

# Fast Approximation Algorithms for Bounded Degree and Crossing Spanning Tree Problems

Chandra Chekuri ✉

University of Illinois at Urbana-Champaign, USA

Kent Quanrud ✉

Purdue University, USA

Manuel R. Torres ✉

University of Illinois at Urbana-Champaign, USA

## Abstract

We develop fast approximation algorithms for the minimum-cost version of the Bounded-Degree MST problem (BD-MST) and its generalization the Crossing Spanning Tree problem (CROSSING-ST). We solve the underlying LP to within a  $(1 + \epsilon)$  approximation factor in near-linear time via the multiplicative weight update (MWU) technique. This yields, in particular, a near-linear time algorithm that outputs an estimate  $B$  such that  $B \leq B^* \leq \lceil (1 + \epsilon)B \rceil + 1$  where  $B^*$  is the minimum-degree of a spanning tree of a given graph. To round the fractional solution, in our main technical contribution, we describe a fast near-linear time implementation of swap-rounding in the spanning tree polytope of a graph. The fractional solution can also be used to sparsify the input graph that can in turn be used to speed up existing combinatorial algorithms. Together, these ideas lead to significantly faster approximation algorithms than known before for the two problems of interest. In addition, a fast algorithm for swap rounding in the graphic matroid is a generic tool that has other applications, including to TSP and submodular function maximization.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Approximation algorithms analysis; Theory of computation  $\rightarrow$  Graph algorithms analysis

**Keywords and phrases** bounded degree spanning tree, crossing spanning tree, swap rounding, fast approximation algorithms

**Digital Object Identifier** 10.4230/LIPIcs.APPROX/RANDOM.2021.24

**Category** APPROX

**Related Version** *Full Version*: <https://arxiv.org/abs/2011.03194v2> [19]

**Funding** *Chandra Chekuri*: Supported in part by NSF grant CCF-1910149.

*Manuel R. Torres*: Supported in part by fellowships from NSF and the Sloan Foundation, and NSF grant CCF-1910149.

## 1 Introduction

Spanning trees in graphs are a fundamental object of study and arise in a number of settings. Efficient algorithms for finding a minimum-cost spanning tree (MST) in a graph are classical. In a variety of applications ranging from network design, TSP, phylogenetics, and others, one often seeks to find a spanning tree with additional constraints. An interesting and well-known problem in this space is the BOUNDED-DEGREE SPANNING TREE (BD-ST) problem in which the goal is to find a spanning tree in a given graph  $G = (V, E)$  that minimizes the maximum degree in the tree. We refer to the minimum-cost version of BD-ST as BD-MST where one seeks a spanning tree of minimum cost subject to a given degree bound  $B$  on the vertices. The decision version of BD-ST (Given  $G, B$  is there a spanning tree with maximum degree  $B$ ?) is already NP-Complete for  $B = 2$  since it captures the Hamilton-Path problem. In an influential paper, Fürer and Raghavachari [24], building on earlier work of Win [49],



© Chandra Chekuri, Kent Quanrud, and Manuel R. Torres;  
licensed under Creative Commons License CC-BY 4.0

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021).

Editors: Mary Wootters and Laura Sanità; Article No. 24; pp. 24:1–24:22



Leibniz International Proceedings in Informatics

LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

described a simple local-search type algorithm that runs in  $\tilde{O}(mn)$  time (here  $m$  is number of edges and  $n$  number of nodes) that outputs a spanning tree with degree at most  $B + 1$ , or certifies that  $G$  does not have a spanning tree with degree at most  $B$  (we use  $\tilde{O}$  notation to suppress poly-logarithmic factors in  $n, m, 1/\epsilon$  for notational simplicity). Their algorithm, in fact, works even in the non-uniform setting where each vertex  $v$  has a specified degree bound  $B_v$ . The Fürer-Raghavachari result spurred a substantial line of work that sought to extend their clean result to the minimum-cost setting. This was finally achieved by Singh and Lau [43] who described a polynomial-time algorithm that outputs a tree  $T$  such that the degree of each  $v$  in  $T$  is at most  $B_v + 1$  and the cost of the tree is at most  $\text{OPT}$ . Their algorithm is based on iterative rounding of a natural LP relaxation. We refer the reader to [35, 13, 28, 43, 22] for several ideas and pointers on BD-ST and BD-MST.

Motivated by several applications, Bilo et al. [9] defined the CROSSING SPANNING TREE problem (CROSSING-ST). In CROSSING-ST the input is a graph  $G = (V, E)$ , a collection of cuts  $C_1, C_2, \dots, C_k$ , and integers  $B_1, B_2, \dots, B_k$ . Each cut  $C_i$  is a subset of the edges though in many applications we view  $C_i$  as  $\delta_G(S_i)$  for some  $S_i \subset V$  (where  $\delta_G(S_i) = \{uv \in E \mid u \in S_i, v \in V \setminus S_i\}$  is the standard definition of a cut set with respect to  $S_i$ ). The goal is to find a spanning tree  $T$  such that  $|E(T) \cap C_i| \leq B_i$ , that is,  $T$  crosses each cut  $C_i$  at most  $B_i$  times. It is easy to see that BD-ST is a special case of CROSSING-ST where the cuts correspond to singletons. We refer to the min-cost version of CROSSING-ST as CROSSING-MST. CROSSING-ST gained substantial prominence in the context of the asymmetric traveling salesman problem (ATSP) — Asadpour et al. [5] showed the importance of thin spanning trees for approximating ATSP and obtained an  $O(\log n / \log \log n)$ -approximation (now we have constant factor approximations for ATSP via other methods [46, 47]). Motivated by the thin tree conjecture and its applications to ATSP (see [5, 2]) and other technical considerations, researchers have studied CROSSING-ST, its generalization to the matroid setting, and various special cases [20, 8, 7, 37, 36]. The best known approximation algorithms for CROSSING-ST and its special cases have mainly relied on the natural LP relaxation. For general CROSSING-ST the best known approximation ratio is  $\min\{O(\log k / \log \log k), (1 + \epsilon)B + O(\log k / \epsilon^2)\}$ . A variety of sophisticated and interesting rounding techniques have been designed for CROSSING-ST and its special cases. An outstanding open problem is whether CROSSING-ST admits a constant factor approximation via the natural LP relaxation. This is challenging due its implications for the thin tree conjecture.

Most of the focus on BD-MST and CROSSING-ST has been on the quality of the approximation. The best known approximation bounds rely on LP relaxations and complex rounding procedures. The overall running times are very large polynomials in the input size and are often unspecified. In this paper we are interested in the design of *fast* approximation algorithms for BD-MST, CROSSING-ST and related problems. In recent years there has been significant progress in designing fast, and often near-linear time, approximation algorithms for a number of problems in discrete and combinatorial optimization. This has been led by, and also motivated, synergy between continuous/convex optimization, numerical linear algebra, dynamic data structures, sparsification techniques, and structural results, among several others. For BD-ST with uniform degree, Duan, He and Zhang [22] described a combinatorial algorithm that for any given  $\epsilon > 0$ , runs in  $O(m \log^7 n / \epsilon^7)$  time, and either outputs a spanning tree with degree  $(1 + \epsilon)B + O(\log n / \epsilon^2)$  or reports that there does not exist a tree with maximum degree  $\leq B$ . This paper is partly motivated by the goal of improving their results: dependence on  $\epsilon$ , a better approximation, handling non-uniform bounds, cost, CROSSING-MST, and connection to the LP relaxation.

