Offline Local Search for Online Stochastic Bandits

GERDUS BENADÈ, Boston University, USA
RATHISH DAS, University of Houston, USA
THOMAS LAVASTIDA, University of Texas at Dallas, USA

Combinatorial multi-armed bandits provide a fundamental online decision-making environment where a decision-maker interacts with an environment across T time steps, each time selecting an action and learning the cost of that action. The goal is to minimize regret, defined as the loss compared to the optimal fixed action in hindsight under full-information. There has been substantial interest in leveraging what is known about offline algorithm design in this online setting. Offline greedy and linear optimization algorithms (both exact and approximate) have been shown to provide useful guarantees when deployed online. We investigate local search methods, a broad class of algorithms used widely in both theory and practice, which have thus far been under-explored in this context. We focus on problems where offline local search terminates in an approximately optimal solution and give a generic method for converting such an offline algorithm into an online stochastic combinatorial bandit algorithm with $O(\log^3 T)$ (approximate) regret. In contrast, existing offline-to-online frameworks yield regret (and approximate regret) which depend sub-linearly, but polynomially on T. We demonstrate the flexibility of our framework by applying it to three online stochastic combinatorial optimization problems: scheduling to minimize total completion time, finding a minimum cost base of a matroid and uncertain clustering.

CCS Concepts: • **Theory of computation → Online learning algorithms**; **Regret bounds**; **Sequential decision making**; *Scheduling algorithms*; *Facility location and clustering*.

Additional Key Words and Phrases: Online learning; bandit algorithms; offline-to-online frameworks; local search

1 Introduction

Online learning is a cornerstone of modern algorithm design. The bandit setting, in which the learner only receives cost feedback, epitomizes the trade-off between exploration and exploitation [5, 6, 34]. Here, an algorithm makes a sequence of decisions $x_1, x_2, \ldots, x_T$ over a series of T discrete time periods. In response to each decision x_t, the environment generates a cost $c_t(x_t)$ (either adversarially [6] or stochastically [5]), and the algorithm's objective is to minimize its cumulative cost across all T periods. As feedback to improve future decisions, the algorithm observes each cost $c_t(x_t)$, which is known as *bandit feedback*. Notably, the algorithm does not observe what the cost would have been if, instead, a different action was chosen at each period, necessitating a trade-off between exploring the action space to find promising solutions and exploiting previously discovered low-cost solutions. *Regret* measures the difference between the cumulative cost achieved by an algorithm and that of the best fixed solution under full information. A common objective is to construct algorithms with regret that is sub-linear in T, i.e., $o(T)$.

In the combinatorial bandit setting [4, 13, 15], the learner must additionally navigate an exponentially large space of actions, rendering some approaches computationally infeasible. There is substantial interest in developing general frameworks for designing online algorithms, especially frameworks that convert offline algorithms (in either a black- or white-box fashion) into effective bandit algorithms; see, for example, Dudík et al. [20], Kakade et al. [31], Kalai and Vempala [32], Niazadeh et al. [41], and Agarwal et al. [1]. Such frameworks allow us to leverage the wealth of knowledge about offline algorithms to construct online learning algorithms. Notable techniques

Authors' Contact Information: Gerdus Benadè, benade@bu.edu, Boston University, Boston, MA, USA; Rathish Das, rathish@uh.edu, University of Houston, Houston, USA; Thomas Lavastida, thomas.lavastida@utdallas.edu, University of Texas at Dallas, Dallas, USA.

Algorithm 1: Generic Local Search

Data: Feasible set $\mathcal{X}$, neighborhood map $\mathcal{N}$, cost function Cost, and parameter $\beta \in (0, 1)$
Result: Approximate locally optimal solution
Procedure `LocalSearch`($\mathcal{X}$, $\mathcal{N}$, Cost):
  $x_0 \leftarrow$ any feasible solution from $\mathcal{X}$
  $t \leftarrow 0$
  while True **do**
    $x_{t+1} \leftarrow \arg\min_{x' \in \mathcal{N}(x_t)} \mathrm{Cost}(x')$
    if $\mathrm{Cost}(x_{t+1}) \leq \beta \, \mathrm{Cost}(x_t)$ **then**
      $t \leftarrow t + 1$ `//Move to the next iteration`
    else
      return x_t `//Approximate local optimum found`

for designing offline algorithms that have proven to be successful in 'offline-to-online' frameworks include greedy algorithms [23, 41, 42] and offline linear optimization in both exact [32] and approximate [31, 32] settings. These all utilize sophisticated algorithm design and analysis techniques to convert the offline algorithm to an effective bandit learning algorithm with regret (or γ-regret [26, 31, 41]) which scales sub-linearly in T, more specifically, with regret which scales as T^{ρ} for some $\rho \in (0, 1)$ (usually $\rho = 1/2$ or $\rho = 2/3$).

The focus of our paper is local search — another technique widely used in the design of offline algorithms. A local search algorithm consists of three components: a set of feasible solutions $\mathcal{X}$, a neighborhood map $\mathcal{N} : \mathcal{X} \rightarrow 2^{\mathcal{X}}$, and a cost function $\mathrm{Cost} : \mathcal{X} \rightarrow \mathbb{R}$. Starting from an initial feasible solution $x_0 \in \mathcal{X}$, local search methods iteratively set $x_{t+1} = \arg\min_{x' \in \mathcal{N}(x_t)} \mathrm{Cost}(x')$, for $t = 1, 2, \ldots$ (for a minimization problem), continuing until a locally optimal solution, i.e., a solution $x \in \mathcal{X}$ such that $\mathrm{Cost}(x) \leq \mathrm{Cost}(x')$ for all $x' \in \mathcal{N}(x)$, is found. Often, to increase efficiency, a local move is accepted only if it improves upon the current solution by a β-factor, for some $\beta \in (0, 1)$, and we instead converge to an approximate locally optimal solution. This method is more formally described in Algorithm 1. By setting β sufficiently close to 1, the resulting local search method can often be shown to converge in a polynomial number of iterations.

In practice, local search methods are appealing due to their ease of implementation, efficiency, and ability to find good solutions quickly [7, 11, 39]. Theoretical approximation guarantees have also been established for local search methods for a variety of problems [2, 3, 8, 12, 14, 25, 30]. Despite this widespread use in offline algorithm design, local search has been under-explored as a framework for designing combinatorial bandit algorithms. In fact, as far as we are aware, the connection between local search and bandit algorithms has been in the opposite direction — several papers use multi-armed bandit algorithms to improve the performance of local search algorithms in practical settings [33, 38, 50, 51]. Thus, the main question we study in this paper is:

Can we convert a good offline local search algorithm into an effective online bandit learning algorithm? If so, what regret guarantees can be achieved using this framework?

1.1 Our Approach and Results

Our main contribution is the development of a general framework for converting an offline local search algorithm to an online stochastic bandit algorithm. It requires that the local search neighborhood must admit (β, γ)-improving moves which, informally, means that the neighborhood of a solution x with $\mathrm{Cost}(x) > \gamma \, \mathrm{Cost}(x^*)$, where x^* is an optimal solution, must contain a solution x' which is a β-factor improvement over x, in other words, $\mathrm{Cost}(x') \leq \beta \, \mathrm{Cost}(x)$. In particular, it is straightforward to verify that any problem on which Algorithm 1 terminates in an approximately optimal solution permits (β, γ)-improving moves for appropriate choices of β and γ. Our main result is that local search algorithms with (β, γ)-improving neighbourhoods can be used online to guarantee γ-regret with $O(\log^3 T)$ dependence on T.

Theorem 1.1 (Informal version of Theorem 3.1). *Suppose a problem admits (β, γ)-improving moves. Then we can construct an algorithm for the stochastic online variant with bandit feedback with*

$$O\left(M \cdot C_{\max} \cdot \mathrm{poly}(\beta^{-1}) \cdot \log^3 T\right) \quad \gamma\text{-regret after } T \text{ rounds,}$$

where $C_{\max} = \max_{x \in \mathcal{X}, z \in \mathcal{Z}} \mathrm{Cost}(x, z)$ is an upper bound on solution costs and $M = \max_{x \in \mathcal{X}} |\mathcal{N}(x)|$ is the largest neighborhood of a feasible solution.

We remark that M, $C_{\max}$, and β^{-1} all can be bounded polynomially in problem specific parameters for each of the applications we consider, and are thus independent of T. Note that β may depend on γ, so the regret bound may deteriorate as we get closer to exact ($\gamma = 1$) regret. Intuitively, more samples are required to distinguish between solutions with objective function value OPT and $\gamma \cdot$ OPT. The poly-logarithmic dependence on T is perhaps most surprising. While direct comparisons between offline-to-online frameworks is difficult, this is a substantially better dependence on T than the $O(T^{2/3})$ γ-regret guarantee obtained by Niazadeh et al. [41] via offline greedy algorithms, at the cost of multiplicative factors that depend on the problem and local search neighborhood.

The resulting algorithm, in essence, attempts to mimic Algorithm 1 while handling the complications that sprout from the fact that only the stochastically realized cost of the solution submitted in each period is observed. First, the cost of a solution must be estimated by sampling for several time periods. The algorithm attempts to maintain, with high probability, a solution with expected cost below a threshold, which decreases geometrically over time. Whenever the estimated cost of the current solution exceeds this threshold, the solutions in its neighborhood is explored. Neighboring solutions must be carefully sampled to ensure that even neighbors with very high costs do not incur large regret, at the same time, we need enough samples to identify β-improving moves with high probability whenever the current solution is not a γ-approximation of the optimal solution. This is done by using a successive-elimination style algorithm [21, 22] on the neighborhood. When a β-improving move is found during exploration it becomes the new current solution. This process repeats until time T or a local optimum is found.

To round out our results, we apply our framework to the repeated online bandit feedback versions of three problems: scheduling to minimize total completion time under stochastic job sizes, finding a minimum cost base in a matroid under stochastic element costs, and uncertain k-median clustering.

Scheduling to Minimize Total Completion Time: When scheduling to minimize total completion time, a feasible solution is a permutation π of n jobs which schedules job $j \in [n]$ in position $\pi(j)$. In each time step the size P_j of each job j is drawn from a distribution. Given job sizes $P_1, \ldots, P_n$, the cost of solution π is $\mathrm{Cost}(\pi, P) = \sum_{j=1}^{n} (n - \pi(j) + 1) P_j$, the total makespan. We show the neighborhood consisting of swapping a single pair of jobs is $(1 - \epsilon/n^2, 1 + \epsilon)$-improving. As a result, we have an algorithm for minimizing online total completion time with $(1 + \epsilon)$-regret in the order of $O\left(n^{12} \log^3 T/\epsilon^4\right)$.

Minimum Cost Base in a Matroid: For the problem of finding a minimum cost base for a matroid, a feasible solution is a base, or maximal independent set, B. Given stochastic realizations of element costs Z, the cost of base B is $\mathrm{Cost}(B) = \sum_{s \in B} Z_s$. We show the neigborhood which consist of bases constructed by adding an arbitrary element to the current solution and removing an element from the resulting circuit is $(1 - \epsilon/(2r), 1 + \epsilon)$-improving, where r is the rank of the matroid. This implies an online algorithm with $(1 + \epsilon)$-regret in the order of $O\left(nr^6 \log^3 T/\epsilon^4\right)$, where n is the size of the ground set. A special case of this result establishes a similar regret bound for the problem of finding minimum cost spanning trees in a graph.

Uncertain k-Median Clustering: In the uncertain k-median clustering problem the locations of n points are sampled in a metric space with diameter 1 and the task is to select k cluster centers from a set of m potential centers to minimize the cost, defined as the sum of distances from each point to its closest cluster center. We use a local search algorithm analysed by Arya et al. [3] and Cormode and McGregor [16] to show the single swap neighbourhood is $(1 - 1/n^2, 5(1 - 1/n))$-improving, implying $5(1 - 1/n))$-regret which is $O\left(n^9 m^2 \log^3 T\right)$.

