Approximation Algorithms for Correlated Knapsack Orienteering

David Alemán Espinosa
Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1, Canada

Chaitanya Swamy
Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1, Canada

Abstract

We consider the *correlated knapsack orienteering* (CorrKO) problem: we are given a travel budget B, processing-time budget W, finite metric space (V, d) with root $\rho \in V$, where each vertex is associated with a job with possibly correlated random size and random reward that become known only when the job completes. Random variables are independent across different vertices. The goal is to compute a ρ-rooted path of length at most B, in a possibly adaptive fashion, that maximizes the reward collected from jobs that processed by time W. To our knowledge, CorrKO has not been considered before, though prior work has considered the uncorrelated problem, *stochastic knapsack orienteering*, and *correlated orienteering*, which features only one budget constraint on the *sum* of travel-time and processing-times.

Gupta et al. [19] showed that the *uncorrelated* version of this problem has a constant-factor adaptivity gap. We show that, perhaps surprisingly and in stark contrast to the uncorrelated problem, the *adaptivity gap of* CorrKO *is is at least* $\Omega\big(\max\{\sqrt{\log B}, \sqrt{\log\log W}\}\big)$. Complementing this result, we devise *non-adaptive* algorithms that obtain: (a) $O(\log\log W)$-approximation in quasi-polytime; and (b) $O(\log W)$-approximation in polytime. This also establishes that the adaptivity gap for CorrKO is at most $O(\log\log W)$. We obtain similar guarantees for CorrKO with cancellations, wherein a job can be cancelled before its completion time, foregoing its reward. We show that an α-approximation for CorrKO implies an $O(\alpha)$-approximation for CorrKO with cancellations.

We also consider the special case of CorrKO where job sizes are weighted Bernoulli distributions, and more generally where the distributions are supported on at most two points (2CorrKO). Although weighted Bernoulli distributions suffice to yield an $\Omega(\sqrt{\log\log B})$ adaptivity-gap lower bound for (uncorrelated) *stochastic orienteering*, we show that they are easy instances for CorrKO. We develop non-adaptive algorithms that achieve $O(1)$-approximation, in polytime for weighted Bernoulli distributions, and in $(n+\log B)^{O(\log W)}$-time for 2CorrKO. (Thus, our adaptivity-gap lower-bound example, which uses distributions of support-size 3, is tight in terms of support-size of the distributions.)

Finally, we leverage our techniques to provide a quasi-polynomial time $O(\log\log B)$ approximation algorithm for correlated orienteering improving upon the approximation guarantee in [2].

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Mathematics of computing → Discrete optimization

Keywords and phrases Approximation algorithms, Stochastic orienteering, Adaptivity gap, Vehicle routing problems, LP rounding algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2024.29

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2408.16566

Funding *David Alemán Espinosa*: Supported in part by NSERC grant 327620-09.
Chaitanya Swamy: Supported in part by NSERC grant 327620-09.

1 Introduction

The *orienteering* problem, first introduced by [16], is a fundamental and widely-studied vehicle-routing problem (VRP). The input to the problem consists of a length/travel bound B, finite metric space (V, d) representing travel times, root vertex $\rho \in V$, and non-negative rewards

associated with the vertices. The goal is to compute a path rooted at ρ of length at most B that collects maximum reward. Orienteering often arises as a subroutine in devising algorithms for other more complex VRPs, both in approximation algorithms [4, 12, 6, 22, 13, 14, 1], as also in computational methods, where it arises as the pricing problem when using a branch-cut-and-price method on a set-covering/configuration LP.

Gupta et al. [19] introduced the following *stochastic* version of orienteering to model settings where one must spend some uncertain amount of time processing a visited node in order to collect its reward. Formally, each vertex corresponds to a job with a random, possibly correlated, processing time and reward, drawn from a given probability distribution. Random variables corresponding to different vertices are independent. The reward and processing time of a job become known only when the job is fully processed. The goal is to devise an algorithm, also called *policy*, that visits a sequence of vertices (starting at ρ) in a possibly *adaptive* fashion that maximizes the expected total reward collected, subject to the constraint that the total time expended in traveling and processing jobs is at most B. Jobs cannot be preempted, and only jobs completed by the time-horizon B yield reward. This is the *correlated orienteering* (CorrO) problem. We refer to the special case where rewards and sizes are independent, simply as *stochastic orienteering*; due to independence, one can move to deterministic rewards by replacing the random rewards with their expectations.

A related problem, and the focus of this paper, is *correlated knapsack orienteering* (CorrKO), wherein there are two separate budgets: B for the (deterministic) travel time, and W for the total time spent in processing jobs. Again, we refer to the uncorrelated problem as *stochastic knapsack orienteering*. Correlated knapsack orienteering can be motivated from a similar perspective as CorrO. Indeed, it is quite natural to decouple the "apples and oranges" entities of travel time and processing time when these may represent disparate resources; e.g., travel time may represent latency of access of jobs in a distributed network, and processing time may present CPU time.

In general, a policy may be *adaptive* and choose the next vertex to visit based on the (size, reward) realizations of the vertices previously visited; unless otherwise stated, the approximation ratio is always measured relative to the maximum reward OPT that can be achieved by an adaptive policy. On the other hand, a *non-adaptive* policy fixes beforehand the sequence of vertices to visit, and only the stopping-point (when the time-horizon B is exceeded) depends on the (size, reward) realizations. While adaptive policies may collect much greater reward, non-adaptive policies are usually easier to implement, specify, and analyze, by virtue of the fact that they admit a much-more compact description compared to the decision tree associated with an adaptive policy whose description may require space that is exponential in the input size. Consequently, much work in stochastic optimization has focused on developing good non-adaptive policies and obtaining bounds on the *adaptivity gap*, which is the supremum, over all problem instances, of $OPT/$(maximum reward achieved by a non-adaptive policy); see e.g., [9, 17, 11, 18, 19, 7, 20].

Prior work has studied stochastic orienteering, CorrO, and stochastic knapsack orienteering, and our current knowledge for these problems can be summarized as follows. (1) The adaptivity gap for stochastic orienteering is $O(\log\log B)$ [19] and $\Omega(\sqrt{\log\log B})$ [2], and there is a non-adaptive algorithm that achieves and $O(1)$-approximation with respect to the *non-adaptive optimum* [19], and hence obtains an $O(\log\log B)$-approximation; the approach leading to the latter result also yields an $O(1)$-approximation for *stochastic knapsack orienteering*. (2) The adaptivity gap for CorrO is also $O(\log\log B)$ [2], but this is established non-constructively; algorithmically, we can obtain $O(\alpha \log B)$-approximation in polytime [19] and $O\big(\alpha \cdot \frac{\log^2 \log B}{\log\log\log B}\big)$ in quasi-polytime [2], where α is the approximation ratio for *deadline-TSP*.

To our knowledge, *there is no prior work on* CorrKO. As noted above, (uncorrelated) stochastic knapsack orienteering and CorrO admit quite different guarantees, and this raises the natural question: where does CorrKO stand in terms of difficulty relative to these two problems? Is it more difficult than the uncorrelated problem? How does it compare in difficulty relative to CorrO?

Our contributions

We initiate a study of correlated knapsack orienteering, and obtain results that, in particular, shed light on these questions. Our chief contributions are as follows.

1. **Adaptivity gap and approximation algorithms.** Somewhat surprisingly, and in stark contrast with (uncorrelated) stochastic knapsack orienteering, we prove that *the adaptivity gap for* CorrKO *is not a constant*, showing that the correlated problem is strictly harder than the uncorrelated problem.

   ▶ **Theorem 1.1** (see Section 3)**.** *The adaptivity gap for* CorrKO *is* $\Omega\big(\max\{\sqrt{\log B}, \sqrt{\log\log W}\}\big)$*, where* B *is the travel budget and* W *is the processing-time budget.*

   Complementing this, we develop various *non-adaptive* approximation algorithms for CorrKO. Our main algorithmic result is a *quasi-polytime* $O(\log\log W)$*-approximation algorithm* for CorrKO, which thus shows that the adaptivity gap is $O(\log\log W)$.

   ▶ **Theorem 1.2.** *There are non-adaptive algorithms for* CorrKO *with the following guarantees:*

   (a) $O(\log\log W)$*-approximation in time* $(n+\log B)^{O(\log W \log\log W)}$ *(Section 4.1);*

   (b) $O(\log W)$*-approximation in polynomial time (Section 4.2).*

   By leveraging the approach leading to Theorem 1a, we also obtain the following guarantee for correlated orienteering, which improves upon the approximation guarantee in [2] (that also runs in quasi-polytime) by an $O\big(\frac{\log\log B}{\log\log\log B}\big)$-factor.

   ▶ **Theorem 1.3.** *Given an* α*-approximation algorithm for deadline TSP with running time* T*, we can obtain a non-adaptive* $O(\alpha\log\log B)$*-approximation algorithm for* CorrO *with running time* $(n+\log B)^{O(\log B\log\log B)}\cdot T$*. Using the algorithm for deadline TSP in [15], we obtain an* $O(\log\log B)$*-approximation in quasi-polytime.*

2. **CorrKO with 2-point distributions.** Our adaptivity-gap lower bound uses distributions of support-size 3, whereas the $\Omega(\sqrt{\log\log B})$ adaptivity-gap lower-bound example for stochastic orienteering in [2] considers *weighted Bernoulli size distributions*. In Section 5, we investigate CorrKO from a fine-grained-complexity perspective to understand this discrepancy. In contrast with stochastic orienteering, we show that when all distributions are supported on at most 2 points – we call this 2CorrKO– *the adaptivity gap becomes* $O(1)$ (Theorem 5.3), and we can obtain a *non-adaptive* $O(1)$*-approximation in time* $(n+\log B)^{O(\log W)}$ (Theorem 5.4). Moreover, for weighted Bernoulli size distributions, we obtain a *polytime non-adaptive* $O(1)$*-approximation* (Theorem 5.5). Our key insight here lies in identifying a novel *deterministic* problem, that we call *orienteering with knapsack deadlines* (OrientKD), which we show is equivalent to 2CorrKO, up constant factors. In OrientKD, in addition to orienteering, each vertex v has a weight and knapsack deadline, and an orienteering-solution P is feasible, if for every $v\in P$, the total weight of all nodes on P up to (and including) v is at most its knapsack deadline. For instance, in a setting where jobs distributed over a network have to be processed on a single machine, travel times could represent the latency involved in accessing a job, and the knapsack deadlines would capture completion-time deadlines on the machine. We obtain

the above approximation guarantee for OrientKD (Theorem 5.1), and hence obtain the same guarantee (up to constant factors) for 2CorrKO.
These results show that our adaptivity-gap example is tight in terms of the support-size: *any* such lower-bound example *must* involve some distribution of support-size at least 3.

3. **CorrKO with cancellations.** In this version (see Section 6), we can *cancel* or discard the current vertex v at any time-step prior to its completion, foregoing its reward, and we are not allowed to return to v. We obtain the same approximation guarantees for this problem as for CorrKO: i.e., quasi-polytime $O(\log\log W)$-approximation, and polytime $O(\log W)$-approximation. En route, we obtain an $O(1)$-approximation for the special case where we obtain non-zero rewards only when jobs instantiate to size at most $W/2$.

Our results paint a nuanced picture of the complexity of CorrKO vis-a-vis CorrO and stochastic knapsack orienteering. While CorrKO is harder than stochastic knapsack orienteering, our algorithmic results suggest that it is easier than CorrO. We obtain similar approximation factors for both problems in quasi-polytime, but in polytime, we obtain $O(\log W)$-approximation for CorrKO, while the current-best polytime factor for CorrO is $O(\log n \log B)$; also, with weighted Bernoulli distributions, CorrKO is provably easier than CorrO.

Technical overview. We now highlight the key technical ideas underlying our results. Let OPT be the optimal reward for CorrKO. Let S_v denote the random size of vertex v. For an integer $j \geq 0$, let $X_v^j := \min\{S_v, 2^j\}$ and $\mu_v^j = \mathbb{E}\left[X_v^j\right]$. The significance of these quantities is that if $\mu^j(P_{\rho,v} - v) \leq c \cdot 2^j$, where P is a rooted path, $v \in P$, and $P_{\rho,v}$ is the $\rho \rightsquigarrow v$ portion of P, then a random subpath P'' of P where we retain each $u \in P$ independently with probability $\frac{1}{2c}$ satisfies $\Pr[v \in P''$ and is processed by time $2^j] \geq \frac{1}{4c}$; this indicates that $\pi_v(2^j)$, which is the expected reward of v if its processing starts by time 2^j, can serve as a good proxy for the expected reward obtained from v.

