



Data-Driven Solution Portfolios

Marina Drygala  

EPFL, Lausanne, Switzerland

Silvio Lattanzi  



Google Research, Barcelona, Spain

Andreas Maggiori  

Columbia University, New York, NY, USA

Miltiadis Stouras¹  

EPFL, Lausanne, Switzerland

Ola Svensson¹  

EPFL, Lausanne, Switzerland

Sergei Vassilvitskii  

Google Research, New York, NY, USA

Abstract

In this paper, we consider a new problem of portfolio optimization using stochastic information. In a setting where there is some uncertainty, we ask how to best select k potential solutions, with the goal of optimizing the value of the best solution. More formally, given a combinatorial problem Π , a set of value functions \mathcal{V} over the solutions of Π , and a distribution \mathcal{D} over \mathcal{V} , our goal is to select k solutions of Π that maximize or minimize the expected value of the *best* of those solutions. For a simple example, consider the classic knapsack problem: given a universe of elements each with unit weight and a positive value, the task is to select r elements maximizing the total value. Now suppose that each element's weight comes from a (known) distribution. How should we select k different solutions so that one of them is likely to yield a high value?

In this work, we tackle this basic problem, and generalize it to the setting where the underlying set system forms a matroid. On the technical side, it is clear that the candidate solutions we select must be diverse and anti-correlated; however, it is not clear how to do so efficiently. Our main result is a polynomial-time algorithm that constructs a portfolio within a constant factor of the optimal.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases solution portfolios, data-driven algorithm design, matroids

Digital Object Identifier 10.4230/LIPIcs.ITCS.2025.46

Related Version *Full Version*: <https://arxiv.org/abs/2412.00717>

1 Introduction

Worst-case analysis has long been the prevailing standard for assessing an algorithm's performance. However, this approach often falls short in capturing the real-world performance of algorithms, as instances encountered in practice often differ significantly from those that define worst-case scenarios. To address this discrepancy, the field of *Data-Driven Algorithm Design* has sought to create algorithms that utilize past data about a problem, either implicitly or explicitly, and provably achieve superior performance on the typical instances that arise in practical applications. For surveys of this area see [12, 1].

¹ Supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number MB22.00054.



Building on this line of research, we study the portfolio optimization problem as a simple framework for speeding up algorithms using historical data; a method with potential applications across various combinatorial problems. Specifically, given a combinatorial problem Π , a solution set \mathcal{I} , and a distribution \mathcal{D} over value functions (which map solutions to values, with each function representing a different scenario), our objective is to compute a portfolio of k solutions that maximizes (or minimizes) the expected value (or cost) of the best among the k solutions. Starting from a different perspective, Kleinberg, Papadimitriou and Raghavan [9] have defined a very similar family of optimization problems, called “Segmentation Problems”. The PORTFOLIO OPTIMIZATION problem can be seen as a stochastic version of the aforementioned family of problems; actually the two definitions are effectively equivalent when the distribution over value functions has a polynomial-sized support (for more details see Related Work, Section 1.2).

The PORTFOLIO OPTIMIZATION problem captures many natural questions. The problem mentioned in the abstract—how to select k bundles of r items each so that the expected value of the best bundle is maximized—is only one of them. Another practical application is finding the shortest path between two points (e.g., home and work) in a city, under varying daily traffic conditions. Here, the solution set consists of all possible source-destination paths and the value functions assign weights to each path based on specific traffic scenarios. The distribution over value functions models the stochastic nature of traffic. Rather than rerunning a shortest-path algorithm for each new instance, one could leverage the statistical knowledge about traffic patterns to precompute a few paths that cover different likely scenarios (e.g., peak morning traffic, light nighttime traffic, etc.). This approach, then, allows us to quickly evaluate the precomputed paths under new traffic conditions and select the best option without the need to resolve the problem from scratch each time.

Interestingly, this stochastic formulation also captures problems outside the area of speeding up algorithms. In particular, our objective captures any stochastic problem where one needs to select a set of dependent objects with the goal of maximizing the expectation of their maximum. A practical example of this is sports betting pools, which was recently explored in [6]. The authors examine the scenario of participating in a betting pool for a basketball tournament, where individuals can pay a fee to submit a prediction for the outcome of all tournament matches, with the potential to win a substantial monetary prize if their predictions are accurate. Given a budget constraint on the number of entries that a person can submit, the technical problem essentially reduces to computing a portfolio of entries that maximizes the probability that one of the selected entries succeeds.

From a technical standpoint, in the maximization variant of our problem, the value of a portfolio is a submodular function. Therefore, one can find an approximately optimal portfolio by running the *Greedy* algorithm, with a running time that is polynomial in the size of the solution set \mathcal{I} . However, for most interesting problems, e.g., spanning trees, entries in betting pools, etc., the solution set is given implicitly and is usually exponentially large in the size of the input, making *Greedy* inefficient. This raises the natural question of whether one can compute approximately optimal portfolios, for interesting problems, with a running time that is polynomial in the size of the input.

1.1 Our contributions

The first important contribution of our paper is conceptual: we formulate the PORTFOLIO OPTIMIZATION problem from a new, stochastic, viewpoint and show that it captures many interesting scenarios.

On the technical side, we focus on constructing solution portfolios for the fundamental problem of optimization under matroid constraints. We examine the simplified case where each element of the matroid’s ground set takes a 0-1 value, independently, with some known probability, and the value of a set is the sum of the values of its elements. Despite its simplicity, this case captures many core challenges of constructing solution portfolios and proves to be technically challenging. As we discuss in Section 3, this problem differs fundamentally from classical problems in the area of randomized algorithms, because constructing effective portfolios requires leveraging the “anti-concentration” properties of various solutions. Additionally, employing standard tools from the literature, such as contention resolution schemes, presents challenges due to the nature of our objective function. In particular, the value of a portfolio depends heavily on “abnormal” events, such as significant deviations from the expected value, occurring in one of its k solutions. However, contention resolution schemes do not offer per-instance guarantees but rather only work on expectation. Therefore, we need to take extra care to apply these results while conditioning on those events.