1.2 Additional Related Work

As discussed above, we study a combinatorial bandit setting [4, 13, 15], and our work is closest to other 'offline-to-online' frameworks including Kalai and Vempala [32] and Dudík et al. [20], where it is assumed that the offline version of the problem can be easily solved, and Niazadeh et al. [41], which considers problems with robust greedy approximation algorithms satisfying a property called *Blackwell reducibility*. We also highlight the recent work of Agarwal et al. [1] which gives a general framework for online learning in monotone stochastic optimization problems achieving $O(\sqrt{T \log T})$ regret against the best approximation algorithm for the offline problem under known distributions. Notably, their results hold in the *semi-bandit* setting in which the algorithm observes realizations of some (but not all) of the underlying random variables in addition to the realized cost. Our results hold for the pure bandit setting where only the cost information is observed. Next, we briefly touch on other related topics.

Logarithmic Regret in Online Learning: Due to the existence of strong lower bounds in many online learning settings [5, 34], it is typically necessary to make additional assumptions concerning the environment, or relax the benchmark, in order to achieve regret bounds improving beyond the typical $O(T^\rho)$ for some $\rho \in (0, 1)$. For example, in online convex optimization, logarithmic regret is achievable if the sequence of convex functions satisfies *strong convexity* [29, 44]. Similarly, logarithmic regret may be possible if certain problem-dependent parameters are bounded appropriately. For example, Xu and Wang [48] achieve logarithmic regret in feature-based dynamic pricing whenever the minimum eigenvalue of a problem-dependent matrix is bounded from below, and Vera et al. [46] achieve logarithmic regret for contextual bandits with knapsacks whenever the weight of an item is bounded from below.

In contrast to assuming additional structure, we give poly-logarithmic regret guarantees against a relaxed benchmark, so-called γ-regret. Our bound is not problem-dependent, and holds for a wide class of instances as long as the local search neighborhood admits (β, γ)-improving moves. For settings where the corresponding offline problem is NP-hard, it is necessary to allow a multiplicative approximation factor. our result is especially interesting in this context, since we achieve γ-regret that scales as $O(\log^3 T)$ while prior frameworks including those due to Kakade et al. [31] and Niazadeh et al. [41] have γ-regret which scales as $T^{2/3}$.

Efficient Bandit Algorithms: The exploration strategies specified by many standard approaches to bandit problems can be computationally intractable for large action spaces [35]. As a result, there is significant interest in developing algorithms with more efficient implementations [17, 40, 42, 47, 49]. Since our algorithm is based on local search and performs exploration locally, it naturally lends itself to a computationally efficient implementation.

1.3 Roadmap

We organize the rest of this paper as follows. Section 2 formally sets up our model and recalls preliminary results we need for our analysis. Then we describe our 'offline-to-online' algorithm utilizing local search and provide a high-level overview of its analysis in Section 3. Following this, Section 4 demonstrates the applicability of our framework on the problems we discussed above. Section 5 concludes the paper and discusses potential directions for future work.

2 Preliminaries

We now formally define the general problem setting we consider — stochastic combinatorial bandits with local search. First, each instance is associated with a set $\mathcal{X}$ of feasible solutions[1] and for each $x \in \mathcal{X}$ there is a random non-negative cost associated with it. We model the randomness as a collection of latent variables taking values in some space $\mathcal{Z}$, for which there is an unknown distribution $\mathcal{D} \in \Delta(\mathcal{Z})$ over the possible values the latent variables can take. Thus we can model the cost as a function $\mathrm{Cost} : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}_+$, which induces a distribution on the realized cost for a fixed $x \in \mathcal{X}$ as $\mathrm{Cost}(x, Z)$, where $Z \sim \mathcal{D}$. To streamline our notation, we let $\mathrm{Cost}(x) := \mathbb{E}_{Z\sim\mathcal{D}}[\mathrm{Cost}(x, Z)]$ be the expected cost of playing $x \in \mathcal{X}$ and set $\mathrm{OPT} := \min_{x\in\mathcal{X}} \mathrm{Cost}(x)$ to be the minimum expected cost of any feasible solution.

In the online bandit setting, we consider making a sequence of decisions $\{x_t\}_{t=1}^T$, where each $x_t \in \mathcal{X}$, in order to optimize an objective over a time horizon of T periods in the presence of bandit (cost) feedback. An algorithm (or policy) $\mathcal{A}$ is a sequence of maps $\{\mathcal{A}_t : \mathcal{H}_{t-1} \to \mathcal{X}\}_{t=1}^T$ which each takes the observed history H^{t-1} up to period $t-1$ and returns a new solution x_t to be used in period t [2]. In the bandit setting, a history up to period t is a sequence $\{X_s, Y_s\}_{s=1}^t$, where $Y_s = \mathrm{Cost}(X_s, Z_s)$ is the observed random cost at step s for some independent sample $Z_s \sim \mathcal{D}$. Notice $\{Z_s\}_{s=1}^T$, the realizations of randomness that underlying the cost or reward, is unobserved. Formally, histories are random variables over $(\mathcal{X} \times \mathbb{R}_+)^t$. We let $\mathcal{H}^t = (\mathcal{X} \times \mathbb{R}_+)^t$ be the set of all length t sequences over solution-cost pairs. For simplicity, we let $\mathcal{A}_1$ be a constant function since no history has been observed at period 1. Our main interest is giving algorithms with low γ-regret, which measures the difference in cumulative expected cost between the algorithm and using a γ-approximate solution for all T periods, which we can now define formally.

Definition 2.1 (γ-Regret). For an algorithm $\mathcal{A}$, latent distribution $\mathcal{D}$, time horizon T, and $\gamma \geq 1$, we define

$$\mathrm{Regret}_\gamma(\mathcal{A}, \mathcal{D}, T) := \sum_{t=1}^{T} \mathbb{E}\left[\mathrm{Cost}(X_t, Z_t)\right] - \gamma \cdot T \cdot \mathrm{OPT} \tag{1}$$

to be the T-period γ-regret of algorithm $\mathcal{A}$ on distribution $\mathcal{D}$, where the expectation is taken with respect to the independent samples $(Z_1, Z_2, \dots, Z_T) \sim \mathcal{D}^T$ and any internal randomness utilized by $\mathcal{A}$. We say that algorithm $\mathcal{A}$ has γ-regret $R(T)$ if for all distributions $\mathcal{D}$, $\mathrm{Regret}_\gamma(\mathcal{A}, \mathcal{D}, T) \leq R(T)$.

We remark that if $\gamma = 1$, then we recover the standard definition of regret used in stochastic multi-armed bandits. As discussed in Section 1.1, our goal is to find algorithms with γ-regret whose dependence on T is not just $o(T)$ but is $O(\mathrm{poly}(\log T))$, which is $o(T^\rho)$ for any $\rho \in (0, 1)$.

Local search will allow us to avoid some of the difficulties that arise with exploring the large action spaces that can arise in stochastic combinatorial bandit problems. For a feasible solution space $\mathcal{X}$, let $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ be its neighborhood map so that for each $x \in \mathcal{X}$, $\mathcal{N}(x)$ is a set of neighboring

[1] We are mainly interested in the case where $\mathcal{X}$ is finite (but potentially large), so minimizing a function over $\mathcal{X}$ is well-defined.
[2] We focus on deterministic algorithms. We can extend to randomized algorithms by letting A_t be a map to $\Delta(\mathcal{X})$ and taking an additional expectation over the realized distribution of actions.

solutions to x. We assume that $M := \max_{x \in \mathcal{X}} |\mathcal{N}(x)|$ is a priori known to the algorithm, which holds in all the examples we consider. The key property of a neighborhood map that we require to achieve strong γ-regret bounds is that any solution which has expected cost worse than γOPT has a neighboring solution which has multiplicative improvement to its cost.

Definition 2.2 ((β, γ)-improving moves). Consider the problem specified by feasible set $\mathcal{X}$, neighborhood map $\mathcal{N}$, and expected cost function Cost : $\mathcal{X} \to \mathbb{R}_+$ as defined above. We say the problem admits (β, γ)-improving moves if, for any $x \in \mathcal{X}$ with $\mathrm{Cost}(x) > \gamma\mathrm{OPT}$, there exists $x' \in \mathcal{N}(x)$ with $\mathrm{Cost}(x') \leq \beta\,\mathrm{Cost}(x)$.

The parameters β and γ can depend on the structure of the problem (e.g., the number of jobs in completion time scheduling) and β can depend on γ (e.g., if $\gamma = 1 + \epsilon$, then β can depend on ϵ). A priori, it is perhaps unclear how to establish whether a problem permits (β, γ)-improving moves. Fortunately, there is a straightforward correspondence showing that all problems where Algorithm 1 terminates in an approximate solution must permit (β, γ)-improving moves.

Observation 1. *Given instance $(\mathcal{X}, \mathcal{N}, \mathrm{Cost})$, whenever Algorithm 1 is guaranteed to terminate in a γ-approximation for some choice of β, then $(\mathcal{X}, \mathcal{N}, \mathrm{Cost})$ permits (β, γ)-improving moves.*

Proof. We proceed by contradiction. Suppose that Algorithm 1 with parameter β terminates in a γ-approximation on a problem defined by $(\mathcal{X}, \mathcal{N}, \mathrm{Cost})$, but the problem does not permit (β, γ) improving moves. Then there exists solution $x \in \mathcal{X}$ such that $\mathrm{Cost}(x) > \gamma\mathrm{OPT}$ and $\mathrm{Cost}(x') > \beta\,\mathrm{Cost}(x)$ for all $x' \in \mathcal{N}(x)$. However, when using x as the starting x_0 in Algorithm 1, the algorithm converges in some solution x° with $\mathrm{Cost}(x^\circ) \leq \gamma\mathrm{OPT}$. Now either $x = x^\circ$, or the algorithm proceeds to a subsequent solution which yields a β-factor improvement en route to x°. In either case, this contradicts $(\mathcal{X}, \mathcal{N}, \mathrm{Cost})$ not permitting (β, γ)-improving moves. □

Finally, we require the following version of the Chernoff bounds, see, e.g., [19] for a reference.

Theorem 2.3. *Suppose that $X_1, X_2, \ldots, X_N$ are independent random variables in the interval $[0, 1]$. Let $\mu = \frac{1}{N}\sum_i \mathbb{E}[X_i]$ and $\bar{X} = \frac{1}{N}\sum_s X_s$, then for all $\delta \in (0, 1)$ and any μ_H, μ_L such that $\mu \in [\mu_L, \mu_H]$ we have:*

$$\Pr\left[\bar{X} > (1+\delta)\mu_H\right] \leq \exp\left(-\frac{\delta^2 N \mu_H}{3}\right) \quad \textit{and} \quad \Pr\left[\bar{X} < (1-\delta)\mu_L\right] \leq \exp\left(-\frac{\delta^2 N \mu_L}{3}\right).$$

3 Stochastic Combinatorial Bandits from Offline Local Search

Our algorithm is formally described in Algorithms 2 and 3. The algorithm operates in a sequence of phases $\ell = 1, 2, \ldots$ which are managed by the "for" loop in Algorithm 2. Each phase keeps track of a solution x_ℓ and a cost threshold $\theta_\ell := \alpha^{\ell-1}C_{\max} = \beta^{(\ell-1)/2}C_{\max}$. The invariant which, due to stochasticity, we wish to maintain with high probability is that $\mathrm{Cost}(x_\ell) \leq \theta_\ell$. Thus, assuming $\mathrm{Cost}(x_\ell) \leq \theta_\ell$, in phase ℓ we wish to either determine that $\mathrm{Cost}(x_\ell) \leq \alpha\theta_\ell = \theta_{\ell+1}$, find some $x' \in \mathcal{N}(x_\ell)$ with $\mathrm{Cost}(x') \leq \alpha\,\mathrm{Cost}(x_\ell) \leq \alpha\theta_\ell = \theta_{\ell+1}$, or determine that no such x' exists. In the first case we may set $x_{\ell+1} = x_\ell$ to maintain the invariant, while in the second we may set $x_{\ell+1} = x'$ to maintain the invariant. In the last case, we will use the fact that $\mathcal{N}$ satisfies Definition 2.2 and no x' was found to conclude that $\mathrm{Cost}(x_\ell) \leq \gamma\mathrm{OPT}$, and thus we use the current solution x_ℓ for all remaining periods.

