Approximation Algorithms for Stochastic Orienteering

Anupam Gupta[*] Ravishankar Krishnaswamy[†] Viswanath Nagarajan[‡] R. Ravi[§]

Abstract

In the Stochastic Orienteering problem, we are given a metric, where each node also has a job located there with some deterministic reward and a *random* size. (Think of the jobs as being chores one needs to run, and the sizes as the amount of time it takes to do the chore.) The goal is to adaptively decide which nodes to visit to maximize total expected reward, subject to the constraint that the total distance traveled plus the total size of jobs processed is at most a given budget of B. (I.e., we get reward for all those chores we finish by the end of the day). The (random) size of a job is not known until it is completely processed. Hence the problem combines aspects of both the stochastic knapsack problem with uncertain item sizes and the deterministic orienteering problem of using a limited travel time to maximize gathered rewards located at nodes.

In this paper, we present a constant-factor approximation algorithm for the best non-adaptive policy for the Stochastic Orienteering problem. We also show a small adaptivity gap—i.e., the existence of a non-adaptive policy whose reward is at least an $\Omega(1/\log\log B)$ fraction of the optimal expected reward—and hence we also get an $O(\log\log B)$-approximation algorithm for the adaptive problem. Finally we address the case when the node rewards are also random and could be correlated with the waiting time, and give a non-adaptive policy which is an $O(\log n \log B)$-approximation to the best adaptive policy on n-node metrics with budget B.

1 Introduction

Consider the following problem: you start your day at home with a set of chores 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 chores in (say, you have from 9am until 5pm, when all these shops close). Each successfully completed chore/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 doing 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.[1] (You may cancel processing the job j prematurely, but in that case you don't get any reward, and you are barred from trying j again.) The goal is now a natural one: given the metric (V, d), the starting point ρ, the time budget B, and the probability distributions for all the jobs, give a strategy for traveling around and doing the jobs that maximizes the expected reward accrued.

The case when all the sizes are deterministic (i.e., $S_j = s_j$ with probability 1) is the orienteering problem, for which we now know a $(2+\epsilon)$-approximation algorithm [BCK+07, CKP08]. Another special case, where all the chores are located at the start node, but the sizes are random, is the stochastic knapsack problem, which also admits a $(2+\epsilon)$-approximation algorithm [DGV08, Bha11]. However, the stochastic orienteering problem above, which combines aspects of both these problems, seems to have been hitherto unexplored in the approximation algorithms literature.

It is known that even for stochastic knapsack, an optimal adaptive strategy may be exponential-sized (and finding the best strategy is PSPACE-hard) [DGV08]. So another set of interesting questions is to bound the

[*]Department of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Research was partly supported by NSF awards CCF-0964474 and CCF-1016799.

[†]Department of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213. Research was partly supported by NSF awards CCF-0964474 and CCF-1016799, and an IBM Graduate Fellowship.

[‡]IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA.

[§]Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213.

[1]To clarify: 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 $(S_j \mid S_j > t)$.

"adaptivity gap", and to find *non-adaptive* solutions whose expected reward is close to that of the best adaptive solution. A non-adaptive solution for stochastic orienteering is simply a tour 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.

One natural algorithm for stochastic orienteering is to replace each random job j by a deterministic job of size $E[S_j]$, and use an orienteering algorithm to find the tour of maximum reward in this deterministic instance.[2] Such an approach was taken by [DGV08] for stochastic knapsack: they showed a constant adaptivity gap, and constant-factor approximation via precisely this idea. However, for stochastic orienteering this gives only an $O(\log B)$ approximation, and indeed there are examples where we get only an $\widetilde{\Omega}(\log B)$ fraction of the optimal expected reward. (See Section 4 for a more detailed discussion.) In this paper we show we can do much better than this logarithmic approximation:

THEOREM 1.1. *There is an $O(\log \log B)$-approximation algorithm for the stochastic orienteering problem.*

Indeed, our proof proceeds by first showing the following structure theorem which bounds the adaptivity gap:

THEOREM 1.2. *Given an instance of the stochastic orienteering problem, then*

- *either there exists a single job which gives an $\Omega(\log \log B)$ fraction of the optimal reward, or*
- *there exists a value W^* such that the optimal non-adaptive tour which spends at most W^* time waiting and $B - W^*$ time traveling, gets an $\Omega(\log \log B)$ fraction of the optimal reward.*

Note that naïvely we would expect only a logarithmic fraction of the reward, but the structure theorem shows we can do better. Indeed, this theorem is the technical heart of the paper, and is proved via a martingale argument of independent interest. Since the above theorem shows the existence of a non-adaptive solution close to the best adaptive solution, we can combine it with the following result to prove Theorem 1.1.

THEOREM 1.3. *There exists a constant-factor approximation algorithm to the optimal non-adaptive policy for stochastic orienteering.*

Note that if we could show an existential proof of a constant adaptivity gap (which we conjecture to be true), the above approximation for non-adaptive problems that we show immediately implies an $O(1)$-approximation algorithm for the adaptive problem too.

Our second set of results are for a variant of the problem, one where both the rewards and the job sizes are random and correlated with each other. For this correlated problem, we show the following results:

THEOREM 1.4. *There is a polynomial-time algorithm that outputs a non-adaptive solution for correlated stochastic orienteering, which is an $O(\log n \log B)$-approximation to the best adaptive solution. Moreover, the correlated problem is at least as hard as the orienteering-with-deadlines problem.*

Recall that we only know an $O(\log n)$ approximation for the orienteering-with-deadlines problem [BBCM04].

Our Techniques: Most of the previous adaptivity gaps have been shown by considering some linear programming relaxation that captures the optimal adaptive strategies, and then showing how to round a fractional LP solution to get a non-adaptive strategy. But since we do not know a good relaxation for even the deterministic orienteering problem, this approach seems difficult to take directly. So to show our adaptivity gap results, we are forced to argue directly about the optimal adaptive strategy: we use martingale arguments to show the existence of a path (a.k.a. non-adaptive strategy) within this tree which gives a large reward. We can then use algorithms for the non-adaptive settings, which are based on reductions to orienteering (with an additional knapsack constraint).

Roadmap: The rest of the paper follows the outline above. We begin with some definitions in Section 2, and then define and give an algorithm for the *knapsack orienteering problem* which will be a crucial sub-routine in all our algorithms in Section 3. Next we motivate our non-adaptive algorithm by discussing a few straw-man solutions and the traps they fall into, in Section 4. We then state and prove our constant-factor non-adaptive algorithm in Section 5, which naturally leads us to our main result in Section 9, the proof of the $O(\log \log B)$-adaptivity gap for StocOrient. Finally, we present in Section 13 the poly-logarithmic approximation for the problem where the rewards and sizes are correlated with each other. For all these results we assume that we are not allowed to cancel jobs once we begin working on them: in Section 18 we show how to discharge this assumption.

[2]Actually, one should really replace the stochastic job with a deterministic one of size $E[\min(S_j, B - d(\rho, j))]$ and reward $r_j \cdot \Pr[S_j + d(\rho, j) \leq B]$, it is very easy to fool the algorithm otherwise.

1.1 Related Work The orienteering problem is known to be APX-hard, and the first constant-factor approximation was due to Blum et al. [BCK$^+$07]. Their factor of 4 was improved by [BBCM04] and ultimately by [CKP08] to $(2+\epsilon)$ for every $\epsilon > 0$. (The paper [BBCM04] also considers the problem with deadlines and time-windows; see also [CK04, CP05].) There is a PTAS for low-dimensional Euclidean space [CHP08]. 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. [CGT11].

The stochastic knapsack problem [DGV08] is a special case of this problem, where all the tasks are located at the root ρ itself. Constant factor approximation algorithms for the basic problem were given by [DGV08, BGK11, Bha11], and extensions where the rewards and sizes are correlated were studied in [GKMR11]. Most of these papers proceed via writing LP relaxations that capture the optimal adaptive policy; extending these to our problem faces the barrier that the orienteering problem is not known to have a good natural LP relaxation.

Another very related body of work is on budgeted learning. Specifically, in the work of Guha and Munagala [GM09], there is a collection of Markov chains spread around in a metric, each state of each chain having an associated reward. When the player is at a Markov chain at j, she can advance the chain one step every unit of time. If she spends at most L time units traveling, and at most C time units advancing the Markov chains, how can she maximize some function (say the sum or the max) of rewards of the final states in expectation? [GM09] give an elegant constant factor approximation to 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. However, it is unclear how to directly use such two-budget approaches to get $O(1)$-factors for our one-budget problem without incurring an $O(\log B)$-factor in the approximation ratio.