The main technical contribution of our work is to design an algorithm with polynomial running time in the size of the matroid’s ground set which constructs a portfolio that is a $\Theta(1)$ -approximation of the optimal portfolio.

We give a high-level description of our techniques in Section 3 and present a simpler algorithm for the case of uniform matroids in Section 4. In the full version of the paper we show how this algorithm can be extended to work for all matroids.

1.2 Related work

The related work can be broadly categorized into three main directions. The first is the study of Segmentation Problems, initially introduced by Kleinberg, Papadimitriou and Raghavan [9]. This work, inspired by data mining techniques for market segmentation, defines a new class of optimization problems named *Segmentation Problems*. For any combinatorial optimization problem and a set S of different cost vectors for this problem, the corresponding segmentation problem asks to partition the set S into several *segments* and pick a separate solution for each segment, so that the total cost is minimized. The PORTFOLIO OPTIMIZATION problem can be seen as a stochastic variant of the aforementioned family of problems, where instead of a fixed set of cost vectors, one has access to a distribution over cost vectors and aims to optimize the expected value of the constructed portfolio. In fact, the two formulations are effectively equivalent when the distribution over cost vectors has a polynomial-sized support. This new, stochastic, viewpoint can accommodate a wider variety of applications where randomness is inherent in the problem at hand (e.g., sports betting, as introduced earlier). Besides that, it also enables the formulation of elegant and technically challenging theoretical questions, like the case where every element of a groundset takes a value independently of the other elements. Another adaptation of the family of “Segmentation Problems” is due to Gupta, Moondra and Singh [8] who examined a “robust” version of the problem. In this variant, the objective is to identify a small set of solutions that guarantees a good approximation for each one of the cost vectors of interest. This approach was motivated by fairness concerns, aiming to guarantee a good approximation across various cost vectors that may arise due to different fairness constraints.

On the technical side, Kleinberg et al. [9] study the Catalogue Segmentation problem, i.e. constructing portfolios for linear maximization with cardinality constraints. For the case where elements take values in $\{0, 1\}$, they design an algorithm that runs in time $O(n^{k \log k} / \delta)$ and produces a portfolio that is a $(1 - \delta)$ -approximation, under the assumption that on every instance a large fraction of the items have value 1. Our result is a $O(1)$ approximation

with a running time that is polynomial in both n and k , without the need of a density assumption, but for the case where the value of each element is independent of the values of other elements. Furthermore, our results extend to more general settings, accommodating any matroid constraints on the ground set.

The second related line of work is the area of Data-Driven Algorithm Design and specifically speeding up an algorithms' execution by utilizing data. Some of these attempts include speeding up: the *Greedy* algorithm for submodular maximization [2, 15], the Hungarian algorithm for calculating maximum matchings [7], primal-dual algorithms for various graph problems [3], algorithms for flow problems [4, 5] and the Bellman-Ford algorithm [11]. The key distinction of our work lies in defining a new set of problems and providing general techniques for solving them.

A third relevant area of research is stochastic probing and online decision-making (for a detailed survey, see [14]). These problems typically involve a ground set of elements, each with an unknown stochastic weight sampled from a known distribution. The algorithm can probe certain elements – often incurring a cost – in order to reveal their weights. Based on these observations, it then selects a feasible subset and is rewarded with the weight of the selected items. A key distinction between this area and the PORTFOLIO OPTIMIZATION problem is that, in our setting, the solutions are chosen entirely offline, without observing any weights, and remain fixed across all possible realizations.

Our problem is more closely aligned with a non-adaptive strategy for stochastic probing. However, the variants of non-adaptive strategies studied in the literature typically impose only partial restrictions on the algorithm's flexibility – for instance, requiring the algorithm to preselect which elements to probe but still allowing it to form its solution after observing the probed weights. In contrast, in the PORTFOLIO OPTIMIZATION problem, the selection is entirely offline, with no adjustments allowed after weights are realized. For this reasons, the benchmarks of the two problems are also different: non-adaptive strategies are evaluated against the optimal value achievable in hindsight, whereas our benchmark is the best offline strategy with the same constraints as the algorithm.

2 Problem statement and preliminaries

2.1 Portfolio Optimization

In this section we define the PORTFOLIO OPTIMIZATION problem in its general form, for any classical combinatorial problem. In the next section, we define the MATROID PORTFOLIO OPTIMIZATION problem, which is the portfolio problem for a special case of maximization with matroid constraints.

We present the maximization variant of our problem, with the minimization version defined in a similar manner. For any combinatorial problem Π , the PORTFOLIO OPTIMIZATION problem is described by a tuple $\mathcal{J} = (\mathcal{I}, \mathcal{V}, \mathcal{D}, k)$ where \mathcal{I} is the set of feasible solutions of problem Π , \mathcal{V} is a set of value functions from \mathcal{I} to $\mathbb{R}_{\geq 0}$, \mathcal{D} is a distribution over \mathcal{V} and k is a natural number that describes the desired size of the portfolio.

Our goal is to select k solutions from \mathcal{I} so as to maximize the expected value of the *best* of those solutions. That is, for a collection of k sets $\mathcal{S} = \{S_1, \dots, S_k\}$ such that $S_i \in \mathcal{I}, \forall i \in \{1, \dots, k\}$ we define its value as:

$$\text{value}(\mathcal{S}) = \mathbf{E}_{v \sim \mathcal{D}} \left[\max_{S_i \in \mathcal{S}} v(S_i) \right].$$

Formally, we want to solve the following optimization problem.

$$\begin{aligned} & \text{maximize value}(\mathcal{S}) \\ & \text{s.t. } \mathcal{S} = \{S_1, \dots, S_k\} \text{ and } S_i \in \mathcal{I}, \forall i \in [k]. \end{aligned}$$

We remark that the solutions S_1, \dots, S_k are chosen *offline*, without observing the realized value function, but rather only by using our knowledge about the distribution \mathcal{D} .

