Random Sampling and Greedy Sparsification for Matroid Optimization Problems.

David R. Karger*
MIT
Laboratory for Computer Science
karger@lcs.mit.edu

Abstract

Random sampling is a powerful tool for gathering information about a group by considering only a small part of it. We discuss some broadly applicable paradigms for using random sampling in combinatorial optimization, and demonstrate the effectiveness of these paradigms for two optimization problems on matroids: finding an optimum matroid basis and packing disjoint matroid bases. Applications of these ideas to the graphic matroid led to fast algorithms for minimum spanning trees and minimum cuts.

An optimum matroid basis is typically found by a *greedy algorithm* that grows an independent set into an the optimum basis one element at a time. This continuous change in the independent set can make it hard to perform the independence tests needed by the greedy algorithm. We simplify matters by using sampling to reduce the problem of finding an optimum matroid basis to the problem of verifying that a given *fixed* basis is optimum, showing that the two problems can be solved in roughly the same time.

Another application of sampling is to packing matroid bases, also known as matroid partitioning. Sampling reduces the number of bases that must be packed. We combine sampling with a greedy packing strategy that reduces the size of the matroid. Together, these techniques give accelerated packing algorithms. We give particular attention to the problem of packing spanning trees in graphs, which has applications in network reliability analysis. Our results can be seen as generalizing certain results from random graph theory. The techniques have also been effective for other packing problems.

1 Introduction

Arguably the central concept of statistics is that of a representative sample. It is often possible to gather a great deal of information about a large population by examining a small sample randomly drawn from it. This has obvious advantages in reducing the investigator's work, both in gathering and in analyzing the data.

We apply the concept of a representative sample to combinatorial optimization. Given an optimization problem, it may be possible to generate a small representative subproblem by random sampling. Intuitively, such a subproblem may form a microcosm of the larger problem. In particular, an optimum solution to the subproblem may be a nearly optimum solution to the problem as a whole. In some situations, such an approximation might be sufficient. In other situations, it may be relatively easy to improve this good solution to a truly optimum solution.

*Some of this work done at Stanford University, supported by National Science Foundation and Hertz Foundation Graduate Fellowships, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation and Xerox Corporation. Research supported in part by ARPA contract N00014-95-1-1246 and NSF contract CCR-9624239

We show that these desirable properties apply to two problems involving matroids: *matroid optimization,* which aims to find a minimum cost matroid basis, and *basis packing,* which aims to find a maximum set of pairwise-disjoint matroid bases.

An early algorithmic use of sampling was in a median finding algorithm due to Floyd and Rivest [FR75]. Their algorithm selects a small random sample of elements from the set; they prove that inspecting this sample gives a very accurate approximation to the value of the median. It is then easy to find the actual median by examining only those elements close to the estimate. This algorithm uses fewer comparisons (in expectation) than any other known median-finding algorithm.

Examination of this algorithm reveals a paradigm involving three parts. The first is a definition of a *randomly sampled subproblem.* The second is an *approximation theorem* that describes how a solution to the subproblem approximates a solution to the original problem. These two components by themselves will typically yield an obvious approximation algorithm with a speed-accuracy tradeoff. The third component is a *refinement algorithm* that takes the approximate solution and turns it into an exact solution. Combining these three components can yield an algorithm whose running time will be determined by the refinement algorithm; intuitively, refinement should be easier than computing a solution from scratch.

This approach been applied successfully to several graph optimization problems. A preliminary version of this paper [Kar93] gave a sampling-based algorithm for the minimum spanning tree problem, after which Klein and Tarjan [KT94] showed that a version of it ran in linear time; a merged journal version appeared in [KKT95]. This author [Kar97a, Kar96, BK96, Kar97b] used these sampling techniques in fast algorithms for finding and approximating minimum cuts and maximum flows in undirected graphs.

In this paper, we extend our techniques from graphs to matroids. Besides increasing our understanding of what is important in the graph sampling results, our matroid sampling techniques have additional applications, in particular to packing spanning trees.

1.1 Matroids

We give a brief discussion of matroids and the ramifications of our approach to them. An extensive discussion of matroids can be found in [Wel76].

The matroid is an abstraction that generalizes both graphs and vector spaces. A matroid M consists of a finite ground set M of which some subsets are declared to be *independent.* The independent sets must satisfy three properties:

- The empty set is independent.
- All subsets of an independent set are independent.
- If U and V are independent, and $\|U\| > \|V\|$, then some element of U can be added to V to yield an independent set.

This definition clearly generalizes the notion of linear independence in vector spaces [VDW37]. However, it was noted [Whi35] that matroids also generalize graphs: in the *graphic matroid* the edges of the graph form the ground set, and the independent sets are the acyclic sets of edges (forests). Maximal independent sets of a matroid are called *bases;* bases in a vector space are the standard ones while the bases in a graph are the spanning forests (spanning trees, if the graph is connected). In the *matching matroid* of a graph [Law76], bases correspond to the endpoints of maximum matchings.

Matroids have rich structure and are the subject of much study in their own right [Wel76]. Matroid theory is used to solve problems in electrical circuit analysis and structural rigidity [Rec89]. A discussion of many optimization problems that turn out to be special cases of matroid problems can be found in [LR92].

1.2 Matroid Optimization

In computer science, perhaps the most natural problem involving matroids is *matroid optimization*. If a weight is assigned to each element of a matroid, and the weight of a set is defined as the sum of its elements' weights, the optimization problem is to find a basis of minimum weight. The minimum spanning tree problem is the matroid optimization problem on the graphic matroid. Several other problems can also be formulated as instances of matroid optimization [CLR90, Law76, LR92].

Edmonds [Edm71] observed that the matroid optimization problem can be solved by the following natural greedy algorithm. Begin with an empty independent set I, and consider the matroid elements in order of increasing weight. Add each element to I if doing so will keep I independent. Applying the greedy algorithm to the graphic matroid yields Kruskal's algorithm [Kru56] for minimum spanning trees: grow a forest by repeatedly adding to the forest the minimum weight edge that does not form a cycle with edges already in the forest. An interesting converse result [Wel76] is that if a family of sets does not satisfy the matroid independent set properties, then there is an assignment of weights to the elements for which the greedy algorithm will fail to find an optimum set in the family.

The greedy algorithm has two drawbacks. First, the elements of the matroid must be examined in order of weight. Thus the matroid elements must be sorted, adding an $\Omega(m \log m)$ term to the running time of the greedy algorithm on an m-element matroid. Second, the independent set under construction is constantly changing, so that the problem of determining independence of elements is a *dynamic* one.

Contrast the optimization problem with that of *verifying* the optimality of a given basis. For matroids, all that is necessary is to verify that no single element of the matroid M "improves" the basis. Thus in verification the elements of M can be examined in any order. Furthermore, the basis that must be verified is *static*. Extensive study of dynamic algorithms has demonstrated that they tend to be significantly more complicated than their static counterparts—in particular, algorithms on a static input can preprocess the input so as to accelerate queries against it.

Applying the sampling paradigm, we show in Section 2 how an algorithm for verifying basis optimality can be used to construct an optimum basis. The reduction is very simple and suggests that the best way to develop a good optimization algorithm for a matroid is to focus attention on developing a good verification algorithm.

1.3 Basis Packing

Packing problems are common in combinatorial optimization. Some well studied problems that can be seen as packing problems include the maximum flow problem (packing paths between a source and sink), computing graph connectivity (equivalent to packing directed spanning trees as discussed by Gabow [Gab95]), matching (packing disjoint edges), and bin packing. Schrijver [Sch79] discusses packing problems in numerous settings.

Many packing problems have the following form: given a collection of implicitly defined *feasible sets* over some universe, find a maximum collection of disjoint feasible sets. Random sampling applies to packing problems. A natural random subproblem is defined by specifying a random subset of the universe and asking for a packing of feasible sets within the random subset. Furthermore, if we partition the universe randomly into two sets, defining two subproblems, the solutions to the two subproblems can be merged to give an approximate solution to the whole problem.

In Section 3, we discuss the problem of packing matroid bases, *i.e.* finding a maximum set of disjoint bases in a matroid. This problem arises in the analysis of electrical networks and in the analysis of the rigidity of physical structures (see [LR92] for details). Edmonds [Edm65] gave an early algorithm for the problem. A simpler algorithm was given by Knuth [Knu73]. Faster algorithms exist for the special case of the graphic matroid [GW92, Bar95] where the problem is to find a maximum collection of disjoint spanning trees (this particular problem is important in network reliability analysis—see for example Colbourn's book [Col87]).

In Section 4, we apply random sampling to the basis packing problem. Let the *packing number* of a matroid be the maximum number of disjoint bases in it. We show that a random sample of a $1/k$ fraction of the elements from a matroid with packing number n has a packing number tightly distributed around n/k. This provides the approximation results needed to apply sampling.

Random sampling lets us reduce algorithms' dependence on the packing number k. A complementary "greedy packing" scheme generalizing Nagamochi and Ibaraki's sparse certificate technique [NI92] for graphs lets us reduce the running time dependence on the size of the matroid. We describe this scheme in Section 5.

In Section 6, we show how combined sampling and greedy packing yields approximate basis packing algorithms whose running times (aside from a linear term) depend only on the rank of the input matroid, and not its size. In section 7, we give an application of this combination to the problem of packing spanning trees in a graph.

1.4 Related work

Polesskii [Pol90] showed a bound on the likelihood that a random sample of the elements of a matroid contains a basis, as a function of the number of disjoint bases in the matroid. His result is related to our study of matroid basis packing in Section 3.

Our packing techniques have had some applications to min-cuts and max-flows in undirected graphs. Among our sampling-based results for an m-edge, n-vertex undirected graph are:

- A randomized algorithm to find minimum cuts in $\tilde{O}(m)$[1] time [Kar96],
- A randomized algorithm to find a packing of c directed spanning trees in $\tilde{O}(m\sqrt{c})$ time, extending Gabow's deterministic $\tilde{O}(mc)$ time algorithm, together with some faster maximum flow algorithms [Kar97a],
- A randomized algorithm for approximating s-t min-cuts in $\tilde{O}(n^2)$ time [BK96]

Our matroid algorithm generalizes of the graph algorithms presented there.

Reif and Spirakis [RS80] studied random matroids; their results generalized existence proofs and algorithms for Hamiltonian paths and perfect matchings in random graphs. Their approach was to analyze the average case behavior of matroid algorithms on random inputs, while our goal is to develop randomized algorithms that work well on all inputs.

Unlike the problem of counting or constructing disjoint bases, the problem of counting the total number of bases in a matroid is hard. This $\sharp\mathcal{P}$-complete generalization of the problem of counting perfect matchings in a graph has been addressed recently; see for example [FM92].

1.5 Preliminaries and Definitions

Throughout this paper, $\log m$ denotes $\log_2 m$.

For any set A, $\|A\|$ denotes the cardinality of A. The complement of A (with respect to a known universe) is denoted by $\overline{A}$.

Our algorithms rely on random sampling, typically from some universe U:

Definition 1.1. For a fixed universe U, U(p) is a set generated from U by including each element independently with probability p. For $A \subseteq U$, $A(p) = A \cap U(p)$.

Note that we are overloading the definition of $A(p)$—it reflects the outcome of the original experiment on U, and not of a new sampling experiment performed only on the elements of A. Which meaning is intended will be clear from context.

Consider a statement that refers to a variable n. We say that the statement holds *with high probability (w.h.p.) in* n if for any constant d, there is a setting of the constants in the statement (typically hidden by O-notation) such that the probability that the statement fails to hold is $O(n^{-d})$.

[1] $\tilde{O}(f)$ denotes $O(f \operatorname{polylog} f)$.

1.5.1 Randomized Algorithms

Many algorithms described in this paper are randomized. We assume these algorithms can call a subroutine `RBIT` that, in $O(1)$-time, returns a random bit that is independent (of all other bits) and unbiased (0 or 1 with equal probability). The outcomes of calls to RBIT defines a probability space over which we measure the probability of events related to the algorithm's execution. When we say that an algorithm works "with high probability" without reference to a variable then the implicit parameter is the size of our problem input—in our case, the number of elements in the matroid.

Although the model allows us to generate only unbiased random bits, a standard construction [KY76] lets us use `RBIT` to generate random bits that are 1 with any specified probability. The expected time to generate such a bit is constant, and the time to generate a sequence of more than $\log n$ such bits is (amortized) linear in the number of bits with high probability in n. All algorithms in this paper use enough random bits to allow this amortization to take place, so we assume throughout that generating a biased random bit takes constant time.

Randomized algorithms can fail to be correct in two ways. A *time $T(n)$ Monte Carlo (MC)* algorithm runs in time $T(n)$ with certainty on problems of size n and gives the correct answer with high probability (in n). A *time $T(n)$ Las Vegas (LV)* algorithm always gives the correct answer but only runs in time $T(n)$ with high probability. Any Las Vegas algorithm can be turned into a Monte Carlo algorithm with the same time bounds—simply give an arbitrary answer if the algorithm takes too long. However, there is no general scheme for turning a Monte Carlo algorithm into a Las Vegas one, since it may not be possible to check whether the algorithm's answer is correct. Our theorems claiming time bounds will append (MC) or (LV) to clarify which type of algorithm is being used.

