\in U} w_e \cdot x_e \leq W$ and parameter $\ell \leq |U|$, there is a polynomial (in ℓ) time computable collection $\mathcal{M}_1, \ldots, \mathcal{M}_T$ of $T = \ell^{O(1)}$ partition matroids such that:*

1. *For every $S \in \cup_{t=1}^{T}\mathcal{I}(\mathcal{M}_t)$ we have $\sum_{e\in S} w_e \le 2\cdot W$.*
2. *For every $S \subseteq U$ with $|S| \le \ell$ and $\sum_{e\in S} w_e \le W$ we have $S \in \cup_{t=1}^{T}\mathcal{I}(\mathcal{M}_t)$.*

This follows directly from Lemma 3.3 in [24]. Although that result is only stated for $\ell = |U|$, it can be extended to any l as follows (our requirement will be for $\ell = k$). Here we only mention the changes required to the proof in [24]. We partition U into $G = \lceil \log_2 \ell \rceil + 1$ groups according to geometrically increasing weights. That is, $V_0 := \{e \in U : w_e \le W/\ell\}$ and $V_j = \{e \in U : \frac{W}{\ell}\cdot 2^{j-1} < w_e \le \frac{W}{\ell}\cdot 2^j\}$ for all $j = 1, \cdots, \lceil \log_2 \ell \rceil$. Then we guess upper bounds $\{n_j : 1 \le j \le \lceil \log_2 \ell \rceil\}$ of the numbers of elements from each part V_j, and define a partition matroid corresponding to this. While a näive enumeration has a total of $\ell^{O(\log \ell)}$ different partition matroids, it can be made polynomial using an enumeration idea from [11]. Using this approach, the number T of partition matroids is polynomial is ℓ.

In our setting, since we know all feasible sets in the intersection $\mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{K})$ are of size at most k, we can use Theorem 8 with $\ell := k$. So we can (approximately) reduce the knapsack constraint to a partition matroid $\mathcal{M}'$ that is obtained by enumerating over $T = poly(k)$ possibilities. Moreover, each part in $\mathcal{M}'$ corresponds to an index $j \in \{0, \ldots, \log_2 k\}$ such that elements in part j of $\mathcal{M}'$ have weight at most $2^j \cdot W/k$. Thus solving (6.5) can be reduced to:

$$\max \quad \{f(S) \quad : \quad S \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{M}')\} \tag{6.6}$$

The solution S^* to this problem does not itself satisfy $\mathcal{K}$, but we have $w(S^*) \le 2\cdot W$. So a greedy partitioning can be used to obtain subsets S_1, S_2 and S_3 such that $S^* = S_1 \cup S_2 \cup S_3$ and $w(S_a) \le W$ for $a = 1, 2, 3$. Choosing the subset with maximum function value gives us $S' \subseteq S^*$ with $w(S') \le W$ and $f(S') \ge f(S^*)/3$, by sub-additivity. Thus an approximation algorithm for (6.6) leads to one for (6.5), at the loss of an additional factor of three.

To solve (6.6) we use the natural greedy algorithm: always add element $e \in U$ that retains independence and (approximately) maximizes the marginal increase in the objective. This is well known to achieve an approximation ratio of $(1 + 2\rho)$ assuming a ρ-approximate oracle for the greedy addition step [9]. We observe below that this greedy step corresponds to the point to point orienteering problem: thus $\rho = \alpha$ and we obtain a $(1 + 2\alpha)$-approximation algorithm for (6.6) and $(3 + 6\alpha)$-approximation algorithm for (6.5).

Recall the greedy step: given $S \subseteq U$ find $\max\{f(S \cup \{e\}) - f(S) : e \in U, S \cup \{e\} \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{M}')\}$. We first enumerate over the part $i \in \{1, \ldots, k-1\}$ of $\mathcal{M}$ and part $j \in \{0, \ldots, \log_2 k\}$ of $\mathcal{M}'$, that correspond to the candidate element e. If the upper bound on either of these parts is tight then e can not be added to S; otherwise $S \cup \{e\} \in \mathcal{I}(\mathcal{M}) \cap \mathcal{I}(\mathcal{M}')$. Assuming the latter, we optimize over all elements corresponding to parts i and j (in $\mathcal{M}$ and $\mathcal{M}'$ respectively); this is just:

$$\max \quad \left\{ \sum_{v\in P} \overline{r}_v \quad : \quad P \text{ is an } s_i - s_{i+1} \text{ path}, \quad d(P) \le d(s_i, s_{i+1}) + 2^j \cdot \frac{W}{k} \right\}$$

Above $\overline{r}_v = r_v$ if vertex v is not already covered by S; and $\overline{r}_v = 0$ otherwise. Note that the constraint that P is an $s_i - s_{i+1}$ path is due to part i of $\mathcal{M}$, and the bound on its length is from part j of $\mathcal{M}'$. Observe that this is precisely an instance of point to point orienteering (from s_i to s_{i+1}), and hence we can use the α-approximation algorithm assumed in the theorem. ■

6.1.2. Approximating Knapsack Sequence Orienteering.

In the knapsack sequence orienteering problem (KnapSeqOrient), we are given a directed metric (V, d) with sequence $\langle s_1, \ldots, s_k \rangle$ of portal vertices, rewards and sizes at each vertex, and separate budgets L and W. The goal is to find a path consistent with the sequence $\langle s_1, \ldots, s_k \rangle$ that maximizes the reward on it such that its length is at most L and total size of its vertices is at most W. Using the Lagrangian relaxation approach (exactly as in Theorem 5), we can reduce the knapsack sequence orienteering problem to an instance of sequence orienteering, while losing a factor 2 in the approximation ratio.

THEOREM 9. *There is an $O(\alpha)$-approximation algorithm for knapsack sequence orienteering, where α denotes the best approximation ratio for directed point-to-point orienteering.*

In the next two subsections, we show how this result can be used within our framework for solving the stochastic versions. Since most of the details are essentially same as in Sections 4 and 5, we only point out the changes required in context of sequence orienteering.

6.2. Non-adaptive Sequence Orienteering

Here we use the $O(\alpha)$-approximation algorithm for $\mathsf{KnapSeqOrient}$ to obtain an $O(\alpha)$-approximation algorithm for non-adaptive sequence orienteering. We define valid $\mathsf{KnapSeqOrient}$ instances $\mathcal{I}_{kso}(W)$ exactly as in Definition 2, which is parametrized by value $W \in \{0,1,\dots,B\}$: recall that the size budget is W and travel budget is $B-W$. The algorithm remains the same as Algorithm 1. Observe that Assumption 1 can be enforced for sequence orienteering as well. Furthermore, implementing a $\mathsf{KnapSeqOrient}$ solution as a non-adaptive policy for stochastic sequence orienteering follows directly by Lemma 1. Therefore, it remains to prove an equivalent of Lemma 2, i.e. for some choice of W the optimal value of $\mathsf{KnapSeqOrient}$ instance $\mathcal{I}_{kso}(W)$ is $\Omega(\mathsf{Opt})$.