For any input tuple \mathcal{J} , we denote by $\mathcal{O}(\mathcal{J}) = \{O_1(\mathcal{J}), \dots, O_k(\mathcal{J})\}$ the optimal portfolio, that is the maximizer of the above optimization problem. To ease notation, whenever the input and the algorithm are clear from the context we use $\mathcal{O} = \{O_1, \dots, O_k\}$ and OPT to denote the optimum solution and its value respectively.

For an algorithm ALG we denote by $\mathcal{S}_{\text{ALG}(\mathcal{J})}$ its output on input \mathcal{J} . We say that ALG is a c -approximation if it always outputs a collection of feasible solutions and:

$$\mathbf{E}_{\text{ALG}} [\text{value}(\mathcal{S}_{\text{ALG}(\mathcal{J})})] \geq c \cdot \text{value}(\mathcal{O}(\mathcal{J})), \forall \mathcal{J}$$

where the expectation is taken over the internal randomness of ALG.

In the maximization variant of the problem, the value of a set of solutions, defined as the expected maximum of their values, is a monotone submodular function, as demonstrated by Kleinberg and Raghunathan [10]. Consequently, the celebrated *Greedy* algorithm of [13] is a $(1 - 1/e)$ approximation that runs in polynomial time in the size of the solution set \mathcal{I} . Without imposing any further restrictions to the problem, this is the best approximation ratio one can achieve, as the Max- k -Cover problem can be reduced to the PORTFOLIO OPTIMIZATION problem. The details of the reduction are deferred to the full version of the paper.

Although Greedy achieves the optimal approximation ratio for this problem, its running time is impractical for most interesting applications like max- k -cover, knapsack, shortest paths, spanning trees, etc. In all of the aforementioned applications, the set of feasible solutions is given implicitly and is exponentially large in the size of the input. Therefore, one needs to have a running time that is polynomial in the description of the solution set, rather than its size. This raises the natural question of whether one can compute approximately optimal portfolios, for interesting problems, with a running time that is polynomial in the description of the solution set \mathcal{I} .

In this work, we focus on the fundamental case of optimization over matroid constraints and answer the latter question affirmatively for a natural distribution over value functions. We formulate this problem in the following section.

2.2 Matroid Portfolio Optimization

The MATROID PORTFOLIO OPTIMIZATION problem is a special case of the PORTFOLIO OPTIMIZATION problem, where the set of feasible solutions, \mathcal{I} , is the family of independent sets of an underlying matroid $M = (E, \mathcal{I})$ (see Definition 1). The set \mathcal{V} of value functions is the set of all *additive* set functions that map the elements of E to $\{0, 1\}$. Formally, for any value function $u \in \mathcal{V}$, element $e \in E$ and subset $S \subseteq E$ we have: $u(\{e\}) \in \{0, 1\}$ and $u(S) = \sum_{e' \in S} u(\{e'\})$. Throughout the paper we use the term “active” to denote an element $e \in E$ that takes value one, under a specific value function, and the term “inactive” to describe an element with value zero.

We assume that \mathcal{D} is such that each element $e \in E$ is active with probability p_e , independently of the values of the rest of the elements of E . Formally,

$$\forall S \subseteq E : \Pr_{u \sim \mathcal{D}} \left[\bigwedge_{e \in S} \{u(\{e\}) = 1\} \right] = \prod_{e \in S} p_e$$

To simplify the notation we define an equivalent distribution \mathcal{D}' over subsets of E , that represent the set of active elements, such that

$$\forall e \in E : \Pr_{u \sim \mathcal{D}} [u(\{e\}) = 1] = \Pr_{A \sim \mathcal{D}'} [e \in A].$$

and

$$\forall S \subseteq E : \Pr_{A \sim \mathcal{D}'} \left[\bigwedge_{e \in S} \{e \in A\} \right] = \prod_{e \in S} p_e$$

In other words, instead of sampling a value function that assigns $\{0, 1\}$ values to the elements, we can equivalently sample the set of active elements and, thus, switch to distributions over subsets of E . Also, the value of any set $S \subseteq E$ will now be equal to the random variable $|S \cap A|$, where A is the random set denoting the active elements.

For the remainder of the paper, the input to the MATROID PORTFOLIO OPTIMIZATION problem will be described by a triplet $\mathcal{J} = (M, k, \mathcal{D})$, where $M = (E, \mathcal{I})$ is a matroid over the ground set E , k a natural number describing the desired size of the portfolio and \mathcal{D} is a distribution over subsets of E such that each element $e \in E$ is included in a sample, independently, with probability p_e . For a collection $\mathcal{S} = \{S_1, \dots, S_k\}$ such that $S_i \in \mathcal{I}, \forall i \in \{1, \dots, k\}$, we define its value as

$$\text{value}(\mathcal{S}) = \mathbf{E}_{A \sim \mathcal{D}} \left[\max_{S_i \in \mathcal{S}} |S_i \cap A| \right].$$

2.3 Preliminaries

In this subsection we introduce notation and give some preliminary definitions. First, for any positive integer ℓ we use the shorthand $[\ell]$ to denote the set $\{1, \dots, \ell\}$. In addition, we give the definition of matroids below.

► **Definition 1.** A pair $M = (E, \mathcal{I})$ is called a matroid if E is a finite ground set and \mathcal{I} is a non-empty collection of subsets of E such that

1. If $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$,
2. If $I, J \in \mathcal{I}$ and $|I| < |J|$, then $I + e \in \mathcal{I}$ for some $e \in J \setminus I$.