1.5.2 The Chernoff Bound

A basic tool in our analysis is the Chernoff bound [Che52]:

Lemma 1.2 (Chernoff [Che52, MR95]). *Let X be a sum of Bernoulli random variables on n independent trials with success probabilities $p_1, \ldots, p_n$ and expected value $\mu = \sum p_i$. Then for $\epsilon \leq 1$,*

$$\Pr[X < (1-\epsilon)\mu] \leq e^{-\epsilon^2\mu/2}$$

and

$$\Pr[X > (1+\epsilon)\mu] \leq e^{-\epsilon^2\mu/3}$$

In particular, if $\mu = \Omega(\log n)$, then $X = \Theta(\mu)$ with high probability in n.

1.5.3 Matroids

We use several standard matroid definitions; details can be found in [Wel76].

Definition 1.3. The *rank* of $S \subseteq M$, written $\mathrm{rk}S$, is the size of any largest independent set in S.

Definition 1.4. A set of elements X *spans* an element e if $\mathrm{rk}(X \cup \{e\}) = \mathrm{rk}X$. Otherwise, e is *independent* of X.

Definition 1.5. The *closure of X in M*, $\mathrm{cl}(X, M)$, is the set of all elements in M spanned by X. If M is understood from context, we write $\mathrm{cl}X$.

Definition 1.6. A set is *closed* if it contains all the elements it spans.

1.5.4 Matroid Constructions

Any subset $S \subset M$ can itself be thought of as a matroid: its independent sets are all the independent sets of M contained in S. Its rank as a matroid is the same as its rank as a subset.

Definition 1.7. Let A be any independent set of M. The *quotient matroid M/A* is defined as follows:

- Its elements are all elements of M independent of A

- Set S is independent in M/A if and only if $S \cup A$ is independent in M.

Note that the elements of M/A are simply the complement of $\mathrm{cl}A$. The rank of M/A, denoted $\mathrm{rk}^C(M/A)$, is $\mathrm{rk}M - \mathrm{rk}A$. In general, the rank of a set in M/A can be different from its rank in M.

Definition 1.8. If B is any subset of M, M/B denotes the quotient M/A where A is any basis of B; this is independent of the choice of A.

2 Finding a Minimum Cost Basis

In this section we address the problem of finding a minimum cost basis in a matroid. We consider sampling at random from a weighted matroid (that is, a matroid in which each element has been assigned a real-valued weight or cost) and formalize the following intuition: a random sample is likely to contain a nearly minimum cost basis. This shows that if we need to find a good but not necessarily optimum basis, it suffices to choose a small random sample from the matroid and construct its optimum basis. This gives an obvious improvement in the running time; we will make the speed-accuracy tradeoff precise. Extending this idea, once we have a good basis, we can use a *verification algorithm* to identify the elements that *improve* it, preventing it from actually being optimum. In Section 2.3 we show how this information can be used to quickly identify the optimum basis, thus reducing the problem of constructing an optimum basis to that of verifying one.

2.1 Definitions

We proceed to formalize our ideas. Fix attention on an m-element rank-r matroid M. Given $S \subseteq M$, let $\mathrm{OPT}(S)$ denote the optimum basis of the set S. By ordering same-weight elements lexicographically, we can assume without loss of generality that all element weights are unique, implying that the optimum basis is unique. Suppose we have a (not necessarily minimum weight or full rank) independent set T of G. We say that an element e *improves* T if $e \in \mathrm{OPT}(T \cup \{e\})$. In other words, an element improves an independent set if adding it allows one to decrease the cost of or enlarge that independent set. We have an equivalent alternative formulation: e improves T if and only if the elements of T weighing less than e do not span e. The proof that these definitions are equivalent can be found in any discussion of optimum bases (e.g. [Wel76]).

It is convenient for the discussion, and follows from the definition, that the elements of any independent set T improve T. It is easy to prove (but not necessary for our discussion) that $\mathrm{OPT}(M)$ is the only independent set with no improving elements other than its own. Furthermore, the optimum basis elements are the only elements that improve all bases. This allows a natural concept of approximation of minimum bases:

Definition 2.1. An independent set is *k-optimum* if at most kr elements improve it.

Our definition diverges from the more common approach of measuring approximation by the ratio of the output value to the optimum. This change is natural since in the comparison model of computation (in which all of our results apply) any approximation algorithm with a bounded relative-cost performance guarantee will necessarily be an exact algorithm. This is because the optimum basis depends only on the order, and not on the values, of elements of the matroid.

Note that our definitions apply to independent sets of any rank, not just bases.

2.2 A Sampling Theorem

In this section we determine the quality of an approximately optimum basis found by random sampling. Let $M(p)$ denote a random submatroid of M constructed by including each element of M independently with probability p. We proceed to prove that with high probability $\mathrm{OPT}(M(p))$ is $(1/p)$-optimum in M. This is a generalization of a result for graphs appearing in [KKT95]. In

preliminary versions of this paper [Kar92, Kar93], we proved the weaker claim that $\mathrm{OPT}(M(p))$ is $O((\log n)/p)$-optimum. Klein and Tarjan [KT94] gave an improved analysis for minimum spanning trees that immediately generalizes to matroids as follows:

Theorem 2.2. *The expected number of elements improving $OPT(M(p))$ is at most r/p. For any constant $\epsilon > 0$, the probability that more than $(1+\epsilon)r/p$ elements improve $OPT(M(p))$ is at most $e^{-\epsilon^2 r/2(1+\epsilon)} = e^{-\Omega(r)}$.*

Corollary 2.3. *$OPT(M(p))$ is $O(1/p)$-optimum in M with high probability in r.*

Proof. Given a matroid M and fixing p, let T denote the random set $\mathrm{OPT}(M(p))$. Let V denote the random set of elements of M that improve T but are not in T. Intuitively and in our algorithms, we determine V by constructing $M(p)$, finding its optimum basis, and then identifying the improving elements. However, to prove the theorem, we interleave the sampling process that constructs $M(p)$ with the process of finding T and V using the standard greedy algorithm for matroids.

The proof uses a variant of the standard greedy algorithm to find the optimum basis and the improving elements of the sample. Recall that the greedy algorithm examines elements in order of increasing weight and adds an element to the growing optimum basis if and only if that element is not spanned by the smaller elements that have already been examined. We modify this algorithm to interleave the sampling process to determine T and V simultaneously. Consider the algorithm `Greedy-Sampling` of Figure 1.

Input: A matroid M with m elements

Output: An instance of $M(p)$,
its optimum basis T, and
the elements V of M that improve T.

Order the elements e_i by increasing weight.
Let $T_0 = V_0 = \emptyset$.
for $i = 1$ **to** m
  if e_i is spanned by T_i **then**
    discard e_i
  else
    Flip a coin to sample e_i with probability p
    if e_i is sampled **then**
      $V_{i+1} = V_i$
      $T_{i+1} = T_i \cup \{e\}$
    else
      $V_{i+1} = V_i \cup \{e\}$
      $T_{i+1} = T_i$
return $T = T_m$ and $V = V_m$.

Figure 1: `Greedy-Sampling`

Note first that if we delete the lines involving V from the algorithm, we are left with the standard greedy algorithm running on the random matroid $M(p)$ (except that we do not bother flipping coins for elements that we know cannot be in the optimum basis). The only change is that instead of performing all the coin flips for the elements before running the algorithm, we flip the coins only when they are needed. This does not effect the outcome, since we could just as easily flip all the coins in advance but keep the results secret until we needed them (this *deferred decision principle* is

discussed in [MR95]). Thus we know that the output T is indeed the optimum basis of the sampled matroid.

We next argue that the output $V \cup T$ is indeed the set of improving elements. To see this, observe that every element e_i is examined by the algorithm. There are two cases (enumerated in the algorithm) to consider. If e_i is spanned by T_i, then since all elements of T_i are smaller than e_i, and since $T_i \subseteq T$, by definition e_i does not improve T. If e_i is not spanned by T_i, then e_i is placed in one of T_{i+1} or V_{i+1} and is thus part of the output. Thus all improving elements are in $V \cup T$. It is also clear that only improving elements are in $V \cup T$. This follows from the fact that any element in $V \cup T$ is not spanned by the elements smaller than itself in the sample.

It remains to bound the value of $s = \|V \cup T\|$. Consider the s elements for which we flip coins in the algorithm. Call each coin flip a *success* if it makes us put an element in T (probability p) and a *failure* if it makes us put an element in V. The number of improving elements is simply s, the number of coins flipped. Note as well that since T is independent, the number of successes is at most r. The question is therefore: if the success probability is p, how many coins will be flipped before we have r successes? This formulation defines the well known *negative binomial distribution* [Fel68, Mul94]. The expected number of coin flips needed is r/p. Furthermore, the probability that more than $(1+\epsilon)r/p$ coin flips are required is equal to the probability that of the first $(1+\epsilon)r/p$ coin flips, less than r successes occur. This probability is exponentially small in r by the Chernoff bound. □

Remark. Note that we cannot say that the expected number of improving elements is exactly r/p, since our experiment above may terminate *before* we have r successes (this will occur if $M(p)$ does not have full rank). Thus we can only give an upper bound.

2.3 Optimizing by Verifying

We use the results of the previous section to reduce the problem of *constructing* the optimum basis to the problem of *verifying* a basis to determine which elements improve it. Suppose that we have two algorithms available: a *construction* algorithm that takes a matroid with m elements and rank r and constructs its optimum basis in time $C(m,r)$; and a *verification* algorithm that takes an m-element, rank-r matroid M and an independent set T and determines which elements of M improve T in time $V(m,r)$. We show how to combine these two algorithms to yield a more efficient construction algorithm when V is faster than C.

We begin with a simple observation we will use to simplify some bounds in our sampling-based algorithm.

Lemma 2.4. *Without loss of generality,* $C(m,r) \leq m \log m + mV(r+1,r)$.

Proof. We give a construction algorithm that uses the verifier as a subroutine. This is an implementation of the standard greedy algorithm. First we sort the elements of M in $O(m \log m)$ time. Then we build the optimum basis incrementally. We maintain an independent set B, initially empty. We go through the elements e of M in increasing order, asking the verifier to verify independent set B against matroid $B \cup \{e\}$. If the verifier says that e improves B, it means that $B \cup \{e\}$ is independent, so we add e to B. After all m elements have been tested, B will contain the optimum basis. □

We now give sampling-based algorithm for the optimum basis. Suppose we sample each element of M with probability $1/k$ and construct the optimum basis $\mathrm{OPT}(M(p))$ of the sample using C. With high probability the sample has size $O(m/k)$ so that this takes $C(m/k,r)$ time. Use the verification algorithm to find the set V of elements of M that improve $\mathrm{OPT}(M(p))$; this takes time $V(m,r)$. Construct $\mathrm{OPT}(V)$; since $\mathrm{OPT}(M) \subseteq V$ we know $\mathrm{OPT}(V) = \mathrm{OPT}(M)$. By Theorem 2.2, V has size $O(kr)$ with high probability; thus this construction takes $C(kr,r)$ time. The overall running time is thus $V(m,r) + C(m/k,r) + C(kr,r)$. If we set $m/k = kr$, the running time becomes

$$V(m,r) + 2C(\sqrt{mr}, r).$$

This is a clear improvement when r is significantly less than m. This new algorithm is just as simple as the original construction and verification algorithms were, since it only consists of two construction calls and a verification call.

At the cost of some additional complexity, we can improve the running time further by applying the reduction recursively. We apply the algorithm `Recursive-Refinement` of Figure 2.

Algorithm `Recursive-Refinement`(M, C, V)

Input: A matroid M with m elements and rank r.
A construction algorithm C
A verification algorithm V.
Output: The optimum basis of M.

if $m < 3r$ **then**
return $C(M)$
else
$S \leftarrow M(1/2)$
$T \leftarrow$ `Recursive-Refinement`(S)
$U \leftarrow$ elements of $M - S$ improving T (using verifier V)
return $C(U)$ (which is the optimum basis of M)

Figure 2: `Recursive-Refinement`

Algorithm `Recursive-Refinement` is clearly correct by induction, since at each level of the recursion T will contain all optimum elements of M that are in S. We now analyze its running time.

We begin with an intuitive argument. Observe that the expected size of $M(1/2)$ is $m/2$. Furthermore, Theorem 2.2 says that U has expected size at most $2r$. This suggests the following recurrence for the running time C' of the new construction algorithm.

$$\begin{aligned} C'(2r, r) &= C(2r, r) \\ C'(m, r) &\leq C'(m/2, r) + V(m, r) + C(2r, r). \end{aligned}$$

If we now make the natural assumption that $V(m, r) = \Omega(m)$, this recurrence solves to

$$C'(m, r) = O(V(m, r) + C(2r, r) \log(m/r)).$$

Unfortunately, this argument is not rigorous. Our rigorous argument must be careful about passing expectations through the recurrence functions, and must allow for the possibility that V and/or C have exponential running times. We now give a rigorous high-probability result; an expected time result follows similarly. The statement of the theorem is simplified significantly if we assume that V is at least linear and at most polynomial in m. After proving the theorem for this case, we state the messier general case.