Algorithms for CorrKO and CorrO. Our quasi-polytime $O(\log\log W)$-approximation for CorrKO builds upon a structural result for CorrO shown by [2]. They show that one can extract a suitable rooted path Q^* from the decision tree representing an optimal adaptive policy and suitable nodes $\varphi_{-1} = \rho, \varphi_0, \varphi_1, \ldots, \varphi_k$ on Q^*, where $k \leq \log W$, such that (roughly speaking): (a) the prefix property $\mu^j(Q^*_{\rho,\varphi_j} - \varphi_j) \leq O(K) \cdot 2^j$ holds for every $j = 0, \ldots, k$, and (b) $\sum_{j=0}^{k} \sum_{v \in Q^*_{\varphi_{j-1},\varphi_j}} \pi_v(2^j) = \Omega(OPT)$, where $K = O(\log\log W)$ (see Theorem 7.1). So if we could find this path Q^*, then using the sampling idea described above, one can easily obtain an $O(K)$-approximation. For CorrO, Bansal and Nagarajan [2] "guess" the *portal nodes* $\varphi_0, \ldots, \varphi_k$ and write a configuration LP to find suitable paths between every pair of consecutive portal nodes. They use randomized rounding to round a fractional solution, which incurs a $\frac{\log k}{\log\log k}$-factor violation of the prefix property due to Chernoff bounds, since for each j, $\mu^j(Q^*_{\rho,\varphi_j} - \varphi_j)$ can be written as a sum of $O(K) \cdot 2^j$-bounded independent random variables. When one combines this with the node-sampling step, one therefore incurs an $O\big(K \cdot \frac{\log k}{\log\log k}\big)$-factor loss relative to the value of the LP solution.

For CorrKO (and CorrO), we proceed similarly, but we guess many more portal vertices. We split each $Q^*_{\varphi_{j-1},\varphi_j}$ into $O(K)$ segments having μ^j-weight at most 2^j, and guess the end-points of all such segments (see Theorems 4.1, 4.2). We then again set up a configuration LP and use randomized rounding; however, we can now ensure that the prefix property holds with $O(1)$ violation, since we can decompose $\mu^j(Q^*_{\rho,\varphi_j} - \varphi_j)$ into a *sum of* 2^j*-bounded random variables* corresponding to the μ^j-weight of each random segment. Thus, an application of Chernoff bounds and the union bound only incurs an $O(1)$-factor violation of the prefix property,

since $K = \Omega(\log k)$; therefore, we lose only an $O(K)$-factor compared to the value of the LP solution. This idea extends to CorrO. The only essential difference between CorrKO and CorrO comes from how well we can solve the corresponding configuration LP; for CorrKO, we can obtain an $O(1)$-approximation to the LP-optimum using an $O(1)$-approximation algorithm for knapsack orienteering (see below), but for CorrO, we obtain an $O(\alpha)$-approximate LP solution given an α-approximation for deadline TSP.

The $O(\log W)$-approximation for CorrKO proceeds by relating the problem to *knapsack orienteering* (KnapOrient), which is orienteering with an additional total-node-weight budget constraint. For each index $j = 0, 1, \ldots, \log W$, we use the portion of the optimal adaptive-policy tree corresponding to nodes processed at some point in $[2^j, 2^{j+1})$, to extract a good *fractional solution to an LP-relaxation* (KO-LP) *for* KnapOrient, where we exploit the LP-relaxation for orienteering [14]. This translation is easy because one can naturally interpret the LP variables as corresponding to certain probabilities obtained from an adaptive policy.

We remark that one can combine the LP-relaxations for orienteering [14] and the *correlated knapsack* problem [18], which is the special case where all nodes are co-located, to obtain an LP for CorrKO. However, the chief impediment in rounding an LP solution is that the rounding algorithms for orienteering and correlated knapsack may give rise to incompatible orderings. Rounding the orienteering-portion of the LP solution yields a node sequence, and we need to stick with a subsequence of this to satisfy the travel-budget constraint. However, forcing one to consider items in a prescribed order for correlated knapsack can drastically reduce the reward obtained, because jobs that instantiate to large sizes (i.e., $> W/2$) may need to be processed in a different incompatible order; see Appendix A. This tension is real, as evidenced by our adaptivity-gap lower bound, and seems challenging to deal with.

2CorrKO. The chief insight here is that the problematic case where we obtain reward only from large-size instantiations becomes quite structured in two ways. (1) There is no adaptivity gap, since only the path in the adaptive-policy tree corresponding to small-size (i.e., $\leq W/2$) instantiations can yield non-zero reward. (2) Given (1), one can infer that the reward obtained from a vertex v is a function of the total small size, and total large-size-instantiation probablity of vertices visited up to v. This allows one to define an instance of *orienteering with knapsack deadlines* (OrientKD) to capture the stochastic problem.

CorrKO with cancellations. The algorithm for CorrKO with cancellations considers two cases. For large-size instantiations, it is not hard to argue that cancellations do not help (as with correlated knapsack [18]). For small-size instantiations, we formulate an LP by combining the LPs for orienteering [14] and *correlated knapsack with cancellations* [18]. We show that from an LP solution, one can define a suitable KnapOrient-instance and extract a good LP solution for this KnapOrient-instance. The KnapOrient-instance is defined in such a way that feasible solutions to this instance can be mapped to fractional solutions to the correlated-knapsack LP. So we can first round the solution to obtain an integral KnapOrient-solution Q, and then utilize the LP-rounding algorithm in [18] for correlated knapsack with cancellations to process vertices, with cancellations, *in the order they appear on* Q. It is crucial here that the algorithm in [18] for small-size instantiations has the flexibility that one can specify a prescribed order for considering vertices (unlike in CorrKO with large-size instantiations).

Related work

As mentioned earlier, *orienteering* is a fundamental problem in combinatorial optimization that finds various applications. Blum et al. [5] devised the first constant-factor approximation algorithm for orienteering, and the current best approximation factor is $(2 + \epsilon)$ for any

$\epsilon > 0$ [8]. Friggstad and Swamy [14] gave the first LP-based $O(1)$-approximation algorithm. Their LP plays an important role for obtaining some of our results. *Deadline TSP*, also known as *deadline orienteering*, is a generalization of orienteering, where nodes now have deadlines, and a path P is feasible if, for every $v \in P$, its travel time along P is at most its deadline; the goal is again to compute a maximum-reward feasible path. Both orienteering and deadline TSP can be considered in the rooted, or *point-to-point* (P2P) setting, where both the start and end nodes of the path are specified. Deadline TSP admits a polytime $O(\log n)$-approximation [1] and an $O(1)$-approximation in time $n^{O(\log(\text{maximum deadline}))}$ [15]. Friggstad and Swamy [15] also consider the more general *monotone-reward TSP*, wherein the the reward of a node v having travel time t is given by $\mathsf{rewd}_v(t)$, where $\mathsf{rewd}(.)$ is a non-increasing function. They showed that this problem is essentially equivalent to deadline TSP.

The literature on stochastic optimization problems is rich, and we discuss below only the work that is most relevant to our work.

- **Stochastic knapsack problems.** Stochastic orienteering and CorrKO generalize respectively *stochastic knapsack*, which was studied in the seminal work of [9], and *correlated knapsack* [18, 21], which correspond to the special case where all nodes are co-located (i.e., the travel budget is irrelevant). The state-of-the-art for stochastic knapsack is a $(2+\epsilon)$-approximation [3]. Gupta et al. [18] obtained the first constant-factor approximation for correlated knapsack, and the constant was improved to $(2+\epsilon)$ by Ma [21].
- **Stochastic VRPs.** We have already mentioned the works of Gupta et al. [19] and [2] that consider (uncorrelated) stochastic orienteering and correlated orienteering. A minimization version of stochastic orienteering, called *stochastic k-TSP* was considered by [11, 20], where instead of a travel budget, we want to collect a reward of at least k, and seek to minimize the expected travel time. Ene et al. [11] gave an adaptive $O(\log k)$-approximation algorithm for this problem, and Jiang et al. [20] obtained a non-adaptive $O(1)$-approximation. The special case where all nodes are co-located is called *stochastic knapsack cover* for which [10] obtained a $(2+\epsilon)$-approximation.
- **Multi-armed bandits with metric switching costs.** A related problem to CorrKO is the *multi-armed bandit* problem with metric switching costs, considered by Guha and Munagala [17], which can be viewed as a setting where each vertex corresponds to a Markov chain (i.e., arm) with known transition probabilities and rewards. Guha and Munagala consider this setting under a crucial *martingale assumption*, which does not hold for CorrO or CorrKO, with separate budgets for the travel-cost and the number of arm-pulls, as in CorrKO. In their setting, one can also abandon a vertex and possibly return to this vertex at a later time. They devise an $O(1)$-approximation algorithm for this problem that is a hybrid between adaptive and non-adaptive policies: it non-adaptively specifies the sequence of arms to visit, but adaptively decides when an arm should be abandoned. They use an elegant Lagrangian-relaxation idea to reduce the problem to orienteering; this Lagrangian-relaxation idea was also later used in [19].

2 Preliminaries and notation

For an integer $n \geq 0$, we use $[n]$ to denote $\{1, \ldots, n\}$, where $[0] := \emptyset$, and $[\![n]\!]$ to denote $\{0\} \cup [n]$. For any universe U, set $S \subseteq U$ and element $e \in U$, we sometimes use $S - e$ and $S + e$ to denote $S \setminus \{e\}$ and $S \cup \{e\}$ respectively.

The problems we consider involve a metric space (V, d) and root $\rho \in V$. The metric $d : V \times V \mapsto \mathbb{Z}_{\geq 0}$ is symmetric and captures travel times between vertices; by scaling we may assume that these are integers. Let $n = |V|$ and Δ be the diameter of the metric space. For

a set S of edges of the underlying complete graph (V, E), we use $d(S)$ to denote $\sum_{e \in S} d(e)$. Similarly, for any $f \in \mathbb{R}^V$ and $U \subseteq V$, $f(U)$ denotes $\sum_{v \in U} f_v$. We say that a path P in G is rooted if it begins at ρ. We always think of the nodes on a rooted path P as being ordered in increasing order of their distance from ρ along the path. For any $u, w \in P$, we say $u \prec_P w$ to denote that u comes before w on P, and $u \preceq_P w$ means that $u = w$ or $u \prec_P w$; we omit the subscript P when P is clear from the context. We will interchangeably think of a path as an edge-set, or a sequence of nodes; the meaning will be clear from the context. For any path P and nodes $a, b \in P$, we use $P_{a,b}$ to denote the a-b portion of P. For a path P starting at node r, and a node $v \in P$, we define the travel time of v as $d(P_{r,v})$.

Deterministic max-reward vehicle routing. The following three vehicle routing problems (VRPs) play a prominent role in the study of stochastic orienteering. All three problems fall in the genre of max-reward VRPs, wherein we have nonnegative node rewards $\{\pi_v\}_{v \in V}$, and we need to select some vertices and find a suitable path visiting these vertices, so as to maximize the reward obtained. The differences in the problems lie in which paths are allowed, and the definition of the reward collected by a path. The problems below can be considered in the *rooted* setting, where we have a root ρ and the feasible paths form a subset of rooted paths, or in the *point-to-point* (P2P) setting, where both a start-node a and end-node b are specified, and the feasible paths are a subset of a-b paths.

- **Orienteering.** We have a budget B, and feasible paths (in the rooted and P2P versions) are those with length at most B; we collect the reward of all nodes on a feasible path.
- **Deadline TSP**, also called **deadline orienteering.** Here nodes have deadlines $\{D_v\}_{v \in V}$. A path P with the appropriate end-points is feasible if the travel time of each node in P is at most its deadline. So in the rooted case, a rooted path P is feasible if $d(P_{\rho,v}) \leq D_v$ for all $v \in P$; in the P2P-case, an a-b path P is feasible if $d(P_{a,v}) \leq D_v$ for all $v \in P$. We collect the reward of all nodes on a feasible path. (Equivalently, one can say that the feasible paths are *all* paths with the prescribed end-points, and we collect the reward from all nodes on the path that are visited *by their deadlines*.)
  Observe that orienteering is the special case where the deadline of each node is the length bound B. Also, the rooted and P2P versions of deadline TSP are equivalent [14].
- **Monotone-reward TSP.** This is a generalization of deadline TSP, where each node v has a non-increasing reward-function $\pi_v : \mathbb{Z}_+ \mapsto \mathbb{R}_+$, where $\pi_v(t)$ gives the reward obtained from v if v is visited at time t. Every path P with the appropriate end-points is feasible, and the reward of P is given by $\sum_{v \in P} \pi_v(\text{travel time of } v) = \sum_{v \in P} \pi_v\big(d(P_{r,v})\big)$, where r is the start node of P.
  Friggstad and Swamy [14] showed that monotone-reward TSP can be reduced to deadline TSP losing a $(1 + \epsilon)$-factor, for an $\epsilon > 0$. Monotone-reward TSP will play a key role in the algorithm for correlated orienteering.