As in the proof of Lemma 2, let the non-adaptive optimum P^* correspond to ordering $v_1,\dots,v_n$ where the portal vertices $\langle s_1,\dots,s_k\rangle$ appear in the prescribed order with $v_1 = s_1$ and $v_n = s_k$. For any $v_j \in V$ recall that $D_j = \sum_{\ell=1}^{j} d(v_{\ell-1}, v_\ell)$ is the travel time to visit v_j; also define

$$\overline{D}_j \ := \ D_j + d(v_j, s_i) + \sum_{\ell=i}^{k-1} d(s_i, s_{i+1})$$

where $i \in \{1,\dots,k-1\}$ is the index such that v_j appears between portals s_i and s_{i+1}. Note that $\overline{D}_j$ is the minimum amount of travel that *must* be incurred if the policy visits vertex v_j; this is because in sequence orienteering, we are required to visit all the portal vertices. Note that by triangle inequality, $\overline{D}_j$ is non-decreasing in j. Analogous to (4.2), let j^* denote the first index such that:

$$\textstyle\sum_{i<j} \mu_{v_i}\left(B - \overline{D}_j\right) \quad \geq \quad K \cdot (B - \overline{D}_j) \tag{6.7}$$

Here K is some constant. Having defined our "stopping point", it is easy to see that Lemma 1 continues to hold and the rest of the proof is completely identical, when we replace D with $\overline{D}$. Thus we obtain an $O(\alpha)$-approximation algorithm for non-adaptive sequence orienteering.

6.3. Adaptive Sequence Orienteering

Here we use the $O(\alpha)$-approximation algorithm for $\mathsf{KnapSeqOrient}$ to obtain an $O(\alpha \cdot \log\log B)$-approximation algorithm for adaptive sequence orienteering. We enforce Assumption 1, and it suffices to show an analog of Lemma 4 that there is some choice of parameter W for which instance $\mathcal{I}_{kso}(W)$ has value $\Omega(\mathsf{Opt}/\log\log B)$. The proof given in Section 5 makes use of the metric being undirected, and we need to generalize that to the directed setting. To this end, we use a different concentration inequality in place of Freedman's inequality, which is better suited in the directed setting.

Note that the optimal adaptive policy is a decision tree Opt, with nodes being vertices that are visited and its branches corresponding to the random instantiation. (Here we do not subdivide Opt as done in Section 5; we also do not assume that jobs are not co-located). For any node x in Opt, we denote by $\mathsf{level}(x)$ the *travel time* spent until node x; to reduce notation we will use x to also denote the vertex in the metric that corresponds to x. For node x define $\overline{\mathsf{lev}}(x) := \mathsf{level}(x) + d(v_j, s_i) + \sum_{\ell=i}^{k-1} d(s_i, s_{i+1})$ where $i \in \{1,\dots,k-1\}$ is the index such that x appears between portals s_i and s_{i+1}. Note that $\overline{\mathsf{lev}}(x)$ is the minimum amount of travel that *must* be incurred if the policy visits node x. By triangle inequality, $\overline{\mathsf{lev}}(\cdot)$ is non-decreasing down the Opt tree.

Analogous to (5.3), node x is called a *star node* if it is the first node along its sample path for which:

$$\textstyle\sum_{x' \prec x} \mu_{x'}\left(B - \overline{\mathsf{lev}}(x)\right) \quad > \quad 8K\left(B - \overline{\mathsf{lev}}(x)\right) \tag{6.8}$$

Here $K = \Theta(\log\log B)$. This condition clearly holds when $\overline{\mathsf{lev}}(x) = B$; so the parent y of any star node x must satisfy $\overline{\mathsf{lev}}(y) \leq B-1$. We define Opt'' by pruning Opt at star nodes (again, rewards at star nodes are not included in Opt''). Leaf-nodes in Opt'' are either leaves in Opt or parents of star nodes. We will prove Lemma 5; this would imply an analog of Lemma 4 (with $\mathsf{KnapSeqOrient}$ instances) exactly as in Section 5 (by replacing level by $\overline{\mathsf{lev}}$).

In proving Lemma 5, we again partition the star nodes x into bands depending on the value $B - \overline{\mathsf{lev}}(x)$. Star node x is in band i if $B - \overline{\mathsf{lev}}(x) \in (B/2^{i+1}, B/2^i]$ for $0 \le i \le \lceil \log B \rceil$, and in band $\lceil \log B \rceil + 1$ if $\overline{\mathsf{lev}}(x) = B$. By Assumption 1, as in Section 5, the loss in reward by truncating at band $\lceil \log B \rceil + 1$ is at most $\mathsf{Opt}/8$. Moreover, an analogue of Claim 4 holds in this setting as well. So in order to bound the loss from other bands, it suffices to prove the analog of Lemma 6:

$$\text{For any } i \in \{0, \ldots, \lceil \log B \rceil\}, \text{ the probability of reaching band } i \text{ is at most } \tfrac{1}{10\lceil \log B \rceil}. \tag{6.9}$$

We obtain the first part of Claim 5 (the second part is not true in the directed setting).

$$\text{For each } i \in \{0, 1, \ldots, \lceil \log B \rceil\} \text{ and star node } x \text{ in band } i\text{:} \quad \sum_{x' \prec x} \mu_{x'}\left(B/2^{i+1}\right) \;\ge\; 2K \cdot \frac{B}{2^i} \tag{6.10}$$

Claim 6 continues to hold,

$$\text{For each } i \in \{0, 1, \ldots, \lceil \log B \rceil\} \text{ and star node } x \text{ in band } i\text{:} \quad \sum_{x' \prec x} \min\left(S_{x'}, \frac{B}{2^{i+1}}\right) \;\le\; \frac{B}{2^i} \tag{6.11}$$

We use these two properties to bound the probability of reaching band i. We also make use of a concentration inequality due to Zhang [29] that is described below (we use this in place of Freedman's inequality because its form suits us better in applying it for directed metrics):

Let $I_1, I_2, \ldots$ be a sequence of possibly dependent random variables; for each $k \ge 1$ 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 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 we have:

THEOREM 10 **(Theorem 1 in [29])**.