A second motivation for this paper is to develop a fast algorithm for swap-rounding in the

spanning tree polytope. It is a dependent rounding technique that has several applications ranging from TSP to submodular function maximization (see [20, 26, 17, 23]). The question of developing a fast swap-rounding procedure for spanning trees was explicitly raised in [17] in the context of Metric-TSP.

## 1.1 Results

In this paper we develop fast approximation algorithms for BD-MST, CROSSING-MST and related problems in a unified fashion via broadly applicable methodology based on the LP relaxation. We consider the following problem with general packing constraints. The input to this problem is an undirected graph  $G = (V, E)$ , a non-negative edge-cost vector  $c : E \rightarrow \mathbb{R}_+$ , a non-negative matrix  $A \in [0, 1]^{k \times m}$ , and a vector  $b \in [1, \infty)^k$ . The goal is to find a spanning tree  $T$  of minimum cost such that  $A\mathbf{1}_T \leq b$  where  $\mathbf{1}_T \in \{0, 1\}^m$  is the characteristic vector of the edge set of  $T$ . This is a special case of a more general problem considered in [20]: min-cost matroid base with packing constraints. Here we restrict attention to spanning trees (graphic matroid). We refer to this slightly more general problem also as CROSSING-MST.

Our first result is a near-linear time algorithm to approximately solve the underlying LP relaxation for CROSSING-MST. For a multigraph  $G$  we let  $\mathcal{T}(G)$  denote the set of all spanning trees of  $G$  and let  $\text{ST}(G)$  denote the spanning tree polytope of  $G$  (which is the convex hull of the characteristic vectors  $\{\mathbf{1}_T \mid T \in \mathcal{T}(G)\}$ ).

► **Theorem 1.** *Let  $G = (V, E)$  be a multigraph with  $m$  edges and  $n$  nodes and consider the linear program  $\min\{c^T x : Ax \leq b, x \in \text{ST}(G)\}$  where  $A \in [0, 1]^{k \times m}$ ,  $b \in [1, \infty)^k$ ,  $c \in [0, \infty)^m$ . Let  $N$  be the maximum of  $m$  and number of non-zeroes in  $A$ . There is a randomized polynomial time algorithm that for any given  $\epsilon \in (0, 1/2]$  runs in  $\tilde{O}(N/\epsilon^2)$  time and with high probability either correctly certifies that the LP is infeasible or outputs a solution  $y \in \text{ST}(G)$  such that  $c^T y \leq (1 + \epsilon)\text{OPT}$  and  $Ay \leq (1 + \epsilon)b$  where  $\text{OPT}$  is the minimum value of a feasible solution.*

► **Remark 2.** We describe a randomized algorithm for the sake of simplicity, however we believe that a deterministic algorithm with similar guarantees can be obtained via ideas in [16].

Solving the LP relaxation quickly enables to estimate the optimum integer solution value via existing rounding results [43, 20, 8, 7, 37, 36]. For instance, when specialized to BD-ST, we obtain a near-linear time algorithm to estimate the optimum value arbitrarily closely (modulo the additive 1).

► **Corollary 3.** *There is a randomized  $\tilde{O}(m/\epsilon^2)$ -time algorithm that outputs a value  $B$  such that  $B \leq B^* \leq \lceil (1 + \epsilon)B \rceil + 1$  where  $B^*$  is the minimum maximum degree over all spanning trees (that is,  $B^* = \min_{T \in \mathcal{T}(G)} \max_{v \in V} \deg_T(v)$  where  $\deg_T(v)$  is the degree of  $v$  in  $T$ ).*

Our second result shows the utility of the LP solution to sparsify the original graph  $G$ .

► **Theorem 4.** *Let  $x \in \text{ST}(G)$  be such that  $Ax \leq b$  for a matrix  $A \in [0, 1]^{k \times m}$  and  $b \in [1, \infty)^k$ . Consider a random subgraph  $G' = (V, E')$  of  $G$  obtained by picking each edge  $e \in G$  with probability  $\alpha_e := \min\{1, \frac{36 \log(k+m)}{\epsilon^2} \cdot x_e\}$ . Then with high probability the following hold: (i)  $|E'| = O(n \ln(k+m)/\epsilon^2)$  (ii) there exists a fractional solution  $z \in \text{ST}(G)$  in the support of  $G'$  such that  $Az \leq (1 + 3\epsilon)b$ .*

One can run a combinatorial algorithm such as the Fürer-Raghavchari algorithm [24] on the sparse graph rather than on the original graph  $G$ . This yields the following corollary which improves the  $\tilde{O}(mn)$  running time substantially when  $G$  is dense.

136 ► **Corollary 5.** *There is a randomized algorithm for BD-ST that given a graph  $G$  on  $n$  nodes*  
 137 *runs in  $\tilde{O}(n^2/\epsilon^2)$  time, and with high probability outputs a spanning tree  $T$  with maximum*  
 138 *degree  $\lceil(1 + \epsilon)B^*\rceil + 2$  where  $B^*$  is the optimal degree bound.*

139 ► **Remark 6.** Corollaries 3 and 5 can be generalized to the non-uniform degree version of  
 140 BD-ST. Input is  $G$  and degree bounds  $B_v, v \in V$ , and the algorithm either decides that  
 141 there is no spanning tree satisfying the degree bounds or outputs a tree that approximately  
 142 satisfies them.

143 Our final result is a fast algorithm to round the LP solution. Several different rounding  
 144 strategies have been developed for BD-MST and CROSSING-MST and they yield different  
 145 guarantees and take advantage of the special structure of the given instance. Iterated  
 146 rounding has been one of the primary and powerful techniques, however it requires basic  
 147 feasible solutions to the LP relaxation; it seems far from obvious how to obtain fast algorithms  
 148 with comparable guarantees and is a challenging open problem. We are here interested  
 149 in *oblivious randomized rounding* strategies that take a point  $x \in \text{ST}(G)$  and round it  
 150 to a random spanning tree  $T \in \mathcal{T}(G)$  such that the coordinates of the resulting random  
 151 edge vector are *negatively correlated*<sup>1</sup>. Negative correlation implies concentration for linear  
 152 constraints as shown by Panconesi and Srinivasan [38]. These strategies, when combined  
 153 with the LP solution, yield bicriteria approximation algorithms for CROSSING-MST of the  
 154 form  $(1 + \epsilon, \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\})$  where the first part is the  
 155 approximation with respect to the cost and the second part with respect to the packing  
 156 constraints. For CROSSING-ST and CROSSING-MST these are currently the best known  
 157 approximation ratios (although special cases such as BD-MST admit much better bounds).  
 158 Several dependent randomized rounding techniques achieving negative correlation in the  
 159 spanning tree polytope are known: maximum entropy rounding [5], pipage rounding and  
 160 *swap rounding* [20]. These rounding techniques generally apply to matroids and have several  
 161 other applications. In this paper we show that given  $x \in \text{ST}(G)$ , one can swap-round  $x$  to  
 162 a spanning tree in near-linear time provided it is given in an implicit fashion; alternately  
 163 one can obtain an implicit *approximate* representation  $x'$  of  $x$  and then apply an efficient  
 164 swap-rounding on  $x'$ . Since swap-rounding is a flexible procedure and does not generate a  
 165 unique distribution, a precise technical statement requires more formal notation and we refer  
 166 the reader to Section 3. Here we state a theorem in a general form so that it can be used in  
 167 other contexts.