Stochastic orienteering problems. In the *correlated knapsack orienteering* (CorrKO) problem, each vertex $v \in V$ is associated with an stochastic job with a random processing time or size $S_v \in \mathbb{Z}_{\geq 0}$ and a possibly *correlated* random reward $R_v \in \mathbb{R}_{\geq 0}$. We use the terms processing time and size interchangeably. These random variables are independent across different vertices, and their distributions are specified in the input. We are given a length or travel-time budget B, and a processing-time budget W. A solution, or policy, for CorrKO visits a sequence of (distinct) vertices starting from the root ρ, in a possibly adaptive fashion, without exceeding the travel-time and processing-time budgets. More precisely, when a vertex v is visited, it's corresponding job is processed non-preemptively, and we get to know

the processing time and reward of the job only upon its completion; the completion time of job v is the total processing time of all jobs up to and including v. So if the adaptive policy visits vertices $v_0 := \rho, \ldots, v_\ell = u$ in that order, then it must be that the total travel-time $\sum_{i=1}^{\ell} d(v_{i-1}, v_i)$ to get to u is at most B, and the total processing time of (the jobs associated with) $v_1, \ldots, v_{\ell-1}$ is at most W. We collect the rewards of $v_1, \ldots, v_{\ell-1}$, and we collect u's reward if its completion time is at most W. The goal is to maximize the expected total reward collected. For notational convenience, we also assign a deterministic value of 0 to the reward and processing time of ρ.

In the *correlated orienteering* (CorrO) problem, the setup is almost the same as in CorrKO, except that there is only one budget B, which is the budget for the *sum* of the travel times and processing times. (That is, we have one notion of time, which advances due to both travel and the processing of jobs.) So if an adaptive policy for CorrO visits vertices $v_0 := \rho, \ldots, v_\ell = u$ in that order, then we must have $\sum_{i=1}^{\ell} d(v_{i-1}, v_i) + \sum_{i=1}^{\ell-1} S_{v_i} \leq B$; that is, the completion time of each v_i for $i = 1, \ldots, \ell - 1$, as also the time when we reach v_ℓ, *taking into account both travel time and processing time*, should be at most B. We collect rewards from $v_1 \ldots, v_{\ell-1}$, and we collect u's reward if $\sum_{i=1}^{\ell} d(v_{i-1}, v_i) + \sum_{i=1}^{\ell} S_{v_i} \leq B$.

Any adaptive policy for CorrKO or CorrO can be represented by a decision tree $\mathcal{T}$ rooted at ρ, whose nodes are labeled by vertices of V, and the branches of a node labeled $v \in V$ correspond to the different size and reward instantiations of v, with each branch specifying the next node to visit under the corresponding instantiation.

A *nonadaptive policy* (for CorrKO or CorrO) fixes a priori the sequence of vertices to potentially visit, *without looking at the size and reward instantiations.* The *adaptivity gap* for an instance is the ratio (optimal expected reward collected by an adaptive policy)/(optimal reward collected by a nonadaptive policy), and the adaptivity gap for a problem is the supremum over all instances of the adaptivity gap for the instance.

Deterministic knapsack-constrained vehicle routing. Algorithms for stochastic orienteering problems frequently utilize knapsack-constrained variants of deterministic VRPs, wherein we seek a feasible solution to the VRP satisfying an additional knapsack constraint on the total vertex-weight of the path. More precisely, suppose we have an underlying "base" max-reward VRP, specified by a collection $\mathcal{I}$ of feasible paths along with nonnegative vertex-rewards $\{\pi_v\}_{v \in V}$, where the goal is to find a maximum-reward path in $\mathcal{I}$. In the *knapsack-constrained version of this* VRP, we also have a knapsack constraint specified by nonnegative knapsack weights $\{\mathsf{wt}_v\}_{v \in V}$ and knapsack budget W, which restricts the set of feasible solutions to $\mathcal{I}^{\mathsf{knap}} := \{\tau \in \mathcal{I} : \sum_{v \in \tau} \mathsf{wt}_v \leq W\}$; the goal is to find a maximum-reward path in $\mathcal{I}^{\mathsf{knap}}$, i.e., a maximum-reward path in $\mathcal{I}$ satisfying the knapsack constraint. When the base VRP is: (i) orienteering, the knapsack-constrained version is *knapsack orienteering* (KnapOrient); (ii) deadline-TSP, the knapsack-constrained version is *knapsack deadline orienteering* (KnapDO). KnapOrient and KnapDO were considered by [19, 2] in the context of stochastic orienteering. We say that the base-VRP is a rooted-VRP, if all paths in $\mathcal{I}$ start at the same vertex, and it is a P2P-VRP, if all paths in $\mathcal{I}$ have the same start and end nodes.

We give a general reduction (Theorem 2.1) showing if the base-VRP is a rooted-VRP or P2P-VRP, and satisfies a certain subpath-closure property, then an α-approximation for the VRP can be used as a black-box to obtain an $(\alpha + 2)$-approximation for the knapsack-constrained VRP. Let τ be a path with ends $a, b \in V$, which we will view as a sequence of nodes. By a P2P-*subpath* of τ, we mean any a-b path whose node-sequence is a subsequence of τ; by a *rooted-subpath* of τ, we mean any path starting at a whose node-sequence is a subsequence of τ. (Note that any subsequence of τ yields a path, since we are working with

a complete graph.) The *subpath-closure property* requires that for every path $\tau \in \mathcal{I}$: (a) for rooted-VRP, every rooted-subpath τ' of τ is also in $\mathcal{I}$, (b) for P2P-VRP, every P2P-subpath τ' of τ is also in $\mathcal{I}$. Most max-reward VRPs – e.g., orienteering, deadline TSP – satisfy the subpath-closure property. (Also, note that if a VRP satisfies the subpath-closure property, then so does the knapsack-constrained VRP.)

The above reduction is based on a Lagrangian-relaxation idea that was also used by [19], specifically to obtain approximation algorithms for KnapOrient and KnapDO. However, their approach results in a constant-factor blowup -in the approximation ratio (factor 2 for KnapOrient, and factor 4 for KnapDO[1] when going from the VRP to the knapsack-constrained VRP; our general reduction yields a better factor, in a somewhat simpler fashion.

▶ **Theorem 2.1.** *Consider a max-reward rooted-*VRP *or* P2P-VRP*, specified by a set $\mathcal{I}$ of feasible solutions satisfying the subpath-closure property. For any $\epsilon > 0$, an α-approximation algorithm $\mathcal{A}$ (where $\alpha \geq 1$) for the* VRP *can be used to obtain an $(\alpha+2)(1+\epsilon)$-approximation for the knapsack-constrained* VRP *by making $O\left(\frac{\log n}{\epsilon}\right)$ calls to $\mathcal{A}$.*

▶ **Corollary 2.2.** *There are algorithms with the following guarantees.*

(a) *$(4+\epsilon)$-approximation, for any $\epsilon > 0$, for rooted- and* P2P- *knapsack orienteering;*

(b) *$O(\log n)$-approximation for the rooted and* P2P *versions of knapsack deadline orienteering, and knapsack monotone-reward TSP;*

(c) *$O(1)$-approximation in $O\left(n^{\log n \Delta}\right)$ time, for the rooted and* P2P *versions of knapsack deadline orienteering, and knapsack monotone-reward TSP.*

LP-relative guarantee for KnapOrient. For rooted KnapOrient, we can utilize Theorem 2.1 to obtain an LP-relative approximation guarantee. This will be useful in devising algorithms for CorrKO. Consider the following LP-relaxation for rooted KnapOrient along the lines of an LP-relaxation for rooted orienteering in [14]. Let ρ be the root node for the KnapOrient instance. We bidirect the edges of the complete graph on V to obtain the arc-set A.

$$\max \quad \sum_{u,v \in V} z_u^v \cdot \pi_u \tag{KO-LP}$$

$$\text{s.t.} \quad x^v\left(\delta^{\text{in}}(u)\right) \geq x^v\left(\delta^{\text{out}}(u)\right) \qquad \forall u \in V - \rho,\ v \in V \tag{O1}$$

$$x^v\left(\delta^{\text{in}}(S)\right) \geq z_u^v \qquad \forall v \in V,\ S \subseteq V - \rho,\ u \in S \tag{O2}$$

$$z_u^v = 0 \qquad \forall u, v \in V : d_{\rho,u} > d_{\rho,v} \tag{O3}$$

$$\sum_{a \in A} d_a \cdot x_a^v \leq B z_v^v, \quad x^v\left(\delta^{\text{out}}(\rho)\right) = z_v^v \qquad \forall v \in V \tag{O4}$$

$$x, z \geq 0$$

$$\sum_{v \in V} z_v^v = 1 \tag{O5}$$

$$\sum_{u,v \in V} z_u^v \cdot \mathsf{wt}_u \leq W. \tag{KN}$$

The x_a^v and z_u^v variables encode the arcs included, and vertices visited, respectively by the KnapOrient-path, provided that v is the node visited that is furthest from ρ, i.e., v maximizes $d(\rho, u)$ among all nodes u on the path: constraints (O3) enforce this semantics; in an integer solution, these variables will be 0 if v is not the furthest visited node from ρ. Constraints

[1] [19] do not explicitly state a result for KnapDO, and instead embed this result within their algorithm for correlated orienteering. We can infer this factor by tracing through their algorithm and analysis.

(O1) and (O2) encode that the $\rho \rightsquigarrow u$-connectivity is z_u^v, and together with (O4) encode that $\{x_a^v\}$ is a ρ-preflow of value z_v^v satisfying the length budget. Constraint (O5) enforces that overall x is a ρ-preflow of value 1. Constraints (O1)–(O5) are from the LP for rooted orienteering in [14]; (KN) is the new constraint encoding the knapsack budget.

▶ **Theorem 2.3.** *We can obtain a* KnapOrient*-solution that obtains reward at least* $OPT_{\text{KO-LP}}/5$.

3 An adaptivity-gap lower bound for CorrKO

We now show that the adaptivity gap for CorrKO is $\Omega\big(\max\{\sqrt{\log B}, \sqrt{\log\log W}\}\big)$, thereby proving Theorem 1.1. We consider the following instance of correlated knapsack orienteering that has a similar spirit as the adaptivity-gap example in [2] for (uncorrelated) stochastic orienteering. The metric is a tree-metric induced by a complete binary tree T on a vertex set V, with root $r \in V$ and $H \geq 4$ levels, where the distances decrease geometrically as we move away from r. To conform to our notation, we include a separate dummy node ρ that serves as the root for CorrKO, with distance 0 to r; but when we say root below, we always mean the root r of the tree T. The knapsack budget is $W := 2^{2^{H+1}}$ and the length/travel budget is $B := 2^{H-1} - 1$. For a node $v \in V$, we use: $\mathsf{lev}(v)$ to denote the level of v, $\mathsf{path}(v)$ to denote the unique $r \rightsquigarrow v$-path in T, and $\mathsf{par}(v)$ to denote the parent of v if $v \neq r$. The root r is at level H and each leaf node is at level 1; for a non-leaf node v at level ℓ, the distance between v and its children is $2^{\ell-2}$. For a rooted path P in T we say that a node $v \in P$ is a right-branching (resp. left-branching) node if the node succeeding v on P is its right-child (resp. left-child). We denote by $\mathsf{rt}(P)$ and $\mathsf{left}(P)$, the right-branching nodes and left-branching nodes of P, respectively. For notational convenience, we assume that the end-node of P other than r is a left-branching node; if $P = r$, then we say that $r \in \mathsf{left}(P)$. The (correlated) (size, reward) distribution of node v is supported on three points:

$$\big(S_v^{(3)}, R_v^{(3)}\big) = (0,0), \qquad \big(S_v^{(2)}, R_v^{(2)}\big) = \Big(2^{2^{\mathsf{lev}(v)}} \cdot \prod_{w \in \mathsf{rt}(\mathsf{path}(v))} 2^{2^{\mathsf{lev}(w)}},\ 0\Big)$$

$$\big(S_v^{(1)}, R_v^{(1)}\big) = \left(W - \sum_{w \in \mathsf{rt}(\mathsf{path}(v))} S_w^{(2)},\ \left(1 - \tfrac{1}{\sqrt{H}}\right)^{|\mathsf{rt}(\mathsf{path}(v))|}\right)$$

and we have $\Pr[S_v = S_v^{(3)}] = 1 - \frac{1}{\sqrt{H}} - \frac{1}{H}$, $\Pr[S_v = S_v^{(2)}] = \frac{1}{\sqrt{H}}$, $\Pr[S_v = S_v^{(1)}] = \frac{1}{H}$; see Fig. 1. Observe that $S_v^{(2)} \leq \big(2^{2^{\mathsf{lev}(v)+1}}\big)/2 \leq W/2$, and $S_v^{(1)} > W/2$ for every node v.