Finally, while most of the work on giving approximations to adaptive problems has proceeded by using LP relaxations to capture the optimal adaptive strategies—and then often rounding them to get non-adaptive strategies, thereby also proving adaptivity gaps [GM07, DGV08], there are some exceptions. In particular, papers on adaptive stochastic matchings [CIK$^+$09], on stochastic knapsack [BGK11], on building decision trees [KPB99, AH08, GNR10], all have had to reason about the optimal adaptive policies directly. We hope that our martingale-based analysis will add to the set of tools used for such arguments.

2 Definitions and Notation

An instance of stochastic orienteering 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 unique stochastic job, which we also call v. For the first part of the paper, each job v has a fixed reward $r_v \in \mathbb{Z}_{\geq 0}$, and a random processing time/size S_v, which is distributed according to a known but arbitrary probability distribution $\pi_v : \mathbb{R}^+ \to [0, 1]$. (In the introduction, this was the amount we had to wait in queue before receiving the reward for job v.) We are also given a starting "root" vertex ρ, and a budget B on the total time available. Without loss of generality, we assume that all distances are integer values.

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., total travel time plus processing time, is at most B), and we can then move to the next job. There are two variants of the basic problem: in the basic variant, we are not allowed to cancel any job that we begin processing (i.e., we cannot leave the queue once we join it). In the version with cancellations, we can cancel any job at any time without receiving any reward, *and we are not allowed to attempt this job again in the future.* Our results for both versions will have similar approximation guarantees—for most of the paper, we focus on the basic version, and in Section 18 we show how to reduce the more general version with cancellations to the basic one.

Note that any strategy consists of a decision tree where each state depends on which previous jobs were processed, and what information we got about their sizes. Now the goal is to devise a strategy which, starting at the root ρ, must decide (possibly adaptively) which jobs to travel to and process, so as to maximize the expected sum of rewards of jobs successfully completed before the total time (travel and processing) reaches the threshold of B.

In the second part of the paper, we consider the setting of *correlated rewards and sizes*: in this model, the job sizes and rewards are both random, and are correlated with each other. (Recall that the stochastic knapsack version of this problem also admits a constant factor approximation [GKMR11]).

We are interested in both adaptive and non-adaptive strategies, and in particular, want to bound the ratio between the performance of the best adaptive and best non-adaptive strategies. An *adaptive strategy* 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 strategy*, on the other hand, is a path P starting at ρ; we just traverse this path, executing the jobs we encounter, until the total (random) size of the jobs and the distance traveled exceeds B. (A *randomized non-adaptive strategy* 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.)

Finally, we use $[m]$ to denote the set $\{0, 1, \ldots, m\}$.

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 ($\mathsf{KnapOrient}$), we are given 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 now also a "size" $\widehat{s}_v$. A feasible solution is a path P of 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 feasible solution of maximum reward $\sum_{v \in P} \widehat{r}_v$.

Theorem 3.1. *There is a polynomial time $O(1)$-approximation* AlgKO *for the* $\mathsf{KnapOrient}$ *problem.*

The idea of the proof is to push the knapsack constraint into the objective function by considering *Lagrangian relaxation* of the knapsack constraint; we remark that such an approach was also taken in [GM09] for a related problem. This way we alter the rewards of items while still optimizing over the set of feasible orienteering tours. 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 and orienteering constraints.

Proof. 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 solution to 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 [CKP08]. 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 $\mathsf{Alg}_o(\lambda) \geq \frac{1}{\alpha}(\mathsf{Opt} - \lambda W) = \frac{\mathsf{Opt}}{2\alpha}$. Thus $\lambda^* \geq \frac{\mathsf{Opt}}{2W}$.

Let σ denote the ρ-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' as our approximate solution to the $\mathsf{KnapOrient}$ instance. It remains to bound the reward we obtain.

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

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 $\widehat{r}(\sigma) \geq \lambda^* \cdot \widehat{s}(\sigma) + \frac{\lambda^* \cdot W}{\alpha} = \lambda^* y W + \frac{\lambda^* W}{\alpha}$ 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 $\widehat{r}(\sigma') \geq \frac{\mathsf{Opt}}{2\alpha}$, thus giving us the desired approximation guarantee.

As an aside, this Lagrangean approach can give a constant-factor approximation for a two-budget version of stochastic orienteering, but it is unclear how to extend it 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 good Lagrange multiplier.

4 A Strawman Approach: Reduction to Deterministic Orienteering

As mentioned in the introduction, a natural idea for $\mathsf{StocOrient}$ is to replace random jobs by deterministic ones with size equal to the expected size $E[S_v]$, 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 strategy for the original $\mathsf{StocOrient}$ instance with expected reward $\Omega(R)$. Indeed, suppose the path P spends time L traveling and W waiting on 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 large reward. 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 have 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 $E[\min(S_j, B)]$ where S_j is the random size of job j. (Of course, we also have to set the reward to be $r_j \Pr[S_j \leq B]$ 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 [DGV08] to get an $O(1)$-approximation and adaptivity gap.

But for $\mathsf{StocOrient}$, 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 get reward from only one job, while the $\mathsf{StocOrient}$ 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 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 would work for the aforementioned example, the following example shows that the deterministic instance gets only an $O(\frac{\log \log B}{\log B})$ fraction of the optimal stochastic reward.

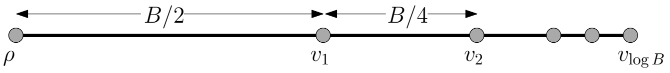

Figure 4.1: Bad example for replacing by expectations.

Consider $n = \log B$ jobs on a line as in Figure 4. For $i = 1, 2, \ldots, \log B$, the i^{th} job is at distance $B(1 - 1/2^i)$, and it takes on sizes $B/2^i$ 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: with probability $(1-p)^{\log B} = 1/e$, all the jobs instantiate to size 0 and we will accrue reward $\log B$.

However, suppose we replace each job by its expected size $\mu_i = B/(2^i \log B)$, and try to pick jobs so that 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), and 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

Step 1: Construct a suitable instance $\mathcal{I}_{ko}$ of *Knapsack Orienteering* ($\mathsf{KnapOrient}$), with the guarantee that the optimal profit from this $\mathsf{KnapOrient}$ instance $\mathcal{I}_{ko}$ is $\Omega(\mathsf{Opt}/\log\log B)$.
Step 2: Use Theorem 3.1 on $\mathcal{I}_{ko}$ to find a path P with reward $\Omega(\mathsf{Opt}/\log\log B)$.
Step 3: Follow this solution P and process the jobs in the order P visits them.

Figure 4.2: High Level Overview

jobs would take. (In fact, since their sizes decrease geometrically, we can indeed get all but a constant number of these jobs.)

This shows that replacing jobs in a $\mathsf{StocOrient}$ 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.

4.1 Why the above approaches failed, and our approach The reason why the determinization 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 spend processing jobs (up to a factor of 2) which gets the largest profit, and use that as the truncation threshold. Such an approach seems like it should lose a $\Omega(\log B)$ fraction of the reward. *Surprisingly, we show that this simple algorithm actually gives a much better reward: it gets a constant factor of the reward of a non-adaptive optimum, and an $O(\log\log B)$-approximation when compared to the adaptive optimum!*

At a high level, our approach is to solve this problem in three steps. Suppose we start off with an instance $\mathcal{I}_{so}$ of $\mathsf{StocOrient}$ with optimal (adaptive) solution having expected reward Opt. The high level overview of all our algorithms is described in Figure 4.2. However, there are many details to be addressed, and we flesh out the details of this algorithm over the next couple of sections. Before we proceed, it would be useful to recall our definition of the $\mathsf{KnapOrient}$ problem from Section 3.

5 The Algorithm for Non-Adaptive Stochastic Orienteering

We first consider the *non-adaptive* $\mathsf{StocOrient}$ problem, and present an $O(1)$-approximation algorithm. This will prove Theorem 1.3, and also serve as a warmup for the more involved analysis in 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 $r_v \in \mathbb{Z}_{\geq 0}$ 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 strategy is an ordering π of the vertices (starting with ρ), which corresponds to visiting vertices (and processing the respective jobs) in the order π. The goal in the non-adaptive $\mathsf{StocOrient}$ 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).