168 ► **Theorem 7.** *Let  $G = (V, E)$  be a multigraph with  $m$  edges and let  $x \in [0, 1]^m$ . For any*  
 169  *$\epsilon \in (0, 1/2)$  there is a randomized algorithm that runs in  $\tilde{O}(m/\epsilon^2)$  time and either correctly*  
 170 *decides that  $x \notin \text{ST}(G)$  or outputs a random vector  $T = (X_1, X_2, \dots, X_m) \in \{0, 1\}^m$  such*  
 171 *that (i)  $T$  is the characteristic vector of a spanning tree of  $G$  (ii)  $\mathbb{E}[X_i] \leq (1 + \epsilon)x_i$  for*  
 172  *$1 \leq i \leq m$  and (iii)  $X_1, X_2, \dots, X_m$  are negatively correlated. In particular  $T$  is obtained as*  
 173 *a swap-rounding of a vector  $y$  such that  $y \leq (1 + \epsilon)x$ .*

174 Combining Theorems 1 and 7 and existing results on swap rounding [20] we obtain the  
 175 following. The approximation ratio matches the best known for CROSSING-MST and the  
 176 algorithm runs in near-linear time.

<sup>1</sup> A collection of  $\{0, 1\}$  random variables  $X_1, X_2, \dots, X_r$  are negatively correlated if, for all subsets  $S \subseteq [r]$ ,  
 $\mathbb{E}[\prod_{i \in S} X_i] \leq \prod_{i \in S} \mathbb{E}[X_i]$  and  $\mathbb{E}[\prod_{i \in S} (1 - X_i)] \leq \prod_{i \in S} (1 - \mathbb{E}[X_i])$ .

177 ► **Corollary 8.** *For the feasibility version of CROSSING-MST, there is a randomized algorithm*  
 178 *that runs in near-linear time and outputs a spanning tree  $T$  such that*

$$179 \quad A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\}$$

181 *with high probability. For the cost version of CROSSING-MST, there is a randomized algorithm*  
 182 *that outputs a*

$$183 \quad (1 + \epsilon, \min\{O(\log k / \log \log k)b_i, (1 + \epsilon)b_i + O(\log k)/\epsilon^2\})$$

185 *bicriteria approximation with probability  $\Omega(\epsilon)$ . After  $\tilde{O}(1/\epsilon)$  independent repetitions of this*  
 186 *algorithm, we can obtain the same guarantees with high probability.*

187 Our algorithm, when specialized to BD-ST and BD-MST is more general than the one  
 188 in [22] in terms of handling cost and non-uniform degrees. In addition we obtain a very close  
 189 estimate of  $B^*$ , a much better dependence on  $\epsilon$ , and also obtain an approximation of the  
 190 form  $O(\log n / \log \log n)B^*$  which is better than  $(1 + \epsilon)B^* + O(\log n)/\epsilon^2$  for small  $B^*$ .

191 We mainly focused on BD-MST and a high-level result for CROSSING-MST. One can  
 192 obtain results for related problems that involve multiple costs, lower bounds in addition to  
 193 upper bounds, and other applications of swap-roundings. We discuss these in more detail in  
 194 Section A.

## 195 1.2 Overview of main ideas

196 Faster approximation algorithms for LPs that arise in combinatorial optimization have been  
 197 developed via several techniques. We follow a recent line of work [16, 18, 39, 14] that utilizes  
 198 features of the multiplicative weight update (MWU) method and data structures to speed  
 199 up *implicit* LPs. In particular, the LP for CROSSING-MST that we seek to solve can be  
 200 addressed by the randomized MWU algorithm from [18] and data structures for dynamic  
 201 MST [29]. The overall approach follows some ideas from past work [16]. The sparsification  
 202 result is inspired by recent applications of similar ideas [16, 15, 11] and utilizes Karger’s  
 203 theorem on random sampling for packing disjoint bases in matroids [30].

204 Our main novel contribution is Theorem 7 which we believe is of independent interest  
 205 beyond the applications outlined here. Dependent randomized rounding techniques have had  
 206 many spectacular applications. In particular maximum entropy rounding in the spanning  
 207 tree polytope gave a strong impetus to this line of work via its applications to ATSP [5] and  
 208 metric-TSP [27]. Swap-rounding is a simpler scheme to describe and analyze, and suffices for  
 209 several applications that only require negative correlation. However, all the known dependent  
 210 rounding schemes are computationally expensive. Recent work has led to fantastic progress  
 211 in sampling spanning trees [4], however the bottleneck for maximum entropy rounding is to  
 212 compute, from a given point  $x \in \text{ST}(G)$ , the maximum entropy distribution with marginals  
 213 equal to  $x$ ; polynomial time (approximation) algorithms exist for this [5, 44] but they are  
 214 rather slow. Swap-rounding [20] requires one to decompose  $x \in \text{ST}(G)$  (or more generally a  
 215 point in the matroid base polytope) into a convex combination of spanning trees; that is  
 216 we write  $x = \sum_{T \in \mathcal{T}} \lambda_T \mathbf{1}_T$  such that  $\sum_T \lambda_T = 1$  and  $\lambda_T \geq 0, T \in \mathcal{T}$ . This is a non-trivial  
 217 problem to do exactly. The starting point here is a theorem in [16] that shows that one can  
 218 solve this decomposition problem *approximately* and deterministically in near-linear time via  
 219 a reduction to the problem of spanning tree packing; this is done via MWU techniques. The  
 220 near-linear time algorithm implies that any  $x \in \text{ST}(G)$  can be decomposed efficiently into an  
 221 *implicit* convex decomposition of total size  $\tilde{O}(m/\epsilon^2)$  where  $\epsilon$  is the approximation parameter  
 222 in the decomposition. To store the convex combination  $\sum_{i=1}^h \lambda_i \mathbf{1}_{T_i}$  implicitly, we store the

223 first tree  $T_1$  explicitly and to obtain  $T_{i+1}$  from  $T_i$  for  $i \in [h - 1]$ , we store the edges in the  
 224 symmetric difference of  $T_{i+1}$  and  $T_i$ . The size of the decomposition is then the sum of the sizes  
 225 of the symmetric differences and the size of  $T_1$ . We give a more formal definition of an implicit  
 226 decomposition in Section 3.2. We show in this paper that this implicit sparse decomposition  
 227 is well-suited to the swap-rounding algorithm. We employ a divide-and-conquer strategy  
 228 with appropriate tree data structures to obtain an implementation that is near-linear in the  
 229 size of the implicit decomposition. Putting these ingredients together yields our result.<sup>2</sup>

230 The seemingly fortuitous connection between the MWU based algorithm for packing  
 231 spanning trees and its implicit representation leading to a fast algorithm for swap-rounding  
 232 is yet another illustration of the synergy between tools coming together in the design of fast  
 233 algorithms.