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

We make use of this result by setting I_k to be the k^{th} node seen in Opt, and

$$\xi_k(I_1, \ldots, I_k) \quad = \quad \min\left\{\frac{S_{I_k}}{B/2^{i+1}}, 1\right\}$$

Recall that for any node x, its instantiated size (i.e. processing time) is denoted S_x. We define the stopping time τ as reaching either a band i star node or a node that has no descendant band i star node. At any band i star node, we have:

$$(6.10) \quad \Longrightarrow \quad \sum_{k=1}^{\tau} \mathbb{E}_{I_k}[\xi_k(I_1, \ldots, I_k)] \;\ge\; 4K$$

$$(6.11) \quad \Longrightarrow \quad \sum_{k=1}^{\tau} \xi_k(I_1, \ldots, I_k) \;\le\; 2$$

Combining these, the probability of reaching a band i star node is at most:

$$\Pr\left[\sum_{k=1}^{\tau} \mathbb{E}_{I_k}[\xi_k(I_1, \ldots, I_k)] \;\ge\; 4 \cdot \left(\sum_{k=1}^{\tau} \xi_k(I_1, \ldots, I_k) + K - 2\right)\right] \quad \le \quad e^{-(K-2)},$$

where we use Theorem 10 with $\delta = K - 2$. Using $K = \Theta(\log \log B)$, we obtain that this probability is at most $1/(10\lceil \log B \rceil + 1)$, which proves (6.9).

7. Stochastic Orienteering with Correlated Rewards

In this section we consider the stochastic orienteering problem where the reward of each job is a random variable that may be correlated with its processing time (i.e. size). The distributions at different vertices are still independent of each other. The input to CorrOrient consists of a metric (V, d) with root vertex ρ and a bound B. At each vertex $v \in V$, there is a stochastic job with a given probability distribution over (size, reward) pairs: for each $t \in \{0, 1, \dots, B\}$, the job at v has size t and reward $r_{v,t}$ with probability $\pi_{v,t}$. Again we consider the non-preemptive setting, so once a job is started it must be run to completion unless the budget B is exhausted. The goal is to devise a (possibly adaptive) policy that maximizes the expected reward of completed jobs, subject to the total budget (travel time + processing time) being at most B. The results of this section apply only to the basic orienteering setting, and not sequence orienteering.

When there is no metric in the problem instance, this is precisely the correlated stochastic knapsack problem, and [22] gave a non-adaptive algorithm which is a constant-factor approximation to the optimal adaptive policy; this used an LP-relaxation that is quite different from that in the uncorrelated setting. The trouble with extending that approach to stochastic orienteering is again that we do not know of LP relaxations with good approximation guarantees even for deterministic orienteering. We circumvented this issue for the uncorrelated case by using a Martingale analysis to bypass the need for an LP relaxation, which gave a direct lower bound. We adopt a similar approach for CorrOrient, but as Theorem 4 says, our approximation ratio is only $O(\log n \log B)$: this is because our algorithm here relies on the "deadline orienteering" problem. Moreover, we show that CorrOrient is at least as hard to approximate as the deadline orienteering problem, for which the best guarantee known is an $O(\log n)$ approximation algorithm [2].

7.1. The Non-Adaptive Algorithm for CorrOrient

We now present our approximation algorithm for CorrOrient, which proceeds via a reduction to suitably constructed instances of the deadline orienteering problem [2]. An instance of *deadline orienteering* (DeadlineOrient) consists of a metric (denoting travel times) with a reward and deadline at each vertex, and a root vertex. The objective is to compute a path starting from the root that maximizes the reward obtained from vertices that are visited before their deadlines.

Our high level approach, much like the earlier sections of the paper, is to reduce the stochastic problem to a deterministic one where there is a travel budget and a size budget, i.e., a knapsack version of a deterministic orienteering problem. In the uncorrelated stochastic orienteering problem, it did not matter when the tour visited a vertex, as long as the job can finish with reasonable probability within the size budget allocated by the tour (the rewards are fixed). Hence the deterministic problem was simply the orienteering problem, with a knapsack constraint. In the correlated case however, the reward could in fact depend on when the job is started with respect to the budget remaining. For example, if a job has reward 1 when its processing time is $B - 1$, and 0 otherwise, and its expected size is 1, then to collect any reward from this job, we'd have to start processing it by a time of 1. Therefore, we use the deadline orienteering problem as a deterministic subproblem for stochastic orienteering with correlated rewards.

We solve a knapsack version of deadline orienteering by taking a *Lagrangian relaxation* of the processing times, and then use an amortized analysis to argue that the reward is high in expectation. (In this it is similar to the ideas of [21].) The crux of our proof is in showing that we can indeed reduce the stochastic problem to the deadline orienteering problem, Namely, what deadlines do we choose for each job, and if we create many copies with different deadlines for each job then how do we ensure that the reward is not over counted?

Notation. Let Opt denote an optimal decision tree. We classify every execution of a job in this decision tree as belonging to one of $(\log_2 B + 1)$ *types*. For $i \in [\log_2 B]$, a *type-i* job execution occurs when the *processing time* spent *before* running the job lies in the interval $[2^i - 1, 2^{i+1} - 1)$. So if t' is the distance spent before reaching a type-i job then its start time lies in $[t' + 2^i - 1, t' + 2^{i+1} - 1)$. Note that the same job might have different types on different sample paths of Opt; but for a fixed sample path down Opt, it can have at most one type. If $\mathsf{Opt}(i)$ is the expected reward obtained from job runs of type i, then we have $\mathsf{Opt} = \sum_i \mathsf{Opt}(i)$, and hence $\max_{i \in [\log_2 B]} \mathsf{Opt}(i) \geq \Omega(\frac{1}{\log B}) \cdot \mathsf{Opt}$. For all $v \in V$ and $t \in [B]$, let $R_{v,t} := \sum_{z=0}^{B-t} r_{v,z} \cdot \pi_{v,z}$ denote the expected reward when job v's size is restricted to being at most $B - t$.

Note that this is the expected reward obtained from job v *if it starts at time* t. Recall that for any $v \in V$, S_v denotes its random size which has distribution $\{\pi_{v,t}\}_{t=0}^{B}$

7.1.1. Reducing CorrOrient **to (deterministic)** DeadlineOrient

The high-level idea is the following: for any fixed i, we create an instance of DeadlineOrient to get an $O(\log n)$ fraction of $\mathsf{Opt}(i)$ as reward; then choosing the best such setting of i gives us the $O(\log n \log B)$-approximation algorithm. To obtain the instance of DeadlineOrient, for each vertex v we create several copies of it: for each time t there is a copy corresponding to starting job v at time t (and hence has reward $R_{v,t}$). However, to prevent the DeadlineOrient solution from collecting reward from many different copies of the same vertex, we make copies of vertices only when the reward changes by a constant factor. The following claim is useful for defining such a minimal set of starting times for each job.

CLAIM 8. *Given any non-increasing function $f : [B] \to \mathbf{R}_+$, we can efficiently find a subset $I \subseteq [B]$:*

$$\frac{f(t)}{2} \le \max_{\ell \in I : \ell \ge t} f(\ell) \qquad \text{and} \qquad \sum_{\ell \in I : \ell \ge t} f(\ell) \le 3 \cdot f(t), \qquad \forall t \in [B].$$

Proof. The set I is constructed as follows.