To achieve this, Algorithm 2 first uses solution x_ℓ for N_ℓ time steps to get a rough estimate of its cost, then compares it to a threshold. The threshold is set so that if $\mathrm{Cost}(x_\ell) \leq \alpha^2\theta_\ell$ then the estimated cost is highly likely to be smaller than the threshold. As discussed above, we then move to phase $\ell + 1$ with $x_{\ell+1} = x_\ell$. If instead, the estimated cost is larger than the threshold, we run the subroutine in Algorithm 3 which explores the neighborhood of x_ℓ. This subroutine returns a solution

x_{New}, which is either some $x' \in \mathcal{N}(x)$ or x_ℓ. If x_{New} is some $x' \in \mathcal{N}(x)$, then we set $x_{\ell+1} = x_{\text{New}}$ and move to phase $\ell + 1$. Critically, we will show that with high probability $\text{Cost}(x_{\text{New}}) \leq \alpha\theta_\ell = \theta_{\ell+1}$ to maintain the invariant in this case. Finally if $x_{\text{New}} = x_\ell$, this indicates that nothing significantly better was found and so we break out of the "for" loop over phase ℓ, setting $x_{\text{Last}} = x_\ell$ and using x_{Last} until period T since, by Definition 2.2, x_{Last} will satisfy $\text{Cost}(x_{\text{Last}}) \leq \gamma\text{OPT}$ with high probability.

As discussed, we want Algorithm 3 to either find a neighboring solution which improves upon x_ℓ by at least an α-factor or indicate that no such improvement exists in $\mathcal{N}(x_\ell)$, and for this to hold with high probability. It is thus inevitable that a significant number of time steps are spent exploring the neighbourhood of x_ℓ. Doing so without incurring high regret requires some care. A first attempt might involve sampling each neighboring solution N_ℓ times, as we did for x_ℓ in Algorithm 2, and comparing to an appropriate threshold. This may potentially incur linear regret in later phases when a neighboring solution could be significantly worse than the current solution x_ℓ. To avoid this, we sample neighboring solutions cautiously over a series of "sub-phases" of increasing length (Algorithm 3, line 4), where we progressively increase the amount of sampling only while the neighbouring solution has the potential to improve over x_ℓ. As a result, poor neighbours are eliminated in early subphases, reminiscent of the successive-elimination algorithm for stochastic multi-armed bandits [21], and sampled geometrically fewer times than the best solutions (and x_ℓ).

Let $\mathcal{A}_{\text{Local}}$ denote the algorithm described by Algorithms 2 and 3. Our main result is that the regret of $\mathcal{A}_{\text{Local}}$ scales poly-logarithmically with T as long as the underlying problem admits (β, γ)-improving moves.

Theorem 3.1. *Suppose that $\mathcal{X}$, $\mathcal{N}$, and the cost function $\text{Cost}(\cdot) = \mathbb{E}_{z\sim\mathcal{D}}[\text{Cost}(\cdot, z)]$ induced by $\mathcal{D}$ admit (β, γ)-improving moves. Then for all distributions $\mathcal{D}$ and all sufficiently large T, we have*

$$\text{Regret}_\gamma(\mathcal{A}_{\text{Local}}, \mathcal{D}, T) = O\left(\frac{MC_{\max}\log^2 T}{\delta^2\alpha^2\log^2\frac{1}{\alpha}}(\log T + \log M)\right),$$

where $\mathcal{A}_{\text{Local}}$ is given by Algorithms 2 and 3, $\alpha = \sqrt{\beta}$, $C_{\max} = \max_{x\in\mathcal{X}, z\in\mathcal{Z}} \text{Cost}(x, z)$, and $\delta = \frac{1-\alpha}{1+\alpha}$.

3.1 Analysis

We now perform the analysis which will yield Theorem 3.1. First, we encapsulate our applications of concentration inequalities into the following lemma which we will apply several times.

Lemma 3.2. *Fix a solution $x \in \mathcal{X}$ and let $z_1, z_2, \ldots, z_N$ be $N := 3C_{\max}(4\log T + \log M)/(\delta^2\alpha^2\theta)$ independent samples from $\mathcal{D}$, where $\ell, M \in \mathbb{N}$, $\theta > 0$, $\alpha \in (0, 1)$, and $\delta = \frac{1-\alpha}{1+\alpha} \in (0, 1)$. Let $\widehat{\text{Cost}}(x) = \frac{1}{N}\sum_{s=1}^{N}\text{Cost}(x, z_s)$. Then*

(a) $\text{Cost}(x) \leq \alpha^2\theta \implies \Pr_{z_1,z_2,\ldots,z_N\sim\mathcal{D}^N}\left[\widehat{\text{Cost}}(x) > \frac{2\alpha^2}{1+\alpha}\theta\right] \leq M^{-1}T^{-4}$, *and*

(b) $\text{Cost}(x) \geq \alpha\theta \implies \Pr_{z_1,z_2,\ldots,z_N\sim\mathcal{D}^N}\left[\widehat{\text{Cost}}(x) \leq \frac{2\alpha^2}{1+\alpha}\theta\right] \leq M^{-1}T^{-4}$.

Proof. For brevity, we write $\Pr[\cdot] := \Pr_{z_1,z_2,\ldots,z_N\sim\mathcal{D}^N}[\cdot]$. Setting $X_s = \text{Cost}(x, z_s)/C_{\max}$ for all $s \in [N]$, we have that $X_s \in [0, 1]$ and $\mathbb{E}[X_s] = \text{Cost}(x)/C_{\max}$. Observe that $\frac{2\alpha^2}{1+\alpha} = (1+\delta)\alpha^2 = (1-\delta)\alpha$ for our choice of δ. For part (a), when $\text{Cost}(x) \leq \alpha^2\theta$, observe that

$$\Pr\left[\widehat{\text{Cost}}(x) > \frac{2\alpha^2}{1+\alpha}\theta\right] = \Pr\left[\frac{\widehat{\text{Cost}}(x)}{C_{\max}} > (1+\delta)\frac{\alpha^2\theta}{C_{\max}}\right] = \Pr\left[\frac{1}{N}\sum_{s=1}^{N}X_s > (1+\delta)\frac{\alpha^2\theta}{C_{\max}}\right].$$

Algorithm 2: Local Search for Stochastic Combinatorial Bandits

Data: Feasible set $\mathcal{X}$, Max cost $C_{\max}$, $M = \sup_{x \in \mathcal{X}} |\mathcal{N}(x)|$, and Parameters β, γ from Definition 2.2

Result: Sequence of solutions with $O(\log^3 T)$ γ-regret

Procedure `BanditLocalSearch`($\mathcal{X}, C_{\max}, \beta, \gamma$):

- $x_1 \leftarrow$ any solution in $\mathcal{X}$
- $\alpha \leftarrow \sqrt{\beta}$
- $\delta \leftarrow (1-\alpha)/(1+\alpha)$
- $\theta_1 \leftarrow C_{\max}$
- **for** phase $\ell = 1, 2, \ldots$ **do**
  - `//Test the solution` x_ℓ `for the current phase` ℓ
  - $N_\ell \leftarrow 3C_{\max}\left(4\log T + \log M\right)/\left(\delta^2\alpha^2\theta_\ell\right)$
  - Use solution x_ℓ for N_ℓ periods
  - $\widehat{\text{Cost}}(x_\ell) \leftarrow$ average cost of using x in these periods
  - **if** $\widehat{\text{Cost}}(x_\ell) > \left(\frac{2\alpha^2}{1+\alpha}\right)\theta_\ell$ **then**
    - $x_{\text{New}} \leftarrow$ `TestNeighborhood`($x_\ell, \ell, \mathcal{N}$)
    - **if** $x_{\text{New}} = x_\ell$ **then**
      - `//`x_ℓ `is locally optimal with high probability`
      - $x_{\text{Last}} \leftarrow x_\ell$
      - **break**
    - **else**
      - `//`x_{New} `is better than` x_ℓ `with high probability`
      - $x_{\ell+1} \leftarrow x_{\text{New}}$
  - **else**
    - $x_{\ell+1} \leftarrow x_\ell$
  - $\theta_{\ell+1} \leftarrow \alpha\theta_\ell$
- Use solution x_{Last} until period T

Since $X_s \in [0,1]$ and $\mathbb{E}[X_s] \le \alpha^2\theta/C_{\max}$, we may bound the right hand side using Theorem 2.3 as

$$\Pr\left[\frac{1}{N}\sum_{s=1}^{N} X_s > (1+\delta)\frac{\alpha^2\theta}{C_{\max}}\right] \le \exp\left(-\frac{\delta^2 N\alpha^2\theta}{3C_{\max}}\right) = M^{-1}T^{-4},$$

where the right hand side follows from our choice of N.

For part (b), when $\text{Cost}(x) \ge \alpha\theta$, it follows that

$$\Pr\left[\widehat{\text{Cost}}(x) \le \frac{2\alpha^2}{1+\alpha}\theta\right] = \Pr\left[\frac{\widehat{\text{Cost}}(x)}{C_{\max}} \le (1-\delta)\frac{\alpha\theta}{C_{\max}}\right] = \Pr\left[\frac{1}{N}\sum_{s=1}^{N} X_s \le (1-\delta)\frac{\alpha\theta}{C_{\max}}\right].$$

Again, since $X_s \in [0,1]$ and $\mathbb{E}[X_s] \ge \alpha\theta/C_{\max}$, we bound the right hand side using Theorem 2.3 as

$$\Pr\left[\frac{1}{N}\sum_{s=1}^{N} X_s \le (1-\delta)\frac{\alpha\theta}{C_{\max}}\right] \le \exp\left(-\frac{\delta^2 N\alpha\theta}{3C_{\max}}\right) \le M^{-1}T^{-4},$$

where the right hand side follows from our choice of N and $\alpha \le 1$. This completes the proof. □

Algorithm 3: Neighboring Solution Tester

Data: Current solution x_ℓ, Phase number ℓ, Neighborhood $\mathcal{N}$
Result: Some solution $x' \in \mathcal{N}(x_\ell)$ or x_ℓ

Procedure `TestNeighborhood`(x_ℓ, ℓ, $\mathcal{N}$):
  for $x' \in \mathcal{N}(x_\ell)$ **do**
    `//Test the solution` x'
    Better $\leftarrow$ True
    for sub-phase $\ell' = 1, 2, \ldots, \ell$ **do**
      $N_{\ell'} \leftarrow 3C_{\max}\left(4\log T + \log M\right) / \left(\delta^2\alpha^2\theta_{\ell'}\right)$
      Use solution x' for $N_{\ell'}$ periods
      $\widehat{\text{Cost}}(x') \leftarrow$ average cost of using x' in these periods
      `//If` x' `is not significantly better than` x`, move on from` x'
      if $\widehat{\text{Cost}}(x') > \left(\frac{2\alpha^2}{1+\alpha}\right)\theta_{\ell'}$ **then**
        Better $\leftarrow$ False
        break
    `//If` x' `passes all tests, update` $x_{\ell+1}$ `to be` x'
    if Better = True **then**
      return x'
  `//Nothing significantly better found`
  return x_ℓ

3.1.1 Handling Bad Events. Let L be the index of the (random) last phase that is encountered in Algorithm 2. To bound the overall regret, we will bound the total regret in phases with index $\ell < L$ and also show that the regret in the last phase L is negligible. Before we can do that, we need to define an appropriate sequence of "bad events" which are exceedingly unlikely, for which will be able to bound the regret when conditioning on their negation, as is standard.

Definition 3.3. For each phase ℓ, let $\mathcal{M}_\ell$ be the event that Algorithm 2 makes it to phase ℓ and let C_ℓ be the event that $\text{Cost}(x_\ell) > \theta_\ell$. Then we define the bad event in phase ℓ as $\mathcal{B}_\ell := \mathcal{M}_\ell \wedge C_\ell$ and $\mathcal{G}_\ell = \neg\mathcal{B}_\ell$.

We will show that these events are unlikely. Intuitively, this will follow inductively by assuming that $\mathcal{B}_\ell$ is unlikely, and then we can use the definition of our algorithm to bound $\Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell]$ and show that $\mathcal{B}_{\ell+1}$ is unlikely via a standard decomposition. The key step involves bounding $\Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell]$.

Lemma 3.4. *For each ℓ, we have* $\Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell] \le T^{-3}$

Proving this lemma requires careful analysis of the `TestNeighborhood` subroutine (Algorithm 3), which we postpone momentarily. For now, we show Lemma 3.4 implies a bound on $\Pr[\mathcal{B}_\ell]$.