### 234 1.3 Other related work

235 We overview some known results on CROSSING-ST and CROSSING-MST and special cases.  
 236 BD-MST can be viewed as a special case of CROSSING-MST where each edge participates in  
 237 2 constraints. Bansal et al. [8] showed that if each edge participates in at most  $\Delta$  constraints  
 238 of  $A$  (and  $A$  is a binary matrix) then one can obtain a  $(1, b + \Delta - 1)$ -approximation generalizing  
 239 the BD-MST result; this was further extended to matroids by Lau, Kiraly and Singh [33].  
 240 It is shown in [7] that for CROSSING-ST one cannot obtain a purely additive approximation  
 241 better than  $O(\sqrt{n})$  via the natural LP relaxation. For this they use a reduction from  
 242 discrepancy minimization; it also implies, via the hardness result in [12] for discrepancy,  
 243 that it is NP-Hard to obtain a purely additive  $o(\sqrt{n})$  bound. Bansal et al. [7] consider  
 244 the *laminar* case of CROSSING-MST where the cuts form a laminar family and obtained  
 245 a  $(1, b + O(\log n))$  approximation via iterative rounding (this problem generalizes BD-  
 246 MST). Olver and Zenklusen [37] consider chain-constrained CROSSING-ST which is a further  
 247 specialization when the laminar family is a chain (a nested family of cuts). For this special  
 248 case they obtained an  $O(1)$ -factor approximation in the unit cost setting; Linhares and  
 249 Swamy [36] considered the min-cost version and obtained an  $(O(1), O(1))$ -approximation.  
 250 [37] also showed that even in the setting of chain-constrained CROSSING-ST, it is NP-Hard  
 251 to obtain a purely additive bound better than  $c \log n / \log \log n$  for some fixed constant  $c$ .

252 Dependent randomized rounding has been an active area of research with many applica-  
 253 tions. Pipeage rounding, originally devoped by Ageev and Sviridenko [1] in a deterministic  
 254 way, was generalized to the randomized setting by Srinivasan [45] and by Gandhi et al. [25]  
 255 and [10, 20] and has led to a number of applications. Maximum entropy rounding satisfies  
 256 additional properties beyond negative correlation and this is important in applications to  
 257 metric-TSP (see [27] and very recent work [31, 32]). There has been exciting recent progress  
 258 on sampling spanning trees and bases in matroids and we refer the reader to some recent  
 259 work [41, 3, 4] for further pointers. Concentration bounds via dependent rounding can also  
 260 be obtained without negative correlation (see [21] for instance) and recent work of Bansal [6]  
 261 combines iterative rounding with dependent rounding in a powerful way.

---

<sup>2</sup> In an earlier version of the paper (see [19]) we described our fast swap rounding using two ideas. The first was a fast near-linear time algorithm to merge two spanning trees using the link-cut tree data structure. We were unaware of prior work of Ene and Nguyễn [23] that had already given such an algorithm in the context of fast algorithms for submodular function maximization in graphic matroids. In this version of the paper we use their algorithm as a black box. We focus on our second idea which exploits the implicit representation. We thank Alina Ene and Huy Nguyễn for pointing out to us their fast algorithm for merging two trees.

262 Fast approximation algorithms for solving positive LPs and SDPs has been an extensive  
 263 area of research starting from the early 90s. Lagrangean relaxation techniques based on  
 264 MWU and other methods have been extensively studied in the past, and continue to provide  
 265 new insights and results for both explicit and implicit problems. Recent work based on  
 266 a convex optimization perspective has led to a number of new results and improvements.  
 267 It is infeasible to do justice to this extensive research area and we refer the reader to two  
 268 recent PhD theses [40, 48]. Spectacular advances in fast algorithms based on the Laplacian  
 269 paradigm, interior point methods, cutting plane methods, spectral graph theory, and several  
 270 others have been made in the recent past and is a very active area of research with frequent  
 271 ongoing developments.

## 272 Organization

273 Section 2 introduces some relevant notation, technical background and tree data structures  
 274 that we rely on. Section 3 describes our fast swap-rounding algorithm and proves Theorem 7.  
 275 Section 4 describes the sparsification process of Theorem 4. Section 5 discusses the LP  
 276 relaxation for CROSSING-ST and Theorem 1. Section A brings together results from previous  
 277 sections to prove some of the corollaries stated in the introduction and provides details of  
 278 some extensions and related problems.

## 279 2 Preliminaries and notation

280 For a set  $S$ , we use the convenient notation  $S - i$  to denote  $S \setminus \{i\}$  and  $S + i$  to denote  
 281  $S \cup \{i\}$ .

### 282 Matroids

283 We discuss some basics of matroids to establish some notation as well as present some useful  
 284 lemmas that will be used later. A *matroid*  $\mathcal{M}$  is a tuple  $(N, \mathcal{I})$  with  $\mathcal{I} \subseteq 2^N$  satisfying  
 285 the following three properties: (1)  $\emptyset \in \mathcal{I}$ , (2) if  $A \in \mathcal{I}$  and  $B \subseteq A$ , then  $B \in \mathcal{I}$ , and (3) if  
 286  $A, B \in \mathcal{I}$  such that  $|A| < |B|$  then there exists  $b \in B \setminus A$  such that  $A + b \in \mathcal{I}$ . We refer to  
 287 the sets in  $\mathcal{I}$  as *independent sets* and say that maximal independent sets are *bases*. The *rank*  
 288 of  $\mathcal{M}$  is the size of a base. For a set  $A \subseteq 2^N$ , we refer to  $r_{\mathcal{M}}(A) = \max\{|S| : S \subseteq A, S \in \mathcal{I}\}$   
 289 as the *rank* of  $A$ .

290 A useful notion that we utilize in our fast implementation of swap rounding is that of  
 291 contraction of a matroid. We say that the *contraction* of  $e$  in  $\mathcal{M}$  results in the matroid  
 292  $\mathcal{M}/e = (N - e, \{I \subseteq N - e : I + e \in \mathcal{I}\})$  if  $r_{\mathcal{M}}(\{e\}) = 1$  and  $\mathcal{M}/e = (N - e, \{I \subseteq N - e : I \in \mathcal{I}\})$   
 293 if  $r_{\mathcal{M}}(\{e\}) = 0$ . This definition extends naturally to contracting subsets  $A \subseteq N$ . It can be  
 294 shown that contracting the elements of  $A$  in any order results in the same matroid, which we  
 295 denote as  $\mathcal{M}/A$ .

296 The following statements are standard results in the study of matroids (e.g. see [42]).  
 297 The following theorem is important in the analysis of swap rounding. It is often called the  
 298 strong base exchange property of matroids.

299 ► **Theorem 9.** *Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $B, B'$  be bases. For  $e \in B \setminus B'$ , there  
 300 exists  $e' \in B' \setminus B$  such that  $B - e + e' \in \mathcal{I}$  and  $B' - e' + e \in \mathcal{I}$ .*

301 The next lemma shows that if one contracts elements of an independent set in a matroid,  
 302 bases in the contracted matroid can be used to form bases in the initial matroid.