Definition 6. (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.*

Definition 7. (Valid $\mathsf{KnapOrient}$ Instances) *Now given an instance $\mathcal{I}_{so}$ of* $\mathsf{StocOrient}$, *and a value $W \leq B$, we define the following* valid $\mathsf{KnapOrient}$ instance *with parameter W to be $\mathcal{I}_{ko}(W) := \mathsf{KnapOrient}(V, d, \{(\widehat{s}_u, \widehat{r}_u) : \forall u \in V\}, L, W, \rho)$ as follows.*

(i) Set $L = B - W$,

(ii) For all $u \in V$, define its deterministic size $\widehat{s}_u = \mu_u(W/2)$.

(iii) For all $u \in V$, set its reward to be $\widehat{r}_u = r_u$ if $\Pr_{S_u\sim\pi_u}[S_u > W/2] \leq \frac{1}{2}$, and to 0 otherwise.

Recall that $\mathsf{KnapOrient}$ has deterministic sizes but two budgets L and W, and that AlgKO is an $O(1)$-approximation algorithm for it. The algorithm for non-adaptive $\mathsf{StocOrient}$ now proceeds as given below in Algorithm 1. Note that the algorithm gives a randomized non-adaptive policy: it chooses a random (sub)path to follow, and just visits all the jobs on that path until time runs out.