Corollary 3.5. *For each ℓ, we have* $\Pr[\mathcal{B}_\ell] \le \ell T^{-3}$.

Proof. We proceed by induction on ℓ. For the base case when $\ell = 1$, we have that $\text{Cost}(x_1) \le C_{\max} \le \theta_1$, and so $\Pr[\mathcal{B}_1] = 0 \le T^{-3}$, proving the base case.

Inductively, we assume that $\Pr[\mathcal{B}_\ell] \leq \ell T^{-3}$ and aim to bound $\Pr[\mathcal{B}_{\ell+1}]$. By the law of total probability we have

$$\begin{aligned}\Pr[\mathcal{B}_{\ell+1}] &= \Pr[\mathcal{B}_{\ell+1} \mid \mathcal{B}_\ell] \Pr[\mathcal{B}_\ell] + \Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell] \Pr[\mathcal{G}_\ell] \\ &\leq \Pr[\mathcal{B}_\ell] + \Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell] \\ &\leq \ell T^{-3} + T^{-3} = (\ell+1) T^{-3}\end{aligned}$$

completing the proof of the inductive case and giving the corollary. □

3.1.2 Analysis of the `TestNeighborhood` *Subroutine.* We now turn to analyzing Algorithm 3. Consider a phase ℓ of Algorithm 2 in which Algorithm 3 is called on solution x_ℓ and fix an iteration of the outer 'for' loop in Algorithm 3 in which solution $x' \in \mathcal{N}(x_\ell)$ is considered. We call each iteration of the inner 'for' loop in Algorithm 3 a *sub-phase*, of which there are at most ℓ for each neighboring solution $x' \in \mathcal{N}(x)$. Ideally, if $\mathrm{Cost}(x') > \alpha\theta_\ell$, then we do not make it past sub-phase ℓ when considering solution x' (and actually, we need to be slightly more careful than this). The tools we develop here are needed to bound the regret incurred by using each neighboring solution x' across each sub-phase. To start, we introduce the following definitions.

Definition 3.6. For each $x' \in \mathcal{N}(x)$, we say that x' is a bad neighbor of x if $\mathrm{Cost}(x') \geq \alpha\theta_\ell$. Additionally, let $\mathcal{N}_B(x) \subseteq \mathcal{N}(x)$ be the set of bad neighbors of x and let $\ell'(x') \in [\ell]$ be the sub-phase index such that $\mathrm{Cost}(x') \in [\alpha\theta_{\ell'(x')}, \theta_{\ell'(x')}]$ for each $x' \in \mathcal{N}_B(x)$.

Definition 3.7. Let $\mathcal{B}'_\ell$ be the event that in phase ℓ some bad neighbor $x' \in \mathcal{N}_B(x_\ell)$ makes it past sub-phase $\ell'(x')$ when Algorithm 3 is called.

Note that if any bad neighbor x' makes it past sub-phase ℓ, then Algorithm 3 returns x'. Further, if any bad neighbor x' makes it past sub-phase $\ell'(x')$, then we may incur too much regret from testing x' in later sub-phases since $\mathrm{Cost}(x') \geq \alpha\theta_{\ell'(x')}$. We show that $\mathcal{B}'_\ell$ is unlikely, and conclude that bad neighbors are highly likely to be removed from contention early enough to avoid incurring high regret.

Lemma 3.8. $\Pr[\mathcal{B}'_\ell] \leq T^{-4}$.

Proof. In the case that Algorithm 3 is not called in phase ℓ, then the probability is at most 0 since $\mathcal{B}'_\ell$ cannot happen. Formalizing this, let $\mathcal{T}_\ell$ be the event that Algorithm 3 is called in phase ℓ, then we have $\Pr[\mathcal{B}'_\ell \mid \neg\mathcal{T}_\ell] = 0$. In the other case where Algorithm 3 is called in phase ℓ, then we need to show that it is unlikely for any bad neighbor x' to make it past sub-phase $\ell'(x')$. To this end, fix a bad neighbor x'. If x' doesn't make it to sub-phase $\ell'(x') \leq \ell$, then it clearly doesn't make it past sub-phase ℓ. Thus in order for event $\mathcal{B}'_\ell$ to happen due to x', we must have that x' makes it to sub-phase $\ell'(x')$ and makes it past this sub-phase. Now, x' only makes it past sub-phase $\ell'(x')$ when $\widehat{\mathrm{Cost}}(x') \leq (2\alpha^2/(1+\alpha))\theta_{\ell'(x')}$, where $\widehat{\mathrm{Cost}}(x')$ is the average of $N_{\ell'(x')}$ independent samples with distribution $\mathrm{Cost}(x', Z)$, where $Z \sim \mathcal{D}$. By part(b) of Lemma 3.2, the probability of this is at most $M^{-1}T^{-4}$. It follows that

$$\begin{aligned}\Pr[\mathcal{B}'_\ell \mid \mathcal{T}_\ell] &\leq \Pr\left[\bigcup_{x' \in \mathcal{N}_B(x)} \left(\widehat{\mathrm{Cost}}(x') \leq \frac{2\alpha^2}{1+\alpha}\theta_{\ell'(x')}\right)\right] \\ &\leq \sum_{x' \in \mathcal{N}_B(x)} \Pr\left[\widehat{\mathrm{Cost}}(x') \leq \frac{2\alpha^2}{1+\alpha}\theta_{\ell'(x')}\right] \leq T^{-4}\end{aligned}$$

Finally, observe that

$$\Pr[\mathcal{B}'_\ell] = \Pr[\mathcal{B}'_\ell \mid \mathcal{T}_\ell] \Pr[\mathcal{T}_\ell] + \Pr[\mathcal{B}'_\ell \mid \neg\mathcal{T}_\ell] \Pr[\neg\mathcal{T}_\ell] \leq T^{-4}(\Pr[\mathcal{T}_\ell] + \Pr[\neg\mathcal{T}_\ell]) = T^{-4}$$

which completes the proof. □

Lemma 3.8 establishes that Algorithm 3 filters out bad neighboring solutions with high probability. In order to prove Lemma 3.4, we also need to show that good neighboring solutions are unlikely to be filtered out. More specifically, there are three situations to consider (all under the condition that $\mathrm{Cost}(x_\ell) \le \theta_\ell$): (1) every solution in $\mathcal{N}(x_\ell)$ is a bad neighbor, (2) some solution $x' \in \mathcal{N}(x_\ell)$ satisfies $\mathrm{Cost}(x') \le \alpha^2\theta_\ell$, and (3) not all neighboring solutions are bad, but all solutions that are not bad have $\mathrm{Cost}(x') \in [\alpha^2\theta_\ell, \alpha\theta_\ell]$.

In situation (1), it is a simple corollary of Lemma 3.8 that Algorithm 3 will output x_ℓ with high probability, indicating that we did not find an improved solution (and therefore $\mathrm{Cost}(x_\ell) \le \gamma\mathrm{OPT}$). In situation (2), where there is a neighboring solution x' with $\mathrm{Cost}(x') \le \alpha^2\theta_\ell$, it is possible that Algorithm 3 considers and returns a different neighboring solution x'' with $\mathrm{Cost}(x'') \in [\alpha^2\theta_\ell, \alpha\theta_\ell]$ (again it is unlikely to return a bad neighbor due to Lemma 3.8). However, if x' is ever considered it will survive past sub-phase ℓ with high probability (Lemmas 3.2 and 3.8) and thus be returned by Algorithm 3. Either outcome is acceptable and guarantees at least an α-factor improvement. In situation (3), Algorithm 3 may output x_ℓ, indicating no improved solution. This follows since, under the event that $\mathrm{Cost}(x_\ell) \le \theta_\ell$, each neighboring solution has $\mathrm{Cost}(x') \ge \alpha\theta_\ell \ge \beta\,\mathrm{Cost}(x_\ell)$, which by the (β, γ)-improving moves condition implies that $\mathrm{Cost}(x_\ell) \le \gamma\mathrm{OPT}$. It is also acceptable for Algorithm 3 to output a solution with cost in the interval $[\alpha^2\theta_\ell, \alpha\theta_\ell]$, since that guarantees an α-factor improvement. The following lemma formalizes this.

Lemma 3.9. *Suppose that Algorithm 3 is run in phase ℓ with solution x_ℓ and let x_{New} be the random solution which it outputs. Define the events $\mathcal{F}_{\ell,1}, \mathcal{F}_{\ell,2}, \mathcal{F}_{\ell,3}$ as follows:*

- $\mathcal{F}_{\ell,1} = \{\mathcal{N}_B(x_\ell) = \mathcal{N}(x_\ell)\}$ *(all neighbors of x_ℓ are bad)*
- $\mathcal{F}_{\ell,2} = \{\exists x' \in \mathcal{N}(x_\ell), \mathrm{Cost}(x') \le \alpha^2\theta_\ell\}$ *(there is a neighbor with a $\beta = \alpha^2$-factor decrease)*
- $\mathcal{F}_{\ell,3} = \{\forall x' \in \mathcal{N}(x_\ell) \backslash \mathcal{N}_B(x_\ell) \ne \emptyset, \mathrm{Cost}(x') \in [\alpha^2\theta_\ell, \alpha\theta_\ell]\}$ *(no neighbor has an $\beta = \alpha^2$-factor decrease, but there is one with an α-factor decrease).*

Then

(a) $\Pr[x_{\mathrm{New}} \ne x_\ell \mid \wedge\mathcal{F}_{\ell,1}] \le T^{-4}$
(b) $\Pr[\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell \mid \mathcal{F}_{\ell,2}] \le (\ell+1)T^{-4}$
(c) $\Pr[(\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell) \wedge (x_{\mathrm{New}} \ne x_\ell) \mid \mathcal{F}_{\ell,3}] \le T^{-4}$

Proof. Part (a): Conditioned on $\mathcal{F}_{\ell,1}$, the only way that $x_{\mathrm{New}} \ne x_\ell$ is if a bad neighboring solution makes it past sub-phase ℓ. This implies that event $\mathcal{B}'_\ell$ occurs, and it follows by Lemma 3.8 that

$$\Pr[x_{\mathrm{New}} \ne x_\ell \mid \mathcal{F}_{\ell,1}] \le \Pr[\mathcal{B}'_\ell] \le T^{-4}.$$

Part (b): There are two ways we can have $\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell$ under condition $\mathcal{F}_{\ell,2}$. One way is that a bad neighbor has been output as x_{New}, and thus $\mathcal{B}'_\ell$ has occurred as discussed previously. The other way is if $x_{\mathrm{New}} = x_\ell$ and $\mathrm{Cost}(x_\ell) > \alpha\theta_\ell$, in which case it must be that the solution x' with $\mathrm{Cost}(x_\ell) \le \alpha^2\theta_\ell$ was considered but did not make it past sub-phase ℓ. Let $\mathcal{B}'_\ell(x')$ denote this latter event. Thus by a union bound we have

$$\Pr[\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell \mid \mathcal{F}_{\ell,2}] \le \Pr[\mathcal{B}'_\ell] + \Pr[\mathcal{B}'_\ell(x')].$$

Again, from Lemma 3.8, $\Pr[\mathcal{B}'_\ell] \le T^{-4}$. It remains to bound $\Pr[\mathcal{B}'_\ell(x')]$. This event happens if x' fails at least one of the checks that occur when x' is considered in the inner 'for' loop of Algorithm 3. By part (a) of Lemma 3.2 and a union bound, we have

$$\Pr[\mathcal{B}'_\ell(x')] \le \ell M^{-1}T^{-4} \le \ell T^{-4}.$$

Combining these bounds completes part (b).

Part (c): We can only have $\mathrm{Cost}(x_\ell) > \alpha\theta_\ell$ and $x_{\mathrm{New}} \neq x_\ell$ if some bad neighbor makes it past sub-phase ℓ, which can only occur if $\mathcal{B}'_\ell$ has occurred. Thus we have

$$\Pr[(\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell) \wedge (x_{\mathrm{New}} \neq x_\ell) \mid \mathcal{F}_{\ell,3}] \leq \Pr[\mathcal{B}'_\ell] \leq T^{-4}$$

by Lemma 3.8. This yields the claim for the last case and completes the proof of the lemma. □

As a direct corollary of Lemma 3.9, we get that Algorithm 3 is unlikely to output a solution with cost more than $\alpha\theta_\ell$ when run in phase ℓ.