Algorithm 2 Computing the support I in Claim 8.

```
1: let $h \leftarrow 0$, $k_0 \leftarrow 0$, $I \leftarrow \emptyset$.
2: while $k_h \in [B]$ and $f(k_h) > 0$ do
3:     $\ell_h \leftarrow \max\left\{\ell \in [B] : f(\ell) \ge \frac{f(k_h)}{2}\right\}$.
4:     $k_{h+1} \leftarrow \ell_h + 1$, $I \leftarrow I \bigcup \{\ell_h\}$.
5:     $h \leftarrow h + 1$.
6: end while
7: output set $I$.
```

Observe that B is always contained in the set I, and hence for any $t \in [B]$, $\min\{\ell \ge t : \ell \in I\}$ is well-defined. To prove the claimed properties let $I = \{\ell_h\}_{h=0}^{p}$. For the first property, given any $t \in [B]$ let $\ell_h = \min\{\ell \ge t : \ell \in I\}$. We must have $\ell_{h-1} < t$, and so $k_h \le t$. Hence $f(\ell_h) \ge f(k_h)/2 \ge f(t)/2$; the first inequality is by the choice ℓ_h in Algorithm 2, and the second inequality uses the fact that f is non-increasing.

We now show the second property. For any index h, we have $k_h \le \ell_h < k_{h+1} \le \ell_{h+1}$. Moreover, $f(k_{h+1}) = f(\ell_h + 1) < f(k_h)/2$ by the choice of ℓ_h. Given any $t \in [B]$ let q be the index such that $\ell_q = \min\{\ell \ge t : \ell \in I\}$. Consider the sum:

$$\sum_{h \ge q} f(\ell_h) = f(\ell_q) + \sum_{h \ge q+1} f(\ell_h) \le f(\ell_q) + \sum_{h \ge q+1} f(k_h) \le f(\ell_q) + 2 \cdot f(k_{q+1}) \le 3 \cdot f(\ell_q)$$

The first inequality uses $f(\ell_h) \le f(k_h)$, the next uses $f(k_{h+1}) < f(k_h)/2$ and a geometric summation, and the last is by $\ell_q \le k_{q+1}$. Finally observe that $t \le \ell_q$, so $\sum_{h \ge q} f(\ell_h) \le 3 \cdot f(t)$. This completes the proof. ∎

Consider any $i \in [\log_2 B]$. Now for each $v \in V$, apply Claim 8 to the function $f(t) := R_{v,t+2^i-1}$ to obtain a subset $I_v^i \subseteq [B]$. These subsets define the copies of each job that we will use.

For each i and parameter $\lambda \ge 0$ we define a deadline orienteering instance as follows.

DEFINITION 3 (DEADLINE ORIENTEERING INSTANCE $\mathcal{I}_i(\lambda)$). The metric is (V, d) with root vertex ρ. For each $v \in V$ and $\ell \in I_v^i$ there is a job $\langle v, \ell \rangle$ located at vertex v with deadline ℓ and reward $\hat{r}_i(v, \ell, \lambda) := R_{v,\ell+2^i-1} - \lambda \cdot \mathbb{E}\left[\min(S_v,\, 2^i)\right]$. The objective in $\mathcal{I}_i(\lambda)$ is to find a path originating at ρ that maximizes the reward of the jobs visited within their deadlines.

Also define $N_i = \{\langle v, \ell \rangle : \ell \in I_v^i,\, v \in V\}$, the set of all jobs in instance $\mathcal{I}_i(\lambda)$. For each job $\langle v, \ell \rangle \in N_i$, define its size $s_i(v, \ell) = s_i(v) := \mathbb{E}\left[\min(S_v,\, 2^i)\right]$, and let $r_i(v, \ell) = R_{v,\ell+2^i-1}$.

The co-located jobs $\{\langle v,\ell\rangle : \ell \in I_v^i\}$ in $\mathcal{I}_i(\lambda)$ are copies of job v in the original $\mathsf{CorrOrient}$ instance, where copy $\langle v,\ell\rangle$ corresponds to running v as a type-i job after distance ℓ. The parameter λ can be thought of as a Lagrangian multiplier, and so $\mathcal{I}_i(\lambda)$ is a Lagrangian relaxation of a $\mathsf{DeadlineOrient}$ instance with an additional constraint that the total size is at most 2^i. It is immediate by the definition of rewards that $\mathsf{Opt}(\mathcal{I}_i(\lambda))$ is a non-increasing function of λ.

The idea of our algorithm is to argue that for the "right" setting of λ, the optimal $\mathsf{DeadlineOrient}$ solution for $I_i(\lambda)$ has value $\Omega(\mathsf{Opt}(i))$, which is shown in Lemma 8. Moreover, as shown in Lemma 9, we can recover a valid solution to $\mathsf{CorrOrient}$ from an approximate solution to $I_i(\lambda)$.

LEMMA 7. *For any $i \in [\log B]$ and $\lambda > 0$, the optimal value of the deadline orienteering instance $\mathcal{I}_i(\lambda)$ is at least $\frac{\mathsf{Opt}(i)}{2} - \lambda \cdot 2^{i+1}$.*

Proof. Consider the optimal decision tree Opt of the $\mathsf{CorrOrient}$ instance, and label every node in Opt by a (dist, size) pair, where dist is the total time spent traveling and size the total time spent processing jobs *before* visiting that node. Note that both dist and size are non-decreasing as we move down Opt. Also, type-i nodes are those where $2^i - 1 \le \mathsf{size} < 2^{i+1} - 1$. We use $\mathsf{Opt}(i)$ to denote the decision tree obtained by retaining only type-i nodes in Opt; $\mathsf{Opt}(i)$ also denotes the expected reward from this decision tree.

For any vertex $v \in V$, let X_v^i denote the indicator random variable that job v is run as type-i in Opt, and S_v be the random variable denoting its instantiated size. Note that X_v^i and S_v are independent: X_v^i is determined by the instantiations at vertices $V \setminus \{v\}$, and S_v depends only on vertex v (which is independent of all other vertices). Also let $Y^i = \sum_{v\in V} X_v^i \cdot \min(S_v, 2^i)$ be the random variable denoting the sum of truncated sizes of type-i jobs. By definition of type-i, we have that $Y^i \le 2 \cdot 2^i$ with probability one, and hence $\mathbb{E}[Y^i] \le 2^{i+1}$. For ease of notation let $q_v = \Pr[X_v^i = 1]$ for all $v \in V$. We now have,

$$2^{i+1} \;\ge\; \mathbb{E}[Y^i] \;=\; \sum_{v\in V} q_v \cdot \mathbb{E}[\min(S_v, 2^i) \mid X_v^i = 1] \;=\; \sum_{v\in V} q_v \cdot \mathbb{E}[\min(S_v, 2^i)] \;=\; \sum_{v\in V} q_v \cdot s_i(v) \qquad (7.12)$$