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 tour to $v$.
3: w.p. $1/2$, just visit the vertex $v$ with the highest $R_v$ and exit.
4: for $W = B, B/2, B/4, \ldots, B/2^{\lceil \log B \rceil}, 0$ do
5:     let $i = \log(B/W)$ if $W \neq 0$, otherwise let $i = \lceil \log B \rceil + 1$.
6:     let $P_i$ be the path returned by AlgKO on the valid KnapOrient instance $\mathcal{I}_{ko}(W)$.
7:     let $\widehat{R}_i$ be the reward of this KnapOrient solution $P_i$.
8: let $P_{i^*}$ be the solution among $\{P_i\}_{i \in [\log B + 1]}$ with maximum reward $\widehat{R}_i$.
9: sample each vertex in $P_{i^*}$ independently w.p. $1/4$, and visit these sampled vertices in order given by $P_{i^*}$.
```

7.1 The Analysis The analysis consists of two conceptual parts. Let Opt denote both the optimal non-adaptive strategy, and its total expected reward.

- We first show that either just a single-vertex tour has reward $\Omega(\mathsf{Opt})$, or one of the *valid* KnapOrient instances for some W of the form $B/2^i$ (or $W = 0$) has reward $\Omega(\mathsf{Opt})$.
- If a single-vertex tour is good, we'll visit it with 50% chance and be done. If not, since AlgKO is an $O(1)$-approximation algorithm for KnapOrient, the solution P_{i^*} selected in Step 8 must have reward $\widehat{R}_{i^*} = \Omega(\mathsf{Opt})$, albeit only as a KnapOrient solution. So we finally show that a solution for *any valid* KnapOrient *instance* with reward $\widehat{R}$ can be converted into a non-adaptive strategy for StocOrient with reward $\Omega(\widehat{R})$.

The following theorem summarizes the first part of this argument:

THEOREM 7.1. (STRUCTURE THEOREM 1) *Given an instance $\mathcal{I}_{so}$ for which an optimal non-adaptive strategy has an expected reward of* Opt*, either there is a single-vertex tour with expected reward* $\Omega(\mathsf{Opt})$*, or there exists $W = B/2^i$ for some $i \in \mathbb{Z}_{\ge 0}$ (or $W = 0$) for which the valid* KnapOrient *instance $\mathcal{I}_{ko}(W)$ has reward* $\Omega(\mathsf{Opt})$.

For now, let us assume Theorem 7.1 and complete the proof of Theorem 1.3; we'll then prove Theorem 7.1 in the following subsection. Firstly, note that if there is a single-vertex tour with expected reward $\Omega(\mathsf{Opt})$, then our algorithm would get this reward in Step 3 with probability 1/2. Therefore, for the remainder of this section, assume that no single-vertex tour is good; hence, by Theorem 7.1, the path P_{i^*} with the highest reward identified in Step 8 has reward $\Omega(\mathsf{Opt})$.

For brevity, denote P_{i^*} as P^*, and let $W^* = 2^{i^*}$ be the associated value of W. If $P^* = \langle \rho, v_1, v_2, \ldots, v_l \rangle$. Hence $\widehat{R}_{i^*} = \sum_{u \in P^*} \widehat{r}_u$. Since P^* is feasible for $\mathcal{I}_{ko}(W^*)$, we know that **(a)** the length of the path P^* is at most $B - W^*$, and **(b)** $\sum_{u \in P^*} \widehat{s}_u \le W^*$. And recall that by the way we defined the valid KnapOrient instances (Definition 7), $\widehat{s}_u = \mu_u\,(W^*/2)$ for all u, and $\widehat{r}_u = r_u$ if $\Pr_{S_u \sim \pi_u}[S_u \ge W^*/2] \le \frac{1}{2}$ and 0 otherwise.

LEMMA 7.1. *For any vertex $v \in P^*$ such that $\widehat{r}_v > 0$, the probability of successfully reaching v and finishing the job at v before violating the budget is at least* $1/16$.

Proof. The proof is an easy application of Markov's inequality. Indeed, consider the sum

$$\textstyle\sum_{u \prec_{P^*} v} \mathbf{1}_u \cdot \min(S_u, W^*/2) \qquad (7.2)$$

over the vertices on P^* that precede v, where $\mathbf{1}_u$ is the indicator random variable for whether u is sampled in Step 9. If this sum is at most $W^*/2$ (the "good" event), then the total time spent processing jobs up to (but not including) v is at most $W^*/2$; And indeed, since we sample each vertex in P^* independently with probability 1/4, and since $\sum_{x \in P^*} \mathbb{E}[\min(S_x, W^*/2)] = \sum_{x \in P^*} \widehat{s}_x \le W$, the expected value of the sum (7.2) is at most $W^*/4$. By Markov's inequality, the "good event" happens with probability at least 1/2.

In this case job v is sampled independently with probability 1/4, and if chosen, it finishes processing within $W^*/2$ with probability at least 1/2 (because $\widehat{r}_{v_i} > 0$ and therefore $\Pr_{S_{v_i} \sim \pi_{v_i}}[S_{v_i} > W^*/2] \le \frac{1}{2}$ by definition). Finally, since the total time spent traveling on the entire path P^* is at most $B - W^*$, we can combine the above events to get that the job successfully completes with probability at least 1/16.

Now, linearity of expectation implies that using P_{i^*} with sub-sampling, as in Step 9, gives us an expected reward of at least $\widehat{R}_{i^*}/16$. Since $\widehat{R}_{i^*}$ is $\Omega(\mathsf{Opt})$, this completes the proof of Theorem 1.3.

7.1.1 Proof of Theorem 7.1 The proof proceeds as follows: assuming that no single-vertex tour has reward $\Omega(\mathsf{Opt})$, we want to show that one of the valid $\mathsf{KnapOrient}$ instances must have large reward. We first note that this assumption lets us make the following structural observation about Opt.

OBSERVATION 8. *If no single-vertex tour has reward $\Omega(\mathsf{Opt})$, we can skip all visits to vertices u such that $Pr_{S_u \sim \pi_u}[S_u > (B - d(\rho, u))] \geq 1/2$, and still preserve a constant factor of the total reward.*

Proof. Say that a vertex u is bad if $Pr_{S_u \sim \pi_u}[S_u > (B - d(\rho, u))] \geq 1/2$. Notice that by visiting a bad vertex u, the probability that Opt can continue decreases geometrically by a factor of $1/2$, because of the high chance of the size being large enough to exceed the total budget. Therefore, the total expected reward that we can collect from all bad jobs is at most, say, twice, that we can obtain by visiting the best bad vertex. Since the expected reward from any bad vertex is small, we can safely skip such vertices in Opt, and still retain $\Omega(\mathsf{Opt})$ reward.

Without loss of generality, let the optimal non-adaptive ordering be $P^* = \{\rho = v_0, v_1, v_2, \ldots, v_n\}$; also denote the expected total reward by Opt. 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 time spent before visiting any vertex is a random quantity, the distance is deterministic, since we deal with non-adaptive strategies. Let v_{j^*} denote the first vertex in P^* such that

$$\textstyle\sum_{i<j} \mu_{v_i}(B - D_j) \quad \geq \quad K \cdot (B - D_j) \tag{8.3}$$

Here K is some constant which we will fix later.

LEMMA 8.1. *The probability that Opt visits some vertex indexed j^* or higher is at most $e^{1-K/2}$.*

Proof. If Opt 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 Opt visits v_{j^*} is upper bounded by the probability that $\sum_{i<j^*} X_i \leq (B - d_{j^*})$. To this end, we have from Inequality (8.3) that $\sum_{i<j^*} \mathbb{E}[X_i] \geq K$. Therefore we can apply a standard Chernoff bound to conclude that

$$\begin{aligned} \Pr[\mathsf{Opt} \text{ visits vertex } v_{j^*}] &\leq \Pr\left[\sum_{i<j^*} X_i \leq 1\right] \\ &\leq e^{1-K/2} \end{aligned}$$

This completes the proof.

We therefore conclude that, for a suitably large constant K (say $K = 10$), we can *truncate* Opt at vertex v_{j^*-1}, and still only lose a constant fraction of the reward: this is because Opt can reach vertex v_{j^*} with probability at most $e^{1-K/2}$ by Lemma 8.1, and conditioned on reaching v_{j^*}, the expected reward it can fetch is at most Opt. In particular we have the following two properties of the prefix of vertices $\Pi^* \equiv \{v_1, v_2, \ldots, v_{j^*-1}\}$.

$$\begin{gathered} \textstyle\sum_{i \leq j^*-1} r_{v_i} = R^* = \Omega(\mathsf{Opt}) \\ \textstyle\sum_{i \leq j^*-1} \mu_{v_i}(B - D_{j^*-1}) = \\ \sum_{i<j^*-1} \mu_{v_i}(B - D_{j^*-1}) + \mu_{v_{j^*-1}}(B - D_{j^*-1}) \\ \leq (K+1)(B - D_{j^*-1}) \end{gathered}$$

The last inequality is by choice of j^* in equation (8.3). Now if $D_{j^*-1} < B$, then let $\ell \in \mathbb{Z}_+$ be such that $B/2^\ell < B - D_{j^*-1} \leq B/2^{\ell-1}$ and $W^* = B/2^\ell$. Otherwise if $D_{j^*-1} = B$, then let $W^* = 0$. Observe that in either case, $D_{j^*-1} \leq B - W^*$. Furthermore, we have

$$\begin{aligned} \sum_{i \leq j^*-1} \mu_{v_i}(W^*/2) &\leq \sum_{i \leq j^*-1} \mu_{v_i}(B - D_{j^*-1}) \\ &\leq (K+1)(B - D_{j^*-1}) \\ &\leq 2(K+1)W^* \end{aligned} \tag{8.4}$$

A Valid Knapsack Orienteering with Large Reward. We now show that a sub-prefix of Π^* (which has length at most $B - W^*$) is a good candidate solution for the valid $\mathsf{KnapOrient}$ instance $\mathcal{I}_{ko}(W^*)$ by applying the following lemma with parameters $R = \Omega(\mathsf{Opt})$, $W = W^*$ and $K' = 2(K+1) = O(1)$, all defined as above. From the lemma, we obtain a solution Π' which is feasible for the valid $\mathsf{KnapOrient}$ instance $\mathcal{I}_{ko}(W = W^*)$ and has reward $\Omega(R/K') = \Omega(\mathsf{Opt})$, and this completes the proof of Theorem 7.1.

LEMMA 8.2. *Given parameters R, W, and K' suppose that no single-vertex tour has expected reward $\Omega(R/K')$. Then, given a path $\Pi \equiv \{v_1, v_2, \ldots, v_j\}$ of length at most $B - W$ such that*

- $\sum_{i \le j} r_{v_i} \ge R$, *and*
- $\sum_{i \le j} \mu_{v_i}(W/2) \le K' \cdot W$.

Then the valid $\mathsf{KnapOrient}$ *instance with parameter W has reward $\Omega(R/K')$.*

Proof. We first partition the vertices in Π into two classes based on their truncated means: $\mathcal{L} = \{u \in \Pi \mid \Pr_{S_u \sim \pi_u}[S_u > W/2] \ge 1/2\}$ and $\mathcal{S}$ be the other vertices visited in Π. Notice that all the vertices in $\mathcal{S}$ have $\widehat{r}_u = r_u$ in the valid instance $\mathcal{I}_{ko}(W)$ of $\mathsf{KnapOrient}$.

Case (i): $\sum_{u \in \mathcal{L}} r_u \ge (1/2) \sum_{u \in \Pi} r_u$. By the definition of $\mathcal{L}$, each node $u \in \mathcal{L}$ has $\mu_u(W/2) \ge (1/4)W$; moreover, $\sum_{u \in Pi} \mu_u(W/2) \le K' \cdot W$ by the second assumption in the lemma (which in turn would follow from equation (8.4)). So $|\mathcal{L}| \le 4K'$, and the node v' in $\mathcal{L}$ with highest reward has reward $r_{v'} \ge \Omega(R/K')$. Now by Observation 8, the job located at v' instantiates to a size at most $B - d(\rho, v')$ with probability at least $1/2$, and therefore the single vertex tour of visiting v' has expected reward $\Omega(R/K')$, a contradiction.

Case (ii): $\sum_{u \in \mathcal{S}} r_u \ge (1/2) \sum_{u \in \Pi} r_u$. In this case we now construct the (natural) feasible solution Π' to the valid $\mathsf{KnapOrient}$ instance with parameter W:

(a) Partition the vertices in $\mathcal{S}$ into at most $2K'$ groups such that the sum of the truncated means of jobs in each group is at most W.

(b) Pick the group which has largest sum of rewards (clearly this is at least $\Omega(\mathsf{Opt}/K')$).

(c) Let Π' visit only the nodes of the group chosen above.

Since Π had length at most $B - W$, the subpath Π' also satisfies the travel budget. Further, the partitioning in step (a) above ensures that $\sum_{u \in \Pi'} \widehat{s}_u \le W$, and the definition of $\mathcal{S}$ ensures that $\widehat{r}_u = r_u$ for all vertices $u \in \Pi'$, so the reward is $\sum_{u \in \Pi'} \widehat{r}_u \ge \Omega(\mathsf{Opt}/K')$. This completes the proof of Lemma 8.2.

9 The Algorithm for 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 to the best adaptive solution. Note that this also establishes an adaptivity gap of $O(\log\log B)$, thus proving Theorem 1.2.

Our proof proceeds by showing that for an optimal adaptive solution with expected reward Opt, either just a single-vertex tour has reward $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$, or one of the *valid* $\mathsf{KnapOrient}$ instances (Definition 7) for some W of the form $B/2^i$ (or $W = 0$) has reward $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$. This is sufficient because we can now plug in the following Structure Theorem 2 (Theorem 9.1) instead of the non-adaptive Structure Theorem 7.1 and the rest of the analysis goes through unchanged. Indeed, we argued in Section 7.1 that any solution for *any valid* $\mathsf{KnapOrient}$ *instance* with reward $\Omega(R)$ can be converted into a non-adaptive strategy for $\mathsf{StocOrient}$ with reward $\Omega(R)$, and therefore this also establishes the adaptivity gap of $\log\lceil\log B\rceil$ for $\mathsf{StocOrient}$.