For a matroid $M = (E, \mathcal{I})$ the elements of \mathcal{I} are called the independent sets of M . The rank function of a matroid M , $r_M : 2^E \rightarrow \mathbb{N}$, maps each set $U \subseteq E$ to the size of the largest independent set contained in U . We use the term “rank of a matroid M ” to denote the rank r of the ground set, i.e. $r = r_M(E)$. A set $B \subseteq E$ is called a base if and only if it’s a maximum size independent set. In addition, the span of a set $S \subseteq E$ is defined as $\text{span}_M(S) = \{e \in E : \text{rank}(S \cup \{e\}) = \text{rank}(S)\}$.

The matroid polytope $\mathcal{P}(M)$, of a matroid M , is a subset of $\mathbb{R}^{|E|}$ that is defined by the following sets of inequalities, where r_M is the rank function of M .

$$\mathcal{P}(M) = \left\{ \begin{array}{ll} \sum_{e \in U} x_e \leq r_M(U), & \forall U \subseteq E \\ x_e \geq 0, & \forall e \in E \end{array} \right\}$$

It is well known that one can check whether some $x \in \mathbb{R}^{|E|}$ lies in the matroid polytope, that is $x \in \mathcal{P}(M)$, in polynomial time in the size of E .

3 Overview of our Techniques

The core difficulty of the MATROID PORTFOLIO OPTIMIZATION problem stems from its objective function. Indeed, calculating or bounding, the expectation of the maximum of k random variables is a difficult task, especially when the random variables are dependent. Apart from that, optimizing under this objective requires to look at the problem from a perspective that differs with what we are used to in the analysis of randomized algorithms. Usually, we argue that our solutions have a good enough expectation and that they reach this expectation with a reasonable probability. However, in order to construct a good portfolio we are interested in the “anti-concentration” properties of the solutions that we pick, meaning that we want each solution to be able to greatly surpass its expected value with a reasonable probability. Intuitively, if the values of our solutions were i.i.d. random variables, we would want them to be somewhat “heavy-tailed” so that after k independent samples there would be a good probability that we observe a high outlier. On the other hand, if the solutions that we pick were highly concentrated around their expected value, then we would observe almost no benefit by taking the maximum of several random variables.

In this section, we first highlight our ideas for the simpler case of MATROID PORTFOLIO OPTIMIZATION for uniform matroids and then explain how they can properly be generalized to work for all matroids. As a reminder, in the MATROID PORTFOLIO OPTIMIZATION problem on uniform matroids, we are given a ground set of n elements and we want to construct a portfolio \mathcal{P} consisting of k subsets of the elements, each of size r . The value of the i -th element is an independent Bernoulli random variable with probability p_i and we are interested in maximizing the value of the portfolio that we construct.

Even in this simple case, understanding the core trade-off of using a higher probability element multiple times across solutions versus replacing it with independent elements of lower probability is a challenging task. In order to build some intuition, in the next subsection, we discuss some natural approaches for this simple case and show why they fail to produce a portfolio that is a constant-factor approximation of the optimal portfolio.

3.1 Natural approaches that fail

The first approach for this problem is simply to pick the k independent sets with the highest expected value. Of course, it makes sense to enforce that these independent sets are *disjoint*, since we want them to “complement” each other. If these sets had large pairwise intersections, we would observe no benefit from selecting k of them instead of one. In other words, a natural strategy is to pick the highest expectation subset, remove it from the ground set, continue by picking the highest expectation subset of the remaining elements and so on. However, this approach will fail to construct a good portfolio.

Intuitively, in some cases it is better to restrict ourselves to considerably less disjoint subsets and complete our portfolio by combining those subsets in a clever way, instead of using new subsets that might have lower expectation. For example, consider the instance where we need to pick k subsets of size $r = k$ and we have $n = r^2$ elements available. Let the first $k \log k$ elements have activation probability $1/k$, and the rest to have activation probability $1/k^2$. The approach described above will form k disjoint subsets, out of which $\log k$ will behave as independent binomials with expectation 1 and the rest will be independent binomials with expectation $1/k$. The value of this portfolio will be dominated by the expectation of the maximum of the first $\log k$ binomials, which is $O(\log \log k)$. Surprisingly, instead of using the lower probability elements, one can pick k solutions by uniformly combining parts of the first $\log k$ subsets and construct a portfolio that has value $\Theta(\log k / \log \log k)$, which is

asymptotically optimal (we describe this construction in the full version of the paper). This example showcases the power of having a clever “mixing” strategy as this can boost the portfolio to achieve an exponentially better value than a portfolio that only uses disjoint solutions. In fact, in this example, a strategy that picks only disjoint solutions would need k disjoint subsets of expectation 1 in order to achieve the same value as the one achieved by uniformly mixing $\log k$ subsets of expectation 1.

3.2 Main ideas

Filtering out elements

From the previous example, it becomes evident that a candidate algorithm should have a filtering procedure that discards some elements of the ground set, and a mixing strategy, which combines the remaining elements to form the desired subsets. It is natural to wonder if one can commit to the simplest possible mixing strategy (i.e. sampling elements uniformly) and try to find a filtering rule that would make this strategy work. Note that the nearly-optimal solution constructed for the previous example actually fits into this framework.

In the simpler case of uniform matroids, designing the filtering procedure can be reduced to first sorting the elements in decreasing order of their probabilities and then selecting a prefix of this order. A key observation in our work is that there always exists a prefix of the elements that admits a good portfolio under the simplest mixing strategy, i.e. forming subsets by uniform sampling. This observation directly gives us a polynomial-time constant-approximation algorithm, as one can try all possible n prefixes, generate the corresponding portfolios through uniform sampling and keep the best one of them after estimating their values. We formally present this algorithm in Section 4 and build up to our algorithm for general matroids in the full version of our paper.