Now consider the expected reward $\mathsf{Opt}(i)$. We can write:

$$\begin{aligned} \mathsf{Opt}(i) \quad &= \quad \sum_{v\in V}\sum_{\ell\in[B]}\sum_{k=2^i-1}^{2^{i+1}-2} \Pr[\mathbf{1}_{v,\mathsf{dist}=\ell,\mathsf{size}=k}] \cdot R_{v,\ell+k} \\ &\le \quad \sum_{v\in V}\sum_{\ell\in[B]} \Pr[\mathbf{1}_{v,\mathsf{type}=i,\mathsf{dist}=\ell}] \cdot R_{v,\ell+2^i-1} \end{aligned} \qquad (7.13)$$

where $\mathbf{1}_{v,\mathsf{dist}=\ell,\mathsf{size}=k}$ is the indicator that Opt visits v with $\mathsf{dist} = \ell$ and $\mathsf{size} = k$, and $\mathbf{1}_{v,\mathsf{type}=i,\mathsf{dist}=\ell}$ is the indicator that Opt visits v as type-i with $\mathsf{dist} = \ell$. The inequality uses the facts that $R_{v,\ell+k}$ is non-increasing in k, and $\Pr[\mathbf{1}_{v,\mathsf{type}=i,\mathsf{dist}=\ell}] = \sum_{k=2^i-1}^{2^{i+1}-2} \Pr[\mathbf{1}_{v,\mathsf{dist}=\ell,\mathsf{size}=k}]$.

Now going back to the metric, let $\mathcal{P}$ denote the set of all possible rooted paths traced by $\mathsf{Opt}(i)$ in the metric (V,d). Now for each path $P \in \mathcal{P}$, define the following quantities.

- $\beta(P)$ is the probability that $\mathsf{Opt}(i)$ traces P.
- For each vertex $v \in P$, $d_v(P)$ is the *travel time* (i.e. dist) incurred in P prior to reaching v. Note that the actual time at which v is visited is $\mathsf{dist} + \mathsf{size}$, which is in general larger than $d_v(P)$.
- $w_\lambda(P) = \sum_{v\in P}\left[\frac{1}{2}\cdot R_{v,d_v(P)+2^i-1} - \lambda \cdot s_i(v)\right]$.

Moreover, for each $v \in P$, let $\ell_v(P) = \min\{\ell \in I_v^i \mid \ell \ge d_v(P)\}$; recall the definition I_v^i using Claim 8, and that the quantity $\ell_v(P)$ is always well-defined.

For any path $P \in \mathcal{P}$, consider P as a solution to the $\mathsf{DeadlineOrient}$ instance $\mathcal{I}_i(\lambda)$ that visits the copies $\{\langle v, \ell_v(P)\rangle : v \in P\}$ within their deadlines. It is feasible for the $\mathcal{I}_i(\lambda)$ because for each vertex $v \in P$, the deadline of its chosen copy $\ell_v(P) \ge d_v(P)$ the time when it is visited by P. Moreover, the objective value of P is precisely

$$\sum_{v\in P} \hat{r}_i(v, \ell_v(P), \lambda) \;=\; \sum_{v\in P}\left[R_{v,\ell_v(P)+2^i-1} - \lambda\cdot s_i(v)\right] \;\ge\; \sum_{v\in P}\left[\frac{1}{2}\cdot R_{v,d_v(P)+2^i-1} - \lambda \cdot s_i(v)\right] \;=\; w_\lambda(P)$$

where the inequality above uses the definition of $\ell_v(P) = \min\{\ell \in I_v^i \mid \ell \geq d_v(P)\}$ and Claim 8. Now,

$$\begin{aligned}
\mathsf{Opt}(\mathcal{I}_i(\lambda)) \;&\geq\; \max_{P \in \mathcal{P}} w_\lambda(P) \;\geq\; \sum_{P \in \mathcal{P}} \beta(P) \cdot w_\lambda(P) \;=\; \sum_{P \in \mathcal{P}} \beta(P) \cdot \sum_{v \in P} \left[\frac{1}{2} \cdot R_{v, d_v(P)+2^i-1} - \lambda \cdot s_i(v)\right] \\
&= \frac{1}{2} \sum_{v \in V} \sum_{\ell \in [B]} \Pr[\mathbf{1}_{v,\text{type}=i,\text{dist}=\ell}] \cdot R_{v,\ell+2^i-1} \;-\; \lambda \cdot \sum_{v \in V} \Pr[X_v^i] \cdot s_i(v) \qquad (7.14) \\
&\geq \frac{\mathsf{Opt}(i)}{2} \;-\; \lambda \cdot 2^{i+1} \qquad (7.15)
\end{aligned}$$

Above, (7.14) is by interchanging summations and splitting the two terms from the previous expression. The first term in (7.15) comes from (7.13), and the second term comes from (7.12) and the fact that $q_v = \Pr[X_v^i] = \Pr[v \text{ visited as type-}i]$. ■

Now let AlgDO denote an α-approximation algorithm for the DeadlineOrient problem. We abuse notation and use $\mathsf{AlgDO}(\mathcal{I}_i(\lambda))$ to denote both the α-approximate solution on instance $\mathcal{I}_i(\lambda)$ as well as its value. We focus on the setting of λ defined thus:

$$\lambda_i^* \quad := \quad \max\left\{\lambda \,:\, \mathsf{AlgDO}(\mathcal{I}_i(\lambda)) \geq \frac{2^i \cdot \lambda}{\alpha}\right\} \qquad (7.16)$$

LEMMA 8. *For any $i \in [\log_2 B]$, we get $\lambda_i^* \geq \mathsf{Opt}(i)/2^{i+3}$, and hence $\mathsf{AlgDO}(\mathcal{I}_i(\lambda_i^*)) \geq \mathsf{Opt}(i)/8\alpha$.*

Proof. Consider the setting $\widehat{\lambda} = \mathsf{Opt}(i)/2^{i+3}$; by Lemma 7, the optimal solution to the DeadlineOrient instance $\mathcal{I}_i(\widehat{\lambda})$ has value at least $\mathsf{Opt}(i)/4 \geq 2^i \cdot \widehat{\lambda}$. Since AlgDO is an α-approximation algorithm for DeadlineOrient, it follows that $\mathsf{AlgDO}(\mathcal{I}_i(\widehat{\lambda})) \geq \mathsf{Opt}(\mathcal{I}_i(\widehat{\lambda}))/\alpha \geq 2^i \cdot \widehat{\lambda}/\alpha$. So $\lambda_i^* \geq \widehat{\lambda} \geq \mathsf{Opt}(i)/2^{i+3}$. ■

7.1.2. Obtaining CorrOrient solution from $\mathsf{AlgDO}(\lambda_i^*)$

It just remains to show that the solution output by the approximation algorithm for DeadlineOrient on the instance $\mathcal{I}_i(\lambda_i^*)$ yields a good non-adaptive solution to the original CorrOrient instance. Recall the notation for the deadline orienteering instance from Definition 3. Let $\sigma = \mathsf{AlgDO}(\lambda_i^*)$ be this solution—hence σ is a rooted path that visits some set $P \subseteq N_i$ of nodes within their respective deadlines. The algorithm below gives a subset $Q \subseteq P$ of nodes that we will visit in the non-adaptive solution; this is similar to the algorithm for KnapOrient in Section 3.