303 ► **Lemma 10.** *Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $A \in \mathcal{I}$ . Let  $B_A$  be a base in  $\mathcal{M}/A$ .  
304 Then  $A \cup B_A$  is a base in  $\mathcal{M}$ .*

### 305 A forest data structure

306 We need a data structure to represent a forest that supports the necessary operations we  
307 need to implement randomized swap rounding in Section 3. The data structure mainly needs  
308 to facilitate the contraction of edges, including being able to recover the identity of the  
309 original edges after any number of contractions. We enable this by enforcing that when  
310 the data structure is initialized, every edge  $e$  is accompanied with a unique identifier. This  
311 identifier will be associated with the edge regardless of the edge's endpoints changing due to  
312 contraction. The implementation of this step is important to guarantee a fast running time.

313 The data structure is initialized via the function `init`, which takes as input the vertices,  
314 edges, and unique edge identifiers of the forest. `init` initializes an adjacency list  $A$ , stores a  
315 mapping  $f$  of edges to their unique edge identifiers, and creates a disjoint-set data structure  
316  $R$  where every vertex initially is in its own set. The operation `contract` contracts an edge  $uv$   
317 in the forest by identifying the vertices  $u$  and  $v$  as the same vertex. This requires choosing  $u$   
318 or  $v$  to be the new representative (suppose we choose  $u$  without loss of generality), merging  
319 the sets corresponding to  $u$  and  $v$  in  $R$  while making  $u$  the representative of the set in  
320  $R$ , and modifying the adjacency list  $A$  to reflect the changes corresponding to contracting  
321  $uv$  and making  $u$  the representative. After an edge is contracted, the vertex set changes.  
322 We need to support the ability to obtain the edge identifier of an edge in the contracted  
323 forest. The data structure maintains  $f$  under edge contractions and returns unique identifiers  
324 with the operation `orig-edge`. Given an edge  $uv$  that was in the contracted forest at some  
325 point in the past, we also need to support the ability to obtain the edge in the vertex  
326 set of the current contracted forest. We do this using  $R$ , which stores all of the vertices  
327 that have been contracted together in disjoint sets. This operation is supported by the  
328 operation `represented-edge`. Finally, we can copy the graph via the operation `copy`, which  
329 simply enumerates over the vertices, edges, and stored unique identifiers of the edges to  
330 create a new data structure.

331 The following lemma formalizes the preceding description of the data structure. We leave  
332 the formal details of a specific implementation and the proof of the following lemma for the  
333 full version.

334 ► **Lemma 11.** *Let  $F = (V, E)$  be a forest and for all  $e \in E$ , let  $id(e)$  be the unique identifier  
335 of  $e$ . The data structure can be initialized via a call to `init` in  $\tilde{O}(|V|)$  time. For  $i = 0, 1, \dots, k$ ,  
336 let  $F_i = (V_i, E_i)$  be the forest after  $i$  calls to `contract`, so  $F_0 = F$  and  $F_k$  is the current state  
337 of the forest. The data structure supports the following operations.*

338 ■ `orig-edge( $e$ )`: *input is an edge  $e \in E_k$ . Output is the identifier  $id(e)$  that was provided  
339 when  $e$  was added to the data structure. Running time is  $\tilde{O}(1)$ .*

340 ■ `represented-edge( $e$ )`: *input is two vertices  $u, v \in V_i$  for some  $i = 0, 1, \dots, k$ . Output is the  
341 pair  $\{u_r, v_r\}$ , where  $u_r$  and  $v_r$  are the vertices in  $V_k$  that correspond to  $u$  and  $v$  in the  
342 contracted forest  $F_k$ . Running time is  $O(1)$ .*

343 ■ `contract( $uv, z \in \{u, v\}$ )`: *input is two vertices  $u, v \in V_k$ . The operation contracts  $uv$  in  
344  $E_k$ , setting the new vertex in  $F_{k+1} = (V_{k+1}, E_{k+1})$  to be  $\{u, v\} \setminus \{z\}$ . The amortized  
345 running time is  $\tilde{O}(\deg_{F_k}(z))$ .*

346 ■ `copy()`: *output is a forest data structure with vertices  $V_k$  and edges  $E_k$  along with the  
347 stored edge identifiers. Running time is  $\tilde{O}(|V_k|)$ .*



```

merge-bases( $\delta, B, \delta', B'$ )
  while  $B \setminus B' \neq \emptyset$  do
     $e \leftarrow$  arbitrary element of  $B \setminus B'$ 
     $e' \leftarrow$  element of  $B' \setminus B$  such that  $B - e + e' \in \mathcal{I}$  and  $B' - e' + e \in \mathcal{I}$ 
     $b \leftarrow 1$  with probability  $\frac{\delta}{\delta + \delta'}$  and 0 otherwise
    if  $b = 1$  then
       $B \leftarrow B - e + e'$ 
    else
       $B' \leftarrow B' - e' + e$ 
    end if
  end while
  return  $B$ 

swap-round( $\delta_1, B_1, \dots, \delta_h, B_h$ )
   $C_1 \leftarrow B_1$ 
  for  $k$  from 1 to  $h - 1$  do
     $C_{k+1} \leftarrow$  merge-bases( $\sum_{i=1}^k \delta_i, C_k, \delta_{k+1}, B_{k+1}$ )
  end for
  return  $C_h$ 

```

■ **Figure 1** The randomized swap rounding algorithm from [20].

### 3 Fast swap rounding in the spanning tree polytope

Randomized swap rounding, developed in [20], is a dependent rounding scheme for rounding a fractional point  $x$  in the base polytope of a matroid to a random base  $X$ . The rounding preserves expectation in that  $\mathbb{E}[X] = x$ , and more importantly, the coordinates of  $X$  are negatively correlated. In this section we prove Theorem 7 on a fast algorithm for swap-rounding in the spanning tree polytope. We begin by describing swap-rounding.

#### 3.1 Randomized swap rounding

Let  $\mathcal{M} = (N, \mathcal{I})$  be a matroid and let  $\mathcal{P}$  be the base polytope of  $\mathcal{M}$  (convex hull of the characteristic vectors of the bases of  $\mathcal{M}$ ). Any  $x \in \mathcal{P}$  can be written as a finite convex combination of bases:  $x = \sum_{i=1}^h \delta_i \mathbf{1}_{B_i}$ . Note that this combination is not necessarily unique. As in [20], we give the original algorithm for randomized swap rounding via two routines. The first is `merge-bases`, which takes as input two bases  $B, B'$  and two real values  $\delta, \delta' \in (0, 1)$ . If  $B = B'$  the algorithm outputs  $B$ . Otherwise the algorithm finds a pair of elements  $e, e'$  such that  $e \in B \setminus B'$  and  $e' \in B' \setminus B$  where  $B - e + e' \in \mathcal{I}$  and  $B' - e' + e \in \mathcal{I}$ . For such  $e$  and  $e'$ , we say that they are a *valid exchange pair* and that we *swap*  $e$  with  $e'$ . The existence of such elements is guaranteed by the strong base exchange property of matroids in Theorem 9. The algorithm randomly retains  $e$  or  $e'$  in both bases with appropriate probability and this increases the intersection size of  $B$  and  $B'$ . The algorithm repeats this process until  $B = B'$ . The overall algorithm `swap-round` utilizes `merge-bases` as a subroutine and repeatedly merges the bases until only one base is left. A formal description is in Figure 1 along with the pseudocode for `merge-bases`.