Importantly, note that any policy for this instance can obtain positive reward from at most one item. This is because for any $v \in V$, $S_v^{(1)} > W/2$. Therefore we can assume that any policy terminates upon observing a size $S_v^{(1)}$ for any visited vertex v. The binary tree is built so that a certain adaptive policy (see the proof of Theorem 3.1) can always reach a leaf-node if no positive reward has been collected in previous levels. The construction of the tree prevents any path from going upward from a node to its parent, as this will cause the length budget to run out. But more importantly, the instance is set up to preclude a policy from going to a left child of a node v if its instantiated size is $S_v^{(2)}$ in the sense that if this happens then one cannot collect positive reward from this point on (Lemma 3.4).

The adaptivity-gap lower bound immediately follows from Theorems 3.1 and 3.2, since $H = \Omega(\log B)$ and $H = \Omega(\log\log W)$ for the above CorrKO instance.

▶ **Theorem 3.1.** *There is an adaptive policy for the above* CorrKO *instance that obtains* $\Omega(1)$ *expected reward.*

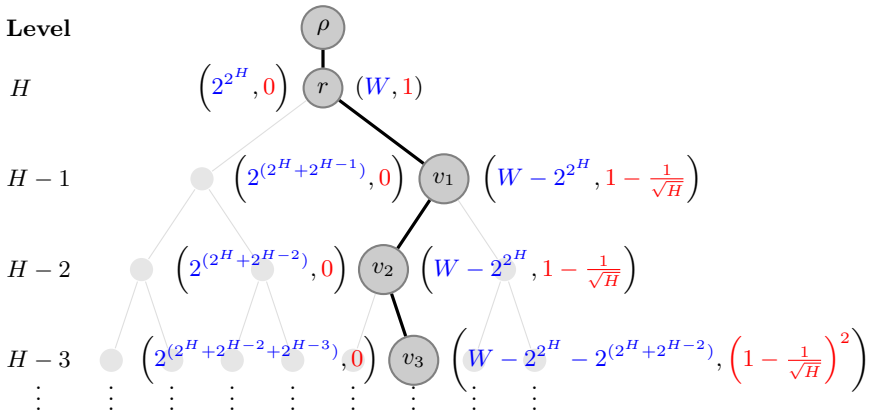

Figure 1 The $(S_v^{(2)}, R_v^{(2)})$, $(S_v^{(1)}, R_v^{(1)})$ pairs are shown respectively on the left and right of each highlighted vertex in the tree.

▶ **Theorem 3.2.** *Any nonadaptive policy for the above* CorrKO *instance obtains expected reward at most* $\frac{2}{\sqrt{H}}$.

Proof of Theorem 3.1. Consider the following adaptive policy $\mathcal{A}$: the policy moves to node r from ρ, and then proceeds as follows. Let v be the current node visited, which is r initially. If v is a leaf, then the policy ends after the instantiation of v. Otherwise, the next node visited by $\mathcal{A}$ is: the left child of v, if $S_v = S_v^{(3)}$, and the right child of v if $S_v = S_v^{(2)}$; if $S_v = S_v^{(1)}$, then $\mathcal{A}$ stops and does not visit any other nodes.

Let P^* denote the (random) path traversed by $\mathcal{A}$, which we may view as a rooted path in T. Let v_{last} be the last vertex visited by $\mathcal{A}$, i.e., v_{last} is the end-node of P^* other than r and $P^* = \mathsf{path}(v_{\mathsf{last}})$.

▷ Claim 3.3. We have $d(P^*) \le B$ and $S(P^*) := \sum_{v \in P^*} S_v \le W$ with probability 1.

We argue that the expected reward collected by P^* is at least $\frac{1-e^{-1}}{4}$. Let $\mathcal{R} = R(P^*) := \sum_{v \in P^*} R_v$ denote the reward obtained by P^*. Note that v_{last} is the only vertex from which P^* can collect positive reward. So $\mathbb{E}[\mathcal{R}] = \Pr[S_{v_{\mathsf{last}}} = S_{v_{\mathsf{last}}}^{(1)}] \cdot \mathbb{E}[R_{v_{\mathsf{last}}}^{(1)}]$. Observe that if $S_{v_{\mathsf{last}}} \neq S_{v_{\mathsf{last}}}^{(1)}$, then v_{last} is a leaf node, and hence the event $\{S_{v_{\mathsf{last}}} \neq S_{v_{\mathsf{last}}}^{(1)}\}$ occurs precisely when $\mathcal{A}$ visits H vertices, one on each level of $\mathcal{T}$, and none of them instantiate to size $S_v^{(1)}$. Since vertex sizes are independent across different vertices, we have $\Pr[S_{v_{\mathsf{last}}} \neq S_{v_{\mathsf{last}}}^{(1)}] = \left(1 - \frac{1}{H}\right)^H \le e^{-1}$, and so $\mathbb{E}[\mathcal{R}] \ge (1 - e^{-1})\mathbb{E}[R_{v_{\mathsf{last}}}^{(1)}]$.

We have $R_{v_{\mathsf{last}}}^{(1)} = \left(1 - \frac{1}{\sqrt{H}}\right)^{|\mathsf{rt}(P^*)|}$, and since $\left(1 - \frac{1}{\sqrt{H}}\right)^x$ is a convex function, we obtain that $\mathbb{E}[R_{v_{\mathsf{last}}}^{(1)}] \ge \left(1 - \frac{1}{\sqrt{H}}\right)^{\mathbb{E}[|\mathsf{rt}(P^*)|]}$. Observe that $v \in P^*$ gets included in $\mathsf{rt}(P^*)$ precisely when S_v instantiates to $S_v^{(2)}$, which happens with probability $\frac{1}{\sqrt{H}}$. So we can upper bound $\mathbb{E}[|\mathsf{rt}(P^*)|]$ by $H \cdot \frac{1}{\sqrt{H}} = \sqrt{H}$. It follows that $\mathbb{E}[R_{v_{\mathsf{last}}}^{(1)}] \ge \left(1 - \frac{1}{\sqrt{H}}\right)^{\sqrt{H}} \ge \frac{1}{4}$, where the last inequality uses the fact that $1 - x \ge 4^{-x}$ for $x \le 0.5$. ◀

Proof of Theorem 3.2. Let σ be some non-adaptive policy, which we may again view as a rooted path in T, since we can always move first to r. We may assume that σ visits vertices in decreasing order of their levels, since any backtracking from a node v to its ancestor would cause one to exceed the travel budget. We say that an execution of σ "cheats" if, for some visited node v, S_v instantiates to $S_v^{(2)}$, and σ proceeds to visit a vertex in the subtree of T rooted at the left-child of v.

▶ **Lemma 3.4.** *σ does not collect any positive reward after cheating.*

Proof. Suppose σ cheats at some vertex u. Let v be any node in the tree rooted at the left-child of u. The residual knapsack budget after visiting u is at most $W - S_u^{(2)}$. It suffices to show that $S_v^{(1)} > W - S_u^{(2)}$. Since $S_v^{(1)} = W - \sum_{w \in \mathsf{rt}(path(v))} S_w^{(2)}$, this amounts to showing that $S_u^{(2)} > \sum_{w \in A} S_w^{(2)}$, where $A = \mathsf{rt}(\mathsf{path}(v))$. We argue that $S_w^{(2)} < S_u^{(2)}$ for every $w \in A$, and the $S_w^{(2)}$'s are distinct for $w \in A$. This, coupled with the fact that $S_u^{(2)}$ and the $S_w^{(2)}$'s are all powers of 2, implies the above inequality.

Recall that for a node z, we have $S_z^{(2)} = 2^{2^{\mathsf{lev}(z)}} \cdot \prod_{w \in \mathsf{rt}(\mathsf{path}(z))} 2^{2^{\mathsf{lev}(w)}}$. Let $A = \{a_1, a_2, \ldots, a_{|A|}\}$, where the nodes are ordered in increasing order of their distance from r. Then, for any $i \geq 2$, we have $S_{a_i}^{(2)} = 2^{2^{\mathsf{lev}(a_i)}} \cdot S_{a_{i-1}}^{(2)}$, showing that each $S_{a_i}^{(2)}$ is a distinct power of 2, and $S_{a_i}^{(2)}$ increases with i. Note that $u \notin \mathsf{rt}(\mathsf{path}(v))$ and $\mathsf{rt}(\mathsf{path}(u)) \subseteq A$. So for $z = a_{|A|}$, we have $S_z^{(2)} = \prod_{w \in A - \mathsf{rt}(\mathsf{path}(u))} 2^{2^{\mathsf{lev}(w)}} \cdot \prod_{w \in \mathsf{rt}(\mathsf{path}(u))} 2^{2^{\mathsf{lev}(w)}}$ and $\prod_{w \in A - \mathsf{rt}(\mathsf{path}(u))} 2^{2^{\mathsf{lev}(w)}} < 2^{2^{\mathsf{lev}(u)}}$. It follows that $S_z^{(2)} < S_u^{(2)}$. ◀

Recall that we view σ also as a rooted path in T. We can show that the total expected reward obtained from $\mathsf{rt}(\sigma)$ and $\mathsf{left}(\sigma)$ are both at most $\frac{1}{\sqrt{H}}$, which completes the proof. For the latter bound, we utilize the fact that, due to Lemma 3.4, we can collect positive reward from a node v only if $S_w = S_w^{(3)}$ for every $w \in \mathsf{left}(\mathsf{path}(v)) - v$. ◀

4 Approximation algorithms for CorrKO

We now devise non-adaptive approximation algorithms for CorrKO. In Section 4.1, we develop an $O(\log \log W)$-approximation algorithm with $(n + \log B)^{O(\log W \log \log W)}$ (i.e., quasi-polynomial) running time, which will prove Theorem 1a, and in Section 4.2, we obtain a polytime $O(\log W)$-approximation algorithm, thereby proving Theorem 1b.

4.1 Quasi-polytime $O(\log \log W)$-approximation algorithm

There are two chief components underlying our algorithm. First, we isolate a key structural result (Theorems 4.1 and 4.2) showing that from an optimal adaptive policy, one can extract a suitable path Q^* and certain "portal" vertices on this path, such that the subpaths of Q^* between these portal vertices satisfy various nice properties. Second, we exploit this structural result algorithmically as follows. The structural result allows us to reduce the problem, at the expense of an $O(\log \log W)$-factor loss, to that of finding the portal vertices, and suitable paths between these portal vertices that satisfy certain knapsack constraints on the total expected truncated size $\mathbb{E}\big[\min\{S_v, 2^j\}\big]$ of nodes on these paths. We "guess" (i.e., enumerate over all possible choices of) these portal vertices and some auxiliary information, and set up a configuration LP (CKO-P) to find paths between these portal vertices. This configuration LP can be solved near-optimally, and we show that a fractional solution can be rounded incurring only an $O(1)$-factor loss in the objective and in the constraints. Finally, we argue that this leads to an $O(\log \log W)$-approximation non-adaptive policy.

Our approach is similar in spirit to the one in [2] for CorrO, and in Section 7, we show that our approach also yields an $O(\log \log B)$-approximation for CorrO, which improves upon the guarantee in [2] by an $O\big(\frac{\log \log B}{\log \log \log B}\big)$-factor. While we borrow various ingredients from [2], the key difference between our approach and theirs is that we extract much more information from the adaptive policy in terms of so-called portal vertices, which enables us to round an underlying configuration LP incurring only a *constant-factor* violation in the knapsack constraints; in contrast, this step in [2] incurs an $O\big(\frac{\log \log B}{\log \log \log B}\big)$-factor violation of the constraints, and this savings is the source of our improved guarantee.

Structural results. Recall that, for a path P and nodes $a, b \in P$, we use $P_{a,b}$ to denote the a-b portion of P. If P is a u-v path, its *regret* is $d^{\text{reg}}(P) := d(P) - d(u,v)$, and the *two-point regret of* P with respect to a node $a \in P$ is $d^{\text{reg}}(P,a) := d(P) - d(u,a) - d(a,v) = d^{\text{reg}}(P_{u,a}) + d^{\text{reg}}(P_{a,v})$. For an index $j \in \{0,1,\ldots,L := \lceil \log W \rceil\}$, recall that we define $X_v^j := \min\{S_v, 2^j\}$ and $\mu_v^j := \mathbb{E}\left[X_v^j\right]$. For any vertex $v \in V$, let $\pi_v(t) := \mathbb{E}\left[R_v \cdot \mathbb{1}_{S_v \le W-t}\right] = \sum_{t'=0}^{W-t} \Pr[S_v = t'] \cdot \mathbb{E}\left[R_v \,|\, S_v = t'\right]$ denote the expected reward obtained from v if its processing starts at time t. Note that $\pi_v(t) = 0$ for any $t > W$. Also, note that $\pi_\rho(t) = 0$ for all t. We may assume that $\pi_v(0) \le OPT/4$ for every $v \in V$, as otherwise, we can obtain $\Omega(OPT)$ reward by going to a single node.

Throughout, let $K = 3\log(6\log W) + 12$, $L = \lceil \log W \rceil$, $N_1 = 2(K+1)$. Define $\varphi_{-1} := \rho$.