Corollary 3.10. *Suppose that Algorithm 3 is run in phase ℓ with solution x_ℓ and let x_{New} be the random solution which it outputs. Then we have that* $\Pr[\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell] \leq (\ell+1)T^{-4}$.

Proof. Under event $\mathcal{F}_{\ell,1}$, the only way for Algorithm 3 to output a bad solution is to output something other than x_ℓ. Similarly, the only for Algorithm 3 to output a bad solution under event $\mathcal{F}_{\ell,3}$ is to output something other than x_ℓ which happens to be bad. Then it follows from the law of total probability and Lemma 3.9 that

$$\begin{aligned}
\Pr[\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell] =& \Pr[x_{\mathrm{New}} \neq x_\ell \mid \mathcal{F}_{\ell,1}] \Pr[\mathcal{F}_{\ell,1}] \\
&+ \Pr[\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell \mid \mathcal{F}_{\ell,2}] \Pr[\mathcal{F}_{\ell,2}] \\
&+ \Pr[(\mathrm{Cost}(x_{\mathrm{New}}) > \alpha\theta_\ell) \wedge (x_{\mathrm{New}} \neq x_\ell) \mid \mathcal{F}_{\ell,3}] \Pr[\mathcal{F}_{\ell,3}] \\
\leq& (\ell+1)T^{-4}(\Pr[\mathcal{F}_{\ell,1}] + \Pr[\mathcal{F}_{\ell,2}] + \Pr[\mathcal{F}_{\ell,3}]) \\
=& (\ell+1)T^{-4}.
\end{aligned}$$

□

3.1.3 Proof of Lemma 3.4. Now we return to the proof of Lemma 3.4 which will utilize the tools we developed from the analysis of Algorithm 3.

Proof of Lemma 3.4. Recall that $\mathcal{G}_\ell = \neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell$, i.e., the good event for phase ℓ happens if either Algorithm 2 doesn't make it to phase ℓ or we have $\mathrm{Cost}(x_\ell) \leq \theta_\ell$. First, we claim that $\Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell] \leq \Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{C}_\ell]$. To see this, we have:

$$\begin{aligned}
\Pr[\mathcal{B}_{\ell+1} \mid \mathcal{G}_\ell] &= \frac{\Pr[\mathcal{B}_{\ell+1} \wedge (\neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell)]}{\Pr[\neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell]} \\
&= \frac{\Pr[(\mathcal{B}_{\ell+1} \wedge \neg\mathcal{C}_\ell) \vee (\mathcal{B}_{\ell+1} \wedge \neg\mathcal{M}_\ell)]}{\Pr[\neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell]} \\
&\leq \frac{\Pr[\mathcal{B}_{\ell+1} \wedge \neg\mathcal{C}_\ell]}{\Pr[\neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell]} + \frac{\Pr[\mathcal{B}_{\ell+1} \wedge \neg\mathcal{M}_\ell]}{\Pr[\neg\mathcal{C}_\ell \vee \neg\mathcal{M}_\ell]} \\
&\leq \frac{\Pr[\mathcal{B}_{\ell+1} \wedge \neg\mathcal{C}_\ell]}{\Pr[\neg\mathcal{C}_\ell]} + \frac{\Pr[\mathcal{B}_{\ell+1} \wedge \neg\mathcal{M}_\ell]}{\Pr[\neg\mathcal{M}_\ell]} \\
&= \Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{C}_\ell] + \Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{M}_\ell] \\
&= \Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{C}_\ell]
\end{aligned}$$

The first line follows from the definition of $\mathcal{G}_\ell$ and conditional probability while the second line follows from the distributive law for $\wedge$ and $\vee$. We use a union bound in the third line and the observation that $\mathcal{G}_\ell$ contains both $\neg\mathcal{C}_\ell$ and $\neg\mathcal{M}_\ell$ in the fourth line. To finish, the fifth line is again the definition of conditional probability and the last step uses the observation that $\Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{M}_\ell] = 0$ since it is impossible for the algorithm to make it to phase $\ell+1$ if it has not made it to phase ℓ.

Thus we may focus on bounding $\Pr[\mathcal{B}_{\ell+1} \mid \neg\mathcal{C}_\ell]$. Under condition $\neg\mathcal{C}_\ell$, $\mathrm{Cost}(x_\ell) \leq \theta_\ell$ by definition. We analyze three cases depending on how $\mathrm{Cost}(x_\ell)$ relates to θ_ℓ.

Case 1: $\mathrm{Cost}(x_\ell) \leq \alpha^2\theta_\ell$**.** In this case, we have $\mathrm{Cost}(x_\ell) \leq \alpha\theta_{\ell+1}$ and the current cost is small enough to move to phase $\ell+1$. Denote the event that $\mathrm{Cost}(x_\ell) \leq \alpha^2\theta_\ell$ by $\mathcal{E}_{\ell,1}$. We will show that $\Pr[\mathcal{B}_{\ell+1} \mid \neg C_\ell \wedge \mathcal{E}_{\ell,1}]$ is small. If Algorithm 2 moves to phase $\ell+1$ with $x_{\ell+1} = x_\ell$, then $\mathrm{Cost}(x_{\ell+1}) < \theta_{\ell+1}$ so $\mathcal{B}_{\ell+1}$ doesn't happen. Thus the only way for $\mathcal{B}_{\ell+1}$ to happen is if the "if" statement on line 10 evaluates to true, which occurs when $\widehat{\mathrm{Cost}}(x_\ell) > 2\alpha^2\theta_\ell/(1+\alpha)$. We will show that this happens with small probability. Since $\widehat{\mathrm{Cost}}(x_\ell)$ is the average of N_ℓ independent samples each with mean $\mathrm{Cost}(x_\ell)$, by the first part of Lemma 3.2 and the discussion above, we have

$$\Pr[\mathcal{B}_{\ell+1} \mid \neg C_\ell \wedge \mathcal{E}_{\ell,1}] \leq \Pr\left[\widehat{\mathrm{Cost}}(x_\ell) > \frac{2\alpha^2}{1+\alpha}\theta_\ell\right] \leq M^{-1}T^{-4} \leq T^{-4}, \tag{2}$$

completing the argument for this case.

Case 2: $\alpha\theta_\ell \leq \mathrm{Cost}(x_\ell) \leq \theta_\ell$**.** For this case, we want Algorithm 3 to be called with high probability and that it succeeds with high probability. Denote the event that $\alpha\theta_\ell \leq \mathrm{Cost}(x_\ell) \leq \theta_\ell$ by $\mathcal{E}_{\ell,2}$. In this case, there are two ways that we could fail to satisfy $\mathrm{Cost}(x_{\ell+1}) \leq \theta_{\ell+1}$. The first is by skipping the step which explores the neighborhood and moving to the next phase with $x_{\ell+1} = x_\ell$. This occurs when the "if" statement in line 10 of Algorithm 2 evaluates to False, which happens with probability at most $M^{-1}T^{-4}$ by part (b) of Lemma 3.2. Assuming Algorithm 3 is called, the second way that we could fail to satisfy $\mathrm{Cost}(x_{\ell+1}) \leq \theta_{\ell+1}$ is if Algorithm 3 returns a bad neighboring solution. Corollary 3.10 shows that this occurs with probability at most $(\ell+1)T^{-4} \leq T^{-3}$. Combining the bounds completes the analysis of this case.

Case 3: $\alpha^2\theta_\ell < \mathrm{Cost}(x_\ell) < \alpha\theta_\ell$**.** This last case is more flexible; we use $\mathcal{E}_{\ell,3}$ to denote the event that $\mathrm{Cost}(x_\ell) \in (\alpha^2\theta_\ell, \alpha\theta_\ell)$. Since $\mathrm{Cost}(x_\ell) \leq \alpha\theta_\ell$, we will be safe in either the situation that $\widehat{\mathrm{Cost}}(x_\ell) > 2\alpha^2\theta_\ell/(1+\alpha)$ (and so we set $x_{\ell+1} = x_\ell$) or not (and so we use Algorithm 3 to determine $x_{\ell+1}$). In the first situation, we clearly have $\mathrm{Cost}(x_\ell) \leq \alpha\theta_\ell = \theta_{\ell+1}$. In the other situation, we use the same argument as in the previous case to argue that the probability of setting $x_{\ell+1}$ to a bad neighbor is small. Thus we conclude that $\Pr[\mathcal{B}_{\ell+1} \mid \neg C_\ell \wedge \mathcal{E}_{\ell,3}] \leq T^{-4}$

Since the three cases above are exhaustive, i.e., $\mathcal{E}_{\ell,1} \vee \mathcal{E}_{\ell,2} \vee \mathcal{E}_{\ell,1} \equiv \mathsf{True}$ when conditioned on C_ℓ, we conclude by the law of total probability that

$$\Pr[\mathcal{B}_{\ell+1} \mid C_\ell] = \sum_{i=1}^{3} \Pr[\mathcal{B}_{\ell+1} \mid C_\ell \wedge \mathcal{E}_{\ell,i}] \Pr[\mathcal{E}_{\ell,i} \mid C_\ell] \leq T^{-3} \sum_{i=1}^{3} \Pr[\mathcal{E}_{\ell,i} \mid C_\ell] = T^{-3},$$

completing the proof of this lemma. □

3.1.4 Bounding the Number of Phases. Let L be the index of the last complete phase, i.e., Algorithm 2 either finds a local solution x_{Last} in phase L (which is then played until period T) or period T is reached in phase $L+1$. We will show that L is logarithmic in T, which will be helpful to establish our regret bound later.

LEMMA 3.11. $L \leq \log(T)/\log(\frac{1}{\alpha})$

Proof. By definition of L we know that $\sum_{\ell=1}^{L} N_\ell \leq T$, since this is just counting the number of plays due to line 7 in Algorithm 2. From the definition of N_ℓ, we have:

$$\begin{aligned} T &\geq \sum_{\ell=1}^{L} N_\ell \\ &= \sum_{\ell=1}^{L} \left(\frac{3C_{\max}}{\delta^2 \alpha^2 \theta_\ell} (4 \log T + \log M) \right) \\ &\geq \left(\frac{12 \log T}{\delta^2 \alpha^2} \right) \sum_{\ell=1}^{L} \left(\frac{1}{\alpha} \right)^{\ell-1} \\ &= \left(\frac{12 \log T}{\delta^2 \alpha^2} \right) \left(\frac{\left(\frac{1}{\alpha}\right)^L - 1}{\frac{1}{\alpha} - 1} \right). \end{aligned}$$

Rearranging this inequality to solve for L and simplifying yields result. □

3.1.5 Final Regret Analysis. To complete the analysis, we separately analyze the γ-regret incurred by the algorithm during phases 1 to L and the γ-regret after phase L.

Regret from Phases 1 *to L:* Consider phase ℓ. We incur regret from playing the solution x_ℓ and also from playing neighboring solutions when Algorithm 3 is called. Let $R_{\ell,1}$ and $R_{\ell,2}$ represent each of these quantities, respectively. We start by bounding $R_{\ell,1}$. Since OPT ≥ 0, by the law of total expectation we have:

$$\begin{aligned} \mathbb{E}[R_{\ell,1}] &= \mathbb{E}[R_{\ell,1} \mid \neg \mathcal{B}_\ell] \Pr[\neg \mathcal{B}_\ell] + \mathbb{E}[R_{\ell,1} \mid \mathcal{B}_\ell] \Pr[\mathcal{B}_\ell] \\ &\leq \theta_\ell N_\ell + C_{\max} N_\ell \ell T^{-3} \\ &\leq \frac{12 C_{\max}}{\delta^2 \alpha^2} (\log T + \log M) + C_{\max} T^{-1}. \end{aligned}$$

The first inequality follows since $\mathrm{Cost}(x_\ell) \leq \theta_\ell$ under $\neg \mathcal{B}_\ell$ (giving the first term) and Corollary 3.5 (which bounds $\Pr[\mathcal{B}_\ell]$). The second inequality follows from the definition of N_ℓ and the fact that N_ℓ and ℓ are both smaller than T.