It is shown in [20] that swap-rounding generates a random base/extreme point  $X \in \mathcal{P}$  (note that the extreme points of  $\mathcal{P}$  are characteristic vectors of bases) such that  $\mathbb{E}[X] = x$  and the coordinates  $X_1, X_2, \dots, X_n$  (here  $|N| = n$ ) are negatively correlated. We observe

372 that swap-rounding does not lead to a unique probability distribution on the bases (that  
 373 depends only  $x$ ). First, as we already noted, the convex decomposition of  $x$  into bases is not  
 374 unique. Second, both `merge-bases` and `swap-round` are non-deterministic in their choices of  
 375 which element pairs to swap and in which order to merge bases in the convex decomposition.  
 376 The key property for negative correlation, as observed in [20], is to view the generation of  
 377 the final base  $B$  as a vector-valued martingale (which preserves expectation in each step)  
 378 that changes only two coordinates in each step. Another rounding strategy, namely pipage  
 379 rounding, also enjoys this property. Nevertheless swap-rounding is a meta algorithm that  
 380 has certain clearly defined features. The flexibility offered by `merge-bases` and `swap-round`  
 381 are precisely what allow for faster implementation in specific settings.

382 We say that  $\bar{B} \stackrel{d}{=} \text{merge-bases}(\delta, B, \delta', B')$  if for some non-deterministic choice of valid  
 383 exchange pairs in the algorithm,  $\bar{B}$  is the random output of `merge-bases`( $\delta, B, \delta', B'$ ). Similarly  
 384 we say that  $B \stackrel{d}{=} \text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h)$  if  $B$  is the random output of the `swap-round`  
 385 process for some non-deterministic choice of the order in which bases are merged and  
 386 some non-deterministic choices in the merging of bases. It follows from [20] that if  $B \stackrel{d}{=} \text{swap-round}(\delta_1, B_1, \dots, \delta_h, B_h)$   
 387 then  $B$  satisfies the property that  $\mathbb{E}[B] = x$  and coordinates  
 388 of  $B$  are negatively correlated.

### 389 3.2 Setup for fast implementation in graphs

390 Let  $G = (V, E)$  be a multigraph with  $|V| = n$  and  $|E| = m$  and let  $x \in \text{ST}(G)$  be a fractional  
 391 spanning tree. Swap rounding requires decomposing  $x$  into a convex combination of spanning  
 392 trees. This step is itself non-trivial; existing algorithms have a high polynomial dependence  
 393 on  $n, m$ . Instead we will settle for an *approximate* decomposition that has some very useful  
 394 features. We state a theorem (in fact a corollary of a theorem) from [16] in a slightly modified  
 395 form suitable for us.

396 ► **Theorem 12** (Paraphrase of Corollary 1.2 in [16]). *Given a graph  $G = (V, E)$  with  $n = |V|$   
 397 and  $m = |E|$  and a rational vector  $x \in [0, 1]^m$  there is a deterministic polynomial-time  
 398 algorithm that runs in  $\tilde{O}(m/\epsilon^2)$  time and either correctly reports that  $x \notin \text{ST}(G)$  or outputs  
 399 an implicit convex decomposition of  $z$  into spanning trees such that  $z \leq (1 + \epsilon)x$ .*

400 The MWU algorithm behind the preceding theorem outputs a convex decomposition  
 401 of  $z = \sum_{i=1}^h \delta_i \mathbf{1}_{T_i}$  for  $h = \tilde{O}(m/\epsilon^2)$  but in an implicit fashion. It outputs  $T = T_1$  and a  
 402 sequence of tuples  $(\delta_i, E_i, E'_i)$  where  $T_{i+1} = T_i - E_i + E'_i$  for  $1 \leq i < h$  and has the property  
 403 that  $\sum_{i=1}^{h-1} (|E_i| + |E'_i|) = \tilde{O}(m/\epsilon^2)$ . Thus the convex decomposition of  $z$  is rather sparse and  
 404 near-linear in  $m$  for any fixed  $\epsilon > 0$ . We will take advantage of this and swap-round  $z$  via this  
 405 implicit convex decomposition. For many applications of interest, including `CROSSING-MST`,  
 406 the fact that we randomly round  $z$  instead of  $x$  does not make much of a difference in the  
 407 overall approximation since  $x$  itself in our setting is the output of an approximate LP solver.

408 ► **Remark 13.** The output of the approximate LP solver based on MWU for `CROSSING-MST`  
 409 has the implicit decomposition as outlined in the preceding paragraph. However, for the sake  
 410 of a self-contained result as stated in Theorem 7, we use the result from [16] which also has  
 411 the advantage of being deterministic.

412 The rest of the section describes a fast implementation for `swap-round`. The algorithm  
 413 is based on a divide and conquer strategy for implementing `swap-round` when the convex  
 414 combination is described in an implicit and compact fashion. An important ingredient is a  
 415 fast black-box implementation of `merge-bases`. For this we use the following result; as we  
 416 remarked earlier, an earlier version of this paper obtained a similar result.

417 ► **Theorem 14** (Ene and Nguyễn [23]). *Let  $T$  and  $T'$  be spanning trees of a graph  $G = (V, E)$   
 418 with  $|V| = n$  and  $E = T \cup T'$  and let  $\delta, \delta' \in (0, 1)$ . There exists an algorithm `fast-merge`  
 419 such that `fast-merge`( $\delta, T, \delta', T'$ )  $\stackrel{d}{=} \text{merge-bases}(\delta, T, \delta', T')$  and the call to `fast-merge` runs in  
 420  $O(n \log^2 n)$  time.*

### 421 3.3 Fast implementation of swap-round

422 In this subsection the goal is to prove the following theorem.

423 ► **Theorem 15.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{T_i}$  be a convex combination of spanning trees of the graph  
 424  $G = (V, E)$  where  $n = |V|$ . Let  $T$  be a spanning tree such that  $T = T_1$  and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$   
 425 be a sequence of sets of edges such that  $T_{i+1} = T_i - E_i + E'_i$  for all  $i \in [h-1]$  and  $E_i \cap E'_i = \emptyset$   
 426 for all  $i \in [h-1]$ . Then there exists an algorithm that takes as input  $T$ ,  $\{(E_i, E'_i)\}_{i=1}^{h-1}$ , and  
 427  $\{\delta_i\}_{i=1}^h$  and outputs a tree  $T_S$  such that  $T_S \stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ . The running  
 428 time of the algorithm is  $\tilde{O}(n + \gamma)$  time where  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ .*

#### 429 A divide and conquer approach

430 We consider the swap rounding framework in the setting of arbitrary matroids for simplicity.  
 431 We work with the implicit decomposition of the convex combination of bases  $\sum_{i=1}^h \delta_i \mathbb{1}_{B_i}$  of  
 432 the matroid  $\mathcal{M} = (N, \mathcal{I})$ , as described in Theorem 15. That is, the input is a base  $B$  such  
 433 that  $B = B_1$ , a sequence of sets of elements  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  such that  $B_{i+1} = B_i - E_i + E'_i$   
 434 and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h-1]$ , and the sequence of coefficients  $\{\delta_i\}_{i=1}^h$ .