Theorem 2.5. *Suppose an m element, rank r matroid has algorithms for constructing an optimum basis in $C(m, r)$ time and for determining the elements that improve a given independent set in $V(m, r)$ time, where V is at least linear and at most polynomial in m. Then with high probability in m, an optimum basis for the matroid can be found in $O(V(m, r) + C(3r, r) \log(m/r))$ time (LV).*

Proof. We prove an $O(1/m)$ probability of failure; a general high probability bound follows with different choice of constants.

We will need different proofs depending on whether or not $r = o(\log m)$. We begin with some observations that apply to both cases. We first determine the size of all problems passed to the verifier. Consider the recursion tree produced by the algorithm. Let M_i be the set of elements input to level i of the recursion tree. Thus $M_{i+1} = M_i(1/2)$ is the set of elements sampled at level i. Let $T_i \subset M_i$ be the set of elements returned to level i by the $(i+1)^{st}$ recursive call. Then the set of elements passed to the verifier at level i is just $(M_i - M_{i+1}) \cup T_i$. A given element e is in $M_i - M_{i+1}$ for exactly one value of i, and each T_i has size at most r. It follows that if the recursion reaches depth d then the total size of problems passed to the verifier is at most $m + rd$. Assuming V is superlinear, it follows that the total time spent verifying is $O(V(m + rd, r))$.

We now prove the time bound for the case $r > 13 \ln m$. Consider the recursion tree produced by the algorithm. In the recursive step, each element of M is pushed down to the next level of the recursion with probability $1/2$. It follows that the probability that an element survives to recursion depth $\log(m/r)$ is $O(r/m)$. Therefore, the expected number of elements at level $\log(m/r)$ in the recursion tree is r. By the Chernoff bound with $\epsilon = 1$, the number of elements at this level at most $2r$ with probability $1 - e^{-r/3} > 1 - m^{-4}$, so the recursion terminates with high probability in m at depth $\log(m/r)$. It follows from the previous paragraph (assuming V is polynomial, and since $r \log m/r = O(m)$) that the time spent verifying is $O(V(m + r\log(m/r))) = O(V(m))$ with high probability.

We now determine the time spent in calls to C. Each set passed to C is known by Theorem 2.2 to have size upper bounded by a negative binomial distribution with mean $2r$. That theorem also shows that the probability that the passed set has size exceeding $3r$ is at most $e^{-(1/2)^2 r/3} = e^{-r/12} < m^{-13/12}$. We have already argued that the recursion terminates at depth $\log(m/r)$, so there are only $\log(m/r)$ calls to C. Thus the probability that any one of the calls involves a set whose size exceeds size $3r$ is $O(1/m)$.

It remains to consider the case $r < 13 \ln m$. We again begin by bounding the recursion depth. Arguing as in the previous case, we note that the expected number of elements at recursion depth $2 \log m$ is $1/m$; it follows from Markov's inequality that the recursion terminates at this depth with probability $1 - 1/m$. Thus the bound on verification time follows as in the previous case.

To bound the time spent in calls to C, note as above that the number of elements passed to C at one level of the recursion is upper bounded by a negative binomial distribution with mean $2r$. Therefore, the number of elements passed to C over *all* recursion levels is upper bounded (with high probability) by a negative binomial distribution with mean $O(r \log m)$ and is therefore $O(r \log m)$ with high probability in m. Recall from Lemma 2.4 that $C(n, r) = O(n \log n + nV(r + 1, r))$. Therefore, the total time spent on construction is upper bounded by $O(r(\log m) \log(r \log m) + V(5r^2 \log m, r)) = O(m + V(m, r))$ and can therefore be absorbed by the bound on verification time. □

We now give a generalization that holds without restriction on C and V.

Corollary 2.6. *An optimum basis can be found in time*

$$O\Big(\sum_{i=1}^{\log m/r} V(m_i + r + \log m, r) + C(3r, r)\log(m/r) + (\log^2 m)V(\log m, r)\Big)$$

with high probability (LV) for some m_i such that $\sum m_i = m$.

Proof. The first term is from the verification time analysis in the theorem; the second from the construction time for the case $r > \log m$, the third for the case $r \leq \log m$. □

Since there is, for example, a linear-time algorithm for verifying minimum spanning trees [DRT92] and a simple $O(m \log n)$ time algorithm for constructing minimum spanning trees [Kru56], the above lemma immediately yields an $O(m + n \log n \log(m/n))$-time algorithm for constructing minimum spanning trees in n-vertex, m-edge graphs. The nonlinear term reflects the need to solve $O(\log(m/n))$

"base cases" on n edges. Karger, Klein and Tarjan [KKT95] observed that this can be improved. By merging vertices connected by minimum spanning tree edges as those edges are discovered, we can reduce the rank of the graphic matroid in the recursive calls. This leads to a linear-time recursive algorithm for the minimum spanning tree problem. A similar technique does not seem to exist for the general matroid optimization problem.

In their survey paper [LR92], Lee and Ryan discuss several applications of matroid optimization. These include job sequencing and finding a minimum cycle basis of a graph. In the applications they discuss, attention has apparently been focused on the construction rather than the verification of the optimum basis. This work suggests that examining verification would be productive. In particular, applying Lemma 2.4 to the main theorem, we have:

Corollary 2.7. *Given a matroid verification algorithm with running time $V(m,r)$, an optimum basis can be found in $O(V(m,r) + r \log r \log(m/r) V(r+1,r))$ time.*

3 Packing Matroid Bases

We now turn to the *basis packing problem.* The objective is to find a maximum collection of pairwise disjoint bases in a matroid M. It is closely related to the problem of finding a basis in a *k-fold matroid sum for M,* which by definition is a matroid whose independent sets are the sets that can be partitioned into k independent sets of M. A rank r matroid contains k disjoint bases precisely when its k-fold sum has rank kr. An early algorithm for this problem was given by Edmonds [Edm65, Law76].

Definition 3.1. The *packing number* $P(M)$ for a matroid M is the maximum number of disjoint bases in it.

Many matroid algorithms access the matroid via calls to an *independence oracle* that tells whether a given set is independent in the matroid. The running time for such an oracle call can be quite large, but special implementations for, e.g., the graphic matroid [GW92] run quickly. For convenience, we will state running times in terms of the number of primitive operations plus the number of oracle calls performed. If we say that an algorithm runs in time T, we are actually saying that, given an oracle with running time T', the algorithm will run in time TT'.

Many matroid packing algorithms use a concept of augmenting paths: the algorithms repeatedly augment a collection of independent sets by a single element until they form the desired number of bases. The augmentation steps generally have running times dependent on the matroid size m, the matroid rank r, and k, the number of bases already found. For example, Knuth [Knu73] gives a "generic" augmenting path algorithm that finds an augmenting path in $O(mrk)$ calls to an independence oracle and can therefore find a set of k disjoint bases of total size rk (if they exist) in $O(mr^2k^2)$ "time." (Knuth claims an augmentation time of $m^2 > mrk$, but an examination of his algorithm's inner loop shows that each of the at most rk elements of the "current" packing is tested against each element of the matroid, yielding the $O(mrk)$ time bound.)

In the following sections, we give two techniques for improving the running times of basis packing algorithms. Random sampling lets us reduce or eliminate the running time dependence on k. A *greedy packing* technique lets us replace m by $kr \ln r \leq m \ln r$ (or in some cases by $kr \leq m$). Combining the two techniques yields algorithms whose running time depends almost entirely on r. We apply them to the problem of packing spanning trees in a graph, previously investigated by Gabow and Westermann [GW92] and Barahona [Bar95].

We believe that the paradigms developed here will be useful for a wide variety of packing problems. We have used similar ideas [Kar97a, BK96, Kar96] in fast new algorithms for maximum flows (packings of paths) and connectivity computations (using a tree packing algorithm developed by Gabow [Gab95]).

3.1 A Quotient Formulation

We first present some of the tools we will use in our development. An alternative characterization of a matroid's packing number is given in the following theorem of Edmonds [Edm65]. Recall (Section 1.5) that any subset A of M has a matroid structure induced by the independent sets of M contained in A. Let $\overline{A}$ denote $M - A$.

Theorem 3.2 (Edmonds). *A matroid on M with rank r has k disjoint bases if and only if for every $A \subseteq M$,*

$$k \operatorname{rk} A + \|\overline{A}\| \geq kr.$$

Recall the definitions of closures and quotients from Section 1.5. In particular, recall that a quotient is any complement of a closed set and that $\operatorname{rk}^C(M/A) = \operatorname{rk} M - \operatorname{rk} A$ is the rank of the quotient matroid M/A. Edmonds' Theorem can be rephrased in quotient language as follows:

Theorem 3.3. *$P(M) \geq k$ if and only if for every quotient Q of M,*

$$\|Q\| \geq k \operatorname{rk}^C Q. \tag{1}$$

Proof. The elements of each quotient Q are a subset of M (and each subset that is a complement of a closed set defines exactly one quotient). Since any independent set of the quotient is independent in M as well, we have $\operatorname{rk}^C Q \leq \operatorname{rk} Q$. Therefore, Edmonds' original formulation implies this one trivially by taking $A = \overline{Q}$. For the other direction, suppose the above statement holds and consider any set $A \subseteq M$. Let $B = \operatorname{cl} A$, so $\overline{B}$ is a quotient with $\operatorname{rk}^C \overline{B} = \operatorname{rk} M - \operatorname{rk} A$. Then

$$\begin{aligned} \|\overline{A}\| &\geq \|\overline{B}\| \\ &\geq k \operatorname{rk}^C \overline{B} \\ &= k(\operatorname{rk} M - \operatorname{rk} A) \end{aligned}$$

□

Motivated by the above theorem, we make the following definitions:

Definition 3.4. The *density* of a quotient Q,

$$\nu(Q) = \frac{\|Q\|}{\operatorname{rk}^C Q}.$$

Definition 3.5. A *sparsest quotient of M* is a quotient of minimum density.

Corollary 3.6. *$P(M)$ is equal to the density of a sparsest quotient, rounded down.*

As an example of quotients, consider the graphic matroid. Given a set of edges, its closure corresponds to the connected components of the edge set. An edge is in the closure if both endpoints are in the same connected component. The complement is the set of edges with endpoints in different components. Thus, it corresponds to a (multiway) *cut* of the graph. The size of the quotient is just the value of the cut. It is worth distinguishing the minimum density quotient from the minimum cut: the minimum density quotient can have many more edges than the minimum cut if it partitions the graph into many components.

4 Sampling for Packing

We now show that randomly sampling from the elements of a matroid "scales" the densities of all the quotients in the matroid. It will immediately follow that sampling also scales the packing number in a predictable fashion. We write $x \in (1 \pm \epsilon)y$ when we mean $(1 - \epsilon)y \leq x \leq (1 + \epsilon)y$.

Theorem 4.1. *Let $P(M) = k$. Suppose each element of M is sampled with probability $p \geq 18(\ln m)/k\epsilon^2$, yielding a matroid $M(p)$. Then with high probability in m, $P(M(p)) \in (1 \pm \epsilon)pk$.*

Clearly this theorem is only interesting when $P(m) = \Omega(\ln m)$, since obviously we want $p < 1$ and $\epsilon < 1$. We will prove it in Section 4.2. A second proof is given in the appendix.

This theorem gives an obvious scheme for approximating $P(M)$: find $P(M(p))$ for some small sampling probability p. We elaborate on this approach, extending it to exactly compute the optimum packing, in Section 4.3.

To prove the theorem, we consider a correspondence between the quotients in M and the quotients in a random sample $M(p)$. We use it to prove that no quotient of $M(p)$ has low density. The result then follows from Corollary 3.6. We now give this correspondence.

Consider a quotient Q of $M(p)$. Let $A = M(p) - Q$. It is easy to verify that $Q = (M/A) \cap M(p) = (M/A)(p)$. In other words, every quotient in $M(p)$ is the sampled part of a *corresponding quotient* of M. In Lemma 4.6 we show that with high probability every quotient in $M(p)$ contains $(1 \pm \epsilon)p$ times as many elements as its corresponding quotient in M. It will follow that

- Every quotient of $M(p)$ has density at least $(1 - \epsilon)pk$.
- The minimum-density quotient in M (with density k) induces a quotient of density at most $(1 + \epsilon)pk$ in $M(p)$.

Combining these facts with Corollary 3.6 will prove the theorem.

4.1 Related Results

Before proving the main theorem, we note a few related results.

4.1.1 Existence of a Basis

The above theorem proves that the number of bases is predictable. We might settle for knowing that at least one basis gets sampled.

Theorem 4.2. *Suppose M contains $a + 2 + k \ln r$ disjoint bases. Then $M(1/k)$ contains a basis for M with probability $1 - O(e^{-a/k})$.*

A proof of this theorem is given in the appendix. Note that Polesskii [Pol90] proved the same fact by giving an explicit construction of the "least reliable" (least likely to contain a basis) matroid containing k disjoint bases, for any k.

This theorem is in a sense orthogonal to Theorem 2.2. That theorem shows that regardless of the number of bases, few elements are likely to be independent of the sample. However, it does not prove that the sample contains a basis. This theorem shows that if the number of bases is large enough, no elements will be independent of the sample (because the sample will contain a basis).