Algorithm 3 Algorithm A_i for CorrOrient given a solution for $\mathcal{I}_i(\lambda_i^*)$ characterized by a path P.

```
1: let $y = \left(\sum_{v \in P} s_i(v)\right)/2^i$.
2: partition vertices of $P$ into $c = \max(1, \lfloor 2y \rfloor)$ parts $P_1, \ldots, P_c$ with $\sum_{\langle v,\ell \rangle \in P_j} s_i(v) \leq 2^i, \forall 1 \leq j \leq c$.
3: set $Q \leftarrow P_k$ where $k = \arg\max_{j=1}^{c} \sum_{\langle v,\ell \rangle \in P_j} r_i(v,\ell)$.
4: for each $v \in V$, define $d_v := \min\{\ell : \langle v,\ell \rangle \in Q\}$.
5: let $\overline{Q} := \{v \in V : d_v < \infty\}$ be the vertices with at least one copy in $Q$.
6: sample vertices in $\overline{Q}$ independently w.p. $1/2$, and visit these sampled vertices in the order given by $P$.
```

At a high level, the algorithm partitions the vertices in P into groups, where each group obeys the size budget of 2^i in expectation. It then picks the most profitable group among these. The main issue with Q chosen in Step 3 is that it may include multiple copies of the same job: recall that the DeadlineOrient instance contains many co-located jobs for each job of the CorrOrient instance. But due to the way we constructed the sets I_i^v (based on Claim 8), we can simply pick the copy which corresponds to the earliest deadline, and by discarding all the other copies, we only lose out on a constant fraction of the reward $r_i(Q)$. Below, Claim 9 bounds the total (potential) reward of the set Q we select in Step 3. Next, Claim 10 shows that we do not lose much of the total reward by retaining only one copy (with deadline d_v) of each $v \in Q$ in Step 4. Finally, Claim 11 shows that for any vertex $v \in \overline{Q}$, with constant probability, Step 6 reaches v by time $d_v + 2^i - 1$ (which corresponds to obtaining reward $r_i(v, d_v)$).

CLAIM 9. *The reward* $r_i(Q) = \sum_{\langle v,\ell\rangle \in Q} r_i(v,\ell)$ *is at least* $\mathsf{Opt}(i)/(8\alpha)$.

Proof. By the choice of set Q in Step 3, $r_i(Q) \geq \frac{r_i(P)}{c}$ and,

$$\frac{r_i(P)}{c} \;\geq\; \frac{\lambda_i^* \cdot y2^i + \lambda_i^* \cdot 2^i/\alpha}{c} \;=\; \lambda_i^* 2^i \cdot \left(\frac{y+1/\alpha}{c}\right) \;\geq\; \lambda_i^* 2^i \cdot \min\left\{y + \frac{1}{\alpha}, \frac{1}{2} + \frac{1}{2\alpha y}\right\} \;\geq\; \frac{\lambda_i^* 2^i}{\alpha} \tag{7.17}$$

The first inequality in (7.17) is by the choice of λ_i^* (7.16), i.e.

$$\frac{2^i \cdot \lambda_i^*}{\alpha} \;\leq\; \mathsf{AlgDO}(\lambda_i^*) \;=\; \sum_{\langle v,\ell\rangle \in P} [r_i(v,\ell) - \lambda_i^* \cdot s_i(v)] \;=\; r_i(P) - \lambda_i^* \cdot s_i(P) \;=\; r_i(P) - \lambda_i^* \cdot y\, 2^i.$$

The second inequality in (7.17) is by $c \leq \max\{1, 2y\}$; and the last inequality uses $\alpha \geq 2$. To conclude we simply use Lemma 8 in the last expression of (7.17). ■

CLAIM 10. $\sum_{v\in\overline{Q}} R_{v,d_v+2^i-1} \geq \mathsf{Opt}(i)\,/\,16\alpha$.

Proof. For each $u \in \overline{Q}$, let $Q_u = Q \bigcap \{\langle u,\ell\rangle : \ell \in [I_u^i]\}$ denote all copies of u in Q. Now by the definition of d_u we have $\ell \geq d_u$ for all $\langle u,\ell\rangle \in Q_u$. So for any $u \in \overline{Q}$,

$$\sum_{\langle u,\ell\rangle \in Q_u} R_{u,\ell+2^i-1} \quad\leq\quad \sum_{\ell\in I_u^i : \ell \geq d_u} R_{u,\ell+2^i-1} \quad\leq\quad 2 \cdot R_{u,d_u+2^i-1}$$

Above, the last inequality uses the definition of I_u^i as given by Claim 8. Adding over all $u \in \overline{Q}$,

$$\sum_{u\in\overline{Q}} R_{u,d_u+2^i-1} \quad\geq\quad \frac{1}{2}\sum_{u\in\overline{Q}}\ \sum_{\langle u,\ell\rangle\in Q_u} R_{u,\ell+2^i-1} \quad=\quad \frac{1}{2}\sum_{\langle v,\ell\rangle\in Q} r_i(v,\ell) \quad\geq\quad \frac{\mathsf{Opt}(i)}{16\alpha}$$

Here, the last inequality uses Claim 9. This completes the proof. ■

CLAIM 11. *For any vertex* $v \in \overline{Q}$*, it holds that* $\Pr[A_i$ *reaches job* v *by time* $d_v + 2^i - 1] \geq \frac{1}{2}$.

Proof. We know that because P is a feasible solution for the $\mathsf{DeadlineOrient}$ instance, the distance traveled before reaching the copy $\langle v, d_v\rangle$ is at most d_v. Therefore in what remains, we show that with probability $1/2$, the total size of previous vertices is at most $2^i - 1$. To this end, let $\overline{U}$ denote the set of vertices sampled in Step 6. We then say that the *bad event* occurs if $\sum_{u\in\overline{U}\setminus v}\min(S_u, 2^i) \geq 2^i$. Indeed if $\sum_{u\in\overline{U}\setminus v}\min(S_u,2^i) < 2^i$, then we would reach v by time $d_v + 2^i - 1$.

We now bound the probability of the bad event. For this purpose, observe that

$$\mathbb{E}\left[\sum_{u\in\overline{U}\setminus v}\min(S_u,2^i)\right] \quad\leq\quad \frac{1}{2}\sum_{u\in\overline{Q}}\mathbb{E}\left[\min(S_u,2^i)\right] \quad=\quad \frac{1}{2}\sum_{u\in\overline{Q}} s_i(u) \quad\leq\quad 2^{i-1}.$$

The first inequality is by linearity of expectation and the fact that each $u \in \overline{Q}$ is sampled into $\overline{U}$ with probability $1/2$. The last inequality uses the size bound on $\overline{Q}$ by the partitioning in Step 2. Hence, the probability of the bad event is at most $1/2$ by Markov's inequality. ■

LEMMA 9. *The expected reward of the algorithm* A_i *is at least* $\mathsf{Opt}(i)/64\alpha$.