Analyzing the value of the produced portfolio

The simplicity of our mixing strategy allows us to bypass dealing with the dependencies of the solutions we form when we want to lower bound the expectation of their maximum. On a high level, to analyze the value of our portfolio, we first fix the randomness of the instance, by conditioning on an outcome for the active elements, and then analyze the expected value of a sampled solution over the internal randomness of the algorithm. Once we have fixed the outcome of the active elements, the *values* of the sampled solutions are simply binomial random variables that only depend on the internal randomness of the sampling procedure. Critically, this makes the values of the sampled solutions *independent* random variables which makes them much easier to work with. On the other hand, if we had first fixed the portfolio produced by the algorithm and then tried to analyze its value over the randomness of the instance, we would have to analyze the dependencies of the picked solutions and how these influence the value of the portfolio, which is a much harder task. This analysis technique, of course, comes with its own challenges as selecting the appropriate event to condition on is not a trivial task.

Generalizing to all matroids

The filtering procedure described above fails to work beyond uniform matroids, simply because it ignores the underlying matroid structure of the elements. For example, consider the case of the graphic matroid of a graph G that consists of a clique on \sqrt{n} vertices and a simple path of $n - \sqrt{n}$ vertices. Let the activation probabilities of the edges of the clique

to be $1/\sqrt{n}$ and the activation probabilities of the edges of the path to be $1/\sqrt{n} - \epsilon$ for some small $\epsilon > 0$. Ordering the elements by their activation probabilities and selecting a prefix of this ordering goes towards the wrong direction as one should prioritize taking edges from the path over taking edges from the clique. Indeed, taking the path as our only solution would give us an expected value of $\Theta(\sqrt{n})$, whereas any portfolio of $k = O(n)$ spanning trees of the clique has value at most $O(\log n / \log \log n)$.

In order to overcome this issue, we change our filtering procedure to select a certain number of disjoint, high-expectation, independent sets of the matroid. In other words, we create an ordering of disjoint independent sets of decreasing expectation and we take a prefix of this ordering. Constructing this ordering can simply be achieved by finding the biggest expectation base of the matroid through the Greedy algorithm, then restricting the matroid to the remaining elements and repeating. As in the case of uniform matroids, we prove that there always exists a prefix of this ordering that admits a nearly-optimal portfolio after sampling elements uniformly and then passing them through a contention resolution scheme.

Finally, the biggest technical challenge of this problem is analyzing the value of the produced portfolios in the case of general matroids. To be more precise, the value of a portfolio heavily relies on “abnormal” events happening in one of its k solutions. For example, if we fix a set of active elements, and analyze the expected value of the sampled solutions over the internal randomness of the algorithm, the expected value of each solution will be the number of active elements we expect to sample in one trial. Crucially, one of the k trials will sample much more active elements than its expectation, and this trial will be responsible for the value of the portfolio. At this point, this solution will be passed through a contention resolution scheme to be trimmed down to an independent set. However, contention resolution schemes do not have per-instance guarantees but only work in expectation. Therefore, there is no guarantee that, when this abnormal event happens, the active elements are not going to be discarded by the contention resolution scheme. We overcome this issue by conditioning on appropriate events that do not entirely fix the randomness of either the instance or the algorithm, but still allow us to use the guarantees of a contention resolution scheme and argue that one of the sampled solutions reaches a near-optimal value. This is the main result of our work which is formally stated below.

► **Theorem 2.** *If \mathcal{D} is a product distribution then there exists an algorithm that achieves $\Theta(1)$ -approximation for the k -portfolio solution problem and has a polynomial time complexity.*

4 Warm-up: An $O(1)$ approximation for Uniform Matroids

In this section we formally introduce some of the ideas described above by presenting our algorithm for the case of uniform matroids. We remind the reader that the input is described by a triplet $\mathcal{J} = (M, k, \mathcal{D})$ where $M = (E, \mathcal{I})$ is a (uniform in this section) matroid with rank r , k is the size of our portfolio and \mathcal{D} is a product distribution where each element $e \in E$ is active with probability p_e . For simplicity, we order elements in E in decreasing order of their probabilities. We denote by e_i the i -th element in this order and by p_i its activation probability. In addition, throughout this section we assume that $\text{OPT} \geq 200$. In the full version of the paper, we prove that lower bounding OPT by a large constant is without loss of generality.

As mentioned in Section 3, our algorithm filters out some elements of the ground set and then produces the k solutions of its portfolio by sampling elements uniformly at random. We prove that for any instance of the problem, there exists a prefix of the elements (when ordered by decreasing activation probability) that admits a near-optimal portfolio under

46:10 Data-Driven Solution Portfolios

uniform sampling. Therefore, the algorithm simply needs to try all n prefixes, estimate the values of the constructed portfolios and output the best one of them. We give the pseudocode of this strategy in Algorithm 1.

■ **Algorithm 1** An algorithm for Uniform Matroids.

```

function CREATE-PORTFOLIO( $\mathcal{J} = (M, k, \mathcal{D})$ )           ▷  $M$ : uniform matroid of rank  $r$ 
  Portfolios  $\leftarrow \emptyset$ 
  for  $i = 1, \dots, n$  do
    Prefix  $\leftarrow \{1, \dots, i\}$                        ▷ Elements are ordered with decreasing  $p_i$ 
    Portfolios[ $i$ ]  $\leftarrow$  PORTFOLIO-FROM-PREFIX(Prefix,  $k, r$ )
  end for
  Estimate the values of the  $n$  portfolios
  Return the portfolio with the biggest estimated value
end function

function PORTFOLIO-FROM-PREFIX(Prefix,  $k, r$ )
   $\mathcal{P} \leftarrow \{\}$ 
  for  $i = 1, \dots, k$  do
    Let  $V_i = \{s_1, \dots, s_r\}$  be  $r$  uniformly random samples from Prefix
     $S \leftarrow$  non-duplicate elements of  $V_i$ 
     $\mathcal{P} \leftarrow \mathcal{P} \cup S$ 
  end for
  return  $\mathcal{P}$ 
end function