Note also that the sampling probability in the theorem is significantly less than that used in our main theorem. At these lower sampling probabilities, we can guarantee one basis, but the number of bases appears to have a large variance. This conforms to our intuition about simpler sampling experiments. When the "expected number of bases" pk is $\ln r$, we expect to find at least one but cannot predict how many. When pk becomes slightly larger (about $\ln m$), we begin to see tight concentration about the mean.

4.1.2 The Graphic Matroid

We also remark on some particular restrictions to the graphic matroid. Erdös and Renyi [ER59, Bol85], proved that if a random graph on n vertices is generated by including each edge independently with probability exceeding $(\ln n)/n$, then the graph is likely to be connected. Rephrased in the language of graphic matroids, if the edges of a complete graph on n vertices are sampled with the given probability, then the sampled edges are likely to contain a spanning tree, *i.e.* a basis. Theorem 4.2 says the same thing, though with weaker constants.

In [Kar97a] we prove the following result analogous to Theorem 4.1 for graphs:

Theorem 4.3. *If a graph G has minimum cut k, then (with p and ϵ as in Theorem 4.1) every cut in $G(p)$ has value $p(1 \pm \epsilon)$ times its value on the original graphs.*

Corollary 4.4. *If a graph G has minimum cut k, then (with p and ϵ as in Theorem 4.1) $G(p)$ has minimum cut $(1 \pm \epsilon)pk$.*

A Theorem of Nash-Williams [NW61] gives a connection between minimum cuts and spanning tree packings—one closely tied to Edmonds' Theorem for matroids. Our sampling theorems for matroids generalize our sampling theorems for graphs. Just as our matroid sampling theorem leads to algorithms for approximating and finding basis packings, the graph sampling theorem on cuts leads to fast algorithms for approximating and exactly finding minimum cuts and s-t minimum cuts and maximum flows [Kar97a].

4.2 Proof of Theorem 4.1

This section is devoted to proving Theorem 4.1. The reader interested only in its applications can skip to the next section. An alternative "constructive" proof is given in the appendix. We actually prove the theorem for a slightly smaller sampling probability $p = 9(\ln mr)/k\epsilon^2 < 18(\ln m)/k\epsilon^2$ since $r \leq m$. It will be clear that using the larger "correct" p can only increase our success probability. As discussed above, we aim to prove that every quotient in $M(p)$ has roughly p times the density of its corresponding quotient in M. To do so, we prove that every quotient has *size* roughly p times that of its corresponding quotient, and then note that (by Theorem 4.2) the rank of $M(p)$ is the same as the rank of M. Density scaling then follows from the definitions.

We first prove that the number of elements sampled from every quotient in M is near its expectation with high probability. For any one quotient, this follows immediately from the Chernoff bound. However, a matroid can have exponentially many quotients, implying that even the exponentially unlikely event of a Chernoff bound violation might occur. To surmount this problem, we show that almost all quotients in a matroid are large. Since the likelihood of deviation from expectation decays exponentially with the size of the quotient, this suffices to prove our result.

Lemma 4.5. *For integer t, the number of quotients of size less than tk is at most $(mr)^t$.*

Proof. First suppose $t > r$. Each quotient corresponds to a closed set, which is the closure of some independent set that spans it, so there is a surjective mapping from independent sets to closures to quotients. Thus the number of quotients is at most the number of independent sets. There are at most $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{r} \leq m^r$ such independent sets, proving the lemma for $t \geq r$.

Now assume $t < r$. We use a proof that essentially generalizes the Contraction Algorithm of [KS96]. We give an algorithm for randomly choosing a small quotient, and show that every small quotient has a large chance of being chosen. This proves that there are only a few such quotients. Consider the following randomized algorithm for selecting a quotient of M:

Let $M_r = M$
For $i = r$ **downto** $t + 1$
    Let x_i be an element selected uniformly at random from M_i
    Let $M_{i-1} = M_i/x_i$
Pick a quotient at uniformly at random from M_t

Consider a particular quotient Q of size less than tk for some real t. We determine the probability this quotient is found by the algorithm. We first find the probability that this quotient is contained in M_t (*i.e.*, contains only elements independent of $x_r, \ldots, x_{t+1}$). Consider the selection of x_r. By hypothesis, Q has size at most tk. However, since M_r has k disjoint bases of rank r, it has at least rk elements. Therefore, $\Pr[x_r \in Q] \leq tk/rk = t/r$. Thus $\Pr[x_r \notin Q] \geq 1 - t/r$. However, if $x_r \notin Q$ then, since Q is a quotient, we know that $Q \subseteq M_{r-1} = M_r/x_r$. Now note that M_{r-1} still has k disjoint bases (the quotients by x_r of the bases in M_r) but has rank $r - 1$. Thus $\Pr[x_{r-1} \in Q \mid x_r \notin Q] = 1 - t/(r-1)$. Continuing, we find that

$$\begin{aligned}\Pr[Q \subseteq M_t] &\geq (1-\frac{t}{r})(1-\frac{t}{r-1})\cdots(1-\frac{t}{t+1}) \\ &\geq \frac{r-t}{r}\cdot\frac{r-t-1}{r-1}\cdots\frac{1}{t+1} \\ &= \binom{r}{t}^{-1} \\ &\geq r^{-t}\end{aligned}$$

Now suppose the we select a quotient uniformly at random from M_t. The number of quotients of M_t is at most the number of independent sets of M_t. But an independent set in M_t can have at most t elements in it (since M_t has rank t). Thus the number of quotients is at most $\sum_{i\leq t}\binom{m}{i} \leq m^{-t}$, so the probability that we pick our particular quotient, given that it is a subset of M_t, is at least m^{-t}.

Combining the above two arguments, we find that the probability of selecting a particular quotient of size tk is at least $(mr)^{-t}$. Therefore, since each quotient of size less than kt has at least this probability of being chosen, and the events of these quotients being chosen are disjoint, we know that the number of such quotients is at most $(mr)^t$. □

Lemma 4.6. *With high probability, for all quotients Q of M we have $\|Q(p)\| \in (1\pm\epsilon)p\|Q\|$.*

Proof. Consider a particular quotient Q of size q in M. The probability that $Q(p)$ has size outside the range $(1\pm\epsilon)pq$ is, by the Chernoff bound, at most $2e^{-\epsilon^2 pq/3} = 2(mr)^{-3q/k}$.

Let $q_1, q_2, \ldots$ be the sizes of M's quotients in increasing order so that $k \leq q_1 \leq q_2, \cdots$. Let p_j be the probability that the j^{th} quotient diverges by more than ϵ from its expected size. Then the probability that some quotient diverges by more than ϵ is at most $\sum p_j$, which we proceed to bound from above.

We have just argued that $p_j \leq 2(mr)^{-3q_j/k}$. According to Lemma 4.5, for integer t there are at most $(mr)^t$ quotients of size less than tk. We now proceed in two steps. First, consider the mr smallest quotients. Each of them has $q_j \geq k$ and thus $p_j \leq 2(mr)^{-3}$, so that

$$\sum_{j\leq mr} p_j \leq 2(mr)(mr)^{-3} = O((mr)^{-2}).$$

Next, consider the remaining larger quotients. Since we have numbered the quotients in increasing order, this means that quotient $q_{(mr)^t+1}, \ldots, q_{(mr)^{t+1}}$ all have size at least tk. Thus $p_{(mr)^t}, \ldots, p_{(mr)^{t+1}} \leq 2(mr)^{-3t}$ It follows that

$$\begin{aligned}\sum_{j>mr} p_j &\leq 2\sum_{t\geq 1}[(mr)^{t+1} - (mr)^t](mr)^{-3t} \\ &\leq 2\sum_{t\geq 1}(mr)^{1-2t} \\ &= O((mr)^{-1}).\end{aligned}$$

□

Lemma 4.7. *With high probability* $\mathrm{rk}(M(p)) = \mathrm{rk}M$.

Proof. This follows from Theorem 4.2. For a direct proof, suppose that $\mathrm{rk}(M(p)) < \mathrm{rk}M$. This implies that $M(p)$ does not span M, meaning that $Q = M/M(p)$ is a nonempty quotient. Edmonds' Theorem implies that in fact such a quotient has at least k elements. From Lemma 4.6 we deduce that $Q(p)$ is nonempty. But from the definition of Q we know that $Q \cap M(p) = \emptyset$, a contradiction. □

Remark. Theorem 4.2 proves that a significantly smaller value of p than that used here is sufficient to ensure full rank.

Lemma 4.8. *With high probability, the density of every quotient in $M(p)$ is $p(1 \pm \epsilon)$ times the density of the corresponding quotient in M.*

Proof. Consider a quotient $M(p)/A$ where $A \subseteq M(p)$. As discussed at the start of the section, it has the form $Q(p)$ for the quotient $Q = M/A$. By Lemma 4.6, we know that $\|Q(p)\| \in (1 \pm \epsilon)p\|Q\|$. At the same time, we know from Lemma 4.7 that $\mathrm{rk}(M(p)) = \mathrm{rk}M$. It follows that the density $\nu(Q(p))$ satisfies

$$\begin{aligned} \nu(Q(p)) &= \frac{\|Q(p)\|}{\mathrm{rk}(M(p)) - \mathrm{rk}A} \\ &\in \frac{(1 \pm \epsilon)p\|Q\|}{\mathrm{rk}(M) - \mathrm{rk}A} \\ &= (1 \pm \epsilon)p\,\nu(Q) \end{aligned}$$

□

Corollary 4.9. *With high probability $P(M(p)) \geq (1 - \epsilon)p \cdot P(M)$*

Proof. Every quotient in $M(p)$ corresponds to a quotient of density at least $P(M)$ in M. □

This gives the lower bound; to prove the upper bound, we just need to show that some quotient in $M(p)$ corresponds to the minimum density quotient of M.

Lemma 4.10. *With high probability $P(M(p)) \leq (1 + \epsilon)p \cdot P(M)$*

Proof. Let $Q = M - A$ be the quotient of minimum density in M. Then consider $Q(p)$ and $A(p)$. Clearly $\mathrm{rk}(A(p)) \leq \mathrm{rk}A$; thus by Lemma 4.7 (or Theorem 4.2) we have $\mathrm{rk}^C(M(p)/A(p)) \geq \mathrm{rk}^C(M/A)$. By the Chernoff bound we have $\|Q(p)\| \leq (1 + \epsilon)\|Q\|$. It follows that

$$\begin{aligned} \nu(Q(p)) &= \frac{\|Q(p\|}{\mathrm{rk}^C(M(p)/A(p))} \\ &\leq \frac{(1 + \epsilon)p\|Q\|}{\mathrm{rk}^C(M/A)} \\ &= (1 + \epsilon)p\,\nu(Q). \end{aligned}$$

□

This completes the proof of Theorem 4.1. We have shown that with high probability every quotient is dense, so $P(M(p)) \geq (1 - \epsilon)k$, and that some quotient is sparse, so that $P(M(p)) \leq (1 + \epsilon)k$.

4.2.1 Capacitated Matroids

In some cases (such as the tree packing problem will will address later) it is natural to assign a *capacity* to each element of the matroid. A basis packing is now one that uses each element no more times than the capacity of that element. We can even allow fractional capacities and talk about the maximum fractional packing in which each basis is given a weight and the total weight of bases using an element can be no more than the capacity of the element.

We would like to apply Theorem 4.1 to capacitated matroids. One approach is to represent an element of capacity c as a set of c identical uncapacitated matroid elements and apply the theorem directly. This has the drawback of putting into the error bound of Theorem 4.1 a term dependent

on the total capacity of matroid elements. In fact, Theorem 4.1 holds for capacitated matroids even when m refers to the number of distinct matroid elements rather than their total capacity.

To see this, note that the m term in the error bound arose from only one place: the $O((mr)^t)$ bound on the number of quotients of size less than tk, which we would like to turn into a bound on the number of quotients of total capacity less than tk. The factor of m in this bound arose in turn from the m^r bound on the number of quotients in an m-element, rank-r matroid, which is at most the number of independent sets in the matroid. But now observe that two independent sets with different copies of the same elements yield the same quotient. This means that replicating elements does not increase the number of independent sets or quotients. Thus the number of independent sets (and quotients) of capacity tk is bounded by $(mr)^t$, where m is the number of *distinct* elements of M, regardless of their capacity.

Corollary 4.11. *Let M be a capacitated matroid with m elements and with $P(M) = k$. Suppose each copy of a capacitated element is sampled with probability $p \geq 18(\ln m)/k\epsilon^2$, yielding a matroid $M(p)$. Then with high probability in m, $P(M(p)) \in (1 \pm \epsilon)pk$.*

To apply this argument to fractional capacities, consider multiplying each capacity by a large integer T, rounding down to the nearest integer, and applying the previous theorem. In the limit as T becomes large, we reduce to the integer capacity case. Note that multiplying by T also scales the packing number by T, which scales the sampling probability of any one element *down* by a factor of T. The limiting distribution on the number of copies sampled from a given capacitated element becomes a Poisson process.