Proof. We know from Claim 11 that for each vertex $v \in \overline{Q}$, algorithm A_i reaches v by time $d_v + 2^i - 1$ with probability at least $1/2$. Moreover, this event is determined by the outcomes at vertices $\overline{Q}\setminus\{v\}$. So, conditioned on this event, v is sampled with probability $1/2$. Therefore, the expected reward collected from v is at least $(1/4)R_{v,d_v+2^i-1}$. The proof is complete by using linearity of expectation and then Claim 10. ■
Since the final algorithm for $\mathsf{CorrOrient}$ takes the best solution over all types $i \in [\log_2 B]$, Lemma 9 implies an $O(\log n \cdot \log B)$-approximation ratio. This proves the first part of Theorem 4.

7.2. Evidence of hardness for CorrOrient

Our approximation algorithm for CorrOrient can be viewed as a reduction to DeadlineOrient, at the loss of an $O(\log B)$ factor. We now provide a reduction in the reverse direction: namely, a β-approximation algorithm for CorrOrient implies a $(\beta - o(1))$-approximation algorithm for DeadlineOrient. In particular this means that a sub-logarithmic approximation ratio for CorrOrient would also improve the best known approximation ratio for DeadlineOrient.

Consider any instance $\mathcal{I}$ of DeadlineOrient on metric (V,d) with root $\rho \in V$ and deadlines $\{t_v\}_{v\in V}$; the goal is to compute a path originating at ρ that visits the maximum number of vertices before their deadlines. We now define an instance $\mathcal{J}$ of CorrOrient on the same metric (V,d) with root ρ. Let $B := 1 + \max_{v\in V} t_v$. Fix parameter $0 < p \ll \frac{1}{n^2}$. The job at each $v \in V$ has the following distribution: size $B - t_v$ and reward $1/p$ with probability p; and size zero and reward 0 with probability $1-p$. To complete the reduction from DeadlineOrient to CorrOrient we will show that:

$$(1-o(1)) \cdot \mathsf{Opt}(\mathcal{I}) \quad \le \quad \mathsf{Opt}(\mathcal{J}) \quad \le \quad (1+o(1)) \cdot \mathsf{Opt}(\mathcal{I}).$$

Let τ be any solution to $\mathcal{I}$ that visits subset $S \subseteq V$ of vertices within their deadline; so the objective value of τ is $|S|$. This also corresponds to a (non-adaptive) solution to $\mathcal{J}$. For any vertex $v \in S$, the probability that zero processing time has been spent prior to v is at least $(1-p)^n$. In this case, the start time of job v is at most t_v (recall that τ visits $v \in S$ by time t_v) and hence the conditional expected reward from v is $p \cdot \frac{1}{p} = 1$ (since v has size $B - t_v$ and reward $1/p$ with probability p). It follows that the expected reward of τ as a solution to $\mathcal{J}$ is at least $\sum_{v\in S}(1-p)^n \ge |S| \cdot (1-np) = (1-o(1)) \cdot |S|$. Choosing τ to be the optimal solution to $\mathcal{I}$, we have $(1-o(1)) \cdot \mathsf{Opt}(\mathcal{I}) \le \mathsf{Opt}(\mathcal{J})$.

Consider now any adaptive policy σ for $\mathcal{J}$, with expected reward $R(\sigma)$. Define path σ_0 as one starting from the root of σ that always follows the branch corresponding to size zero instantiation. Consider σ_0 as a feasible solution to the DeadlineOrient instance $\mathcal{I}$. Let $S_0 \subseteq V$ denote the vertices on path σ_0 that are visited prior to their respective deadlines. Clearly $\mathsf{Opt}(\mathcal{I}) \ge |S_0|$. When policy σ is run, every size zero instantiation gives zero reward; so if positive reward is obtained then the sample path must diverge from σ_0. Moreover, if there is positive reward, the sample path must have positive size instantiation at some vertex in S_0: this is because a positive size instantiation at any $(V \setminus S_0)$-vertex along σ_0 would violate the bound B (by definition of sizes and set S_0). Hence,

$$\Pr[\sigma \text{ gets positive reward}] \quad \le \quad p \cdot |S_0| \tag{7.18}$$

Moreover, since the reward is always an integral multiple of $1/p$,

$$\begin{aligned} R(\sigma) &= \frac{1}{p} \cdot \sum_{i=1}^{n} \Pr[\sigma \text{ gets reward at least } i/p] \\ &= \frac{1}{p} \cdot \Pr[\sigma \text{ gets positive reward}] \; + \; \frac{1}{p} \cdot \sum_{i=2}^{n} \Pr[\sigma \text{ gets reward at least } i/p] \end{aligned} \tag{7.19}$$

Furthermore, for any $i \ge 2$, we have:

$$\Pr[\sigma \text{ gets reward at least } i/p] \;\le\; \Pr[\text{at least } i \text{ jobs instantiate to positive size}] \;\le\; \binom{n}{i} \cdot p^i \le (np)^i.$$

It follows that the second term in (7.19) can be upper bounded by $\frac{1}{p} \cdot \sum_{i=2}^{n}(np)^i \le 2n^2p$ since $np < \frac{1}{2}$. Combining this with (7.18) and (7.19), we obtain that $R(\sigma) \le |S_0| + 2n^2p = |S_0| + o(1)$ since $n^2p \ll 1$. Since this holds for any adaptive policy σ for $\mathcal{J}$, we get $\mathsf{Opt}(\mathcal{I}) \ge (1-o(1)) \cdot \mathsf{Opt}(\mathcal{J})$.

This proves the second part of Theorem 4.

8. Stochastic Orienteering with Cancelations

Throughout the paper, we considered the non-preemptive model for processing jobs. In this section, we observe that those results also extend to a different model where a policy can cancel/abort jobs during processing. However once a job is aborted, it can not be attempted again. Even in the special case of stochastic knapsack, there are instances that demonstrate an arbitrarily large gap in the expected reward for policies which can cancel and those that can not [22]. Our algorithms for usual StocOrient extend easily to StocOrient with cancelation, with the same guarantees: $O(\log\log B)$ for the uncorrelated version and $O(\log n \log B)$ for the correlated version.