```

In order to prove that Algorithm 1 is an $O(1)$ -approximation, it suffices to show that there exists a prefix on which sampling elements uniformly creates a portfolio with value at least $\Theta(1) \cdot \text{OPT}$. Indeed, the value of each solution can be estimated using standard techniques within a small factor with polynomially many samples with high probability. Since there is only n solutions the estimate is close for all of them with good probability (by union bound) and we output the best solution. The prefix on which we will focus, is the largest prefix whose expected value is at most $\text{OPT}/2$. More formally, let M be the largest index such that

$$\sum_{i=1}^M p_i < \text{OPT}/2.$$

By the definition of M , we know that $\sum_{i=1}^{M+1} p_i \geq \text{OPT}/2$, therefore we also get that

$$\sum_{i=1}^M p_i \geq \text{OPT}/2 - p_{M+1} \geq \text{OPT}/2 - 1 \geq \text{OPT}/3,$$

where in the last inequality we used that $\text{OPT} \geq 6$.

Our first observation is that there exists a portfolio which only uses elements outside of the selected prefix, i.e. elements with lower activation probability than what our algorithm has picked, and that achieves value at least $\text{OPT}/2$. This portfolio is the restriction of the optimal one to the elements outside of the prefix. Let $H = \{e_1, \dots, e_M\}$ and $L = \{e_{M+1}, \dots, e_n\}$. The aforementioned claim is stated formally in the following lemma:

► **Lemma 3.** $\mathbf{E}_{A \sim \mathcal{D}} [\max_{O_i \in \mathcal{O}} |O_i \cap L \cap A|] \geq \frac{\text{OPT}}{2}$.

Proof of Lemma 3.

$$\text{OPT} = \mathbf{E}_{A \sim \mathcal{D}} \left[\max_{O_i \in \mathcal{O}} |O_i \cap A| \right] \quad (1)$$

$$= \mathbf{E}_{A \sim \mathcal{D}} \left[\max_{O_i \in \mathcal{O}} |O_i \cap H \cap A| + |O_i \cap L \cap A| \right] \quad (2)$$

$$\leq \mathbf{E}_{A \sim \mathcal{D}} \left[|A \cap H| + \max_{O_i \in \mathcal{O}} |O_i \cap L \cap A| \right] \quad (3)$$

$$\leq \frac{\text{OPT}}{2} + \mathbf{E}_{A \sim \mathcal{D}} \left[\max_{O_i \in \mathcal{O}} |O_i \cap L \cap A| \right] \quad (4)$$

$$\Rightarrow \mathbf{E}_{A \sim \mathcal{D}} \left[\max_{O_i \in \mathcal{O}} |O_i \cap L \cap A| \right] \geq \frac{\text{OPT}}{2}, \quad (5)$$

where from (2) to (3) we used that $O_i \cap H \cap A \subseteq H \cap A$ for all i and from (3) to (4) that $\mathbf{E}_{A \sim \mathcal{D}} [|A \cap H|] = \sum_{i=1}^M p_i \leq \frac{\text{OPT}}{2}$. ◀

The next observation we make is that if someone had access to $k \cdot r$ independent copies of the element e_{M+1} , then by using these elements they could construct a portfolio that has value at least $\text{OPT}/2$.

► **Lemma 4.** *Let B_1, \dots, B_k be k i.i.d random variables following $\text{Bin}(r, p_{M+1})$. Then,*

$$\mathbf{E} \left[\max_{i \in [k]} B_i \right] \geq \frac{\text{OPT}}{2}.$$

Proof of Lemma 4. From Lemma 3, we know that there exist k dependent Poisson Binomials, namely the solutions picked by the optimal portfolio restricted to L , each of which has at most r trials, trial probabilities at most p_{M+1} , and whose expected maximum is at least $\text{OPT}/2$. The proof follows from the fact that one construct the desired Binomials, B_1, \dots, B_k , by starting from the Poisson Binomials and doing the following transformations:

1. Increase the number of trials of all Poisson Binomials to r by augmenting new independent Bernoulli random variables each having probability p_{M+1} .
2. Make the Poisson Binomials independent by introducing new independent copies for the elements that are shared across many solutions.
3. Transform the Poisson Binomials into Binomials by increasing all probabilities to p_{M+1} .

All of the above transformations do not decrease the expectation of the maximum of the random variables, since (1) the augmented variables stochastically dominate the previous ones, (2) making the random variables independent can only increase the expectation of their maximum, because solutions with shared elements are positively correlated and “fail” together and (3) increasing the probabilities of the trials can also only increase the expectation of their maximum. ◀

The previous lemma gives us a “target” for analyzing the expected value of our portfolio. After conditioning on an appropriate constant probability event for the activation of elements, we argue that the *values* of the sampled solutions are independent Binomial random variables with trial probability close to p_{M+1} . We define such event as H having at least $\text{OPT}/12$ active elements and prove that the latter happens with probability at least $1/2$. To that end, let $W = \{\tilde{E} \subseteq E : |\tilde{E} \cap H| \geq \text{OPT}/12\}$.

► **Lemma 5.** $\Pr_{A \sim \mathcal{D}} [A \in W] \geq 1/2$.