Corollary 4.12. *Suppose M has fractional packing number k. Build a new matroid M' by sampling from each capacity w element of M according to a Poisson process with mean pw for $p \geq 18(\ln m)/k\epsilon^2$. Then with high probability, $P(M') \in (1 \pm \epsilon)pk$.*

Further details of this approach to sampling capacitated elements can be found in [Kar97a].

4.3 Packing by Sampling

Having developed approximation theorems for basis packing, we now use sampling with refinement to develop algorithms for exact and approximate basis packing. We first note an obvious approximation scheme. To approximate the packing number, we can sample matroid elements with some probability p and compute the packing number of $M(p)$ using Knuth's algorithm. By Theorem 4.1, $P(M(p))$ will be roughly pk, so Knuth's algorithm will run faster (by a factor of p^2) on the sample. Dividing the sample's packing number by p will give an estimate for $P(M)$. Before giving the details of this scheme, we present a speedup of Knuth's algorithm for finding an exact packing. Using this exact algorithm instead of Knuth's will yield a faster approximation algorithm based on the above scheme.

Theorem 4.13. *A maximum packing of k matroid bases in an m element, rank r matroid can be found with high probability in $O(mr^2k^{3/2}\sqrt{\log m})$ time (LV).*

Recall that the suffix (LV) denotes a Las Vegas algorithm. This result "accelerates" Knuth's algorithm [Knu73] by a factor of $\sqrt{k/\log m}$.

Proof. Start by running Knuth's algorithm to find out if the packing number is less than $\log m$—return the packing if so. Otherwise, randomly partition the matroid elements into 2 groups by flipping an unbiased coin to decide which group each element goes in. Find a maximum basis packing in each of the two groups recursively. Combining these two packings gives a (not necessarily maximum) packing of bases in M. Use Knuth's algorithm to augment this packing to a maximum one.

The algorithm is clearly correct; we now analyze its running time. First suppose that the matroid contains $k \leq \log m$ bases. Then the first test takes $O(mr^2k^2) = O(mr^2k^{3/2}\sqrt{\log m})$ time and we then halt (and our proof is done for this case). Otherwise, the first test takes $O(mr^2 \log^2 m) = O(mr^2k^{3/2}\sqrt{\log m})$ time and we then go on to the (recursive) phase. If the recursive phase executes, we can assume $k \geq \log m$.

We begin with an intuitive argument. Consider a particular node in the recursion tree. Each of the two submatroids we construct looks like a random sample with $p = 1/2$, though the two groups are not independent. We can apply Theorem 4.1 with $p = 1/2$ to each submatroid; solving for ϵ yields $\epsilon = \sqrt{36(\ln m)/k}$, implying that with high probability in m each group contains $k/2 - O(\sqrt{k \log m})$ disjoint bases. Thus, Knuth's algorithm needs to augment the union of the recursive calls by an additional $O(r\sqrt{k \log m})$ elements, which takes $O(mr^2 k\sqrt{k \log m})$ time. This gives a recurrence in which the final cleanup phase dominates the running time.

Unfortunately there is a flaw in this argument: the many leaves in the recursion tree are small and might fail to obey the high probability arguments. We therefore unroll the recursion to prove the theorem. We add up the work (cleanup augmentations) performed at each *node* of the recursion tree. We separately bound two different kinds of work at a given node: *useful* augmentations are augmentations that eventually contribute to an increase in the number of bases returned; *useless* augmentations are those last few which do not so contribute. At any given node there can be at most $r - 1$ useless augmentations since a sequence of r augmentations adds a basis.

We first analyze the shape of the recursion tree. Note that each node at depth d of the recursion tree contains a random subset of M in which each element appears with probability $1/2^d$. It follows that at depth $d > 3 \log m$, every node has at most 1 element with high probability (this can be seen by considering throwing the m elements randomly into the m^3 recursion nodes at this depth). This means the recursion depth is $O(\log m)$ and thus that the number of recursion nodes is polynomial in m (so that events that are high probability in m can be assumed to happen at every node).

Now we separately analyze "shallow" and "deep" recursion tree nodes. First consider the "shallow" nodes at depth d with $2^d = O(k/\log m)$. Applying Theorem 4.1 with $p = 2^{-d}$ to such a node, we deduce that with high probability in m, any such node contains $(1 \pm \epsilon_d)k/2^d$ bases, where $\epsilon_d = O(\sqrt{2^d(\log m)/k} < 1$. In other words, the node contains $k/2^d \pm O(\sqrt{k(\log m)/2^d})$ bases. Since the number of nodes in the recursion tree is $m^{O(1)}$ with high probability, this argument holds with high probability at every (sufficiently shallow) node.

It follows that after merging the bases from its two children, each shallow node at depth d must augment at most

$$\left(k/2^d + \sqrt{36k(\ln m)/2^d}\right) - 2 \cdot \left(k/2^{d+1} - \sqrt{36k(\ln m)/2^{d+1}}\right) = O\left(\sqrt{k(\log m)/2^d}\right)$$

additional bases bases to finish. Since with high probability in m, each of the nodes has $O(m/2^d)$ elements, this takes Knuth's augmentation algorithm

$$O\left(\frac{m}{2^d} r^2 \sqrt{k(\log m)/2^d}\right)$$

time. Since there are 2^d nodes at level d, the total time spent at level d is

$$O\left(mr^2\sqrt{k(\log m)/2^d}\right).$$

To find the time spent at all relevant levels, we sum the times at each. This is a geometric series summing to

$$O\left(mr^2\sqrt{k(\log m)}\right).$$

The above argument bounded the number of useful augmentations. It can also cover the time spent in useless augmentations. Since every recursion node is assumed in the above analysis to augment by at least one basis, we can charge the r useless augmentations to the last r useful ones.

It remains to bound the time at "deep" levels with $2^d = \Omega(\log(k/\log m)))$. To bound useful augmentations, note that each node at this level or below has $O(m(\log m)/k)$ elements and $O(\log m)$ bases with high probability. Furthermore, among all the nodes at or below this level there can be

at most k bases added, since each added base contributes to the total output of bases. Therefore, the total cost of adding bases below this level is

$$\begin{aligned} O(k \cdot mr^2(\log^2 m)/k) &= O(mr^2 \log^2 m) \\ &= O(mr^2 k^{3/2}\sqrt{(\log m)}) \end{aligned}$$

Finally we must account for the useless augmentations. Consider a node with m_i elements. It can perform at most r useless augmentations, each costing at most $O(m_i r \log m)$ time. Thus the time spent at a given level is $O(\sum m_i r^2 \log m) = O(mr^2 \log m)$. Thus the total time spent over all $O(\log m)$ deep levels is $(mr^2 \log^2 m)$, which is negligible since $k = \Omega(\log m)$. □

We now use this faster exact packing algorithm to implement a faster approximation algorithm.

Theorem 4.14. *For any ϵ, the packing number k of an m-element matroid of rank r can be estimated to within $(1 \pm \epsilon)$ in $O(mr^2(\log^3 m)/k\epsilon^5)$ time (MC).*

Proof. Let $\hat{z} = (18 \ln m)/\epsilon^2$, and consider sampling with probability $\hat{p} = \hat{z}/k$. Theorem 4.1 tells us that with high probability, $M(\hat{p})$ has packing number $z \in (1 \pm \epsilon)\hat{z}$. Thus if we compute z in the sampled matroid, $z/\hat{p}$ provides an estimate of k with the desired accuracy.

As described the algorithm requires a knowledge of k. But this is easy to simulate with a repeated doubling scheme. Start with $p = 1/m < \hat{p}$. Repeatedly double p and pack $M(p)$ until $M(p)$ has packing number $z \geq 2\hat{z}$. The probability that this happens when $p < \hat{p}$ is negligible. To see this, consider the following scheme for sampling elements with probability $p < \hat{p}$. First, tentatively sample each element with probability $\hat{p}$. Second, sample each tentatively sampled element with probability $p/\hat{p} < 1$. As required, each element gets sampled with probability p. But consider the tentatively sampled elements. By Theorem 4.1, the packing number of the tentative sample $M(\hat{p})$ is at most $(1+\epsilon)\hat{z} \leq z$. Sampling a subset of these elements clearly cannot increase the packing number, so the claim holds. It follows that with high probability, when the doubling process terminates we have $p \geq \hat{p}$, so $z/p \in (1 \pm \epsilon)k$.

To determine the running time, note that at trial i of doubling p we are working with a matroid that (with high probability) has size $O(pm)$ and packing number $O(\hat{z})$. Thus the time to find its optimum packing is $O(pmr^2\hat{z}^{3/2}\sqrt{\log m})$ by Theorem 4.13. We stop before $p > 2\hat{z}/k$. Thus the work over all trials is a geometric series summing to $O(mr^2\hat{z}^{5/2}\sqrt{\log m}/k)$.

As a technical detail, note that if $\hat{p}$ is much too small, we might get a sample with smaller rank than the original matroid. Bases in this sample will not be bases of M, and the sampling theorem will not apply. But this case is easy to avoid. To start with, we compute the rank of M in $O(m)$ time by greedily constructing a basis. Before packing a sample, we make sure that it has the same rank (taking time linear in the sample size). The total work is negligible.

As another technical detail, note that if $\hat{p} > 1$, we just run Knuth's algorithm on the original matroid. Since $\hat{p} > 1$ implies $\hat{z} > k$, meaning $k = O(\epsilon^{-2} \ln m)$, the original algorithm's running time of $O(mr^2k^2)$ is better than the one we claim. Thus the claimed bound is certainly achievable. □

Corollary 4.15. *In the same time bounds as above, we can identify a quotient of M with density at most $(1+\epsilon)$ times the minimum.*

Proof. The above algorithm will also identify a sparse quotient in the sample. This quotient, by Theorem 4.1, corresponds to a sparse quotient in the original matroid. □

Theorem 4.16. *A packing of $(1-\epsilon)k$ disjoint bases can be found in $O(mr^2(\log^2 m)/\epsilon^3)$ time (LV).*

Proof. After approximating k with the previous algorithm, partition the elements of M randomly into $1/p = \epsilon^2 k/18 \ln m$ groups. Each looks like a sample of the kind discussed in Theorem 4.16 and

therefore contains $pk(1-\epsilon)$ bases. Run the basis packing algorithm on each group and take the union of the results. The running time is the time for $1/p$ executions of the previous algorithm.

The algorithm as stated is Monte Carlo in that there is a small probability that the various groups will not contain $(1-\epsilon)k$ bases among them. But if we combine with the previous algorithm, we can guarantee the correctness of the solution. By Corollary 4.15, we can find a sparse quotient in the original graph; this provides an upper bound on the value of the packing number. By Theorem 4.16, we can find a nearly maximum packing, thus lower bounding the packing number. So long as the upper and lower bounds disagree by more than $(1+\epsilon)/(1-\epsilon)$, we can rerun both algorithms. Once they agree this closely we will have pinned down the correct value to within ϵ. □

5 Greedy Packings

The sampling techniques we have just described let us effectively reduce the dependence of a packing algorithm on the packing number k, partially replacing k with $O(\log m)$. We now give a different, deterministic scheme that lets us reduce the dependencies of these algorithms on the matroid size m. We effectively replace m by rk for basis packing algorithms. In particular, we give a deterministic basis packing algorithm with running time

$$O(m + (rk)^3).$$

Sampling then lets us nearly eliminate the newly introduced factor of k, so we are left only with a dependence on the rank r, the error parameter ϵ, and $\log k$.

The idea behind this approach is a connection between a *maximum* packing and a *maximal* packing. Maximal packings are much easier to construct, but can be made to contain as many bases as the maximum packing. The maximum packing algorithms can then extract those bases from the maximal packing instead of from the entire matroid. This approach was explored for *graphs* by Nagamochi and Ibaraki [NI92], who showed how a maximal packing of forests could be used in minimum cut computations.

Definition 5.1. A *greedy packing* of M is a partition of the elements of M into independent sets B_i, $i > 0$ such that each B_j is a basis of $M - \cup_{i<j} B_i$. The "prefix" $B_1, \ldots, B_t$ is a *greedy t-packing.*

A greedy t-packing has at most tr elements and may therefore be smaller than the original matroid. However, it still contains a large number of disjoint bases:

Lemma 5.2. *Let S be a greedy $k(1+\ln r)$-packing of M. If $P(M) \geq k$, then $P(S) \geq k$.*

Proof. Let $S = \{B_i\}$ be the greedy packing. We show that for every quotient Q of S, its density $\nu Q \geq k$. The result then follows from the quotient version of Edmonds' Theorem. We use the same correspondence as we did for the random sampling proofs. A quotient S/A has the same elements as $S \cap (M/A)$. Since S contains a basis B_1 of M, we know that $\mathrm{rk}^C(S/A) = \mathrm{rk}^C(M/A)$. Therefore, to show the density claim, we need only prove that the necessary number of elements is selected in S from each quotient $Q \subseteq M$.