Bounding $R_{\ell,2}$ is similar, but we have to account for each sub-phase. We see that:

$$\begin{aligned} \mathbb{E}[R_{\ell,2}] &= \mathbb{E}[R_{\ell,2} \mid \neg \mathcal{B}'_\ell] \Pr[\neg \mathcal{B}'_\ell] + \mathbb{E}[R_{\ell,2} \mid \mathcal{B}'_\ell] \Pr[\mathcal{B}'_\ell] \\ &\leq M \sum_{\ell'=1}^{\ell} \theta_{\ell'} N_{\ell'} + C_{\max} \cdot T \cdot T^{-4} \\ &\leq \frac{12 M \ell C_{\max}}{\delta^2 \alpha^2} (\log T + \log M) + C_{\max} T^{-3}. \end{aligned}$$

The first term in the first inequality follows since in the worst case we play each neighboring solution in all sub-phases, but under event $\neg \mathcal{B}'_\ell$ we never play a solution with cost more than $\theta_{\ell'}$ in each sub-phase ℓ'. The second term follows since, under $\mathcal{B}'_\ell$, we can't play a solution of cost more than $C_{\max}$ for more than T steps and $\Pr[\mathcal{B}'_\ell] \leq T^{-4}$ by Lemma 3.8.

Regret Incurred after Phase L: Let $R_{>L}$ denote the regret incurred after phase L. We need to consider both the case where Algorithm 2 terminated with a solution x_{Last} and when it did not. Denote the former event by $\mathcal{L}$. Then we have

$$\mathbb{E}[R_{>L}] = \mathbb{E}[R_{>L} \mid \mathcal{L}] \Pr[\mathcal{L}] + \mathbb{E}[R_{>L} \mid \neg \mathcal{L}] \Pr[\neg \mathcal{L}].$$

We now focus on each conditional expectation and show that the regret is small in each.

Consider the first case where we condition on $\mathcal{L}$, and so we have that Algorithm 2 terminated with some solution x_{Last} which is used for all remaining rounds. This happens when Algorithm 3 is run in phase L and returns the solution x_L. We want to show that it is unlikely for x_L to satisfy $\text{Cost}(x_L) > \gamma\text{OPT}$. Suppose $\text{Cost}(x_L) > \gamma\text{OPT}$, then by Definition 2.2, we have that $\mathcal{N}(x_\ell)$ contains a solution x' with $\text{Cost}(x') \leq \beta \, \text{Cost}(x_L) = \alpha^2 \, \text{Cost}(x_L)$. This means that event $\mathcal{F}_{\ell,2}$ has occurred, and by Lemma 3.9 the probability of Algorithm 3 yielding x_L as output is at most $(\ell+1)T^{-4} \leq T^{-3}$. If, instead, $\text{Cost}(x_L) \leq \gamma\text{OPT}$, we incur zero γ-regret. Together, $\mathbb{E}[R_{>L} \mid \mathcal{L}] \leq C_{\max} T \cdot T^{-3}$.

Now suppose $\mathcal{L}$ does not occur. In this case, period T was reached before finding an approximate local optimum, midway through executing an incomplete phase $L+1$. We can use similar techniques to bound $R_{>L}$ for this incomplete phase as we for R_ℓ above, in fact, we can simply consider $R_{>L}$ to be R_{L+1}. It follows from the same arguments as before that

$$\mathbb{E}[R_{>L} \mid \neg\mathcal{L}] = \mathbb{E}[R_{L+1}] \leq \frac{12(M+1)(L+1)C_{\max}}{\delta^2\alpha^2}(\log T + \log M) + 2C_{\max}T^{-1}.$$

Putting everything together,

$$\mathbb{E}[R_{>L}] \leq \frac{12(M+1)(L+1)C_{\max}}{\delta^2\alpha^2}(\log T + \log M) + 3C_{\max}T^{-1}.$$

Combining the Bounds: Summing all the terms together, we have:

$$\begin{aligned}
\text{Regret}_\gamma(\mathcal{A}, \mathcal{D}, T) &\leq \sum_{\ell=1}^{L} \left(\mathbb{E}[R_{\ell,1}] + \mathbb{E}[R_{\ell,2}]\right) + \mathbb{E}[R_{>L}] \\
&\leq O\left(\frac{MC_{\max}}{\delta^2\alpha^2}(\log T + \log M)\right)\sum_{\ell=1}^{L}\ell + O\left(\frac{MC_{\max}L}{\delta^2\alpha^2}(\log T + \log M)\right) \\
&\leq O\left(\frac{MC_{\max}L^2}{\delta^2\alpha^2}(\log T + \log M)\right) \\
&\leq O\left(\frac{MC_{\max}\log^2 T}{\delta^2\alpha^2\log^2\frac{1}{\alpha}}(\log T + \log M)\right),
\end{aligned}$$

where in the last step we used the bound on L from Lemma 3.11. This establishes Theorem 3.1.

4 Applications of Our Framework

4.1 Stochastic Completion Time Scheduling on a Single Machine

We consider the problem of scheduling stochastic jobs to minimize total completion time on a single machine. Let $\mathcal{D}$ be a distribution over job sizes which is unknown to the scheduler. There are n jobs with stochastic sizes $P = (P_1, P_2, \ldots, P_n) \sim \mathcal{D}$. Let $\mu_j := \mathbb{E}[P_j]$ denote the expected size of job j. We only require independence across time periods; a pair of jobs j and j' may have correlated size distributions. We also make a standard assumption that the job size distribution is bounded and normalized to be in $[0, 1]$.

A schedule is given by a permutation $\pi : [n] \to [n]$ specifying that job j is scheduled in position $\pi(j)$. Let Π_n be the permutations on $[n]$, the feasible schedules. Given a schedule π, let $C_j(\pi, P) := P_j + \sum_{j':\pi(j')<\pi(j)} P_{j'}$ be the completion time of job j under schedule π and sizes P. After reordering the summation, the cost of a schedule π is

$$\text{Cost}(\pi, P) = \sum_{j=1}^{n} C_j(\pi, P) = \sum_{j=1}^{n}(n - \pi(j) + 1)P_j. \tag{3}$$

Let OPT $:= \min_\pi \mathbb{E}_{P\sim\mathcal{D}}[\mathrm{Cost}(\pi, P)]$. We focus on learning from limited feedback using fixed, non-preemptive schedules. In the single machine case full-information case, it is well known that the optimal schedule is given by applying Smith's rule to the expected job sizes: π^* orders the jobs in non-decreasing order of μ_j, or shortest expected processing time first [43, 45].

When learning to schedule with bandit feedback, the scheduler gains information by interacting with unknown $\mathcal{D}$ through the following process which occurs in each of T discrete time periods. In period $t \in [T]$, the scheduler commits to a schedule π^t before job sizes are realized according to an independent sample $P^t \sim \mathcal{D}$. Then the scheduler observes $\mathrm{Cost}(\pi^t, P^t)$, the realized total completion time of schedule π^t, but does not observe the realized job sizes P^t.

For a schedule π, define $\mathrm{error}(\pi) := \mathbb{E}[\mathrm{Cost}(\pi, P)] - \mathrm{OPT}$ to measure the expected regret we incur by using π for one step. Recall that in this case an optimal schedule π^* must schedule j after j' if and only if $\mu_j > \mu_{j'}$. We say that π *inverts* a pair of jobs j, j' with $\mu_j > \mu_{j'}$ if instead it has $\pi(j) < \pi(j')$. The following proposition shows that we can write $\mathrm{error}(\pi)$ in terms of inversions in π from an optimal schedule.

Proposition 4.1 (Lindermayr and Megow [37]). *For any schedule π,*

$$\mathrm{error}(\pi) = \sum_{j,j':\mu_j>\mu_{j'}} (\mu_j - \mu_{j'})\, \mathbf{1}\{\pi(j) < \pi(j')\}.$$

Let $\mathcal{N}(\pi) = \{\pi' \in \Pi_n \mid \exists i \neq j, \pi(i) = \pi'(j), \pi(j) = \pi'(i), \text{ and } \pi(k) = \pi'(k) \text{ for all } k \notin [n] \setminus \{i, j\}\}$ be the the neighborhood of schedules that consists of swapping a single pair of jobs. We show this admits (β, γ)-improving moves, with $\gamma = 1 + \epsilon$ for any $\epsilon > 0$ and $\beta = 1 - \epsilon/n^2$.

Lemma 4.2. *Let π be any schedule with $\mathrm{Cost}(\pi) > (1 + \epsilon)\mathrm{OPT}$ with $\epsilon \in (0, 1)$, then there exists a schedule π' obtainable from π by swapping a single pair of jobs such that*

$$\mathrm{Cost}(\pi') \le \left(1 - \frac{\epsilon}{n^2}\right) \mathrm{Cost}(\pi).$$

Proof. First we lower bound $\mathrm{Cost}(\pi) - \mathrm{OPT}$ by $\epsilon \cdot \mathrm{Cost}(\pi)/2$ since

$$\mathrm{Cost}(\pi) - \mathrm{OPT} \ge \mathrm{Cost}(\pi) - \frac{\mathrm{Cost}(\pi)}{1+\epsilon} = \frac{\epsilon \cdot \mathrm{Cost}(\pi)}{1+\epsilon} \ge \frac{\epsilon \cdot \mathrm{Cost}(\pi)}{2}.$$

Next, we upper bound $\mathrm{Cost}(\pi) - \mathrm{OPT} = \mathrm{error}(\pi)$ as

$$\mathrm{error}(\pi) = \sum_{j,j':\mu_j>\mu_{j'}} (\mu_j - \mu_{j'})1\{\pi(j) < \pi(j')\} \le \binom{n}{2} \cdot \max_{j,j':\mu_j>\mu_{j'}} (\mu_j - \mu_{j'})1\{\pi(j) < \pi(j')\}$$

where we use the fact that there are at most $\binom{n}{2}$ terms in the sum. Combining these bounds, we conclude that there exists a pair j, j' with $\mu_j > \mu'_j$ and $\pi(j) < \pi(j')$ such that

$$\mu_j - \mu_{j'} \ge \frac{\epsilon \cdot \mathrm{Cost}(\pi)}{2\binom{n}{2}} \ge \frac{\epsilon \cdot \mathrm{Cost}(\pi)}{n^2}.$$

For the new schedule π' created by swapping the positions of j and j' in π, we observe that

$$\mathrm{Cost}(\pi) - \mathrm{Cost}(\pi') = (\pi(j') - \pi(j))(\mu_j - \mu_{j'}) \ge (\mu_j - \mu_{j'}) \ge \frac{\epsilon \cdot \mathrm{Cost}(\pi)}{n^2},$$

which uses the fact that $\pi(j') - \pi(j) \ge 1$ by construction. The lemma follows after rearranging. □

Corollary 4.3. *Consider the problem of scheduling to minimize total completion time with stochastic job sizes drawn from distribution $\mathcal{D}$. For any $\epsilon > 0$ there is an algorithm $\mathcal{A}$ (Algorithm 2) with*

$$\mathrm{Regret}_{1+\epsilon}(\mathcal{A}, \mathcal{D}, T) = O\left(\frac{n^{12} \log^3 T}{\epsilon^4}\right).$$

Proof. We apply Theorem 3.1 and simplify with problem-specific parameters. For the swap neighborhood, the maximum neighborhood size is $M \leq n^2$. Job sizes are at most 1, so $C_{max} \leq n^2$. We can bound

$$\log(1/\alpha) = -\frac{1}{2}\log(1-\epsilon/n^2) \geq \frac{1}{2}\frac{\epsilon}{n^2}$$

since $\log(1-x) \leq -x$ for $x > -1$. We can also bound

$$\delta^2\beta = \frac{(1-\epsilon/n^2)(\epsilon/n^2)^2}{(1+\sqrt{1-\epsilon/n^2})^4} \geq 16(1-\epsilon/n^2)(\epsilon/n^2)^2 \geq 8\epsilon^2/n^4$$

since $1-\epsilon/n^2 < 1$, implying $(1+\sqrt{1-\epsilon/n^2})^4 \leq 16$. The result follows after substitution and observing $\log n \leq \log T$. □

4.2 Minimum Cost Base of a Matroid

We now apply our framework to the online stochastic bandit setting for finding a minimum cost base in a matroid. Let $M = (\mathcal{S}, \mathcal{I})$ be a matroid on ground set $\mathcal{S}$ of cardinality $|\mathcal{S}| = n$ with family of independent sets $\mathcal{I}$. Let $\mathcal{B}$ be its family of bases, or maximal independent sets, so that each that $B \in \mathcal{B}$ satisfies $B \in \mathcal{I}$ and there is no $s \in S \setminus B$ such $B \cup s \in \mathcal{I}$. The *rank* of set S, denoted $r(S)$, is the cardinality of the largest independent subset of S. The rank of matroid M is the cardinality of a base, so $r(M) = |B|$.