THEOREM 9.1. (STRUCTURE THEOREM 2) *Given an instance $\mathcal{I}_{so}$ for which an optimal strategy has an expected reward* Opt, *either there is a single-vertex tour with expected reward $\Omega(\mathsf{Opt}/\log\log B)$, or there exists $W = B/2^i$ for some $i \in \mathbb{Z}_{\ge 0}$ (or $W = 0$) for which the valid* $\mathsf{KnapOrient}$ *instance $\mathcal{I}_{ko}(W)$ has reward $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$.*

Before we begin, recall the typical instance of $\mathsf{StocOrient}$ $\mathcal{I}_{so} := \mathsf{StocOrient}(V, d, \{(\pi_u, r_u) : \forall u \in V\}, B, \rho)$ of the stochastic orienteering problem, where the metric is (V, d), B is the total budget of our tour, the job at vertex u has a reward r_u and a probability distribution $\pi_u : \mathbb{R}^+ \to [0, 1]$ over sizes, and ρ is the starting vertex.

Roadmap. We begin by giving a roadmap of the proof. Let us view the optimal adaptive strategy 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 played until x_P exceeds the "budget remaining"—here, if $L_{x,P}$ is the total distance traveled from the root ρ to this vertex 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 would equal the expected sizes), then we would not be able to get any reward from portions of the decision tree on or below the frontier nodes. Unfortunately, since job sizes are random for us, this may not be the case.

The main idea in the proof is to show that we don't lose too much reward by truncation: even if we truncate Opt along this frontier in a $\mathsf{StocOrient}$ instance, we still get an expected reward of $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$ from the truncated tree. Now an averaging argument says that there exists *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 $O(B-L)$ and this gives us the candidate $\mathsf{KnapOrient}$ solution. However, as we elaborated earlier in Section 4 , we need to be very careful in the way we define our deterministic sizes and truncation thresholds.

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 matters, 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', and note the amount of time spent traveling is exactly the number of edges traveled down this tree. (All this is only for analysis.) 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 after going t hops depends on past outcomes). We stop this process 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 timestep 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* t of a node x in Opt' is the number of hops in the decision tree from 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 and π_x to denote the size, reward, and size distribution for node x—hence $S_x = S_{\mathsf{label}(x)}$, $R_x = R_{\mathsf{label}(x)}$ and $\pi_x = \pi_{\mathsf{label}(x)}$. We use $x' \preceq x$ to denote that x' is an ancestor of x.

Now to begin the proof of Theorem 9.1. Firstly, 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 in Section 12.1.) The proof proceeds by assuming that no single-vertex tour has reward $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$, and then going on to show that one of the valid $\mathsf{KnapOrient}$ instances must have large reward. Like in Section 5 we can make the following observation about the optimal solution (the statement is identical to Observation 8, we simply state it for continuity).

Observation 10. *If no single-vertex tour has reward $\Omega(\mathsf{Opt}/\log\lceil\log B\rceil)$, we can skip all visits to vertices u such that $Pr_{S_u \sim \pi_u}[S_u > (B - d(\rho,u))] \geq 1/2$, and still preserve a constant factor of the total reward.*

Proof. See proof of Observation 8.

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:

$$\textstyle\sum_{x' \preceq x} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x))\right) \geq 8K\left(B - \mathsf{level}(x)\right) \tag{10.5}$$

If no such node exists for some sample path, we define the last node on this sample path to be a star node. Observe that the collection $\{x \in \mathsf{Opt}' \text{ s.t } x \text{ is a star node}\}$ satisfies the properties that no star node is an ancestor of another star node, and every sample path has some star node marked. 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 *level*(x) jobs on the sample path until x sum to at least $8K(B - \mathsf{level}(x))$. But since we've spent $\mathsf{level}(x)$ time traveling to reach x, the process only goes below 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 K 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. The following lemma is the main technical component of the reduction, and bounds the probability that the random process $\mathcal{T}$ visits nodes that are descendants of star nodes.

LEMMA 10.1. *The process $\mathcal{T}$ continues beyond a star node with probability at most $1/10$, if $K = \Omega(\log\lceil \log B\rceil)$.*

Proof. We group the star nodes into $\lceil \log B\rceil + 1$ *bands* based on the value of $(B - \mathsf{level}(x))$. Say that a 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 $\mathsf{level}(x) = B$. Firstly note that Opt can never continue beyond a star node at band $\lceil \log B\rceil + 1$ since it would run out of budget by then. Therefore we focus on a particular band $i \in \{0, 1, \ldots, \lceil \log B\rceil\}$ and bound the probability of $\mathcal{T}$ reaching a star node in band i by $1/(10(\lceil \log B\rceil + 1))$; then a trivial union bound over all i suffices.

In order to bound the probability, 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 10.3.

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 corresponding probability of continuing beyond a band-i star node in $\mathcal{T}_i$. Therefore, let us fix a particular band i.

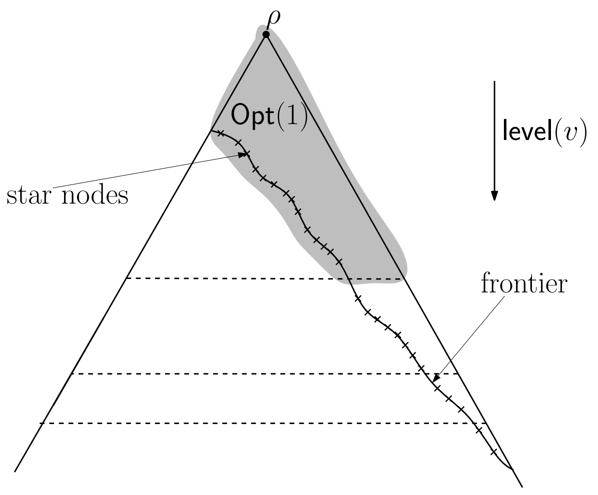

Figure 10.3: Optimal decision tree example: dashed lines indicate the bands, $\times$ indicates star nodes.

CLAIM 11. *For each $i \in \{0, 1, \ldots, \lceil \log B\rceil\}$, and any star node x in band i, the following inequality holds:*

$$4K\frac{B}{2^i} \le \sum_{x' \preceq x} \mu_{x'}\left(B/2^{i+1}\right) \le 34K\frac{B}{2^i}$$

Proof. By the definition of a star node (10.5), and the fact that a band-i star node x has $B - \mathsf{level}(x) > B/2^{i+1}$,

$$\sum_{x' \preceq x} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x))\right) \ge 8K(B - \mathsf{level}(x)) \ge 4K\frac{B}{2^i}. \tag{11.6}$$

Moreover, since x'', the parent node of x, is not a star node, it satisfies $\sum_{x' \preceq x''} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x''))\right) < 8K(B - \mathsf{level}(x'')) = 8K(B - \mathsf{level}(x) + 1)$. But since we are currently not considering band number $\lceil \log B\rceil + 1$

(and all distances are at least 1), $\mathsf{level}(x) \leq B-1$, and hence $B - \mathsf{level}(x) + 1 \leq 2(B - \mathsf{level}(x))$.

$$\begin{aligned}\sum_{x' \preceq x''} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x))\right) &\leq \sum_{x' \preceq x''} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x''))\right) \\ &\leq 8K(B - \mathsf{level}(x) + 1) \\ &\leq 16K(B - \mathsf{level}(x))\end{aligned}$$

Now we can use the naive bound that $\mu_x\left(\frac{1}{2}(B - \mathsf{level}(x))\right) \leq \frac{1}{2}(B - \mathsf{level}(x)) \leq (B - \mathsf{level}(x))$. Adding this to both sides of the above expression, we get that for any node x in band i,

$$\sum_{x' \preceq x} \mu_{x'}\left(\frac{1}{2}(B - \mathsf{level}(x))\right) \leq 17K(B - \mathsf{level}(x)) \leq 17K\frac{B}{2^i}. \tag{11.7}$$

Finally, remember that nodes of band i have $(B - \mathsf{level}(x)) \in (B/2^{i+1}, B/2^i]$; hence if we use a uniform truncation of $B/2^{i+1}$ instead of $\frac{1}{2}(B - \mathsf{level}(x))$, the lower bound of (11.6) still holds, and the upper bound of (11.7) at most doubles.

Claim 12. *For any i and any star node x in band i, if $\mathcal{T}_i$ crosses x, then*

$$\sum_{x' \preceq x} \min(S_{x'}, \frac{B}{2^{i+1}}) \leq \frac{B}{2^i}$$

Proof. Clearly, if the process $\mathcal{T}_i$ crosses the node x, it must mean that $\sum_{x' \preceq x} S_{x'} \leq (B - \mathsf{level}(x))$, else we would have run out of budget. The claim follows because truncation can only decrease the LHS, and for all $i \in \{0, 1, \lceil \log B \rceil\}$, node x in band i implies $(B - \mathsf{level}(x)) \leq B/2^i$.

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

$$Z_t = \sum_{t'=0}^{t}\left(\min(\mathbf{S}_{t'}, \frac{B}{2^{i+1}}) - \mu_{\mathbf{S}_{t'}}\left(B/2^{i+1}\right)\right). \tag{12.8}$$

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 (i) either the process reaches a band-i star node, or (ii) there is no way to get to a band-i star node in the future.

Claim 12 says that when $\mathcal{T}_i$ crosses some fixed star node x, the sum over the first terms in (12.8) is at most $B/2^i$, whereas Claim 11 says the sums of the means is at least $4K\frac{B}{2^i}$. Because $K \geq 1$, we can infer that the probability of the event $\{Z_\tau \leq -3K(B/2^i)\}$ is an upper bound on the probability of $\mathcal{T}_i$ reaching a star node in band i. To this end, we appeal to Freedman's concentration inequality for martingales, which substantially generalizes the Azuma-Hoeffding inequality.

Theorem 12.1. (Freedman [FRE75] (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*

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

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

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

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 $\min(S_t, \frac{B}{2^{i+1}}) - \mu_{S_t}\left(B/2^{i+1}\right)$, we get that $\mathbb{E}\left[X_t \mid X_{t-1}, \ldots\right] = 0$ by definition of $\mu_{S_t}\left(B/2^{i+1}\right)$. Moreover, since the sizes and means are both truncated at $B/2^{i+1}$, we have $|X_t| \leq B/2^{i+1}$ w.p 1; hence we can set $M = B/2^{i+1}$. Finally in order to bound the variance term W_t we appeal to Claim 11. Indeed, consider a single r.v $X_t = \min(S_t, \frac{B}{2^{i+1}}) - \mu_{S_t}\left(B/2^{i+1}\right)$, and suppose we abbreviate $\min(S_t, \frac{B}{2^{i+1}})$ by Y for convenience. Then we have

$$\begin{aligned}\mathbb{E}\left[X_t^2 \mid X_{t-1}, \ldots\right] &= \mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right] \\ &= \mathbb{E}\left[Y^2\right] - \mu_{S_t}\left(B/2^{i+1}\right)^2 \\ &\leq \mu_{S_t}\left(\frac{B}{2^{i+1}}\right)\frac{B}{2^{i+1}}\end{aligned}$$

Here, the last inequality above comes from the fact that the worst case distribution for Y in order to maximize $E[Y^2]$ is $Y = B/2^{i+1}$ w.p $\mathbb{E}[Y]/(B/2^{i+1})$ and $Y = 0$ otherwise. Hence the term W_t is at most $(B/2^{i+1})\sum_{t' \leq t}\mu_{S'_t}\left(B/2^{i+1}\right)$ for the process at time t. Now from Claim 11 we have that for any star node (say at level t) in band i, we have $\sum_{t' \leq t}\mu_{t'}\left(B/2^{i+1}\right) \leq 34K(B/2^i)$. Therefore we can bound W_τ for star nodes by $17K(B/2^i)^2$, and we set σ^2 to be this quantity.

So by setting $l = 3K(B/2^i)$, $\sigma^2 = 17K(B/2^i)^2$, and $M = B/2^{i+1}$, we get that:

$$\Pr\left[|Z_\tau| \geq 3K(B/2^i) \textit{ and } W_\tau \leq 17K(B/2^i)^2\right] \leq \exp(-K/4).$$

This in turn upper bounds the probability of reaching a star node of band i. Setting $K = \Omega(\log\lceil\log B\rceil)$ and performing a simple union bound calculation completes the proof of Lemma 10.1.