We begin with $Q = M$, and show that $\|S\| = \sum \|B_i\| \geq kr$. Write $d_i = kr - \sum_{j \leq i} \|B_j\|$; d_i is the *deficit* in the greedy packing after we have added i sets to it. Once the deficit reaches 0, the greedy packing contains kr elements. We derive a recurrence for the d_i. After removing B_i, we have removed a total of $kr - d_i$ elements. By Edmonds' Theorem (taking $\overline{A}$ in Theorem 3.2 to be the set of removed elements), the rank of the remaining elements is at least d_i/k. Therefore, $\|B_{i+1}\| \geq d_i/k$, meaning that $d_{i+1} \leq d_i - d_i/k$. In other words, we have the following recurrence:

$$\begin{aligned} d_0 &= kr \\ d_{i+1} &\leq (1-1/k)d_i \end{aligned}$$

It follows that $d_i \leq (1-1/k)^i kr$, meaning that $d_{k \ln r} \leq k$. Now consider the next k greedy bases. Each of them has size at least 1, unless we have exhausted all the elements of M. Therefore, after adding these k bases, we will either have kr elements, or else $\|M\|$ elements. But $\|M\| \geq kr$, which proves the claim for $Q = M$.

Now consider any other quotient $Q \subset M$. We know that Q contains k disjoint bases (contained in the quotients of the k disjoint bases of M). We therefore aim to apply the previous argument to the matroid Q. To do so, we show that any greedy t-packing $\{B_i\}$ constructed in M contains a greedy t-packing of Q. This is almost trivial, as it is immediate that for all $i < j$, $B_i \cap Q$ spans $B_j \cap Q$. The only detail to fix is that possibly $B_i \cap Q$ is not independent in Q, meaning that it is not a basis.

We fix this as follows. Begin with $B_1 \cap Q$ which certainly spans Q. If this set is not independent in Q, we remove spanned elements from it, one at a time, until the set becomes independent. For each element e that we remove, we find the smallest i such that $B_i \cap Q$ does not span e, and place e in B_i. This preserves the invariant that $B_i \cap Q$ spans $B_j \cap Q$ whenever $i \leq j$. Eventually, we will have made $B_1 \cap Q$ independent while preserving the invariant; we now move on to do the same with the remaining B_i. When we finish, we have a greedy packing of Q.

This construction shows that our greedy packing of M contains a greedy packing of every quotient of M. But every quotient of M is itself a matroid, and we started by proving that a greedy packing of such a matroid contains the claimed number of elements. Thus, the greedy packing of M contains the required numbers of elements of every quotient of M. □

Corollary 5.3. *A greedy $k \ln(1/\epsilon)$-packing of M contains $k(1-\epsilon)$ disjoint bases.*

Proof. Follows from the value of $d_{k \ln(1/\epsilon)}$ in the previous proof. □

5.1 Using Greedy Packings—First Attempt

The most obvious way to build a greedy packing is greedily:

Lemma 5.4. *A greedy t-packing can be constructed in $O(tm)$ time. A greedy packing can be found in $O(m^2)$ time.*

Proof. Construct one B_i at a time by scanning all as-yet unused elements of M. Constructing a basis requires one independence test per (unused) element. Each independent set we delete contains at least one element, so we use at most m bases. □

Lemma 5.5. *Given a greedy $k(1+\ln r)$-packing of M, a maximum packing of k matroid bases can be found in $O((kr)^3 \log r)$ time.*

Proof. Run Knuth's [Knu73] algorithm, which takes mr^2k^2 time on an m element matroid, on the $O(rk \log r)$ elements of the greedy packing. □

Corollary 5.6. *A maximum packing of k bases can be found in $O(km \log r + (rk)^3 \log r)$ time.*

Proof. We need only determine k. To do so, use repeated doubling. Start by guessing $\hat{k} = 1$, find a greedy $\hat{k}(1+\ln r)$-packing in $O(\hat{k}m \log r)$ time, and use the previous lemma. If we find less than $\hat{k}$ bases in the maximum packing, we know that the maximum packing has value $k < \hat{k}$. It follows that our greedy packing contains k disjoint bases, and that the packing we found in it is a maximum packing. If, on the other hand, we find more than $\hat{k}$ bases, we double $\hat{k}$ and try again.

Clearly we will stop at some value of $\hat{k}$ with $k \leq \hat{k} \leq 2k$. The time spent is a geometric sum dominated by that spent on the final guess for $\hat{k}$, which is at most $2k$. The claimed time bound follows. □

5.2 Faster Greedy Packing

To improve the previous algorithm, we perform a more efficient construction of greedy t-packings. Instead of building each basis in turn, we grow all of the bases at once: start with all B_i empty, consider each element of M in turn, and add it to the first B_i (with the smallest index i) that does not span the element. This produces exactly the same greedy packing as the naive algorithm. Checking each B_i in turn could take t independence tests, yielding the same time bound as before. But we can do better by observing the following:

Lemma 5.7. *For any $i < j$, at all times during the new construction, B_i spans all elements of B_j.*

Proof. Since an element is added to the first B_j that does not span it, all B_i with $i < j$ must span the new element. The claim follows by induction. □

Lemma 5.8. *During the new construction, when we consider adding new element x, there exists a j for which x is spanned by $B_1, \ldots, B_j$ but independent of $B_{j+1}, \ldots, B_t$.*

Proof. Let j be the largest index for which B_j spans x. For any $i < j$, we have just argued that B_i spans each element of B_j. It is an elementary fact of matroid theory that if B_i spans B_j and B_j spans x, then B_i spans x. Thus, B_i spans x for every $i \leq j$, but (by choice of j) the bases $B_{j+1}, \ldots, B_t$ do not span x. □

Corollary 5.9. *A greedy t-packing can be constructed in $O(m + rt \log t)$ time. A greedy packing can be constructed in $O(m \log m)$ time.*

Proof. As we add each element x to the greedy packing, as a first step, we check whether B_t spans x. If so, we can immediately discard x since it cannot be in any B_i. Otherwise, we can perform a binary search on $B_1, \ldots, B_t$ to find the largest j for which B_j spans x, and add x to B_{j+1}. The binary search works because if we test B_i and it spans x, then $j \geq i$, otherwise $j < i$. We need to perform $\log t$ independence tests for this binary search. We perform at most tr binary searches since each adds one element to the greedy packing. For each element that is not added to the t-packing, we perform only the one independence test against B_t. □

5.2.1 Using the Improved Algorithm

Corollary 5.10. *A maximum packing of k matroid bases can be found deterministically in $O(m \log k + (rk)^3 \log r)$ time.*

Proof. If we knew k, we could compute the greedy $k(1 + \ln r)$-packing we need and run Knuth's algorithm on it. The greedy packing time is $O(m + r(k \log r) \log(k \log r)) = O(m + rk \log^{O(1)} rk)$. Of course, k is not known. But this is easy to rectify by repeatedly doubling our guess for k until we fail to pack a number of bases exceeding our guess. At this point we will clearly have built a large-enough greedy packing to find the maximum.

Overall the running time is dominated by the time to run the algorithm with the final guess, which will be at most $2k$ and yield the claimed time bound. There is one exception: the $O(m)$ term in the time for a greedy packing. Since we recompute a greedy packing each time our estimate doubles, we compute $O(\log k)$ greedy packings and pick up an $O(m \log k)$ term in our running time. □

5.2.2 Another Small Improvement

We can make one more minor improvement. The $O(m \log k)$ term in the above time bounds arises because we perform $\log k$ greedy packing constructions to reach the correct value of k for our greedy packing. We now discuss a scheme for reducing this term to $O(m)$. To start with, we set

$$t = \frac{m}{r \log m}$$

and construct a greedy t-packing in $O(rt \log t) = O(m)$ time. This provides us with free access to greedy t'-packings in the algorithm of Corollary 5.10 for all $t' < t$. If our algorithm terminates before needing a larger greedy packing (which happens so long as $2k(1 + \ln r) < t$), all is well. Otherwise, $t = O(k \ln r)$. We argue that in this case the $O(m \log k)$ time bound is dominated by the $O((rk)^3)$ term in our running time above and can therefore be ignored. For $t = O(k \ln r)$ implies that

$$\begin{aligned} \frac{m}{r \log m} &= O(k \log r) \\ \frac{m}{\log m} &= O(kr \log r) \end{aligned}$$

Now it is straightforward that $m/\log m = O(z)$ implies $m = O(z \log z)$. In other words,

$$\begin{aligned} m &= O((kr \log r) \log(kr \log r)) \\ m \log k &= O((kr \log r) \log(kr \log r) \log k) \\ &= kr \log^{O(1)} kr. \end{aligned}$$

Corollary 5.11. *A maximum packing can be found in $O(m + (rk)^3 \log r)$ time. The total time spent constructing greedy packings is $O(m + rk \log^{O(1)} rk)$.*

5.3 Interleaving Greedy Packings with Augmentation

We now show that if we interleave the greedy packing process with the augmentations, we can shave an additional $\log r$ factor from the running time. This small improvement is not used later and can be skipped. Our first step in removing the $\log r$ factor will temporarily introduce an extra factor of k in our running time, but we then show how to get rid of that.

Lemma 5.12. *Let $S \subseteq M$ contain $b < k - 1$ disjoint bases, and let T be a greedy $\frac{k}{k-b-1}$-packing of $M - S$. Then $S \cup T$ contains $b + 1$ disjoint bases of M. If $b = k - 1$, then a greedy $(k(1 + \ln r/k))$-packing of T achieves the same goal.*

Proof. As in the initial analysis of greedy packings, we show that every quotient in $S \cup T$ has density at least $b+1$. Consider a quotient Q with $\mathrm{rk}^C Q = q$. We need to show that $\|Q \cap (S \cup T)\| \geq q(b+1)$. As we construct a greedy packing of $M - S$, we apply Edmonds' Theorem as before to see that until $\|Q \cap (S \cup T)\|$ increases to $q(b+1)$, the rank of the remaining elements will be at least $\frac{kq-(b+1)q}{k}$, so that each basis of the greedy packing will contain at least $(1 - \frac{b+1}{k})q$ elements of $Q - S$.

Since S contains b disjoint bases of M, it contains at least qb elements of Q. Thus, we only need to add q elements of $M - S$ to reach our goal of $q(b+1)$ elements. Since we add $(\frac{k-b-1}{k})q$ elements with each basis of T, we need only add $\frac{k}{k-b-1}$ bases.

If $b = k - 1$, the above bound is infinite. However, we can use the original deficit reduction argument. When $b = k - 1$, the quotient's deficit is at most r. Since the deficit reduces by a $(1 - 1/k)$ factor with each greedy basis, we only need $k \ln(r/k)$ bases to reduce the deficit to k, and another k bases to reduce it to 0. □

Lemma 5.13. *A packing of k matroid bases can be found deterministically (if one exists) in $O(mk \log k + (rk)^3 + r^3k^2 \log r)$ time.*

Proof. We maintain a set S of disjoint bases. We operate in phases; in phase b we increment the number of disjoint bases in S from b to $b + 1$. To do so, we compute a greedy $\frac{k}{k-b-1}$-packing T of $M - S$. By the previous theorem, $S \cup T$ contains $b + 1$ bases, of which we already have b in S. Therefore, we can find the $(b+1)^{st}$ basis by running r iterations of Knuth's augmentation algorithm

on the $rb + r\frac{k}{k-b-1}$ elements of $S \cup T$. Since one augmentation takes $O(mrb)$ time in an m-element matroid with b bases, and we perform r augmentations, the running time of phase b is

$$O(m + r\frac{k}{k-b-1}\log\frac{k}{k-b-1} + r \cdot r(b + \frac{k}{k-b-1})rb)$$

For $b < k-1$ (so $k-b-1 \geq 1$), we worsen this time bound to

$$O(m + rk\log k + r^2(b+k)rb) = O(m + rk\log k + r^3k^2) = O(m + r^3k^2)$$

since it will not affect our analysis. Thus the time for the $k-1$ phases is k times the above, namely

$$O(km + r^3k^3)$$

as claimed.

In the final $b = k-1$ phase, when the above bound is infinite, we use a greedy $k\ln(r/k)$-packing for T; the r augmentations then take $O(r \cdot rk\ln(r/k)rk) = O(r^3k^2\log r)$ time. □

Theorem 5.14. *A maximum packing of k bases can be found in $O(m + (rk)^3 + r^3k^2\log r = O(m + r^3k^2\max(k, \log r))$ time. The time spent on greedy packing computations is $O(m + rk\log^{O(1)} rk)$.*

Proof. Assuming that we know k, build a greedy $k(1+\ln r)$-packing G that contains k bases in $O(m + r(k\ln r)\log(k\ln r))$ time. Run the fast packing algorithm of Lemma 5.13 on G. Since G has $O(rk\log r)$ elements, this takes $O((rk\log r)k\log k + (rk)^3 + r^3k^2\log r) = O((rk)^3 + r^3k^2\log r)$ time.

As before, repeatedly doubling a guess for k allows us to assume that we know k. As with Corollary 5.11, the total time spent on greedy packing can be limited to $O(m + rk\log^{O(1)} rk)$. The claimed time bound follows. □

6 Combining Greedy Packing with Sampling

Greedy packings have reduced our algorithms' running-time dependence on m. Using sampling, we can simultaneously reduce or eliminate the running-time dependence on k.