In the minimum cost base problem, each element $s \in \mathcal{S}$ is associated a cost and the objective is to find a base of minimum total cost. In the online stochastic bandit setting, the learner acts over a sequence of T time periods, selecting base B^t in period $t = 1, \ldots, T$. In each period t, stochastic costs $Z^t = \{Z^t_s\}_{s \in \mathcal{S}} \sim \mathcal{D}$ realize for each element $s \in \mathcal{S}$, where $\mathcal{D}$ is a joint distribution over edge costs. In time step t, given underlying realizations z^t, the learner observes only the cost of base B^t, which is $\text{Cost}(B^t, z^t) = \sum_{s \in B^t} z^t_s$. Let $\mu_e = \mathbb{E}_{Z \sim \mathcal{D}}[Z_e]$ denote the expected cost of edge e, and let $B^* = \arg\min_{B' \in \mathcal{B}} \mathbb{E}_{Z \sim \mathcal{D}}[\text{Cost}(B', Z)]$ be the base with lowest expected cost.

We now define the local search neighborhood which will allow us achieve low regret online. A *circuit* C is a minimally dependent set: $C \notin \mathcal{I}$ but every subset of C is independent.

Definition 4.4. Given a base B of matroid M and $s \in \mathcal{S} \setminus B$, there is a unique circuit $C(s, B)$ such that $x \in C(s, B) \subseteq B \cup s$. Furthermore, for each $t \in C(s, B)$, $(B \cup s) \setminus t$ is also a base.

Definition 4.5. Given matroid M with bases $\mathcal{B}$, let $\mathcal{N} : \mathcal{B} \to 2^{\mathcal{B}}$ be the circuit swap neighborhood map. More formally, for each $B \in \mathcal{B}$, let

$$\mathcal{N}(B) = \{B' \in \mathcal{B} \mid B' = (B \cup s) \setminus t \text{ for some } s \in \mathcal{S} \setminus B, t \in C(s, B)\},$$

so $\mathcal{N}(B)$ is the set of bases reachable from B by adding an element $s \notin B$ to B, followed by removing an element t from the resulting circuit $C(s, B)$.

We will make use of the following well-known property of matroids.

Lemma 4.6 (Frank [24]). *If $B_1, B_2 \in \mathcal{B}$, then there exists a bijection $f : (B_1 \setminus B_2) \to (B_2 \setminus B_1)$ such that $B_1 \setminus \{x\} \cup f(x) \in \mathcal{B}$ for every $x \in B_1 \setminus B_2$.*

We can now show that the circuit swap neighborhood admits (β, γ)-improving moves.

Lemma 4.7. *Given matroid M, the circuit swap neighborhood admits (β, γ)-improving moves with $\gamma = 1 + \epsilon$ for any $\epsilon > 0$ and $\beta = 1 - \epsilon/(2r(M))$.*

Proof. Consider $B \in \mathcal{B}$ satisfying $\text{Cost}(B) > (1+\epsilon)\,\text{Cost}(B^*)$ for $\epsilon \in (0, 1)$.It follows that ,

$$\Delta = \text{Cost}(B) - \text{Cost}(B^*) \geq \text{Cost}(B) - \frac{\text{Cost}(B)}{1+\epsilon} = \frac{\epsilon \cdot \text{Cost}(B)}{1+\epsilon} \geq \frac{\epsilon \cdot \text{Cost}(B)}{2}.$$

We want to show there exists $B' \in \mathcal{N}(B)$ satisfying $\mathrm{Cost}(B') \le \beta\,\mathrm{Cost}(B)$. It is easy to see that

$$\Delta = \mathrm{Cost}(B) - \mathrm{Cost}(B^*) = \sum_{e \in B \setminus B^*} \mu_e - \sum_{e \in B^* \setminus B} \mu_e.$$

From Lemma 4.6, construct a bijection $f : (B \setminus B^*) \to (B^* \setminus B)$ such that $B \setminus \{s\} \cup f(s)$ is a base for every $s \in B \setminus B^*$. If we start from B, then delete s and add $f(s)$ from B for all $s \in B \setminus B^*$, we transform from base B to B^*. Since there are at most $|B| = r(M)$ items in $B \setminus B^*$, there must exist $s' \in B \setminus B^*$, for which $B \setminus s' \cup f(s')$ improves the cost by at least $\Delta/\rho(M)$. Let $T' = B \setminus s' \cup f(s') \in \mathcal{N}(B)$. Then

$$\mathrm{Cost}(B) - \mathrm{Cost}(B') \ge \frac{\Delta}{\rho(M)} \ge \frac{\epsilon \cdot \mathrm{Cost}(B)}{2r(M)}.$$

□

Corollary 4.8. *Consider the problem of finding a minimum cost base in a matroid $M = (\mathcal{S}, \mathcal{I})$ with $|\mathcal{S}| = n$ and rank $r(M) = r$ when element costs are stochastically drawn from $\mathcal{D}$. For any $\epsilon > 0$, there is an algorithm $\mathcal{A}$ (Algorithm 2) with*

$$\mathrm{Regret}_{1+\epsilon}(\mathcal{A}, \mathcal{D}, T) = O\left(\frac{nr^6 \log^2 T}{\epsilon^4}(\log T + \log n)\right) = O\left(\frac{nr^6 \log^3 T}{\epsilon^4}\right).$$

Proof. We apply Lemma 4.7 and Theorem 3.1 with $(\beta, \gamma) = (1 - \epsilon/2r, 1 + \epsilon)$ for $\epsilon > 0$. Thus we just need to bound M, $C_{\max}$, $\frac{1}{\alpha^2}$, $\frac{1}{\delta^2}$, and $\log \frac{1}{\alpha}$. It is easy to see that $M \le nr$ and that $C_{\max} \le r$. Next, we can see that $\frac{1}{\alpha^2} = O(1)$ for $\epsilon \le r$ since $\alpha^2 = 1 - \epsilon/2r \ge 1/2$. For $\frac{1}{\delta^2}$, we have:

$$\delta = \frac{1-\alpha}{1+\alpha} = \frac{1-\alpha^2}{(1+\alpha)^2} = \frac{\epsilon/2r}{1 + 2\alpha + \alpha^2} \ge \frac{\epsilon}{8r}.$$

Finally, for $\log(1/\alpha)$ we have:

$$\log \frac{1}{\alpha} = \log\left(\frac{1}{1 - \epsilon/2r}\right)^{1/2} = -\frac{1}{2}\log(1 - \epsilon/2r) \ge -\frac{1}{2}\log\exp(-\epsilon/2r) = \frac{\epsilon}{4r}.$$

Combining in Theorem 3.1 yields the bound. □

4.3 Uncertain k-Median Clustering

Our final application concerns k-median clustering where the set of points to be clustered are drawn at random from an unknown distribution. The following setup was studied by Cormode and McGregor [16] and Guha and Munagala [27] for a several of k-clustering objectives in the offline setting. An instance consists of n points arriving from a metric space $(\mathcal{M}, d)$ of diameter 1. The location of the i'th point is uncertain and the objective is to choose k cluster centers which minimize the total expected distance between each arriving point and its closest cluster center. We let $Z_i \in \mathcal{M}$ denote the random realization of the i'th point

For a fixed location $z \in \mathcal{M}$ and a fixed set C of cluster centers, let $d(z, C) = \min_{c \in C} d(z, c)$ denote the distance of z to its nearest cluster center in C. Then the cost of a solution C on points $Z = (Z_1, Z_2, \ldots, Z_n)$ is

$$\mathrm{Cost}(C, Z) = \sum_{i \in [n], z \in \mathcal{Z}} \mathbf{1}\{Z_i = z\} \cdot d(z, C).$$

Thus, the expected cost of solution C is given by

$$\mathbb{E}[\mathrm{Cost}(C, Z)] = \sum_{i \in [n], z \in \mathcal{Z}} \Pr[Z_i = z] \cdot d(z, C),$$

and OPT denotes the expected cost of a set of cluster centers which minimizes this expected cost.

In the bandit setting, we do not observe the realizations of the point locations. In each round t a set of cluster center locations C_t is submitted, after which the learner only observes $\mathrm{Cost}(C_t, Z)$.

Let $\mathcal{N}(C) = \{C' \in \mathcal{X} \mid C \cup \{b\} \setminus \{a\}, a \in C, b \in \mathcal{X} \setminus C\}$ be the swap-neighborhood of cluster centers C. Arya et. al. [3] propose a local search algorithm for the k-medians problem which starts from an arbitrary set of k cluster centers C and repeatedly moves to neighboring $C' \in \mathcal{N}(C)$ satisfying $\mathrm{Cost}(C') \leq (1-\epsilon)\,\mathrm{Cost}(C)$ until no local improvement is found. Cormode and McGregor [16] shows that this is equally effective for the *uncertain* k-medians problem.

Lemma 4.9 (Arya et al. [3] and Cormode and McGregor [16]). *For the uncertain k-median clustering problem, local search with ϵ-improvements terminates in $C \in \mathcal{X}$ satisfying $\mathrm{Cost}(C) \leq (5/(1-n\epsilon))\,\mathrm{Cost}(C^*)$.*

We can leverage this to show that uncertain k-median clustering permits (β, γ)-improving moves.

Lemma 4.10. *The Minimum Cost K-Median problem with n input points admits (β, γ)-improving moves where $\gamma = 5/(1-\frac{1}{n})$ and $\beta = (1 - 1/n^2)$.*

Proof. We prove this lemma by contradiction. Suppose that the local search algorithm does not admit (β, γ)-improving moves where $\gamma = 5/(1-\frac{1}{n})$ and $\beta = (1-1/n^2)$. This implies that there exists $C \in \mathcal{X}$ with $\mathrm{Cost}(C) > \gamma\,\mathrm{Cost}(C^*)$ and that $\mathrm{Cost}(C') > \beta\,\mathrm{Cost}(C)$ for all $C' \in \mathcal{N}(C)$. It follows that the local search algorithm of [16] may terminate with C as a local optimum. However, setting $\epsilon = 1/n^2$ in Lemma 4.9 gives a $5/(1-1/n)$-factor approximation algorithm. This implies that $\mathrm{Cost}(C) \leq \frac{5}{1-\frac{1}{n}}\,\mathrm{Cost}(C^*)$, a contradiction. □

Corollary 4.11. *Consider the uncertain k-median clustering with candidate set of m potential cluster centers where the locations of n points in a metric space of diameter 1 are drawn from distribution $\mathcal{D}$. There exists algorithm $\mathcal{A}$ (Algorithm 2) achieving*

$$\mathrm{Regret}_{5n/(n-1)}(\mathcal{A}, \mathcal{D}, T) = O\left(n^9 m^2 \log^3 T\right),$$

when $m^2 \leq T$.

Proof. We apply Theorem 3.1 and simplify with problem-specific parameters. For the swap neighborhood, the maximum neighborhood size is $M \leq |\mathcal{X}|^2 = m^2$. The metric space has diameter 1, so $C_{max} \leq n$. As in the proof of Corollary 4.3, We can bound $\log(1/\alpha) \geq 1/2n^2$ and $\delta^2\beta \geq 8/n^4$. The result follows after substitution and observing $\log n \leq \log T$. □

5 Discussion

We conclude by discussing a couple of relevant issues not raised elsewhere.

For ease of exposition we assumed the algorithm knows T, and that there is an upper bound $C_{\max}$ on costs. It is straightforward but tedious to modify the algorithms and analysis to not require explicit knowledge of T. Similarly, we can replace the assumption of bounded costs with suitable distributional assumptions about the cost. For example, we could require a sub-gaussian like condition on the tails of the distribution or even just finite variance. In the latter case, we could make use of tools for handling heavy-tailed distributions in the bandit setting [10].

It remains to computationally demonstrate the benefits of our algorithm. All three applications are instances of linear bandits[9, 18, 36], so a natural starting point may be to compare the regret of our algorithm with that of a linear bandit style algorithm which, in each step, uses the bandit feedback to estimate problem parameters (eg. mean job sizes for completion time scheduling), then computes the solution to an offline instance of the problem with those parameters to submit in the next time step. For the scheduling and matroid applications the offline instances can be solved efficiently, for clustering we have to resort to an approximate solution due to NP-hardness.