435 The pseudocode for our divide and conquer algorithm `divide-and-conquer-swap` is given in  
 436 Figure 2. The basic idea is simple. We *imagine* constructing an explicit convex decomposition  
 437  $B_1, B_2, \dots, B_h$  from the implicit one. The high-level idea is to recursively apply swap rounding  
 438 to  $B_1, \dots, B_{h/2}$  to create a base  $B$ , and similarly create a base  $B'$  by recursively applying swap  
 439 rounding to  $B_{h/2+1}, \dots, B_h$ , and then merging  $B$  and  $B'$ . The advantage of this approach is  
 440 manifested in the implicit case. To see this, we observe that in `merge-bases`( $\delta, B, \delta', B'$ ), the  
 441 intersection  $B \cap B'$  is always in the output, and this implies that the intersection  $\bigcap_{i=1}^h B_i$  will  
 442 always be in the output of `swap-round`( $\delta_1, B_1, \dots, \delta_h, B_h$ ). Therefore, at every recursive level,  
 443 we simply contract the intersection prior to merging any bases. Note that this is slightly  
 444 complicated by the fact that the input is an implicit representation. However, we note that  
 445  $B \cap \bigcup_{i=1}^{h-1} (E_i \cup E'_i) = B \setminus \bigcap_{i=1}^h B_i$  as  $E_i \cup E'_i = B_i \Delta B_{i+1}$  for all  $i \in [h-1]$  where  $S \Delta U$   
 446 denotes the symmetric difference of the sets  $S, U$  (see full version for more details). (We  
 447 note later how the contraction of elements helps in the running time when specializing to the  
 448 graphic matroid.) After contracting the intersection, the algorithm recursively calls itself on  
 449 the first  $h/2$  bases and the second  $h/2$  bases, then merges the output of the two recursive  
 450 calls via `merge-bases`. With the given implicit representation, this means that the input to  
 451 the first recursive call is  $B_1, \{(E_i, E'_i)\}_{i=1}^{h/2-1}, \{\delta_i\}_{i=1}^{h/2}$  and the input to the second recursive  
 452 call is  $B_{h/2+1}, \{(E_i, E'_i)\}_{i=h/2+1}^{h-1}, \{\delta_i\}_{i=h/2+1}^h$  (note we can easily construct  $B_{h/2+1}$  via the  
 453 implicit representation). The underlying matroid in the call to `merge-bases` is the matroid  
 454  $\mathcal{M}$  with the intersection  $\bigcap_{i=1}^h B_i$  contracted.

455 The following lemma shows that `divide-and-conquer-swap` is a proper implementation of  
 456 `swap-round`. We leave the details of the proof for the full version of the paper.

457 ► **Lemma 16.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{B_i}$  be a convex combination of bases in the matroid  $\mathcal{M}$  and  
 458  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of elements such that  $B_{i+1} = B_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i$ .  
 459 Then `swap-round`( $\delta_1, B_1, \dots, \delta_h, B_h$ )  $\stackrel{d}{=} \text{divide-and-conquer-swap}(B_1, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h)$ .*

```

divide-and-conquer-swap( $B, \{(E_i, E'_i)\}_{i=s}^{t-1}, \{\delta_i\}_{i=s}^t$ )
  if  $s = t$  then
    return  $B$ 
  end if
   $\ell \leftarrow \max \left\{ \ell' \in \{s, s+1, \dots, t\} : \sum_{i=s}^{\ell'-1} |E_i| \leq \frac{1}{2} \sum_{i=s}^{t-1} |E_i| \right\}$ 
   $\hat{B} \leftarrow B \cap \bigcup_{i=s}^{t-1} (E_i \cup E'_i)$ 
   $\hat{B}_C \leftarrow \hat{B}$ 
  for  $i$  from  $s$  to  $\ell$  do
     $\hat{B}_C \leftarrow \hat{B}_C - E_i + E'_i$ 
  end for
   $\hat{B}_L \leftarrow \text{divide-and-conquer-swap}(\hat{B}, \{(E_i, E'_i)\}_{i=s}^{\ell-1}, \{\delta_i\}_{i=s}^{\ell})$ 
   $\hat{B}_R \leftarrow \text{divide-and-conquer-swap}(\hat{B}_C, \{(E_i, E'_i)\}_{i=\ell+1}^{t-1}, \{\delta_i\}_{i=\ell+1}^t)$ 
   $\hat{B}_M \leftarrow \text{merge-bases}(\sum_{i=s}^{\ell} \delta_i, \hat{B}_L, \sum_{i=\ell+1}^t \delta_i, \hat{B}_R)$ 
  return  $\hat{B}_M \cup (B \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i))$ 

```

■ **Figure 2** A divide-and-conquer implementation of swap rounding with an implicit representation.

#### 460 A fast implementation of divide-and-conquer-swap for spanning trees

461 The pseudocode for our fast implementation `fast-swap` of `divide-and-conquer-swap` is given in  
 462 Figure 4.

463 As in `divide-and-conquer-swap`, the algorithm `fast-swap` contracts the intersection of the  
 464 input. Suppose we contract the intersection  $\bigcap_{i=1}^h T_i$  in  $T_j$  and call this contracted tree  $\hat{T}_j$ .  
 465 Then  $|\hat{T}_j| \leq |T_j \setminus \bigcap_{i=1}^h T_i|$ . A simple argument shows that  $T_j \setminus \bigcap_{i=1}^h T_i \subseteq \bigcup_{i=1}^{h-1} (T_i \Delta T_{i+1}) =$   
 466  $\bigcup_{i=1}^{h-1} (E_i \cup E'_i)$  (see full version for more details). Thus, the size of the contracted tree is  
 467 bounded by the size of the implicit representation  $\gamma := \sum_{i=1}^{h-1} |E_i| + |E'_i|$ . One can write a  
 468 convex combination of bases in any matroid using the implicit representation, and contraction  
 469 could even be implemented quickly as is the case in the graphic matroid. The main point for  
 470 improving the running time is having an implementation of `merge-bases` that runs in time  
 471 proportional to the size of the *contracted* matroid. This is key to the speedup achieved for the  
 472 graphic matroid. `fast-merge` runs in time proportional to the size of the input trees, which  
 473 have been contracted to have size  $O(\min\{n, \gamma\})$ , which yields a running time of  $\tilde{O}(\min\{n, \gamma\})$ .  
 474 This speedup at every recursive level combined with the divide-and-conquer approach of  
 475 `fast-swap` is sufficient to achieve a near-linear time implementation of `swap-round`.

476 Recall that as we are working with contracted trees, an edge in the contracted trees  
 477 might have different endpoints than it did in the initial trees. The identifiers of edges do not  
 478 change, regardless of whether the endpoints of the edge change due to contraction of edges  
 479 in a tree. We therefore will refer to id's of edges throughout the algorithm `fast-swap` to work  
 480 from contracted edges back to edges in the initial trees. This extra bookkeeping will mainly  
 481 be handled implicitly.