Proof of Lemma 5. The left hand side of the desired inequality can be written as:

$$\Pr_{A \sim \mathcal{D}} [A \in W] = \Pr_{A \sim \mathcal{D}} \left[|A \cap H| \geq \frac{\text{OPT}}{12} \right] \quad (1)$$

$$\geq \Pr_{A \sim \mathcal{D}} \left[|A \cap H| \geq \frac{\mathbf{E}[|A \cap H|]}{4} \right] \quad (2)$$

$$\geq \Pr_{A \sim \mathcal{D}} \left[\left| |A \cap H| - \mathbf{E}[|A \cap H|] \right| \leq \frac{3}{4} \mathbf{E}[|A \cap H|] \right] \quad (3)$$

$$\geq 1 - 2e^{-\left(\frac{3}{4}\right)^2 \cdot \frac{1}{3} \cdot \mathbf{E}[|A \cap H|]} \quad (4)$$

$$\geq \frac{1}{2} \quad (5)$$

where for (1) and (4) we used that $\mathbf{E}_{A \sim \mathcal{D}} [|A \cap H|] = \sum_{i=1}^M p_i \geq \frac{\text{OPT}}{3} \geq 10$ and for (3) we used a Chernoff Bound for the Binomial random variable $|A \cap H|$. ◀

We continue by proving Lemma 6 for the expected value of the sampled multisets V_i . For simplicity, we slightly abuse notation and for any set $\tilde{E} \subseteq E$ we use $|V_i \cap \tilde{E}|$ to denote the sum $\sum_{x \in V_i} \mathbb{1}\{x \in \tilde{E}\}$.

► **Lemma 6.** *For any $\tilde{E} \in W$, it holds that*

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| \right] \geq \text{OPT}/24.$$

Proof of Lemma 6. When we have fixed an outcome $\tilde{E} \in W$ for the active elements, the values of the sampled multisets, V_i , depend only on the internal randomness of our sampling procedure and are, thus, independent random variables. In addition, the algorithm samples elements uniformly at random from H . Therefore the probability that a sampled element is active is

$$\frac{|\tilde{E} \cap H|}{|H|} \geq \frac{\text{OPT}}{12 \cdot |H|} \geq \frac{\sum_{i \in H} p_i}{12 \cdot |H|} \geq \frac{p_{M+1}}{12},$$

where for the first inequality we used that the definition of W , for the second inequality we used that $\sum_{i \in H} p_i \leq \text{OPT}/2$ from the definition of the prefix H , and for the third inequality we used that $\forall i \in H : p_i \geq p_{M+1}$.

Therefore, the random variables $|V_i \cap \tilde{E}|$ are independent Binomials with trial probability at least $p_{M+1}/12$ for all i . Intuitively, this means that the expectation of their maximum should be close to the expectation of the maximum of k independent binomials with trial probability p_{M+1} , which by Lemma 4 is at least $\Theta(1) \cdot \text{OPT}$.

More formally, let B_1, \dots, B_k be iid random variables following $\text{Bin}(r, p_{M+1})$ and B'_1, \dots, B'_k be iid random variables following $\text{Bin}(r, p_{M+1}/12)$. Then for any $\tilde{E} \in W$, it holds that

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| \right] \geq \mathbf{E} \left[\max_{i \in [k]} B'_i \right] \geq \frac{1}{12} \mathbf{E} \left[\max_{i \in [k]} B_i \right] \geq \frac{\text{OPT}}{24},$$

where the first inequality holds because the values of the sampled multisets dominate the random variables B'_i . For the second inequality, intuitively, one can view the process of sampling the random variables B'_i as first sampling the Binomials B_1, \dots, B_k and then discarding every Bernoulli random variable that succeeded, independently, with probability $1/12$. In this way, for every outcome of the Binomials B_1, \dots, B_k , the Binomials B'_1, \dots, B'_k will retrieve, on expectation, a $1/12$ factor of their corresponding random variables B_i . Finally, for the third inequality we used Lemma 4. ◀

We continue by proving that for any “good” outcome $\tilde{E} \in W$, the expected value of the maximum of the sets S_1, \dots, S_k , that consist of the unique elements of the sampled multisets, V_i , is still $\Theta(1) \cdot \text{OPT}$.

► **Lemma 7.** *For any $\tilde{E} \in W$, it holds that*

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \right] \geq \frac{e}{240(e-1)} \cdot \text{OPT}.$$

Proof of Lemma 7. In order to analyze the expected value of the portfolio for the events $\tilde{E} \in W$, we will condition on the value of the maximum of the multisets V_1, \dots, V_k . By the law of total expectation we get that

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \right] = \sum_x \mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \mid \max_{i \in [k]} |V_i \cap \tilde{E}| = x \right] \cdot \Pr_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| = x \right]. \quad (\ddagger)$$

Lower bounding the following expression,

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \mid \max_{i \in [k]} |V_i \cap \tilde{E}| = x \right],$$

can be seen as a balls and bins question. Specifically, we know that one of the k solutions formed by the algorithm sampled x items from $|\tilde{E} \cap H|$. Since the algorithm is sampling elements uniformly at random, those x items are also distributed uniformly at random inside $|\tilde{E} \cap H|$. We are interested in calculating the expected number of distinct items that were sampled. This question is equivalent to counting the non-empty bins after throwing x balls uniformly into $|\tilde{E} \cap H|$ bins. Using a standard result about the expected number of non-empty bins, we get that

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \mid \max_{i \in [k]} |V_i \cap \tilde{E}| = x \right] \geq \min \left\{ \frac{x}{2}, \frac{3|\tilde{E} \cap H|}{10} \right\}.$$