An exciting direction for future work is exploring further applications for our framework where local search is effective offline. Some applications may require generalizing our current approach, for example, the local search guarantees for set cover [28] rely on finding neighboring solution which improves a specialized potential function, and not the objective function directly. It is an open question whether a version of (β, γ)-improving moves defined on such potential functions can provide guarantees in the bandit online setting.

References

[1] Arpit Agarwal, Rohan Ghuge, and Viswanath Nagarajan. 2024. Semi-Bandit Learning for Monotone Stochastic Optimization. In *65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024, Chicago, IL, USA, October 27-30, 2024*. IEEE, 1260–1274. doi:10.1109/FOCS61266.2024.00083

[2] Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. 2013. Local-Search based Approximation Algorithms for Mobile Facility Location Problems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, Sanjeev Khanna (Ed.). SIAM, 1607–1621. doi:10.1137/1.9781611973105.115

[3] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. 2001. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*. 21–29.

[4] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. 2014. Regret in Online Combinatorial Optimization. *Math. Oper. Res.* 39, 1 (2014), 31–45. doi:10.1287/MOOR.2013.0598

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Mach. Learn.* 47, 2-3 (2002), 235–256. doi:10.1023/A:1013689704352

[6] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2002. The Nonstochastic Multiarmed Bandit Problem. *SIAM J. Comput.* 32, 1 (2002), 48–77. doi:10.1137/S0097539701398375

[7] Thierry Benoist, Frédéric Gardi, Antoine Jeanjean, and Bertrand Estellon. 2011. Randomized local search for real-life inventory routing. *Transportation Science* 45, 3 (2011), 381–398.

[8] Paul Bogdan, Thomas Sauerwald, Alexandre Stauffer, and He Sun. 2013. Balls into Bins via Local Search. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, Sanjeev Khanna (Ed.). SIAM, 16–34. doi:10.1137/1.9781611973105.2

[9] Sébastien Bubeck, Nicolò Cesa-Bianchi, and Sham M. Kakade. 2012. Towards Minimax Policies for Online Linear Optimization with Bandit Feedback. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland (JMLR Proceedings, Vol. 23)*, Shie Mannor, Nathan Srebro, and Robert C. Williamson (Eds.). JMLR.org, 41.1–41.14. http://proceedings.mlr.press/v23/bubeck12a/bubeck12a.pdf

[10] Sébastien Bubeck, Nicolò Cesa-Bianchi, and Gábor Lugosi. 2013. Bandits With Heavy Tail. *IEEE Trans. Inf. Theory* 59, 11 (2013), 7711–7717. doi:10.1109/TIT.2013.2277869

[11] Sara Ceschia and Andrea Schaerf. 2013. Local search for a multi-drop multi-container loading problem. *Journal of Heuristics* 19, 2 (2013), 275–294.

[12] Ke Chen. 2008. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, Shang-Hua Teng (Ed.). SIAM, 826–835. http://dl.acm.org/citation.cfm?id=1347082.1347173

[13] Wei Chen, Yajun Wang, and Yang Yuan. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013 (JMLR Workshop and Conference Proceedings, Vol. 28)*. JMLR.org, 151–159. http://proceedings.mlr.press/v28/chen13a.html

[14] Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. 2022. An Improved Local Search Algorithm for k-Median. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, Joseph (Seffi) Naor and Niv Buchbinder (Eds.). SIAM, 1556–1612. doi:10.1137/1.9781611977073.65

[15] Richard Combes, Mohammad Sadegh Talebi, Alexandre Proutière, and Marc Lelarge. 2015. Combinatorial Bandits Revisited. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.). 2116–2124. https://proceedings.neurips.cc/paper/2015/hash/0ce2ffd21fc958d9ef0ee9ba5336e357-Abstract.html

[16] Graham Cormode and Andrew McGregor. 2008. Approximation algorithms for clustering uncertain data. In *Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 191–200.

[17] Thibaut Cuvelier, Richard Combes, and Eric Gourdin. 2021. Statistically Efficient, Polynomial-Time Algorithms for Combinatorial Semi-Bandits. *Proc. ACM Meas. Anal. Comput. Syst.* 5, 1 (2021), 09:1–09:31. doi:10.1145/3447387

[18] Varsha Dani, Sham M Kakade, and Thomas Hayes. 2007. The Price of Bandit Information for Online Optimization. In *Advances in Neural Information Processing Systems*, Vol. 20. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/bf62768ca46b6c3b5bea9515d1a1fc45-Paper.pdf

[19] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press.

[20] Miroslav Dudík, Nika Haghtalab, Haipeng Luo, Robert E. Schapire, Vasilis Syrgkanis, and Jennifer Wortman Vaughan. 2020. Oracle-efficient Online Learning and Auction Design. *J. ACM* 67, 5 (2020), 26:1–26:57. doi:10.1145/3402203

[21] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2002. PAC Bounds for Multi-armed Bandit and Markov Decision Processes. In *Computational Learning Theory, 15th Annual Conference on Computational Learning Theory, COLT 2002, Sydney, Australia, July 8-10, 2002, Proceedings (Lecture Notes in Computer Science, Vol. 2375)*, Jyrki Kivinen and Robert H. Sloan (Eds.). Springer, 255–270. doi:10.1007/3-540-45435-7_18

[22] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. 2006. Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. *J. Mach. Learn. Res.* 7 (2006), 1079–1105. https://jmlr.org/papers/v7/evendar06a.html

[23] Fares Fourati, Christopher John Quinn, Mohamed-Slim Alouini, and Vaneet Aggarwal. 2024. Combinatorial Stochastic-Greedy Bandit. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 12052–12060. doi:10.1609/AAAI.V38I11.29093

[24] András Frank. 2011. *Connections in combinatorial optimization.* Vol. 38. Oxford University Press Oxford.

[25] Shayan Oveis Gharan and Luca Trevisan. 2014. Partitioning into Expanders. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, Chandra Chekuri (Ed.). SIAM, 1256–1266. doi:10.1137/1.9781611973402.93

[26] Margalit Glasgow and Alexander Rakhlin. 2023. Tight Bounds for γ-Regret via the Decision-Estimation Coefficient. arXiv:2303.03327 [cs.LG]

[27] Sudipto Guha and Kamesh Munagala. 2009. Exceeding expectations and clustering uncertain data. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, Jan Paredaens and Jianwen Su (Eds.). ACM, 269–278. doi:10.1145/1559795.1559836

[28] Anupam Gupta, Euiwoong Lee, and Jason Li. 2023. A Local Search-Based Approach for Set Covering. In *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*, Telikepalli Kavitha and Kurt Mehlhorn (Eds.). SIAM, 1–11. doi:10.1137/1.9781611977585.CH1

[29] Elad Hazan, Amit Agarwal, and Satyen Kale. 2007. Logarithmic regret algorithms for online convex optimization. *Mach. Learn.* 69, 2-3 (2007), 169–192. doi:10.1007/S10994-007-5016-8

[30] Kazuo Iwama, Shuichi Miyazaki, and Naoya Yamauchi. 2007. A 1.875: approximation algorithm for the stable marriage problem. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, Nikhil Bansal, Kirk Pruhs, and Clifford Stein (Eds.). SIAM, 288–297. http://dl.acm.org/citation.cfm?id=1283383.1283414

[31] Sham M. Kakade, Adam Tauman Kalai, and Katrina Ligett. 2009. Playing Games with Approximation Algorithms. *SIAM J. Comput.* 39, 3 (2009), 1088–1106. doi:10.1137/070701704

[32] Adam Tauman Kalai and Santosh S. Vempala. 2005. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.* 71, 3 (2005), 291–307. doi:10.1016/J.JCSS.2004.10.016

[33] Felipe Lagos and Jordi Pereira. 2024. Multi-armed bandit-based hyper-heuristics for combinatorial optimization problems. *Eur. J. Oper. Res.* 312, 1 (2024), 70–91. doi:10.1016/J.EJOR.2023.06.016

[34] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* 6, 1 (March 1985), 4–22. doi:10.1016/0196-8858(85)90002-8

[35] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms.* Cambridge University Press.

[36] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti (Eds.). ACM, 661–670. doi:10.1145/1772690.1772758

[37] Alexander Lindermayr and Nicole Megow. 2022. Permutation Predictions for Non-Clairvoyant Scheduling. In *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11 - 14, 2022*, Kunal Agrawal and I-Ting Angelina Lee (Eds.). ACM, 357–368. doi:10.1145/3490148.3538579

[38] Ole Jakob Mengshoel, Tong Yu, and Ming Zeng. 2020. Stochastic Local Search and Machine Learning: From Theory to Applications and Vice Versa. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September*

2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020) (Frontiers in Artificial Intelligence and Applications, Vol. 325), Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.). IOS Press, 2919–2920. doi:10.3233/FAIA200453

[39] Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. 2004. Local search for shift design. *European journal of operational research* 153, 1 (2004), 51–64.

[40] Gergely Neu and Gábor Bartók. 2016. Importance Weighting Without Importance Weights: An Efficient Algorithm for Combinatorial Semi-Bandits. *J. Mach. Learn. Res.* 17 (2016), 154:1–154:21. https://jmlr.org/papers/v17/15-091.html

[41] Rad Niazadeh, Negin Golrezaei, Joshua R. Wang, Fransisca Susan, and Ashwinkumar Badanidiyuru. 2023. Online Learning via Offline Greedy Algorithms: Applications in Market Design and Optimization. *Manag. Sci.* 69, 7 (2023), 3797–3817. doi:10.1287/MNSC.2022.4558

[42] Pierre Perrault, Vianney Perchet, and Michal Valko. 2019. Exploiting structure of uncertainty for efficient matroid semi-bandits. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 5123–5132. http://proceedings.mlr.press/v97/perrault19a.html

[43] Michael H. Rothkopf. 1966. Scheduling with Random Service Times. *Management Science* 12, 9 (1966), 707–713. http://www.jstor.org/stable/2627947

[44] Shai Shalev-Shwartz and Sham M. Kakade. 2008. Mind the Duality Gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou (Eds.). Curran Associates, Inc., 1457–1464. https://proceedings.neurips.cc/paper/2008/hash/bd686fd640be98efaae0091fa301e613-Abstract.html

[45] Wayne E. Smith. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3, 1-2 (1956), 59–66. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800030106 doi:10.1002/nav.3800030106

[46] Alberto Vera, Siddhartha Banerjee, and Itai Gurvich. 2021. Online Allocation and Pricing: Constant Regret via Bellman Inequalities. *Oper. Res.* 69, 3 (2021), 821–840. doi:10.1287/OPRE.2020.2061

[47] Zheng Wen, Branislav Kveton, and Azin Ashkan. 2015. Efficient Learning in Large-Scale Combinatorial Semi-Bandits. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 1113–1122. http://proceedings.mlr.press/v37/wen15.html

[48] Jianyu Xu and Yu-Xiang Wang. 2021. Logarithmic Regret in Feature-based Dynamic Pricing. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 13898–13910. https://proceedings.neurips.cc/paper/2021/hash/742141ceda6b8f6786609d31c8ef129f-Abstract.html

[49] Shuo Yang, Tongzheng Ren, Sanjay Shakkottai, Eric Price, Inderjit S. Dhillon, and Sujay Sanghavi. 2022. Linear Bandit Algorithms with Sublinear Time Complexity. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*. PMLR, 25241–25260. https://proceedings.mlr.press/v162/yang22m.html

[50] Tong Yu, Branislav Kveton, and Ole J. Mengshoel. 2017. Thompson Sampling for Optimizing Stochastic Local Search. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10534)*, Michelangelo Ceci, Jaakko Hollmén, Ljupco Todorovski, Celine Vens, and Saso Dzeroski (Eds.). Springer, 493–510. doi:10.1007/978-3-319-71249-9_30

[51] Jiongzhi Zheng, Kun He, Jianrong Zhou, Yan Jin, Chu-Min Li, and Felip Manyà. 2022. BandMaxSAT: A Local Search MaxSAT Solver with Multi-armed Bandit. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 1901–1907. doi:10.24963/IJCAI.2022/264