▶ **Theorem 4.1.** *There exists a rooted path Q^* with $d(Q^*) \le B$, vertices $\varphi_0 \preceq \varphi_1 \preceq \ldots \preceq \varphi_k$ on Q^* for some $k \le L$, and, for each $j \in [\![k]\!]$, a vertex-set $\mathsf{Por}_j \subseteq Q^*_{\varphi_{j-1},\varphi_j}$ containing nodes φ_{j-1}, φ_j, with $|\mathsf{Por}_j| \le N_1$, whose vertices are ordered by the order they appear on Q^*, satisfying the following properties.*

(a) $\sum_{j=0}^{k} \sum_{v \in Q^*_{\varphi_{j-1},\varphi_j} - \varphi_j} \pi_v(2^j - 1) \ge OPT/4$.

(b) $\mu^j(Q^*_{\rho,\varphi_j} - \varphi_j) \le (K+1)2^j$ *for all* $j \in [\![k]\!]$.

(c) *For every* $j \in [\![k]\!]$ *and consecutive nodes* $a, b \in \mathsf{Por}_j$, *we have* $\mu^j(Q^*_{a,b} - b) \le 2^j$.

As mentioned earlier, in our quasi-polytime algorithm, we utilize Theorem 4.1 to construct a good rooted path, by using enumeration to guess $\bigcup_j \mathsf{Por}_j$, and an LP to then obtain suitable paths between consecutive nodes of $\bigcup_j \mathsf{Por}_j$. In order to ensure that the total path length is at most the travel budget B, we will also need to obtain some information about the lengths $d(Q^*_{a,b})$ for consecutive nodes a, b in $\bigcup_j \mathsf{Por}_j$. Naively guessing these lengths would yield incur a large $B^{O(LN_1)}$-factor in the running time; to do better, and reduce the dependence to $(\log B)^{O(LN_1)}$, we instead guess the two-point regret of each $Q^*_{a,b}$ with respect to a "mid-point" node, within a factor of 2, which suffices (see Fig. 2). We refine Theorem 4.1 to incorporate these estimates as follows.

▶ **Theorem 4.2** (Main structural result)**.** *Let the node-sequence $\varphi_0, \ldots, \varphi_k$, where $k \le L$, and for each $j \in [\![k]\!]$, the ordered node sequence Por_j of at most N_1 nodes, be as given by Theorem 4.1. Define $\mathsf{Por} := \bigcup_{j=0}^{k} \mathsf{Por}_j$, which we call "portal nodes", where the ordering of nodes in Por is Por_0, Por_1, …, Por_k; for $a \in \mathsf{Por}$, $a \ne \varphi_k$, let $\mathsf{next}(a)$ be the next node in Por after a. For each $a \in \mathsf{Por} - \varphi_k$, there exists an a-$\mathsf{next}(a)$ path $Q^*_{a,\mathsf{next}(a)}$, auxiliary node m_a, and integer $\gamma_a \ge 0$, such that the following properties hold.*

(P1) (Distance) $d(Q^*_{a,b}) \le D_a := 2^{\gamma_a} - 1 + d(a,m_a) + d(m_a,b)$ *for every pair of consecutive nodes* $a, b \in \mathsf{Por}$.

(P2) (Total-length) $\sum_{a \in \mathsf{Por} - \varphi_k} D_a \le B$.

(P3) (Reward) $\sum_{j=0}^{k} \sum_{a \in \mathsf{Por}_j - \varphi_j} \sum_{v \in Q^*_{a,\mathsf{next}(a)} - \mathsf{next}(a)} \pi_v(2^j - 1) \ge OPT/8$.

(P4) (Prefix-size) $\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_h} \mu^j\left(Q^*_{a,\mathsf{next}(a)} - \mathsf{next}(a)\right) \le (K+1)2^j$ *for all* $j \in [\![k]\!]$.

(P5) (Size) $\mu^j(Q^*_{a,b} - b) \le 2^j$ *for every* $j \in [\![k]\!]$ *and pair of consecutive nodes* $a, b \in \mathsf{Por}_j$.

Configuration LP and non-adaptive algorithm. Now assume that we have found, by enumeration, nodes $\varphi_0, \ldots, \varphi_k$, where $k \le L$, ordered node-sets Por_j for $j \in [\![k]\!]$, and length bounds D_a for every pair of consecutive nodes $a, b \in \mathsf{Por} := \bigcup_{j=0}^{k} \mathsf{Por}_j$, as stipulated by Theorem 4.2. (We also need to enumerate for $\{m_a, \gamma_a\}_{a \in \mathsf{Por} - \varphi_k}$; we do not use these quantities directly, but these are used to specify the D_a length bounds.) That is, these objects are compatible with suitable $Q^*_{a,b}$ paths such that P1–P5 hold. Clearly, this enumeration takes $(n \log B)^{O(N_1 L)} = (n \log B)^{O(\log W \log\log W)}$ time, which is the source of the running time in Theorem 1a.

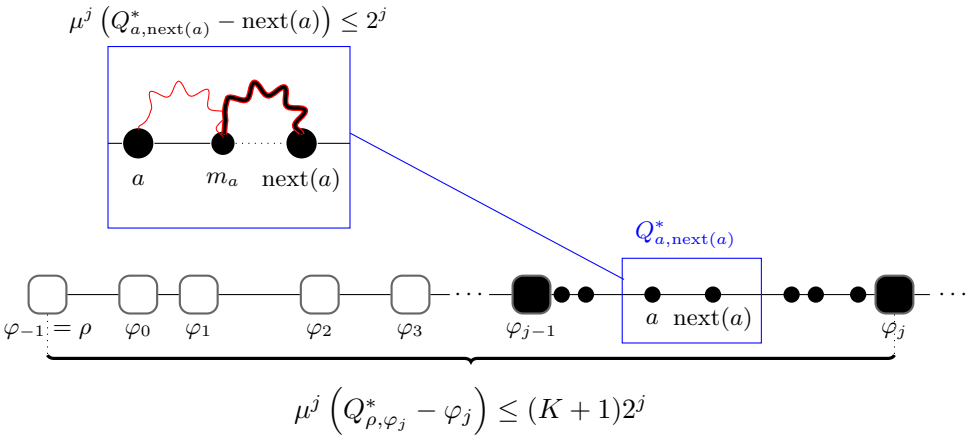

Figure 2 Portal nodes Por and paths between portal nodes. The solid nodes depict Por_j.

We formulate a configuration LP to find a-b paths, for every pair of consecutive nodes $a, b \in \mathsf{Por}$, satisfying properties P1, P3–P5. To this end, fix some $j \in [\![k]\!]$ and $a \in \mathsf{Por}_j - \varphi_j$, and let $b = \mathsf{next}(a)$. The valid a-b paths (i.e., the configurations) are the solutions to the following (deterministic) *point-to-point knapsack orienteering* (KnapOrient) problem: the end-nodes are a, b, the length budget is D_a, the knapsack weights are μ^j_v for all $v \in V - b$ and $\mu^j_b = 0$, and the knapsack-budget is 2^j. Let $\mathcal{I}_a$ denote the set of all feasible solutions to this KnapOrient instance.

The configuration LP has variables x^a_τ, for every $a \in \mathsf{Por} - \varphi_k$ and $\tau \in \mathcal{I}_a$, indicating the a-$\mathsf{next}(a)$ paths that are chosen.

$$\max \quad \sum_{j=0}^{k} \sum_{a \in \mathsf{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a} x^a_\tau \cdot \Big(\sum_{v \in \tau - \mathsf{next}(a)} \pi_v (2^j - 1) \Big) \tag{CKO-P}$$

$$\text{s.t.} \quad \sum_{\tau \in \mathcal{I}_a} x^a_\tau = 1 \qquad \forall a \in \mathsf{Por} - \varphi_k \tag{1}$$

$$\sum_{a \in \mathsf{Por} - \varphi_k} \sum_{\tau \in \mathcal{I}_a : v \in \tau - \mathsf{next}(a)} x^a_\tau \le 1 \qquad \forall v \in V \tag{2}$$

$$\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_h} \sum_{\tau \in \mathcal{I}_a} x^a_\tau \cdot \mu^j\big(\tau - \mathsf{next}(a)\big) \le (K+1)2^j \qquad \forall j \in [\![k]\!] \tag{3}$$

$$x \ge 0.$$

Constraints (1) encodes that we select an a-b path for every consecutive pair of nodes $a, b \in \mathsf{Por}$, and constraints (2) ensure that each node v lies on at most one of these a-b paths; constraint (3) encodes the (Prefix-size) property P4. (Note that if $\varphi_{h-1} = \varphi_h$, then $\mathsf{Por}_h = \{\varphi_h\}$, so we do not have any term for index h in the objective function, and on the LHS of (3).)

To gain some intuition, notice that Theorem 4.2 shows that there is a feasible integral solution to (CKO-P) of objective value at least $OPT/8$: we set $x^a_\tau = 1$ for $\tau = Q^*_{a,\mathsf{next}(a)}$ for every $a \in \mathsf{Por} - \varphi_k$. Properties P1 and P5 show that $Q^*_{a,\mathsf{next}(a)} \in \mathcal{I}_a$; property P4 shows that (3) holds, and P3 shows that the objective value is at least $OPT/8$.

We can solve (CKO-P) approximately, given an approximation algorithm for KnapOrient, since this can be used to obtain an approximate separation oracle for the dual of (CKO-P).

▷ Claim 4.3. The optimal value of (CKO-P), $OPT_{\mathsf{CKO\text{-}P}}$, is at least $OPT/8$.

▶ **Lemma 4.4.** *Given an α-approximation algorithm for* KnapOrient*, we can compute in polytime a solution $\overline{x}$ to* (CKO-P) *of objective value at least $OPT_{\mathsf{CKO\text{-}P}}/\alpha$.*

We use randomized rounding to round the solution $\overline{x}$ obtained by Lemma 4.4, and Chernoff bounds yield that this only incurs an $O(1)$-factor loss in the objective, and in the violation of constraints (3); here is where we crucially exploit property P5. We then obtain an $O(K)$-approximate non-adaptive policy for CorrKO from the rounded solution.

Algorithm CSKO-Alg. // Rounding $(\overline{x}, \overline{y})$ and obtaining a non-adaptive policy

1. Independently, for each $a \in \mathsf{Por} - \varphi_k$, letting $b = \mathsf{next}(a)$, do the following: pick an a-b path by choosing $\tau \in \mathcal{I}_a$ with probability $\overline{x}^a_\tau/2$, and choosing the "direct" path a, b with the remaining probability 0.5; let $P_{a,b}$ denote the path picked.
2. If for any $j \in [\![k]\!]$, we have $\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_h} \mu^j\big(P_{a,\mathsf{next}(a)} - \mathsf{next}(a)\big) > 5(K+1)2^j$, then **return** the empty policy that does not visit any node.
3. Consider the concatenated sequence of nodes $\{P_{a,\mathsf{next}(a)}\}_{a \in \mathsf{Por} - \varphi_k}$ (where Por is ordered as in Theorem 4.2). If a non-portal node is repeated in this sequence, then shortcut the $P_{a,\mathsf{next}(a)}$ paths so as to retain only the first occurrence of each node. Let $P'_{a,\mathsf{next}(a)}$ denote the shortcut version of $P_{a,\mathsf{next}(a)}$ (which is still an a-$\mathsf{next}(a)$ path). Let P' be the rooted path given by the node-sequence $\{P'_{a,\mathsf{next}(a)}\}_{a \in \mathsf{Por} - \varphi_k}$, where we retain only one copy of each portal node.
4. Sample each $v \in P' - \rho$ independently with probability $\frac{1}{10(K+1)}$ to obtain the rooted path, P''. **return** the non-adaptive policy P''.

Analysis overview. The key observation is that since for any $j \in [\![k]\!]$, any $h \le j$, any $a \in \mathsf{Por}_h - \varphi_h$, and any $\tau \in \mathcal{I}_a$, we have $\mu^j\big(\tau - \mathsf{next}(a)\big) \le 2^j$, we obtain that $\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_h} \mu^j\big(P_{a,\mathsf{next}(a)} - \mathsf{next}(a)\big)$ is the sum of a collection of independent 2^j-*bounded* random variables,[2] whose expectation is $O\big((K+1)\cdot 2^j\big)$, due to constraint (3). It follows from Chernoff bounds that the probability that this sum exceeds $5(K+1)2^j$, for any fixed index j, is $\exp -\Omega(K)$, and so by a union bound, step 2 succeeds with high probability (Lemma 4.5).