The main idea is to modify the deterministic subproblems slightly, i.e. KnapOrient for uncorrelated StocOrient and DeadlineOrient for the correlated case. Specifically, we create up to B co-located copies of each job v, each of which corresponds to canceling the job v after a certain time t of processing it (the size and reward of copy $\langle v, t\rangle$ are defined to reflect this). It is easy to see that any adaptive optimal solution, when it visits a vertex in fact just plays *some copy* of it (which copy might depend on the history of the sample path taken to reach this vertex). So exactly as before, we can find a good deterministic solution with suitably large reward (this is the KnapOrient problem for the uncorrelated case and the DeadlineOrient problem for the correlated case). Now the only issue is when we translate back from the deterministic instance to a non-adaptive solution for the StocOrient instance: the deterministic solution might collect reward from multiple copies of the same job. We can bound this gap by again using the geometric scaling idea (Claim 8), i.e., if there are two copies of roughly the same reward, we only keep the one with the earlier cancelation time. This way, we can ensure that for all copies of a particular job, the rewards are geometrically decreasing. Now, even if the deterministic solution collects reward from multiple copies of a job, we can simply use the one amongst them with highest reward.

9. Conclusion

In this paper we studied stochastic variants of the orienteering problem, where jobs with random processing times are located at vertices in a metric space. We obtained an $O(\log\log B)$-approximation algorithm and adaptivity gap for the basic stochastic orienteering problem. Very recently, Bansal and Nagarajan [4] showed an $\Omega(\sqrt{\log\log B})$ lower bound on the adaptivity gap for this problem. Closing this gap remains the main open question. For the correlated stochastic orienteering problem, where job rewards are also random and correlated with processing times, we obtained an $O(\log n \cdot \log B)$-approximation algorithm. We also showed that this problem is at least as hard to approximate as the deadline orienteering problem, for which the best approximation ratio known is $O(\log n)$. Improving the approximation ratio for correlated stochastic orienteering is another interesting open question.

Acknowledgment: Anupam Gupta's research was partly supported by NSF awards CCF-0964474 and CCF-1016799. Ravishankar Krishnaswamy's research was partly supported by NSF awards CCF-0964474 and CCF-1016799, and an IBM Graduate Fellowship. R. Ravi's research was partly supported by NSF awards CCF-1143998 and CCF-1218382. We thank an anonymous SODA 2012 referee for raising the question of stochastic orienteering on directed metrics, that led to the results in Section 6.

References

[1] Micah Adler and Brent Heeringa, *Approximating optimal binary decision trees*, Algorithmica **62** (2012), no. 3-4, 1112–1121.

[2] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson, *Approximation algorithms for deadline-TSP and vehicle routing with time-windows*, ACM Symposium on Theory of Computing (STOC), 2004, pp. 166–174.

[3] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra, *When LP is the cure for your matching woes: Improved bounds for stochastic matchings*, Algorithmica **63** (2012), no. 4, 733–762.

[4] Nikhil Bansal and Viswanath Nagaraan, *On the adaptivity gap of stochastic orienteering*, To appear in the Conference on Integer Programming and Combinatorial Optimization (IPCO), 2014.

[5] Dimitris Bertsimas and Jose Nino-Mora, *Conservation laws, extended polymatroids and multiarmed bandit problems; a polyhedral approach to indexable systems*, Mathematics of Operations Research **21** (1996), no. 2, 257–306.

[6] Anand Bhalgat, *A* $(2+\epsilon)$*-approximation algorithm for the stochastic knapsack problem*, At `http://www.seas.upenn.edu/~bhalgat/2-approx-stochastic.pdf`, 2011.

[7] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna, *Improved approximation results for stochastic knapsack problems*, ACM-SIAM Symposium on Discrete Algorithms (SODA), 2011, pp. 1647–1665.

[8] Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff, *Approximation algorithms for orienteering and discounted-reward TSP*, SIAM J. Comput. **37** (2007), no. 2, 653–670.

[9] Gruia Călinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák, *Maximizing a monotone submodular function subject to a matroid constraint*, SIAM J. Comput. **40** (2011), no. 6, 1740–1766.

[10] Ann Melissa Campbell, Michel Gendreau, and Barrett W. Thomas, *The orienteering problem with stochastic travel and service times*, Annals OR **186** (2011), no. 1, 61–81.

[11] Chandra Chekuri and Sanjeev Khanna, *On multidimensional packing problems*, SIAM J. Comput. **33** (2004), no. 4, 837–851.

[12] Chandra Chekuri, Nitish Korula, and Martin Pál, *Improved algorithms for orienteering and related problems*, ACM Transactions on Algorithms **8** (2012), no. 3, 23.

[13] Chandra Chekuri and Amit Kumar, *Maximum coverage problem with group budget constraints and applications*, Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX), 2004, pp. 72–83.

[14] Chandra Chekuri and Martin Pál, *A recursive greedy algorithm for walks in directed graphs*, IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 245–253.

[15] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen, *Dependent randomized rounding via exchange properties of combinatorial structures*, IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 575–584.

[16] Ke Chen and Sariel Har-Peled, *The Euclidean orienteering problem revisited*, SIAM J. Comput. **38** (2008), no. 1, 385–397. MR 2399532 (2009j:68195)

[17] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra, *Approximating matches made in heaven*, International Colloquium on Automata, Languages and Programming (ICALP), 2009, pp. 266–278.

[18] Brian C. Dean, Michel X. Goemans, and Jan Vondrák, *Approximating the stochastic knapsack problem: The benefit of adaptivity*, Math. Oper. Res. **33** (2008), no. 4, 945–964.

[19] David A. Freedman, *On tail probabilities for martingales*, Annals of Probability **3** (1975), 100–118.

[20] Sudipto Guha and Kamesh Munagala, *Approximation algorithms for budgeted learning problems*, ACM Symposium on Theory of Computing (STOC), 2007, pp. 104–113.

[21] ______, *Multi-armed bandits with metric switching costs*, International Colloquium on Automata, Languages and Programming (ICALP), 2009, pp. 496–507.

[22] Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi, *Approximation algorithms for correlated knapsacks and non-martingale bandits*, IEEE Symposium on Foundations of Computer Science (FOCS), 2011, pp. 827–836.

[23] Anupam Gupta, Viswanath Nagarajan, and R. Ravi, *Approximation algorithms for optimal decision trees and adaptive TSP problems*, International Colloquium on Automata, Languages and Programming (ICALP), 2010, pp. 690–701.

[24] ______, *Robust and maxmin optimization under matroid and knapsack uncertainty sets*, CoRR **abs/1012.4962** (2010).

[25] E.G. Coffman Jr. and I. Mitrani, *A characterization of waiting time performance realizable by single-server queues*, Operations Research **28** (1980), no. 3, 810–821.

[26] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom, *On an optimal split tree problem*, Workshop on Algorithms and Data Structures (WADS), 1999, pp. 157–168.

[27] Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz, *Approximation in stochastic scheduling: the power of LP-based priority policies*, J. ACM **46** (1999), no. 6, 924–942.

[28] Viswanath Nagarajan and R. Ravi, *The directed orienteering problem*, Algorithmica **60** (2011), no. 4, 1017–1030.

[29] Tong Zhang, *Data dependent concentration bounds for sequential prediction algorithms*, Conference on Learning Theory (COLT), 2005, pp. 173–187.