Eq. \ddagger can now be re-written as

$$\mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \right] \geq \sum_x \min \left\{ \frac{x}{2}, \frac{3|\tilde{E} \cap H|}{10} \right\} \cdot \Pr_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| = x \right] \quad (1)$$

$$\geq \sum_x \min \left\{ \frac{x}{2}, \frac{\text{OPT}}{40} \right\} \cdot \Pr_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| = x \right] \quad (2)$$

$$\geq \min \left\{ \frac{\text{OPT}}{300}, \frac{\text{OPT}}{40} \right\} \cdot \Pr_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| \geq \frac{\text{OPT}}{150} \right] \quad (3)$$

$$\geq \frac{\text{OPT}}{300} \cdot \Pr_{\text{ALG}} \left[\max_{i \in [k]} |V_i \cap \tilde{E}| \geq \frac{\mathbf{E}_{\text{ALG}} [\max_{i \in [k]} |V_i \cap \tilde{E}|]}{5} \right] \quad (4)$$

$$\geq \frac{e}{300(e-1)} \cdot \text{OPT}, \quad (5)$$

where for (2) we used the fact that $|\tilde{E} \cap H| \geq \text{OPT}/12$ for $\tilde{E} \in W$, to get (3) we restricted the sum to the terms $x \geq \text{OPT}/150$, to get (4) we used that $\mathbf{E}_{\text{ALG}} [\max_{i \in [k]} |V_i \cap \tilde{E}|] \geq \text{OPT}/24$ from Lemma 6 and to get (5) we used a concentration inequality for the maximum of independent Binomial random variables and the assumption that $\text{OPT} \geq 800$ to get that $\mathbf{E}_{\text{ALG}} [\max_{i \in [k]} |V_i \cap \tilde{E}|]$ is at least some constant. ◀

Finally, we are ready to prove the main theorem of the section.

► **Theorem 8.** *Algorithm 1 is a $\Theta(1)$ -approximation algorithm for the k -solution portfolio problem when the given matroid is uniform and \mathcal{D} is a product distribution.*

Proof of Theorem 8. The value of the constructed portfolio \mathcal{P} can be written as:

$$\text{value}(\mathcal{P}) = \mathbf{E}_{\text{ALG}, A \sim \mathcal{D}} \left[\max_{i \in [k]} |S_i \cap A| \right] \quad (1)$$

$$\geq \sum_{\tilde{E} \in W} \mathbf{E}_{\text{ALG}} \left[\max_{i \in [k]} |S_i \cap \tilde{E}| \right] \cdot \Pr_{A \sim \mathcal{D}} [A = \tilde{E}] \quad (2)$$

$$\geq \frac{e}{300(e-1)} \cdot \text{OPT} \cdot \sum_{\substack{A \sim \mathcal{D} \\ \tilde{E} \in W}} \Pr [A = \tilde{E}] \quad (3)$$

$$\geq \frac{e}{300(e-1)} \cdot \text{OPT} \cdot \Pr_{A \sim \mathcal{D}} [A \in W] \quad (4)$$

$$\geq \frac{e}{600(e-1)} \cdot \text{OPT}, \quad (5)$$

where to get (3) we applied Lemma 7 and (5) follows from Lemma 5. ◀

References

- 1 Maria-Florina Balcan. Data-driven algorithm design, 2020. [arXiv:2011.07177](https://arxiv.org/abs/2011.07177).
- 2 Eric Balkanski, Baharan Mirzasoleiman, Andreas Krause, and Yaron Singer. Learning sparse combinatorial representations via two-stage submodular maximization. In *International Conference on Machine Learning*, pages 2207–2216. PMLR, 2016. URL: <http://proceedings.mlr.press/v48/balkanski16.html>.
- 3 Justin Chen, Sandeep Silwal, Ali Vakilian, and Fred Zhang. Faster fundamental graph algorithms via learned predictions. In *International Conference on Machine Learning*, pages 3583–3602. PMLR, 2022. URL: <https://proceedings.mlr.press/v162/chen22v.html>.
- 4 Sami Davies, Benjamin Moseley, Sergei Vassilvitskii, and Yuyan Wang. Predictive flows for faster ford-fulkerson. In *International Conference on Machine Learning*, pages 7231–7248. PMLR, 2023. URL: <https://proceedings.mlr.press/v202/davies23b.html>.
- 5 Sami Davies, Sergei Vassilvitskii, and Yuyan Wang. Warm-starting push-relabel. *arXiv preprint*, 2024. doi:10.48550/arXiv.2405.18568.
- 6 Jeff Decary, David Bergman, Carlos Cardonha, Jason Imbrogno, and Andrea Lodi. The madness of multiple entries in march madness, 2024. doi:10.48550/arXiv.2407.13438.
- 7 Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. *Advances in neural information processing systems*, 34:10393–10406, 2021. URL: <https://proceedings.neurips.cc/paper/2021/hash/5616060fb8ae85d93f334e7267307664-Abstract.html>.
- 8 Swati Gupta, Jai Moondra, and Mohit Singh. Balancing notions of equity: Approximation algorithms for fair portfolio of solutions in combinatorial optimization. *arXiv preprint*, 2023. doi:10.48550/arXiv.2311.03230.
- 9 Jon Kleinberg, Christos Papadimitriou, and Prabhakar Raghavan. Segmentation problems. *Journal of the ACM (JACM)*, 51(2):263–280, 2004. doi:10.1145/972639.972644.
- 10 Jon Kleinberg and Maithra Raghun. Team performance with test scores. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):1–26, 2018. doi:10.1145/3274644.
- 11 Silvio Lattanzi, Ola Svensson, and Sergei Vassilvitskii. Speeding up bellman ford via minimum violation permutations. In *International Conference on Machine Learning*, pages 18584–18598. PMLR, 2023. URL: <https://proceedings.mlr.press/v202/lattanzi23a.html>.
- 12 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022. doi:10.1145/3528087.

- 13 George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions – I. *Mathematical programming*, 14:265–294, 1978. doi:10.1007/BF01588971.
- 14 Sahil Singla. Combinatorial optimization under uncertainty: Probing and stopping-time algorithms. *Unpublished doctoral dissertation, Carnegie Mellon University*, 2018.
- 15 Serban Stan, Morteza Zadimoghaddam, Andreas Krause, and Amin Karbasi. Probabilistic submodular maximization in sub-linear time. In *International Conference on Machine Learning*, pages 3241–3250. PMLR, 2017. URL: <http://proceedings.mlr.press/v70/stan17a.html>.