Since by Lemma 10.1 we cross star nodes with probability at most $1/10$, truncating Opt' at these star nodes results in a net lost reward of at most $\mathsf{Opt}/10$. So by an averaging argument, there exists a sample path P^* from the root node to some star node x^* (say belonging to band i^*) of length $\mathsf{level}(x^*)$ with (a) the sum of rewards of nodes visited along this path being $9\mathsf{Opt}/10$ and (b) the sum of (truncated) means of jobs satisfies the following condition with $W = B/2^{i^*+1}$ (or with $W = 0$, if i^* is the final band $\lceil\log B\rceil + 1$):

$$\textstyle\sum_{x \in P^*}\mu_x\left(W/2\right) \leq 34K \cdot W. \tag{12.9}$$

For $i^* < \lceil\log B\rceil + 1$, we used the fact that x^* lies in band i^* and applied Claim 11 for x^* to get the above bound; if $i^* = \lceil\log B\rceil + 1$, it follows trivially since $W = 0$. Now we can apply Lemma 8.2 on path P^* (which has length equal to $L = \mathsf{level}(x^*) \leq B - B/2^{i^*+1} = B - W$) with parameters $R = 9\mathsf{Opt}/10$, W, and $K' = 34K = O(\log\lceil\log B\rceil)$ to complete the proof of Theorem 9.1.

12.1 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 tour has traveled a distance of l. In particular, this meant that there is at most one stochastic job per node (because if Opt is done with processing some job, it has to travel at least 1 timestep in the decision tree because its depth increases). However, if we are more careful in defining the truncations of any node (in equation (10.5)) by its *actual length along the tour*, instead of simply its depth/level in the tree, then we will be able to 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.

13 Stochastic Orienteering with Correlated Rewards (CorrOrient)

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 size. Crucially, the distributions at different vertices are still independent of each other. For any vertex v, we are given a probability distribution over (size, reward) pairs as follows: for each $t \in \{0, 1, \ldots, B\}$, the job at v has size t and reward $r_{v,t}$ with probability $\pi_{v,t}$. Again we consider the setting without cancellations, 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) strategy that maximizes the expected reward of completed jobs, subject to the total budget (travel time + processing time) being at most B.

When there is no metric in the problem instance, it is a correlated stochastic knapsack instance, and [GKMR11] show a non-adaptive algorithm which is a constant-factor approximation to the optimal adaptive strategy; this is using an LP-relaxation that is quite different from that in the uncorrelated setting. The problem with extending that approach to stochastic orienteering is that we do not know of LP relaxations with good approximation guarantees even for the *deterministic* orienteering problem. 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 technique here, but as Theorem 1.4 says, our approximation ratio is only a $O(\log n \log B)$ factor here. Moreover, we show that $\mathsf{CorrOrient}$ is at least as hard to approximate as the deadline orienteering problem, for which the best known guarantee is an $O(\log n)$ approximation [BBCM04].

13.1 The Non-Adaptive Algorithm for $\mathsf{CorrOrient}$ We now get into describing our approximation algorithm, which proceeds via a reduction to a suitably constructed instance of the *deadline orienteering problem.* This reduction proceeds via taking a *Lagrangian relaxation* of the processing times, and then performing an amortized analysis to argue that the reward is high and the budget is met with constant probability. (In this it is similar to the ideas of [GM09].) However, new ideas are required to handle correlations between sizes and rewards: indeed, this is where the deadline orienteering problem is needed. Also, we use an interesting counting property (Claim 14) to avoid double-counting reward.

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* thus: 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 $[2^i-1, 2^{i+1}-1)$. (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 we can get from job v *if it starts at time t*.

13.1.1 Reducing $\mathsf{CorrOrient}$ to $\mathsf{DeadlineOrient}$ The high-level idea is the following: for any fixed i, we create an instance of $\mathsf{DeadlineOrient}$ 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. To get the instance of $\mathsf{DeadlineOrient}$, for each vertex v, we create several copies of it, where the copy $\langle v, t\rangle$ corresponds to the situation where the job starts at time t, and hence has reward $R_{v,t}$. However, to prevent the $\mathsf{DeadlineOrient}$ solution from collecting reward from many different copies of the same vertex, we make copies of vertices *only when the reward changes by a geometric factor.* Indeed, for $t_1 < t_2$, it it holds that $R_{v,t_1} \leq 2R_{v,t_2}$, then we do not include the copy $\langle v, t_1\rangle$ in our instance. The following claim is useful for defining such a minimal set of starting times for each vertex/job.

CLAIM 14. *Given any non-increasing function $f : [B] \to \mathbf{R}_+$, we can efficiently find a subset $I \subseteq [B]$ such that:*

$$\frac{f(t)}{2} \leq \max_{\ell\in I:\ell\geq t} f(\ell)\,,$$

and

$$\sum_{\ell\in I:\ell\geq t} f(\ell) \leq 2\cdot f(t),\ \forall t \in [B].$$

Proof. The set I is constructed as follows. Observe that B is always contained in the set I, and hence for any $t \in [B]$, $\min\{\ell \geq t : \ell \in I\}$ is well-defined.

To prove the claimed properties let $I = \{\ell_i\}_{i=0}^p$. For the first property, given any $t \in [B]$ let $\ell(t) = \min\{\ell \geq t : \ell \in I\}$. Observe that if this set is empty then it must be that $f(t) = 0$ and the claim trivially holds. Now assume that value $\ell(t)$ exists and that $\ell(t) = \ell_i$. By the definition of $\ell(t)$ it must be that $\ell_{i-1} < t$, and so $k_i \leq t$. Hence $f(\ell_i) \geq f(k_i)/2 \geq f(t)/2$; the first inequality is by choice ℓ_i, and the second as f is non-increasing.

We now show the second property. For any index i, we have $\ell_{i-1} < k_i \leq \ell_i < k_{i+1} \leq \ell_{i+1}$. Note that $f(k_{i+1}) = f(\ell_i+1) < f(k_i)/2$ by choice of ℓ_i. So $f(\ell_{i+1}) \leq f(k_{i+1}) < f(k_i)/2 \leq f(\ell_{i-1})/2$. Given any $t \in [B]$ let

Algorithm 2 Computing the support I in Claim 14.

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

q be the index such that $\ell_q = \min\{\ell \geq t : \ell \in I\}$. Consider the sum:

$$\begin{aligned}\sum_{i\geq q} f(\ell_i) &= \sum_{j\geq 0} f(\ell_{q+2j}) + \sum_{j\geq 0} f(\ell_{q+1+2j}) \leq f(\ell_q) + f(\ell_{q+1}) \\ &\leq 2 \cdot f(t)\end{aligned}$$

The first inequality uses $f(\ell_{i+1}) < f(\ell_{i-1})/2$ and a geometric summation; the last is by $t \leq \ell_q < \ell_{q+1}$.

Consider any $i \in [\log_2 B]$. Now for each $v \in V$, apply Claim 14 to the function $f(t) := R_{v,t+2^i-1}$ to obtain a subset $I_v^i \subseteq [B]$, which defines which copies we use—namely, we create $|I_v^i|$ co-located copies $\{\langle v, \ell\rangle : \ell \in I_v^i\}$ where node $\langle v, \ell\rangle$ has deadline ℓ. Define the size $s_i(\langle v, \ell\rangle) = \mathbb{E}\left[\min(S_v,\, 2^i)\right]$, and $r_i(\langle v, \ell\rangle) = R_{v,\ell+2^i-1}$. Finally, define $N_i = \{\langle v, \ell\rangle : \ell \in I_v^i,\, v \in V\}$.

Now we are ready to describe the DeadlineOrient instance $\mathcal{I}_i(\lambda)$ for any $\lambda \geq 0$. The metric is (V, d), and for each vertex $v \in V$, the copies I_v^i are all co-located at v. The copy $\langle v, \ell\rangle \in I_v^i$ has deadline ℓ, and a profit of $\hat{r}_i(v, \ell, \lambda) = r_i(v, \ell) - \lambda \cdot s_i(v)$. The goal is to find a path which maximizes the reward of the copies that are visited within their deadlines. Here, λ should be thought of as a Lagrangean multiplier, and hence the above instance is really a kind of Lagrangean relaxation of a DeadlineOrient instance which has a side constraint that the total processing time is at most $\approx 2^i$. 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 DeadlineOrient solution for $I_i(\lambda)$ has value $\Omega(\mathsf{Opt}(i))$; moreover that we can recover a valid solution to the stochastic orienteering problem from an approximate solution to $I_i(\lambda)$.

LEMMA 14.1. *For any $i \in [\log B]$ and $\lambda > 0$, the optimal value of the* DeadlineOrient *instance $\mathcal{I}_i(\lambda)$ is at least* $\mathsf{Opt}(i)/2 - \lambda\, 2^{i+1}$.

Proof. Consider the optimal decision tree Opt of the 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 $2^i - 1 \leq \mathsf{size} < 2^{i+1} - 1$, so zeroing out rewards from all but type-i nodes yields expected reward $\mathsf{Opt}(i)$.

For any vertex $v \in V$, let X_v^i denote the indicator r.v. that job v is run as type-i in Opt, and S_v be the r.v. denoting its instantiated size—note that these are independent. 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 $Y^i \leq 2 \cdot 2^i$ with probability one, and hence $\mathbb{E}[Y^i] \leq 2^{i+1}$. For ease of notation let $q_v = \Pr[X_v^i = 1]$ for all $v \in V$. We now have,

$$\begin{aligned}2^{i+1} &\geq \mathbb{E}[Y^i] \qquad (14.10)\\ &= \sum_{v\in V} q_v \cdot \mathbb{E}[\min(S_v, 2^i) \mid X_v^i = 1]\\ &= \sum_{v\in V} q_v \cdot \mathbb{E}[\min(S_v, 2^i)]\\ &= \sum_{v\in V} q_v \cdot s_i(v)\end{aligned}$$

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

$$\begin{aligned}
\mathsf{Opt}(i) &= \sum_{v\in V}\sum_{\ell\in[B]}\sum_{k=2^i-1}^{2^{i+1}-2} \Pr[\mathbf{1}_{v,\mathsf{dist}=l,\mathsf{size}=k}]\cdot R_{v,\ell+k} \\
&\leq \sum_{v\in V}\sum_{\ell\in[B]} \Pr[\mathbf{1}_{v,\text{type}=i,\mathsf{size}=k}]\cdot R_{v,\ell+2^i-1} \qquad (14.11)
\end{aligned}$$

where $\mathbf{1}_{v,\mathsf{dist}=l,\mathsf{size}=k}$ is the indicator for whether Opt visits v with $\mathsf{dist}=l$ and $\mathsf{size}=k$, and $\mathbf{1}_{v,\text{type}=i,\mathsf{size}=k}$ is the indicator for whether Opt visits v as type-i with $\mathsf{size}=k$. 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 $\ell_v(P)=\min\{\ell\in I_v^i\mid \ell\geq d_v(P)\}$; recall that we defined $\mathcal{I}_v^i$ on the previous page using Claim 14, and since $B\in\mathcal{I}_v^i$, the quantity $\ell_v(P)$ is well-defined.

For any path $P\in\mathcal{P}$, consider P as a solution to $\mathcal{I}_i(\lambda)$ that visits the copies $\{\langle v,\ell_v(P)\rangle : v\in P\}$ within their deadlines. It is feasible for the instance $\mathcal{I}_i(\lambda)$ because for each vertex in P, we defined $\ell_v(P)\geq d_v(P)$ and therefore we would reach the chosen copy within its deadline. Moreover, the objective value of P is precisely

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

where the second inequality above uses the definition of $\ell_v(P)$ and Claim 14. Now

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

Above the second-to-last equatility is by interchanging summations and splitting the two terms from the previous expression, the first term in the final inequality comes from (14.11), and the second term comes from (14.10) 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 $\mathsf{DeadlineOrient}$ problem.[3] We focus on the "right" setting of λ defined thus:

$$\lambda_i^* = \max\left\{\lambda : \mathsf{AlgDO}(\mathcal{I}_i(\lambda)) \geq \frac{2^i\cdot\lambda}{\alpha}\right\} \qquad (14.12)$$

[3]We will slightly 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.

Lemma 14.2. *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 14.1, 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 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 $\lambda_i^* \geq \widehat{\lambda} \geq \mathsf{Opt}(i)/2^{i+3}$, hence proved.

14.0.2 Obtaining CorrOrient solution from AlgDO(λ_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. 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 tour; 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 the vertices of $P$ into $c = \max(1, \lfloor 2y \rfloor)$ parts $P_1, \ldots, P_c$ s.t $\sum_{v\in P_j} s_i(v) \leq 2^i$ for all $1 \leq j \leq c$.
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 each vertex in $\overline{Q}$ independently w.p. 1/2, and visit these sampled vertices in order given by $P$.
```