To bound the reward obtained, consider a node v and index $j \in [\![k]\!]$, and define $\overline{y}^j_v := \sum_{a \in \mathsf{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a : v \in \tau - \mathsf{next}(a)} \overline{x}^a_\tau$. (Note that $\sum_{h=0}^{k} \overline{y}^h_v \le 1$.) We say that v is "visited by segment j" if $v \ne \varphi_j$ and $v \in \bigcup_{a \in \mathsf{Por}_j - \varphi_j} P_{a,\mathsf{next}(a)}$; we say that v is "retained by segment j" if $v \ne \varphi_j$ and v remains on $\bigcup_{a \in \mathsf{Por}_j - \varphi_j} P'_{a,\mathsf{next}(a)}$ after the shortcutting in step 3. Note that the latter events are disjoint, for different js. (Note that for a portal node in $\mathsf{Por}_j - \varphi_j$, both events happen with probability 1.) Clearly, v is retained by segment j only if it is visited by segment j. For convenience of analysis, we will view step 3 as being executed even if step 2 fails, so we can talk about the event "v retained by segment j" regardless of the outcome of step 2. It is not hard to argue that $\Pr[v \text{ is retained by segment } j] = \Omega(\overline{y}^j_v)$, but we need some care to show that this holds even when we condition on the event that step 2 succeeds, as subtle dependencies between events arise here. Nevertheless, we show that this indeed holds (Lemma 4.6).

[2] This the key difference from [2]. They guess only the φ_j nodes, and so in their case, the corresponding sum gets decomposed into the sum of $(K+1)2^j$-bounded random variables, and so an application of Chernoff bounds incurs an additional $\frac{\log k}{\log\log k} = \frac{\log\log W}{\log\log\log W}$-factor.

Finally, given that step 2 succeeds and the rounded path P' satisfies (3) with $O\big((K+1)2^j\big)$ on the RHS, due to the random sampling in step 4, we can argue that, for any node v retained by segment j, the non-adaptive policy processes v by time 2^j-1 with probability $\frac{1}{O(K)}$ (Lemma 4.7). Thus, the expected reward of the non-adaptive policy is $\frac{1}{O(K)} \cdot \sum_{j=0}^{k} \sum_{v \in V} \overline{y}_j^v \cdot \pi_v(2^j-1) \geq \frac{OPT}{O(K)}$.

For an index $j \in [\![k]\!]$, let $\mathcal{B}_j$ be the event that $\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_j} \mu^j\big(P_{a,\mathsf{next}(a)} - \mathsf{next}(a)\big) > 5(K+1)2^j$. So $\mathcal{B} := \bigvee_{j=0}^{k} \mathcal{B}_j$ is the event that step 2 fails; let $\mathcal{B}^c$ denote the complement of $\mathcal{B}$. Recall that $K = 3\log\log(6W) + 12$.

▶ **Lemma 4.5.** $\Pr[\mathcal{B}_j] \leq e^{-(K+1)}$ *for all* $j \in [\![k]\!]$. *Hence,* $\Pr[\mathcal{B}] \leq 1/\operatorname{poly}(\log W)$.

▶ **Lemma 4.6.** *For any node* $v \in V$ *and any* $j \in [\![k]\!]$, *we have* $\Pr[\{v \text{ is retained by segment } j\} \wedge \mathcal{B}^c] \geq \frac{\overline{y}_v^j}{16}$.

▶ **Lemma 4.7.** *Consider any node* $v \in V - \varphi_k$. *Suppose that* v *is retained by segment* j *in step 3. Then* $\Pr[$*non-adaptive policy* P'' *processes* v *by time* $2^j - 1] \geq \frac{1}{20(K+1)}$, *where the probability is over both the random sampling in step 4 and the random execution of* P''.

Proof of Theorem 1a. Combining Lemmas 4.6 and 4.7, and since for any $v \in V - \varphi_k$, the events "v is retained by segment j" are disjoint across different js, the expected reward obtained from a node v is at least $\frac{\sum_{j=0}^{k} \pi_v(2^j-1)\overline{y}_v^j}{320(K+1)}$. So the total expected reward obtained by P'' is at least $\frac{1}{320(K+1)} \cdot \big(\text{objective value of } \overline{x}\big) = OPT/O(K)$.

The running time is polynomial in the time needed to enumerate the quantities in Theorem 4.2, which is $\operatorname{poly}\big((n\log B)^{O(\log W \log\log W)}\big) = O\big((n+\log B)^{O(\log W \log\log W)}\big)$. ◀

4.2 Polynomial-time $O(\log W)$-approximation algorithm

The polytime algorithm also proceeds by gleaning some structural insights from an optimal adaptive policy that enable one to reduce the problem to rooted knapsack orienteering, losing an $O(\log W)$-factor. Recall that $L = \lceil \log W \rceil$.

▶ **Theorem 4.8.** *There exists an index* $j \in [\![L]\!]$ *such that, for the* KnapOrient*-instance with start node* ρ, *travel budget* B, *knapsack budget* 2^{j+1}, *knapsack weights* $\{\mu_v^j\}_{v \in V}$, *and rewards* $\{\pi_v(2^j-1)\}_{v \in V}$, *the optimal value of the LP-relaxation* (KO-LP), *is at least* $OPT/(L+1)$.

Proof of Theorem 1b. Theorem 4.8 leads to the following simple algorithm. For the index j in the theorem statement, we solve (KO-LP) and round it to an *integer solution* P losing a factor of 5 (see Theorem 2.3). We sample each non-root node in P independently with probability $\frac{1}{4}$, and return the resulting rooted path P''. To analyze this, for any $v \in P$, we have that the probability that the non-adaptive policy P'' processes v by time $2^j - 1$ is at least $\frac{1}{8}$. The claim follows because $\Pr\big[\sum_{w \prec_{P''} v} S_w \geq 2^j\big] = \Pr\big[\sum_{w \prec_{P''} v} X_w^j \geq 2^j\big]$, which is at most

$$\frac{\mathbb{E}\big[\sum_{w \prec_{P''} v} X_w^j\big]}{2^j} = \frac{1}{4} \cdot \frac{\mathbb{E}\big[\sum_{w \prec_P v} X_w^j\big]}{2^j} \leq \frac{1}{4} \cdot \frac{\sum_{w \in P} \mu_w^j}{2^j} \leq \frac{1}{2}.$$

The probability of the stated event is therefore at least $\Pr[v \in P'']/2 \geq 1/8$. Therefore, the expected reward obtained is at least $\frac{1}{8} \cdot \sum_{v \in P} \pi_v(2^j-1) \geq \frac{OPT}{L+1} \cdot \frac{1}{5} \cdot \frac{1}{8} = OPT/O(L)$. ◀

5 Refined approximation guarantees and hardness results for CorrKO

In this section, we perform a fine-grained-complexity study of CorrKO. Motivated by the fact that our adaptivity-gap lower bound for CorrKO utilizes distributions of support-size 3, whereas the adaptivity-gap lower-bound example for stochastic orienteering [2] considers *weighted Bernoulli distributions*, we investigate the complexity of CorrKO when we have distributions supported on at most 2 points – we call this special case 2CorrKO– as also the further special case where the vertex-size distributions are weighted Bernoulli distributions.

In stark contrast with stochastic orienteering, we show that the *adaptivity gap is a constant* for 2CorrKO. Moreover, we obtain non-adaptive $O(1)$-approximation algorithms that run in polynomial time for weighted Bernoulli distributions (Theorem 5.5), and in time $(n+\log B)^{O(\log W)}$ for general 2CorrKO (Theorem 5.4).

The chief insight underlying the above results is that one can isolate a novel *deterministic* VRP, that we call *orienteering with knapsack deadlines* (OrientKD), that governs the complexity of 2CorrKO. In OrientKD, we are given an (rooted or P2P) orienteering instance, along with nonnegative knapsack weights $\{\mathsf{wt}_v\}_{v\in V}$ and *knapsack deadlines* $\{\mathsf{KD}_v\}$. A path P with start node a is feasible, if it is feasible for the orienteering instance, and $\sum_{u\in P_{a,v}} \mathsf{wt}_u \le \mathsf{KD}_v$ for every node $v \in P$; the goal is to find a feasible path P that obtains the maximum reward. For this problem, we obtain the following approximation results.

▶ **Theorem 5.1.** *We can obtain the following approximation guarantees for* OrientKD*:*

(a) $O(1)$*-approximation in* $(n+\log B)^{O(\log W)}$ *time;*

(b) *polytime* $O\big(\log(\frac{\max_v \mathsf{KD}_v}{\mathsf{KD}_{\min}})\big)$*-approximation, where* $\mathsf{KD}_{\min}$ *is the minimum non-zero knapsack deadline.*

We show that, up to constant factors, OrientKD is *equivalent to* 2CorrKO *in terms of approximability* (Theorem 5.4). The $O(1)$-approximation for 2CorrKO, and the polytime $O(1)$-approximation for weighted Bernoulli distributions both fall out as direct consequences of this equivalence: the former, because we can devise an $(n+\log B)^{O(\log W)}$-time $O(1)$-approximation for OrientKD (Theorem 5.1); the latter, because the OrientKD instance that one needs to solve for weighted Bernoulli distributions is in fact a KnapOrient instance. Another corollary is a hardness result for CorrKO showing that an α-approximation for CorrKO relative to the *non-adaptive optimum* implies an $O(\alpha)$-approximation for OrientKD (Theorem 5.6); this follows because such an approximation guarantee for CorrKO implies an $O(\alpha)$-approximation for 2CorrKO (since the adaptivity gap for 2CorrKO is $O(1)$).

Difficult instances of CorrKO. We begin by distilling the key source of difficulty for CorrKO (Lemma 5.2). This will prove to be useful when we study 2CorrKO, as it will allow us to focus on the core of the problem. We define the size instantiation S_v of a vertex v to be large if $S_v > W/2$, and small otherwise. We argue that the difficulty of CorrKO stems from instances where most of the optimal reward comes from *vertices that instantiate to a large size with small probability.*

To make this precise, we introduce some notation. For a vertex v, we can split its reward R_v as $R_v = {R_v}^{>W/2} + {R_v}^{\le W/2}$, where ${R_v}^{>W/2} := R_v \mathbb{1}_{S_v > W/2}$ and ${R_v}^{\le W/2} := R_v \mathbb{1}_{S_v \le W/2}$. We can consider the modified CorrKO instances $\mathcal{I}^{>W/2}$ and $\mathcal{I}^{\le W/2}$, where the rewards are given by $\{{R_v}^{>W/2}\}_{v\in V}$ and $\{{R_v}^{\le W/2}\}_{v\in V}$ respectively; so in $\mathcal{I}^{>W/2}$, we only collect non-zero reward from large instantiations, and in $\mathcal{I}^{\le W/2}$, we only collect non-zero reward from small instantiations. For $p \in [0,1]$, define $\mathcal{I}^{>W/2}(p)$ to be the instance with vertex set $V(p) := \{v \in V : \Pr[S_v > W/2] \le p\}$ (note that $\rho \in V(p)$). Thus, in instance $\mathcal{I}^{>W/2}(p)$, we only consider vertices that instantiate to a large size with probability at most p (i.e., small probability), and collect reward only from large instantiations.

▶ **Lemma 5.2.** *Suppose we have an α-approximation algorithm for* CorrKO *instances of the form $\mathcal{I}^{>W/2}(0.5)$. Then, we can obtain an $\big(\alpha + O(1)\big)$-approximation algorithm for all* CorrKO *instances.*

Proof. A CorrKO instance $\mathcal{I}$ can be decomposed into three instances, $\mathcal{I}_1 = \mathcal{I}^{\leq W/2}$, $\mathcal{I}_2 = \mathcal{I}^{>W/2}(0.5)$, and $\mathcal{I}_3$ with vertex set $V_3 := \{v \in V : \Pr[S_v > W/2] > 0.5\}$ and rewards $\{{R_v}^{>W/2}\}_{v \in V_3}$. Any vertex v yields positive reward in at most one of these 3 instances for any size instantiation, and so $OPT = OPT(\mathcal{I}) \leq OPT(\mathcal{I}_1) + OPT(\mathcal{I}_2) + OPT(\mathcal{I}_3)$.

We can obtain non-adaptive polices that yield approximation guarantees of $\beta_1 = O(1)$, $\beta_3 = O(1)$ for $\mathcal{I}_1$ and $\mathcal{I}_3$ respectively. Any adaptive policy for $\mathcal{I}_3$ can collect positive reward from at most one vertex, which is the first vertex that instantiates to a large size; after this the policy may as well stop. The expected number of nodes visited by an adaptive policy is at most $\sum_{i \geq 1} 2^{-(i-1)} \leq 4$, so simply visiting the node in V_3 with largest expected reward, yields an $O(1)$-approximation to $OPT(\mathcal{I}_3)$. For $\mathcal{I}_1$, one can argue that an $O(1)$-approximation follows by solving the KnapOrient instance with rewards $\{\mathbb{E}\big[{R_v}^{\leq W/2}\big]\}_{v \in V}$, travel budget B, knapsack weights $\{\mathbb{E}\big[\min\{S_v, W\}\big]\}_{v \in V}$, and knapsack budget $2W$.