Corollary 6.1. *A maximum packing of k matroid bases can be found in $O(m + r^3k^{5/2}\log^{3/2} kr))$ time with high probability (LV).*

Proof. As before, repeated doubling lets us assume we know k. So we can run the algorithm of Theorem 4.13 on a greedy $k(1+\ln r)$-packing. As before, the time to construct the greedy packing is dominated (aside from the linear term) by the time to construct the packing. □

Corollary 6.2. *An ϵ-approximation to the value of a matroid's packing number can be computed in $O(m + r^3\log^{O(1)}(kr)/\epsilon^5)$ time (MC).*

Proof. Use a greedy packing in the approximation algorithm of Corollary 4.15. □

Greedy packings can also be used to find approximate sparse quotients. We take care to avoid finding a quotient that is sparse in the greedy packing but not in the original matroid.

Lemma 6.3. *Let $S = B_1 \cup \cdots \cup B_t$ be a greedy t-packing of M with $t = k(2+\ln r)$. Suppose Q is a quotient of S with density at most $(1+\epsilon)k$. Then $M/B_t/(S-Q)$ is a quotient of M with density at most $(1+\epsilon)k/(1-\epsilon) = (1+\frac{2\epsilon}{1-\epsilon})k$ in M.*

Proof. Let $t = k(\ln r + 1) + k$ and write $A = S - Q$. Note that $M/B_t = S/B_t$; it follows that we need to prove the theorem only for the case $S = M$. That is, we must prove that if S/A has density at most $(1+\epsilon)k$ then $(S/A)/B_t = (S/B_t)/A$ has density at most $(1+\epsilon)k/(1-\epsilon)$.

We first suppose $A = \emptyset$ so $S/A = S$. We have already shown that the first $k(1+\ln r)$ bases of the greedy packing contain kr elements of S. Now consider the remaining k bases. If $\mathrm{rk}(B_t) > \epsilon r$, then the last k bases of our greedy $k(2+\ln r)$ packing (all at least this large) contribute an additional ϵkr elements to S, implying that S has size exceeding $kr + \epsilon kr$ and thus density exceeding $(1+\epsilon)k$, a contradiction. So $\mathrm{rk}(B_t) \leq \epsilon r$. It follows that $\mathrm{rk}^C(S/B_t) \geq r - \epsilon r$. Thus the density of S/B_t is less than $(1+\epsilon)kr/(r-\epsilon r) = (1+\epsilon)k/(1-\epsilon)$.

Now suppose $A \neq \emptyset$. We must bound the density of $(S/A)/B_t$. We consider S/A as a matroid S' with rank r'. As in Lemma 5.2, we transform $\{B_i/A\} = \{B_i \cap S'\}$ into a greedy packing $\{B'_i\}$ of S' by shifting elements to larger-indexed sets until each set is independent. That construction has two important properties. First, since elements are shifted to larger-indexed sets, we know that for any j,

$$\sum_{i \leq j} \|B_i\| \geq \sum_{i \leq j} \|B'_i\|.$$

Second, since we only shift "spanned" elements out of a set, we never reduce its rank. That is, in matroid S' we have $\mathrm{rk}B_i \leq \mathrm{rk}B'_i$. It follows that

$$\begin{aligned} \nu(S'/B_t) &= \frac{\|S'/B_t\|}{\mathrm{rk}(S') - \mathrm{rk}(B_t)} \\ &\leq \frac{\|S'/B'_t\|}{\mathrm{rk}(S) - \mathrm{rk}(B'_t)} \\ &= \nu(S'/B'_t) \\ &\leq (1+\epsilon)k/(1-\epsilon) \end{aligned}$$

where the last line follows by applying our argument above for the $A = \emptyset$ case to matroid S'. □

Corollary 6.4. *For any $\epsilon < 1$, an ϵ-approximation to the sparsest quotient can be found in $O(m + r^3(\log^{O(1)} kr)/\epsilon^5)$ time (MC).*

Proof. Construct a greedy $k(2+\ln r)$-packing S in M. Note that S has $s = O(rk \ln r)$ elements. By the above lemma, if we find a quotient of density less than $(1+\epsilon/3)k$ in S, it will yield a quotient of density $(1+\epsilon/3)k/(1-\epsilon/3) \leq (1+\epsilon)k$ in M. To find an approximately sparsest quotient of S, apply Theorem 4.1: sample each element with probability $O((\log s)/k\epsilon^2)$ and find a sparsest quotient of the sample (using the algorithm of Corollary 6.1). The running time follows. □

Corollary 6.5. *A $(1-\epsilon)$ times maximum packing of disjoint bases can be found in $\tilde{O}(r^3k/\epsilon^3)$ time.*

Proof. Apply the algorithm of Theorem 4.16 to a greedy packing. □

7 Application: Packing Spanning Trees

A particular instance of matroid basis packing is the problem of packing spanning trees in an undirected graph. This problem has applications to fault tolerant communication [IR88, Gus83]. It can be formulated on uncapacitated graphs (where each edge can be in at most one tree) or capacitated graphs (where the number of trees using a given edge must be less than the capacity of that edge). The corresponding "quotient" problem is to find a (multiway) partition of the graph's vertices that minimizes the ratio between the number (or capacity) of edges cut by the partition and

one less than the number of sets in the partition; a variant of Edmonds' Theorem on the duality of these quantities in graphs was first proven by Nash-Williams [NW61].

Gabow and Westermann [GW92] give an algorithm that solves the tree packing problem in $\tilde{O}(\min(mn, m^2/\sqrt{n}))$ time on m-edge, n-vertex uncapacitated graphs. Barahona [Bar92] gave an $\tilde{O}(mn^3)$-time algorithm for finding the sparsest graph quotient in capacitated graphs, and later [Bar95] followed with a tree packing algorithm that runs in $\tilde{O}(mn^3)$ time.

Theorem 7.1. *A $(1+\epsilon)$-sparsest quotient can be found in $\tilde{O}(m + n^{3/2}/\epsilon^4)$ time on uncapacitated or capacitated graphs.*

Proof. We use the fact that Nagamochi and Ibaraki [NI92] give an algorithm that constructs a greedy packing of the graphic matroid in $O(m)$ time on uncapacitated graphs, and in $O(m + n \log n)$ time on capacitated graphs.

First consider uncapacitated graphs. As with general basis packing, we use sampling to produce a subgraph of G with packing number $\hat{z} = O(\epsilon^{-2} \log m)$ that accurately approximates the quotients of G. We then use Nagamochi and Ibaraki's algorithm to extract a greedy $(\hat{z} \ln n)$-packing G' from the sampled graph. This packing has $m_1 = O(n\hat{z} \ln n)$ edges. Therefore, Gabow and Westermann's algorithm [GW92] finds the optimum packing (and identifies a sparsest quotient) in $\tilde{O}(m_1^2/\sqrt{n}) = \tilde{O}(n^{3/2}/\epsilon^4)$ time.

As noted in Section 4.2.1, the algorithm for capacitated graphs is the same, except that we "sample" from each capacitated edge by generating a Poisson random variable with the appropriate parameters.

We have ignored the determination of the correct sampling probability. By Nash-Williams theorem on graph quotients [NW61], any graph with minimum cut c has packing number between $c/2$ and c. Therefore, we can determine a good sampling probability by using the linear-time minimum cut approximation algorithms of Matula [Mat87] or Karger [Kar97a]. □

Lemma 7.2. *A $(1-\epsilon)$-maximum tree packing of k trees can be found in $\tilde{O}(kn^{3/2}/\epsilon^2)$ time (LV)*

Proof. After computing a greedy packing, we apply Theorem 4.16, but use Gabow and Westermann's algorithm [GW92] to solve each of the random subproblems. □

Remark. Gabow and Westermann's algorithm is based on augmentations, but does not benefit from having already found a large collection of bases. It is analogous to the blocking flow method for maximum flows—it cheaply finds a near optimal solution and then spends most of its time augmenting to an exact solution. Therefore, the recursive algorithm of Theorem 4.13 does not accelerate their exact algorithm. An interest open question is whether sampling can be used to speed up the blocking-flow type algorithms, for example by proving that only a small number of additional blocking flows is needed to clean up the output of the approximation algorithm.

8 Conclusion

This paper has suggested an approach to random sampling for optimization and given results that apply to matroids as models for greedy algorithms and as combinatorial objects. Two future directions suggest themselves.

In the realm of combinatorics, how much of the theory of random graphs can be extended to the more general matroid model? There is a well defined notion of connectivity in matroids [Wel76]; is this relevant to the basis packing results presented here? What further insight into random graphs can be gained by examining them from a matroid perspective? Erdös and Renyi showed a tight threshold of $p = (\ln n)/n$ for connectivity in random graphs, whereas our result gives a looser result of $p = \Omega((\log n)/n)$ for existence of sampled matroid bases. Is there a 0-1 law for bases in a random submatroid?

In the realm of optimization, we have investigated a natural paradigm that works particularly well for matroid problems: generate a small random representative subproblem, solve it quickly, and use the information gained to home in on the solution to the entire problem. In particular, an optimum solution to the subproblem may be a good solution to the original problem which can quickly be improved to an optimum solution. The obvious opportunity: apply this paradigm to other optimization problems.

9 Acknowledgments

Thanks to Don Knuth for help with a troublesome summation, and to Daphne Koller for her extensive assistance in clarifying this paper.

A Alternative Sampling Theorems for Packing

This section presents alternative proofs of our theorems about sampling matroid bases. We begin by proving a theorem on the *existence* of a basis in the sample, and we then generalize this theorem by estimating the *number* of disjoint bases we will find in the sample.

A.1 Finding a Basis

We begin with some definitions needed in the proof.

We use slightly unusual terminology. For the following definitions, fix some independent set T in M.

Definition A.1. A set A is *T-independent* or *independent of T* if $A \cup T$ is independent in M.

Definition A.2. If A is an independent set, then A/T is any maximal T-independent subset of A.

Definition A.3. $B(n,p)$ is a binomial random variable:

$$\Pr[B(n,p) = k] = \binom{n}{k} p^k (1-p)^{n-k}.$$

Lemma A.4. *If A is a basis of M, then A/T is a basis for M/T.*

Lemma A.5. *If B is a basis of M/T, then $B \cup T$ is a basis of M.*

Theorem A.6. *Suppose M contains $a + 2 + k \ln r$ disjoint bases. Then $M(1/k)$ contains a basis for M with probability $1 - O(e^{-a/k})$.*

Proof. Let $p = 1/k$. Let $\{B_i\}_{i=1}^{a+2+k\ln r}$ be disjoint bases of M. We construct the basis in $M(p)$ by examining the sets $B_i(p)$ one at a time and adding some of their elements to an independent set I (initially empty) until I is large enough to be a basis. We invert the problem by asking how many bases must be examined before I becomes a basis. Suppose we determine $U = B_1(p)$, the set of elements of B_1 contained in $M(p)$. Note that the size u of U is distributed as $B(r,p)$; thus $E[u] = rp$. Consider the contraction M/U. By Lemma A.4, this matroid contains disjoint bases $B_2/U, B_3/U, \ldots$, and has rank $r-u$. We ask recursively how many of these bases we need to examine to construct a basis B for the contracted matroid. Once we have done so, we know from Lemma A.5 that $U \cup B$ is a basis for M. This gives a *probabilistic recurrence* for the number of bases we need to examine:

$$T(r) = 1 + T(r-u), \qquad u = B(r,p).$$

If we replaced random variables by their expected values, we would get a recurrence of the form $S(r) = 1 + S((1-p)r)$, which solves to $S(r) = \log_b r$, where $b = 1/(1-p)$. Probabilistic recurrences are studied by Karp [Kar91]. His first theorem exactly describes our recurrence, and proves that for any a,

$$Pr[T(r) \geq \lfloor \log_b r \rfloor + a + 2] \leq (1 - 1/k)^a.$$

In our case, $\log_b r \approx k \ln r$. □

A.2 Counting Bases

Theorem A.7. *If $P(M) = n$ then the probability that $M(p)$ fails to contain k disjoint bases of M is at most $r \cdot \Pr[B(n,p) \leq k]$.*

Proof. We generalize the technique of the previous section. We line up the bases $\{B_i\}$ and pass through them one by one, adding some of the sampled elements from each basis to an independent set I that grows until it is itself a basis. For each B_i, we set aside some of the elements because they may be dependent on elements already added to I; we then examine the remaining elements of B_i to find out which ones were actually sampled and add those sampled elements to I. The change in the procedure is that we do this more than once: to construct the next basis, we examine those elements set aside the previous time.

Consider a series of phases; in each phase we will construct one basis. At the beginning of phase k, there will be a *remaining portion* R_n^k of basis B_n; the elements of R_n^k are those elements of B_n that were not examined in any of the previous phases. We construct an independent set I^k by processing each of the R_n^k in order. Let I_{n-1}^k be the portion of I^k that we have constructed before processing R_n^k. To process R_n^k, we split it into two sets: R_n^{k+1} are those elements that are set aside until the next phase, while $E_n^k = R_n^k - R_n^{k+1}$ is the set of elements we *examine* in this phase. The elements of E_n^k will be independent of I_{n-1}^k. Thus as in the single-basis case, we simply check which elements of E_n^k are in the sampled set, identifying the set $U_n^k = E_n^k(p)$ of elements we *use*, and add them to our growing basis. Formally, we let $I_n^k = I_{n-1}^k \cup U_n^k$; by construction I_n^k will be independent.