At a high level, the algorithm partitions the vertices in σ into groups, where each group obeys the size budget of 2^i in expectation. It then picks the most profitable group of them. The main issue with Q chosen in Step 3 is that it may include multiple copies of the same vertex. But because of the way we constructed the sets I_i^v (based on Claim 14), 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 $r_i(Q)$. Our first claim bounds the total (potential) reward of the set Q we select in Step 3.

Claim 15. *The sum $r_i(Q) = \sum_{\langle v,\ell\rangle\in Q} r_i(v,\ell)$ is at least $\mathsf{Opt}(i)/(8\alpha)$.*

Proof. Consider the following chain of inequalities:

$$\begin{aligned} r_i(Q) &\geq \frac{r_i(P)}{c} \\ &\geq \frac{\lambda_i^* y 2^i + \lambda_i^* 2^i/\alpha}{c} \\ &= \lambda_i^* 2^i \cdot \left(\frac{y + 1/\alpha}{c}\right) \\ &\geq \lambda_i^* 2^i \cdot \min\left\{y + \frac{1}{\alpha}, \frac{1}{2} + \frac{1}{2\alpha y}\right\} \\ &\geq \frac{\lambda_i^* 2^i}{\alpha} \end{aligned}$$

The second inequality is due the the fact that $2^i \lambda_i^*/\alpha \leq \mathsf{AlgDO}(\lambda_i^*) = \hat{r}_i(P) = r_i(P) - \lambda_i^* \cdot s^i(P)$ due to the choice of λ_i^* in equation (14.12); the third inequality is by $c \leq \max\{1, 2y\}$; and the last inequality uses $\alpha \geq 2$. To conclude we simply use Lemma 14.2.