Let $\beta_2 = \alpha$. Consider now the algorithm that With probability $\frac{\beta_j}{\beta_1+\beta_2+\beta_3}$, runs the corresponding algorithm for instance $\mathcal{I}_j$, for $j = 1, 2, 3$. The expected reward obtained via this is at least $\sum_{j=1}^{3} \frac{\beta_j}{\beta_1+\beta_2+\beta_3} \cdot \frac{OPT(\mathcal{I}_j)}{\beta_j} \geq \frac{OPT}{\beta_1+\beta_2+\beta_3} = \frac{OPT}{\alpha+O(1)}$. ◀

5.1 2CorrKO: CorrKO with distributions of support-size at most 2

Recall that 2CorrKO denotes the special case of CorrKO where, for each vertex v, the distribution of S_v is supported on at most 2 values, denoted $S_v^{(1)}$, $S_v^{(2)}$ with $S_v^{(1)} \geq S_v^{(2)}$. By Lemma 5.2, to obtain an $O(1)$-approximation for 2CorrKO, it suffices to consider the instance $\mathcal{I}_2 = \mathcal{I}^{>W/2}(0.5)$, and we focus on such instances in the sequel. To keep notation simple, we continue to use V to denote the vertex set of $\mathcal{I}_2$. Then we may assume that the (size, reward) distribution for each $v \in V$ is $(S_v^{(1)}, R_v)$ with probability p_v, and $(S_v^{(2)}, 0)$ with probability $1 - p_v$, where (i) $S_v^{(1)} > W/2 \geq S_v^{(2)}$ and (ii) $p_v \leq 0.5$. Property (i) holds because if $S_v^{(1)} \leq W/2$, then v yields 0 reward for $\mathcal{I}_2$, so may be discarded; if $S_v^{(2)} > W/2$, then $\Pr[S_v > W/2] = 1$, which means that v would not be considered for $\mathcal{I}_2$. Given (i) the reward when the size is $S_v^{(2)}$ must be 0, and (ii) holds because $p_v = \Pr[S_v > W/2]$. We first argue that the adaptivity gap for such instances is 1.

▶ **Theorem 5.3.** *The adaptivity gap for* 2CorrKO *(instances of the form $\mathcal{I}^{>W/2}(0.5)$) is* 1.

Proof. Let $\mathcal{T}$ be the decision tree of an optimal adaptive policy. Consider the (rooted) path σ of $\mathcal{T}$ corresponding to the $S_v^{(2)}$ size instantiations. Then $\mathcal{T}$ cannot collect any reward outside of σ, since the residual knapsack budget when we reach any node $v \in \mathcal{T} \setminus \sigma$ is less than $W/2$. So the non-adaptive policy represented by σ has the same expected reward as $\mathcal{T}$. ◀

We now show that the resulting 2CorrKO problem is equivalent to OrientKD, up to constant-factor approximation losses. By "equivalent", we always mean equivalent up a multiplicative $O(1)$ factor. We actually show that 2CorrKO is equivalent to another problem, *knapsack orienteering with knapsack deadlines* (KnapOrientKD), which is the knapsack-constrained version of OrientKD; by Theorem 2.1, OrientKD and its knapsack-constrained version KnapOrientKD are equivalent, so this implies that 2CorrKO and OrientKD are equivalent.

▶ **Theorem 5.4.** *Given an α-approximation algorithm for one of the problems,* KnapOrientKD *or* 2CorrKO*, one can obtain an $O(\alpha)$-approximation algorithm for the other. Hence, the problems* 2CorrKO *and* OrientKD *are equivalent. This implies an $O(1)$-approximation algorithm for* 2CorrKO *with running time $(n + \log B)^{O(\log W)}$.*

The approximation guarantee above follows from the guarantee for OrientKD stated in Theorem 5.1. We briefly sketch how to reduce 2CorrKO to KnapOrientKD. By essentially "inverting" this reduction, we obtain the opposite reduction, from KnapOrientKD to 2CorrKO.

Let $\mathcal{I}$ be a 2CorrKO instance. By Theorem 5.3, we can focus on non-adaptive policies for $\mathcal{I}$. Let τ be a ρ-rooted path representing a non-adaptive policy. It is not hard to show that the expected reward from a node $v \in \tau$ is $p_v R_v \prod_{w \prec_\tau v}(1 - p_w)$ if $\sum_{w \prec_\tau v} S_w^{(2)} \leq W - S_v^{(1)}$, and is 0 otherwise. Also, one can argue that the total expected reward from nodes $v \in \tau$ with $\sum_{w \prec_\tau v} p_w > 1$ is a small fraction of $OPT(\mathcal{I})$. This motivates the following reduction to KnapOrientKD. We set rewards $\{\pi_v R_v\}_{v \in V}$. The constraint $\sum_{w \prec_\tau v} S_w^{(2)} \leq W - S_v^{(1)}$ can be encoded by a knapsack deadline, by considering knapsack weights $\{S_w^{(2)}\}_{w \in V}$ and knapsack deadlines $\{W - S_w^{(1)} + S_w^{(2)}\}_{w \in V}$. The additional knapsack constraint will encode that the total p_w-weight of the path should be at most 1, so that the expected reward obtained for $\mathcal{I}$ from each vertex v on the KnapOrientKD-solution is $\Omega(p_v R_v)$.

For weighted Bernoulli size distributions, which is the special case of 2CorrKO where $S_v^{(2)} = 0$ for all $v \in V$, the above reduction actually crates a KnapOrient instance, since the knapsack-deadlines are trivially satisfied by any rooted path. Since we have a polytime $O(1)$-approximation for KnapOrient, we obtain the following.

▶ **Theorem 5.5** (Weighted Bernoulli size distributions)**.** *There is a polytime $O(1)$-approximation for* CorrKO *with weighted Bernoulli size distributions.*

As noted earlier, combining the equivalence of 2CorrKO and OrientKD, and the $O(1)$ adaptivity gap for 2CorrKO, yields the following hardness result.

▶ **Theorem 5.6** (Hardness of approximating the non-adaptive optimum)**.** *Given an α-approximation algorithm for* CorrKO *with respect to the non-adaptive optimum, we can obtain an $O(\alpha)$-approximation algorithm for* OrientKD*.*

6 CorrKO with cancellations

In CorrKO *with cancellations* (CorrKO-Cancel), the input is the same as in CorrKO, but we are now allowed to *cancel* the processing of the current vertex v at any (integer) timestep before its size and reward get fully realized; if v is cancelled, then no reward is collected from v and we cannot process v again. As with CorrKO, we only collect reward from vertices that complete by the processing-time horizon W. Gupta et al. [18] showed that even for correlated knapsack (which is the special case of CorrKO where all vertices are co-located), the optimal reward when we allow cancellations can be substantially larger than the optimal reward without cancellations, so we need to develop new algorithms to handle cancellations.

We obtain the same guarantees for CorrKO-Cancel as for CorrKO: that is, $O(\log \log W)$-approximation in $(n + \log B)^{O(\log W \log \log W)}$ time, and a polytime $O(\log W)$-approximation.

We proceed as follows. Recall that for a vertex v, we define ${R_v}^{>W/2} := R_v \mathbb{1}_{S_v > W/2}$ and ${R_v}^{\leq W/2} := R_v \mathbb{1}_{S_v \leq W/2}$. Let $\mathcal{I}^{>W/2}$ and $\mathcal{I}^{\leq W/2}$ denote the CorrKO-Cancel instances where the rewards are given by $\{{R_v}^{>W/2}\}_{v \in V}$ and $\{{R_v}^{\leq W/2}\}_{v \in V}$ respectively. As observed by [18], cancellations do not help for the instance $\mathcal{I}^{>W/2}$, i.e., the optimal reward is the same both with and without cancellations. This is because if a policy cancels a vertex v after it has

run for some $t \leq W/2$ time steps, we can modify the policy to not process v at all, without decreasing the reward accrued from subsequently-processed vertices; if v is cancelled after it has run for more than $W/2$ time steps, then both with and without cancellation, the policy cannot collect any further reward.

We show that we can obtain an $O(1)$-approximation for $\mathcal{I}^{\leq W/2}$. With probability 0.5 each, we can work on the instance $\mathcal{I}^{\leq W/2}$, where we utilize this $O(1)$-approximation, or the instance $\mathcal{I}^{>W/2}$, where we utilize the approximation results for CorrKO. So this yields: an $O(\log \log W)$-approximation in quasi-polytime, and a polytime $O(\log W)$-approximation.

So we focus on obtaining an $O(1)$-approximation for CorrKO-Cancel instances of the form $\mathcal{I}^{\leq W/2}$. Our approach is based on LP-rounding, by combining the LP-rounding approaches for orienteering in [14] and the correlated knapsack problem with cancellations in [18]. We combine the LP-relaxations for these two problems to obtain the following LP, whose optimal value yields an upper bound on the optimal reward. We use $R_u(t)$ to denote the reward R_u when the size S_u is t; this is 0 if $\Pr[S_u = t] = 0$. Note that $R_u(t) = 0$ for all $t > W/2$, since we are considering $\mathcal{I}^{\leq W/2}$.

$$\max \quad \sum_{u \in V} \sum_{t=1}^{W/2} z_{u,t} \cdot \Pr[S_u = t \,|\, S_u \geq t] \cdot R_u(t) \tag{CKOC-LP}$$

$$\text{s.t.} \qquad (\text{O1}) - (\text{O5})$$

$$\sum_{v \in V} z_u^v = z_{u,0} \qquad \forall u \in V \tag{4}$$

$$z_{u,t} = s_{u,t} + z_{u,t+1} \qquad \forall u \in V,\, t \in [\![W]\!] \tag{CK1}$$

$$s_{u,t} \geq \Pr[S_u = t \,|\, S_u \geq t] \cdot z_{u,t} \qquad \forall u \in V,\, t \in [\![W]\!] \tag{CK2}$$

$$\sum_{u \in V} \sum_{t=0}^{W} t \cdot s_{u,t} \leq W \tag{CK3}$$

$$x, z, s \geq 0.$$

The x_a^v and z_u^v variables, and constraints (O1)–(O5) are from the LP for rooted orienteering (and also present in LP (KO-LP) for KnapOrient). They encode the arcs included, and vertices visited, respectively by the rooted path, provided that v is the furthest node from ρ that is visited. Constraints (O1)–(O5) are valid because any rooted path Q, corresponding to an execution of an adaptive policy, satisfies these constraints, where the superscript v in the non-zero variables is the furthest node from ρ on Q.

The $z_{u,t}$ and $s_{u,t}$ variables, constraints (CK1)–(CK3), and the objective function are from the LP in [18] for correlated knapsack with cancellations. For any vertex u and $t \geq 0$, variable z_t^u encodes that u is processed for a least t time units, and $s_{u,t}$ encodes that u is processed for *exactly* t time units. Thus, variable $z_{u,0}$ encodes that u is visited, and constraint (4) links the orienteering and correlated knapsack LPs. Gupta et al. [18] show (see Theorem 3.1 in [18]) that constraints (CK1)–(CK3) are valid for correlated knapsack with cancellations, and that the objective function provides an upper bound on the expected reward obtained.

We remark that [18] showed that one can replace (CK1)–(CK3) with a polynomial-size formulation losing an $O(1)$-factor, which applies here as well.

We round an optimal solution $(\overline{x}, \overline{z}, \overline{s})$ to (CKOC-LP) in two phases. We first *extract a suitable knapsack orienteering instance from the LP solution*, and use Theorem 2.3 to obtain a good rooted path Q for this KnapOrient instance. Now, we select a subsequence of Q to visit by solving a correlated knapsack with cancellations problem involving only vertices in Q. The KnapOrient-instance is set up so that from $(\overline{x}, \overline{z}, \overline{s})$, one can extract a good LP solution to the

correlated knapsack problem restricted to vertices in Q. We utilize the LP-rounding result in [18] to round this solution to obtain a CorrKO-Cancel solution that visits the vertices in Q in order, potentially cancelling some vertices along the way. Thus, we obtain a non-adaptive policy for CorrKO-Cancel. In the second phase, we crucially leverage an important aspect of the LP-rounding algorithm in [18] for correlated knapsack with cancellations, namely that it is *order oblivious*: it's guarantee does not depend on the order in which the vertices (i.e., items in correlated knapsack) are considered. This flexibility allows us to consider vertices in Q in the order they are visited, and thereby ensure that the travel-budget constraint is satisfied. (We remark that for the correlated knapsack without cancellations, we do not have this flexibility, when considering large instantiations; see Appendix A. This lack of flexibility is the main obstacle in obtaining a good solution from large instantiations in CorrKO.)

7 $O(\log\log B)$-approximation for CorrO

Our approach in Section 4 for CorrKO can be utilized to yield the guarantees mentioned in Theorem 1.3: that is, an $O(\alpha \log\log B)$-approximation algorithm for CorrO in time $(n+\log B)^{O(\log B \log\log B)} \cdot T$, where T is the running time of the given α-approximation algorithm for deadline TSP. The algorithm in [15] for deadline TSP translates to an $O(1)$-approximation in $n^{O(\log B)}$ time, so this implies an $O(\log\log B)$-approximation for CorrO in quasi-polytime. We also simplify the exposition significantly by making use of monotone-reward TSP [15] as a subroutine.