I_n^k	Independent set so far.
R_n^k	Remainder of n^{th} basis.
E_n^k	Elements examined for use.
U_n^k	Elements actually used from E_n^k, namely $E_n^k(p)$.

Figure 3: Variables describing n^{th} basis in k^{th} phase

We now explain precisely how we determine the split of R_n^k into R_n^{k+1} and E_n^k. Let r_n^k, i_n^k, e_n^k, and u_n^k be the size of R_n^k, I_n^k, E_n^k, and U_n^k respectively. Suppose that we have I_{n-1}^k in hand, and wish to extend it by examining elements of R_n^k. We assume by induction that $i_{n-1}^k \leq r_n^k$. It follows from the definition of matroids that there must exist a set $E_n^k \subseteq R_n^k$ such that $I_{n-1}^k \cup E_n^k$ is independent and has size r_n^k. Defining E_n^k this way determines $R_n^{k+1} = R_n^k - E_n^k$. We then set $U_n^k = E_n^k(p)$, and $I_n^k = I_{n-1}^k \cup U_n^k$.

To justify our inductive assumption we use induction on k. To prove it for $k+1$, note that our construction makes $r_n^{k+1} = i_{n-1}^k$. Thus the fact that $i_{n-2}^k \leq i_{n-1}^k$ implies that $r_{n-1}^{k+1} \leq r_n^{k+1}$. Our construction forces $i_{n-1}^{k+1} \leq r_{n-1}^{k+1}$; thus $i_{n-1}^{k+1} \leq r_n^{k+1}$ as desired.

We now use the just noted invariant $r_n^{k+1} = i_{n-1}^k$ to derive recurrences for the sizes of the various sets. As before, the recurrences will be probabilistic in nature. Recall that u_n^k is the size of U_n^k, so $u_n^k = B(e_n^k, p)$. Thus

$$\begin{aligned} r_n^{k+1} &= i_{n-1}^k \\ &= i_{n-2}^k + u_{n-1}^k \\ &= r_{n-1}^{k+1} + B(e_{n-1}^k, p). \end{aligned}$$

It follows that

$$\begin{aligned} e_n^k &= r_n^k - r_n^{k+1} \\ &= [r_{n-1}^k + B(e_{n-1}^{k-1}, p)] - [r_{n-1}^{k+1} + B(e_{n-1}^k, p)] \\ &= e_{n-1}^k - B(e_{n-1}^k, p) + B(e_{n-1}^{k-1}, p). \end{aligned}$$

Now let $f_n^k = E[e_n^k]$. Linearity of expectation applied the recurrence shows that

$$f_n^k = (1-p)f_{n-1}^k + pf_{n-1}^{k-1}.$$

Since we examine the entire first basis in the first phase, $e_0^0 = r$ and $e_0^k = 0$ for $k > 0$. Therefore this recurrence is solved by

$$f_n^k = \binom{n}{k} p^k (1-p)^{n-k} r.$$

We now ask how big n needs to be to give us a basis in the k^{th} phase. As in Section A.1, it simplifies matters to assume that we begin with an infinite set of disjoint bases, and ask for the value of n such that in the k^{th} phase, we finish constructing the k^{th} sampled basis I^k before we reach the n^{th} original basis B_n. Recall the variable u_n^k denoting the number of items from B_n used in I^k. Suppose that in the k^{th} phase we use no elements from any basis after B_n. One way this might happen is if we never finish constructing I^k. However, this is a probability 0 event. The only other possibility is that we have finished constructing I^k by the time we reach B_n so that we examine no more bases.

It follows that if $u_j^k = 0$ for every $j \geq n$, then we must have finished constructing I^k before we examined B_n. Since the u_j^k are non-negative, this is equivalent to saying that $\sum_{j \geq n} u_j^k = 0$. It follows that our problem can be solved by determining the value n such that $\sum_{j \geq n} u_j^k = 0$ with high probability.

From the Markov inequality [MR95], which says that for positive integer random variables $\Pr[X > 0] \leq E[X]$, and from the fact that $E[u_j^k] = pE[e_j^k] = pf_j^k$, we deduce that the probability that we fail to construct I^k before reaching B_n is at most

$$s_n^k = E\left[\sum_{j \geq n} u_j^k\right] = p \sum_{j \geq n} f_j^k.$$

One way to bound s_n^k is to sum by parts. However, an easier method is to consider an infinite sequence of coin flips with heads probability p. The quantity pf_j^k is then r times the probability that the $(k+1)^{st}$ head occurs on the j^{th} flip. Then s_n^k is just r times the probability that the $(k+1)^{st}$ head appeared after the n^{th} flip, which is equivalent to the probability that the first n flips had at most k heads, *i.e.* $\Pr[B(n,p) \leq k]$. This yields the theorem. □

The probability of finding no bases is thus at most $s_n^0 \approx re^{-np}$; this is exactly the result proven in the Section A.1.

We also consider the converse problem, namely to upper bound the number of bases that survive.

Corollary A.8. *If $P(M) \leq n$, and $k > np$, then the probability that $M(p)$ contains more than k disjoint bases of M is at most* $\Pr[B(n,p) \geq k]$.

Proof. By Edmonds' theorem, there must exist some $A \subset M$ such that

$$(n+1)\,\mathrm{rk}A + \|\overline{A}\| < (n+1)r.$$

If $\mathrm{rk}A = r$, the above statement cannot be true. Thus $\mathrm{rk}A \leq r-1$. It follows that in fact

$$(n+1)\,\mathrm{rk}A + \max(\|\overline{A}\|, n) < (n+1)r.$$

Now consider what happens in $M(p)$. The rank of $A(p)$ is certainly at most $\mathrm{rk}A$. To bound $\|\overline{A}(p)\|$, consider two cases. If $\|\overline{A}\| \geq n$, then $\Pr[\|\overline{A}(p)\| > (k/n)\|\overline{A}\|] < \Pr[B(n,p) > k]$. On the other hand, if $\|\overline{A}\| < n$, then $\Pr[\|\overline{A}(p)\| > k] < \Pr[B(n,p) > k]$. In other words, with the desired probability $\|\overline{A}(p)\| \leq \max((k/n)\|\overline{A}\|, k) < \frac{k+1}{n+1}\max(\|\overline{A}\|, n)$. It follows that with the desired probability,

$$\begin{aligned}(k+1)\,\mathrm{rk}(A(p)) + \|\overline{A}(p)\| &< (k+1)\,\mathrm{rk}A + \frac{k+1}{n+1}\max(\|\overline{A}\|, n) \\ &= \frac{k+1}{n+1}[(n+1)\,\mathrm{rk}(A) + \max(\|\overline{A}\|, n)] \\ &< \frac{k+1}{n+1}(n+1)r \\ &= (k+1)r\end{aligned}$$

If $M(p)$ contains no basis of M, then certainly it contains fewer than k bases. On the other hand, if $M(p)$ does contain a basis then $M(p)$ has rank r, in which case $A(p)$ demonstrates through Edmonds' Theorem that $M(p)$ contains at most k bases. □

Applying the Chernoff bound [Che52] to the previous two theorems yields Theorem 4.1.

References

[ACM91] ACM. *Proceedings of the* 23^{rd} *ACM Symposium on Theory of Computing*. ACM Press, May 1991.

[ACM94] ACM. *Proceedings of the* 26^{th} *ACM Symposium on Theory of Computing*. ACM Press, May 1994.

[ACM96] ACM. *Proceedings of the* 28^{th} *ACM Symposium on Theory of Computing*. ACM Press, May 1996.

[Bar92] Francisco Barahona. Separating from the dominant of the spanning tree polytope. *Operations Research Letters*, 12:201–203, 1992.

[Bar95] Francisco Barahona. Packing spanning trees. *Mathematics of Operation Research*, 20(1):104–115, February 1995.

[BK96] András A. Benczúr and David R. Karger. Approximate s–t min-cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the* 28^{th} *ACM Symposium on Theory of Computing* [ACM96], pages 47–55.

[Bol85] Béla Bollobás. *Random Graphs*. Harcourt Brace Janovich, 1985.

[Che52] H. Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[Col87] Charles J. Colbourn. *The Combinatorics of Network Reliability*, volume 4 of *The International Series of Monographs on Computer Science*. Oxford University Press, 1987.

[DRT92] Brandon Dixon, Monika Rauch, and Robert E. Tarjan. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184–1192, 1992.

[Edm65] Jack Edmonds. Minimum partition of a matroid into independents subsets. *Journal of Research of the National Bureau of Standards*, 69:67–72, 1965.

[Edm71] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:126–136, 1971.

[ER59] Pàl Erdös and A. Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[Fel68] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, 3 edition, 1968.

[FM92] Tomas Feder and Milena Mihail. Balanced matroids. In *Proceedings of the* 24^{th} *ACM Symposium on Theory of Computing*, pages 26–38. ACM, ACM Press, May 1992.

[FR75] Robert W. Floyd and Ronald L. Rivest. Expected time bounds for selection. *Communications of the ACM*, 18(3):165–172, 1975.

[Gab95] Harold N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *Journal of Computer and System Sciences*, 50(2):259–273, April 1995. A preliminary version appeared in STOC 1991.

[Gus83] Dan Gusfield. Connectivity and edge disjoint spanning trees. *ipl*, 16:87–89, 1983.

[GW92] Harold N. Gabow and Herbert H. Westermann. Forests, frames, and games: Algorithms for matroid sums and applications. *Algorithmica*, 7(5):465–497, 1992.

[IR88] A. Itai and M. Rodeh. The multi-tree approach to reliability in distributed networks. *Information and Control*, 79:43–59, 1988.

[Kar91] Richard M. Karp. Probabilistic recurrence relations. In *Proceedings of the* 23^{rd} *ACM Symposium on Theory of Computing* [ACM91], pages 190–197.

[Kar92] David R. Karger. Approximating, verifying, and constructing minimum spanning forests. Manuscript., 1992.

[Kar93] David R. Karger. Random sampling in matroids, with applications to graph connectivity and minimum spanning trees. In *Proceedings of the* 34^{th} *Annual Symposium on the Foundations of Computer Science*, pages 84–93. IEEE, IEEE Computer Society Press, November 1993. Submitted for publication..

[Kar96] David R. Karger. Minimum cuts in near-linear time. In *Proceedings of the* 28^{th} *ACM Symposium on Theory of Computing* [ACM96], pages 56–63.

[Kar97a] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 1997. To appear. A preliminary version appeared in STOC 94.

[Kar97b] David R. Karger. Using random sampling to find maximum flows in uncapacitated undirected graphs. In *Proceedings of the* 29^{th} *ACM Symposium on Theory of Computing*, pages 240–249. ACM, ACM Press, May 1997.

[KKT95] David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995.

[Knu73] Donald E. Knuth. Matroid partitioning. Technical Report STAN-CS-73-342, Stanford University, 1973.

[Kru56] J. B. Kruskal, Jr. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[KS96] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, July 1996. Preliminary portions appeared in SODA 1992 and STOC 1993.

[KT94] Philip N. Klein and Robert E. Tarjan. A randomized linear-time algorithm for finding minimum spanning trees. In *Proceedings of the* 26^{th} *ACM Symposium on Theory of Computing* [ACM94], pages 9–15.

[KY76] Donald E. Knuth and Andrew C. Yao. The complexity of nonuniform random number generation. In Joseph F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 357–428. Academic Press, 1976.

[Law76] Eugene L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Reinhardt and Winston, New York, 1976.

[LR92] Jon Lee and Jennifer Ryan. Matroid applications and algorithms. *ORSA Journal on Computing*, 4(1):70–96, 1992.

[Mat87] D. W. Matula. Determining edge connectivity in $O(nm)$. In *Proceedings of the* 28^{th} *Annual Symposium on the Foundations of Computer Science*, pages 249–251. IEEE, IEEE Computer Society Press, 1987.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, 1995.

[Mul94] Ketan Mulmuley. *Computational Geometry*. Prentice Hall, 1994.

[NI92] Hiroshi Nagamochi and Toshihide Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM Journal of Discrete Mathematics*, 5(1):54–66, February 1992.

[NW61] C. St. J. A. Nash-Williams. Edge disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, 36:445–450, 1961.

[Pol90] V. P. Polesskii. Bounds on the connectedness probability of a random graph. *Problems of Information Transmition*, 27:86–97, 1990.

[Rec89] András Recski. *Matroid Theory and its Applications In Electric Network Theory and in Statics*. Number 6 in Algorithms and Combinatorics. Springer-Verlag, 1989.

[RS80] John H. Reif and P. Spirakis. Random matroids. In *Proceedings of the* 12^{th} *ACM Symposium on Theory of Computing*, pages 385–397, 1980.

[Sch79] Alexander Schrijver, editor. *Packing and Covering in Combinatorics*. Number 106 in Mathematical Centre Tracts. Mathematical Centre, 1979.

[VDW37] B. L. Van Der Waerden. *Moderne Algebra*. Springer, 1937.

[Wel76] D. J. A. Welsh. *Matroid Theory*. London Mathematical Society Monographs. Academic Press, 1976.

[Whi35] Hassler Whitney. On the abstract properties of linear independence. *American Journal of Mathematics*, 57:509–533, 1935.