Next, we show that we do not lose much of the total reward by discarding duplicate copies of vertices in Q.

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

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 $u \in \overline{Q}$,

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

Above, the last inequality uses the definition of I_u^i as given by Claim 14. Adding over all $u \in \overline{Q}$,

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

Here, the last inequality avove uses Claim 15. This completes the proof.

We now argue that the algorithm A_i reaches any vertex $v \in \overline{Q}$ before the time $d_v + 2^i - 1$ with constant probability.

Claim 17. *For any vertex $v \in \overline{Q}$, it holds that* $\Pr\left[A_i \text{ reaches job } v \text{ by time } d_v + 2^i - 1\right] \geq \frac{1}{2}$.

Proof. We know that because P is a feasible tour for the 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 $\overline{Q}'$ denote the set of vertices sampled in Step 6. We then say that the *bad event* occurs if $\sum_{u\in\overline{Q}'\setminus v}\min(S_u, 2^i) \geq 2^i$. Indeed if $\sum_{u\in\overline{Q}'\setminus v}\min(S_u, 2^i) < 2^i$, then certainly we will reach v by time $d_v + 2^i - 1$.

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

$$\mathbb{E}\Big[\sum_{u\in\overline{Q}'\setminus v}\min(S_u, 2^i)\Big] \leq \sum_{u\in\overline{Q}}\frac{1}{2}\,\mathbb{E}\left[\min(S_u, 2^i)\right],$$

by linearity of expectation and the fact that each u is sampled with probability 1/2. Now the latter sum is at most $(1/2)\sum_{u\in\overline{Q}} s_i(u) \leq 2^{i-1}$, by the choice of Q in the partition Step 3. Hence, the probability of the bad event is at most 1/2 by Markov's inequality.

Theorem 17.1. *The expected reward of the algorithm A_i is at least $\mathsf{Opt}(i)/64\alpha$.*

Proof. We know from the above claim that for each vertex $v \in \overline{Q}$, A_i reaches v by time $d_v + 2^i - 1$ with probability at least 1/2. In this event, we sample v 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 16.

17.1 Evidence of hardness for CorrOrient Our approximation algorithm for CorrOrient can be viewed as a reduction to DeadlineOrient. 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 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. 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 $(1 - o(1)) \cdot \mathsf{Opt}(\mathcal{I}) \leq \mathsf{Opt}(\mathcal{J}) \leq (1 + o(1)) \cdot \mathsf{Opt}(\mathcal{I})$.

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 $\sum_{v\in S}(1 - p)^n \geq |S| \cdot (1 - np) = (1 - o(1)) \cdot |S|$. Choosing τ to be the optimal solution to $\mathcal{I}$, we have $(1 - o(1)) \cdot \mathsf{Opt}(\mathcal{I}) \leq \mathsf{Opt}(\mathcal{J})$.

Consider now any adaptive strategy 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 $\mathcal{I}$; let $S_0 \subseteq V$ denote the vertices on path σ_0 that are visited prior to their respective deadlines. Clearly $\mathsf{Opt}(\mathcal{I}) \geq |S_0|$. When strategy σ is run, every size zero instantiation gives zero reward, and no $(V \setminus S_0)$-vertex visited on σ_0 can fit into the knapsack (with positive size instantiation); so

$$\Pr[\sigma \text{ gets positive reward}] \leq p \cdot |S_0| \tag{17.13}$$

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

$$\begin{aligned} R(\sigma) &= \frac{1}{p} \cdot \sum_{i=1}^{n} \Pr[\sigma \text{ gets reward at least } i/p] \\ &= \frac{1}{p} \cdot \Pr[\sigma \text{ gets positive reward}] \qquad (17.14) \\ &+ \frac{1}{p} \cdot \sum_{i=2}^{n} \Pr[\sigma \text{ gets reward at least } i/p] \end{aligned}$$

Furthermore, for any $i \geq 2$, we have: $\Pr[\sigma \text{ gets reward at least } i/p] \leq \Pr[\text{at least } i \text{ jobs instantiate to positive size}] \leq \binom{n}{i} \cdot p^i \leq (np)^i$. It follows that the second term in (17.14) can be upper bounded by $\frac{1}{p} \cdot \sum_{i=2}^{n} (np)^i \leq 2n^2 p$. Combining this with (17.13) and (17.14), we obtain that $R(\sigma) \leq |S_0| + 2n^2 p = |S_0| + o(1)$. Since this holds for any adaptive strategy σ for $\mathcal{J}$, we get $\mathsf{Opt}(\mathcal{I}) \geq (1 - o(1)) \cdot \mathsf{Opt}(\mathcal{J})$.

18 Stochastic Orienteering with Cancellations

We now describe a black-box way to handle the setting where an optimal strategy can actually abort a job during its processing (say, if it turns out to be longer than expected). We first note that even for the basic case of Stochastic Knapsack, there are very simple examples that demonstrate an arbitrarily large gap in the expected reward for strategies which can cancel and those that can't. The idea is simple — essentially we create up to B co-located copies of every job, and imagine that copy $\langle v, t \rangle$ has expected reward $\sum_{t'=0}^{t} \pi_{v,t'} r_{v,t'}$, i.e., it fetches reward only when the size instantiates to at most t. 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). Therefore 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 solution for the StocOrient instance: the deterministic solution might collect reward from multiple copies of the same job. We can fix this by again using the geometric scaling idea (using Claim 14), i.e., if there are two copies of the (roughly the) same reward, we only keep the one with the earlier cancellation time. This way, we can ensure that for all remaining copies of a particular job, the rewards are all at least geometric factors away. Now, even if the deterministic solution collects reward from multiple copies of a job, we can simply play the one amongst them with best reward, and we'll have collected a constant fraction of the total.

References

[AH08] Micah Adler and Brent Heeringa. Approximating Optimal Binary Decision Trees. In *APPROX*, pages 1–9, 2008.

[BBCM04] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 166–174 (electronic), New York, 2004. ACM.

[BCK+07] 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(2):653–670 (electronic), 2007.

[BGK11] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In *SODA '11*. Society for Industrial and Applied Mathematics, 2011.

[Bha11] Anand Bhalgat. A $(2+\epsilon)$-approximation algorithm for the stochastic knapsack problem. At `http://www.seas.upenn.edu/~bhalgat/2-approx-stochastic.pdf`, 2011.

[CGT11] Ann Melissa Campbell, Michel Gendreau, and Barrett W. Thomas. The orienteering problem with stochastic travel and service times. *Annals OR*, 186(1):61–81, 2011.

[CHP08] Ke Chen and Sariel Har-Peled. The Euclidean orienteering problem revisited. *SIAM J. Comput.*, 38(1):385–397, 2008.

[CIK$^+$09] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 266–278, Berlin, Heidelberg, 2009. Springer-Verlag.

[CK04] Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *APPROX-RANDOM*, pages 72–83, 2004.

[CKP08] Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 661–670, New York, 2008. ACM.

[CP05] Chandra Chekuri and Martin Pál. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, pages 245–253, 2005.

[DGV08] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.

[Fre75] David A. Freedman. On tail probabilities for martingales. *Annals of Probability*, 3:100–118, 1975.

[GKMR11] Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. *To appear in FOCS 2011*, arxiv: abs/1102.3749, 2011.

[GM07] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *STOC'07—Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 104–113. ACM, New York, 2007. Full version as *Sequential Design of Experiments via Linear Programming*, `http://arxiv.org/abs/0805.2630v1`.

[GM09] Sudipto Guha and Kamesh Munagala. Multi-armed bandits with metric switching costs. In *ICALP*, pages 496–507, 2009.

[GNR10] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive tsp problems. In *ICALP (1)*, pages 690–701, 2010.

[KPB99] S. Rao Kosaraju, Teresa M. Przytycka, and Ryan S. Borgstrom. On an Optimal Split Tree Problem. In *WADS*, pages 157–168, 1999.