Let $K = 3\log(6\log B)+12$, $L = \lceil \log B \rceil$, $N_1 = 2(K+1)$. Define $\varphi_{-1} := \rho$, and $\pi_v(t) := \mathbb{E}\big[R_v \cdot \mathbb{1}_{S_v \le B-t}\big] = \sum_{t'=0}^{B-t} \Pr[S_v = t'] \cdot \mathbb{E}\big[R_v \,|\, S_v = t'\big]$. Let OPT^{CorrO} denote the optimal reward for the CorrO instance. We may assume that $\pi_v(d_{\rho,v}) \le OPT^{\mathsf{CorrO}}/4$ for every $v \in V$, as otherwise, we can obtain $\Omega(OPT^{\mathsf{CorrO}})$ reward by going to a single node. The algorithm in [2] is based on the following structural result, which we have paraphrased (and corrected slightly) to conform to our notation.

▶ **Lemma 7.1** (Lemma 3.6 in [2]). *There exists a rooted path P with $d(P) \le B$, and vertices $\varphi_0 \preceq \varphi_1 \preceq \ldots \preceq \varphi_k$ on P for some $k \le L$, such that:*
(a) $\sum_{j=0}^{k} \sum_{v \in P_{\varphi_{j-1},\varphi_j} - \varphi_j} \pi_v\big(d(P_{\rho,v}) + 2^j - 1\big) \ge OPT^{\mathsf{CorrO}}/4$*; and*
(b) $\mu^j(P_{\rho,\varphi_j} - \varphi_j) \le (K+1)2^j$ *for all $j \in [\![k]\!]$.*

We refine this by subdividing each $P_{\varphi_{j-1},\varphi_j}$ subpath into at most $2(K+1)$ segments each of μ^j-weight at most 2^j, and by guessing the two-point regrets of these segments, to obtain a structural result analogous to Theorem 4.2.

▶ **Theorem 7.2** (Structural result for CorrO). *Let the rooted path P and node-sequence $\varphi_0, \ldots, \varphi_k$, where $k \le L$, be as in Lemma 7.1. For each $j \in [\![k]\!]$, there is a vertex-set $\mathsf{Por}_j \subseteq P_{\varphi_{j-1},\varphi_j}$ containing φ_{j-1}, φ_j, with $|\mathsf{Por}_j| \le N_1$, whose nodes are ordered by the order they appear on P, and for every node $a \in \big(\bigcup_{j=0}^{k} \mathsf{Por}_j\big) - \varphi_k$, there is a path $\overline{Q}^a$, node m_a, integer $\gamma_a \ge 0$, satisfying the following properties. For $a, b \in \mathsf{Por} := \bigcup_{j=0}^{k} \mathsf{Por}_j$, let $\mathsf{next}(a)$ be the next node in Por after a, for $a \ne \varphi_k$; let $b \prec a$ if b comes before a in Por.*
(C1) (Distance) $d(\overline{Q}^a) \le D_a := 2^{\gamma_a} - 1 + d(a, m_a) + d\big(m_a, \mathsf{next}(a)\big)$ *for every $a \in \mathsf{Por} - \varphi_k$.*
(C2) (Total-length) $\sum_{a \in \mathsf{Por} - \varphi_k} D_a \le B$.
(C3) (Reward) $\sum_{j=0}^{K} \sum_{a \in \mathsf{Por}_j - \varphi_j} \sum_{v \in \overline{Q}^a - \mathsf{next}(a)} \pi_v\big(\sum_{b \prec a} D_b + d(\overline{Q}^a_{a,v}) + 2^j - 1\big) \ge OPT^{\mathsf{CorrO}}/8$.
(C4) (Prefix-size) $\sum_{h=0}^{j} \sum_{a \in \mathsf{Por}_h - \varphi_h} \mu^j\big(\overline{Q}^a - \mathsf{next}(a)\big) \le (K+1)2^j$ *for all $j \in [\![k]\!]$.*
(C5) (Size) $\mu^j\big(\overline{Q}^a - \mathsf{next}(a)\big) \le 2^j$ *for every $j \in [\![k]\!]$ and every $a \in \mathsf{Por}_j - \varphi_j$.*

We now exploit this structural result in much the same way as in Section 4.1 by setting up a configuration LP to find the $\overline{Q}^a$-paths. Note that only the (Reward) property C3 is different from the (Reward) property in Theorem 4.2 for CorrKO, and correspondingly, we now exploit monotone-reward TSP to capture the reward of an $\overline{Q}^a$ path. Assume that we have found the "portal nodes" Por and length bounds D_a for all $a \in \mathsf{Por} - \varphi_k$ satisfying Theorem 7.2. To capture the reward obtained from an a-next(a) path, the configurations $\mathcal{I}_a$ for a node $a \in \mathsf{Por} - \varphi_k$ will now consist of feasible solutions to P2P *knapsack monotone-reward TSP*, which is the knapsack-constrained version of P2P-monotone-reward TSP: they are simply all a-next(a) paths τ with $\mu^j\big(\tau - \mathsf{next}(a)\big) \le 2^j$. The length budget D_a will be captured implicitly, by defining the reward function of a node $v \neq \mathsf{next}(a)$ to be $\pi_v\big(\sum_{b \prec a} D_b + d(\tau_{a,v}) + 2^j - 1\big)$ if $d(\tau_{a,v}) \le D_a - d(v, \mathsf{next}(a))$ and 0 otherwise, which is a non-increasing function of $d(\tau_{a,v})$; for $v = \mathsf{next}(a)$, the reward function is defined to be identically 0. For notational convenience, define $\pi^{a,j}(\tau)$ to be the total reward obtained from nodes in τ under the above rewards.

The configuration LP for CorrO now has the same constraints as (CKO-P), but the objective function changes to max $\sum_{j=0}^{k} \sum_{a \in \mathsf{Por}_j - \varphi_j} \sum_{\tau \in \mathcal{I}_a} x^a_\tau \cdot \pi^{a,j}(\tau)$. Let (CO-P) denote this LP, and $OPT_{\mathsf{CO\text{-}P}}$ denote its optimal value.

We now have $OPT_{\mathsf{CO\text{-}P}} \ge OPT^{\mathsf{CorrO}}/8$, and one can argue that an α-approximation algorithm for deadline TSP (and hence, monotone-reward TSP [15]) can be used to obtain a (CO-P)-solution $\overline{x}$ of value at least $OPT_{\mathsf{CO\text{-}P}}/O(\alpha)$. The LP-rounding algorithm and conversion to a non-adaptive policy are *exactly* as in Algorithm CSKO-Alg. The analysis is similar, but we now analyze the reward on a path-by-path basis, considering the reward obtained from paths $\tau \in \mathcal{I}_a$, for each $a \in \mathsf{Por} - \varphi_k$. We can again argue that step 2 succeeds with high probability, and moreover that the expected reward obtained is large conditioned on this. Hence, we obtain an $O(K)$ approximation.

References

1. 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 (STOC)*, pages 166–174, 2004. doi:10.1145/1007352.1007385.
2. Nikhil Bansal and Viswanath Nagarajan. On the adaptivity gap of stochastic orienteering. *Mathematical Programming*, 154(1-2):145–172, December 2015. doi:10.1007/s10107-015-0927-9.
3. Anand Bhalgat. A (2+ ϵ)-approximation algorithm for the stochastic knapsack problem. *Unpublished manuscript*, 2011.
4. Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 163–171, 1994. doi:10.1145/195058.195125.
5. Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM Journal on Computing*, 37(2):653–670, January 2007. doi:10.1137/050645464.
6. Deeparnab Chakrabarty and Chaitanya Swamy. Facility Location with Client Latencies: Linear Programming Based Techniques for Minimum Latency Problems. *Mathematics of Operations Research*, 41(3):865–883, 2016. doi:10.1287/moor.2015.0758.
7. Shuchi Chawla, Evangelia Gergatsouli, Yifeng Teng, Christos Tzamos, and Ruimin Zhang. Pandora's box with correlations: Learning and approximation. In *Proceedings of the 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1214–1225, 2020.
8. Chandra Chekuri, Nitish Korula, and Martin Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms*, 8(3):1–27, July 2012. doi:10.1145/2229163.2229167.

9 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33(4):945–964, November 2008. doi:10.1287/moor.1080.0330.

10 Amol Deshpande, Lisa Hellerstein, and Devorah Kletenik. Approximation Algorithms for Stochastic Submodular Set Cover with Applications to Boolean Function Evaluation and Min-Knapsack. *ACM Transactions on Algorithms*, 12(3):1–28, June 2016. doi:10.1145/2876506.

11 Alina Ene, Viswanath Nagarajan, and Rishi Saket. Approximation Algorithms for Stochastic k-TSP. In *Proceedings of the 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 27:27–27:14, 2017. arXiv:1610.01058.

12 Jittat Fakcharoenphol, Chris Harrelson, and Satish Rao. The k -traveling repairmen problem. *ACM Transactions on Algorithms*, 3(4):40, November 2007. doi:10.1145/1290672.1290677.

13 Zachary Friggstad and Chaitanya Swamy. Approximation algorithms for regret-bounded vehicle routing and applications to distance-constrained vehicle routing. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 744–753, 2014. doi:10.1145/2591796.2591840.

14 Zachary Friggstad and Chaitanya Swamy. Compact, Provably-Good LPs for Orienteering and Regret-Bounded Vehicle Routing. In *Proceedings of 19th IPCO*, pages 199–211, 2017.

15 Zachary Friggstad and Chaitanya Swamy. Constant-Factor Approximation to Deadline TSP and Related Problems in (Almost) Quasi-Polytime. In *Proceedings of 48th ICALP*, pages 67:1–67:21, 2021.

16 Bruce L. Golden, Larry Levy, and Rakesh Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987. doi:10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.

17 Sudipto Guha and Kamesh Munagala. Multi-armed Bandits with Metric Switching Costs. In *Proceedings of 36th ICALP*, pages 496–507, 2009. doi:10.1007/978-3-642-02930-1_41.

18 Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and Ramamoorthi Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In *52nd Annual Symposium on Foundations of Computer Science*, pages 827–836, 2011.

19 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running Errands in Time: Approximation Algorithms for Stochastic Orienteering. *Mathematics of Operations Research*, 40(1):56–79, 2015. doi:10.1287/moor.2014.0656.

20 Haotian Jiang, Jian Li, Daogao Liu, and Sahil Singla. Algorithms and Adaptivity Gaps for Stochastic k-TSP. In *Proceedings of 11th ITCS*, pages 45:1–45:25, 2020. arXiv:1911.02506.

21 Will Ma. Improvements and Generalizations of Stochastic Knapsack and Multi-Armed Bandit Approximation Algorithms: Extended Abstract. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1154–1163, 2014. doi:10.1137/1.9781611973402.85.

22 Viswanath Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, March 2012. doi:10.1002/net.20435.

A Adversarial orderings can be arbitrarily bad for correlated knapsack

Consider an instance of correlated stochastic knapsack on the set of items $[n]$ with budget $W > 2^{n+1}$. Let S_i and R_i denote respectively the random size and random reward of i, which follows the following distribution.

$$(S_i, R_i) = \begin{cases} \left(S_i^{(1)} := W - 2^{n-i} + 1,\ R_i^{(1)} := 1\right) & \text{with probability } \frac{1}{n} \\ \left(S_i^{(2)} := 2^{n-i},\ R_i^{(2)} := 0\right) & \text{with probability } 1 - \frac{1}{n}. \end{cases}$$

At most one item can obtain positive reward since $W/2 < W - 2^{n-i+1} + 1$ for all $i \in [n]$.

Suppose that we are forced to process the items in the ordering $1, \ldots, n$ deciding at each step whether we attempt to insert the current item into the knapsack or abandon it forever. Let j be the first item that we choose to insert into the knapsack. It instantiates to size 2^{n-j} with probability $1 - 1/n$. If this happens we get zero total reward: the residual budget becomes $W - 2^{n-j}$, which is less than the $S_i^{(1)}$-sizes of items $j+1, \ldots, n$ (which yield positive reward). Therefore, by processing the items in this ordering the expected reward is at most $1/n$. But suppose we process the items in the reverse order $n, \ldots, 1$. If we attempt to insert items $n, n-1, \ldots, j$ and get zero reward from all of them, the residual budget is $W - \sum_{k=j}^{n} 2^{n-k} = W - 2^{n-j+1} + 1 > S_{j-1}^{(1)}$, which means that item $j-1$ can be inserted and would yield reward 1 with probability $\frac{1}{n}$. Thus, the probability that no item gives positive reward is $(1 - 1/n)^n \leq e^{-1}$ and the expected reward we obtain in this case is at least $(1 - e^{-1})$. This is $\Omega(n)$ times larger than the expected reward that can be obtained by any policy that is forced to process items in the order $1, \ldots, n$.