482 Contraction of the intersection of the input trees in `fast-swap` using only the implicit  
 483 representation is handled by the algorithm `shrink-intersection` and we give the pseudocode  
 484 in Figure 3. Consider spanning trees  $T_s, T_{s+1}, \dots, T_t$ . The input to `shrink-intersection`  
 485 is  $T_s$  and a sequence of sets of edges  $\{(E_i, E'_i)\}_{i=s}^{t-1}$  such that  $T_{i+1} = T_i - E_i + E'_i$  and  
 486  $E_i \cap E'_i = \emptyset$  for  $i \in \{s, s+1, \dots, t-1\}$ . Then `shrink-intersection` contracts  $\bigcap_{i=s}^t T_i$  in  $T_s$ .  
 487 It is then easy to see that one can compute the intersection via the edges  $\{(E_i, E'_i)\}_{i=s}^{t-1}$  as  
 488  $\bigcap_{i=s}^t T_i = T_s \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i)$  (see full version for more details). Let  $\hat{T}_s$  be  $T_s$  with  $\bigcap_{i=s}^t T_i$

```

shrink-intersection( $T, \{(E_i, E'_i)\}_{i=s}^{t-1}$ )
 $\hat{T} \leftarrow T.\text{copy}()$ 
for  $e \in T \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i)$  do
     $uv \leftarrow \hat{T}.\text{represented-edge}(e)$ 
    assume  $\deg_{\hat{T}}(u) \leq \deg_{\hat{T}}(v)$  (otherwise rename)
     $\hat{T}.\text{contract}(uv, u)$ 
end for
let  $id(e)$  denote the unique identifier of an edge  $e$ 
for  $i$  from  $s$  to  $t - 1$  do
     $\hat{E}_i \leftarrow \bigcup_{e \in E_i} (\hat{T}.\text{represented-edge}(e), id(e))$ 
     $\hat{E}'_i \leftarrow \bigcup_{e \in E'_i} (\hat{T}.\text{represented-edge}(e), id(e))$ 
end for
return  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{t-1})$ 

```

■ **Figure 3** A subroutine used in our fast implementation `fast-swap` of randomized swap rounding; used to implicitly contract the trees of the given convex combination.

489 contracted. The vertex set of  $\hat{T}_s$  is different than the vertex set of  $T_s$ . Then as the sets of  
490 edges  $E_i$  and  $E'_i$  for all  $i$  are defined on the vertices in  $T_s$ , we need to map the endpoints of  
491 edges in  $E_i$  to the new vertex set of  $\hat{T}_s$ . Using the data structure presented in Lemma 11,  
492 this is achieved using the operations `represented-edge` and `orig-edge`, which handle mapping  
493 the edge to its new endpoints and maintaining the edge identifier, respectively.

494 The following lemma shows that `shrink-intersection` indeed contracts the intersection of  
495 the trees via the implicit representation. We leave the details of the proof for the full version.

496 ► **Lemma 17.** *Let  $T_1, \dots, T_h$  be spanning trees and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$  be a sequence of edge  
497 sets defined on the same vertex set such that  $T_{i+1} = T_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  
498  $i \in [h - 1]$ . Contract  $\bigcap_{i=1}^h T_i$  in  $T_1, \dots, T_h$  to obtain  $\hat{T}_1, \dots, \hat{T}_h$ , respectively.*

499 *Let  $n_{T_1} = |T_1|$  and  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ . Then `shrink-intersection`( $T_1, \{(E_i, E'_i)\}_{i=1}^{h-1}$ )  
500 runs in time  $\tilde{O}(n_{T_1} + \gamma)$  and outputs  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=1}^{h-1})$  where  $\hat{T} = \hat{T}_1$  and  $\hat{T}_{i+1} = \hat{T}_i - \hat{E}_i + \hat{E}'_i$   
501 for all  $i \in [h - 1]$ . Moreover,  $|E_i| = |\hat{E}_i|$  and  $|E'_i| = |\hat{E}'_i|$  for all  $i \in [h - 1]$  and  $|\hat{T}| \leq$   
502  $\min\{n_{T_1}, \gamma\}$ .*

503 We use the algorithm in Theorem 14 for merge-bases. In `fast-swap`, the two trees that  
504 are merged  $\hat{T}_L$  and  $\hat{T}_R$  are the return values of the two recursive calls to `fast-swap`. The  
505 algorithm at this point has explicit access to the adjacency lists of both  $\hat{T}_L$  and  $\hat{T}_R$ , which  
506 are used as input to the algorithm `fast-merge`. The output of `fast-merge` will be the outcome  
507 of merging the two trees  $\hat{T}_L$  and  $\hat{T}_R$ , which are edges of potentially contracted trees from  
508 the original convex combination. We can use the operation `orig-edge` of the forest data  
509 structure of Lemma 11 for  $\hat{T}_L$  and  $\hat{T}_R$  to obtain the edges from the trees of the original  
510 convex combination. This extra bookkeeping will be handled implicitly.

511 We next prove that `fast-swap` is implementing `swap-round` and that it runs in near-linear  
512 time.

513 ► **Lemma 18.** *Let  $\sum_{i=1}^h \delta_i \mathbb{1}_{T_i}$  be a convex combination of spanning trees of the graph  
514  $G = (V, E)$  where  $n = |V|$ . Let  $T$  be a spanning tree such that  $T = T_1$  and let  $\{(E_i, E'_i)\}_{i=1}^{h-1}$   
515 be a sequence of sets of edges such that  $T_{i+1} = T_i - E_i + E'_i$  and  $E_i \cap E'_i = \emptyset$  for all  $i \in [h - 1]$ .*

516 *Then `fast-swap`( $T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h$ )  $\stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$  and the call to  
517 `fast-swap` runs in  $\tilde{O}(n_T + \gamma)$  time where  $n_T = |T|$  and  $\gamma = \sum_{i=1}^{h-1} (|E_i| + |E'_i|)$ .*

```

fast-swap( $T, \{(E_i, E'_i)\}_{i=s}^{t-1}, \{\delta_i\}_{i=s}^t$ )
  if  $s = t$  then
    return  $T$ 
  end if
   $\ell \leftarrow \max \left\{ \ell' \in \{s, s+1, \dots, t\} : \sum_{i=s}^{\ell'-1} |E_i| \leq \frac{1}{2} \sum_{i=s}^{t-1} |E_i| \right\}$ 
   $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{t-1}) \leftarrow \text{shrink-intersection}(T, \{(E_i, E'_i)\}_{i=s}^{t-1})$ 
   $\hat{E}_C \leftarrow E(\hat{T})$ 
  for  $i$  from  $s$  to  $\ell$  do
     $\hat{E}_C \leftarrow \hat{E}_C - \hat{E}_i + \hat{E}'_i$ 
  end for
   $\hat{T}_C \leftarrow \text{init}(V(\hat{T}), \hat{E}_C)$ 
   $\hat{T}_L \leftarrow \text{fast-swap}(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=s}^{\ell-1}, \{\delta_i\}_{i=s}^{\ell})$ 
   $\hat{T}_R \leftarrow \text{fast-swap}(\hat{T}_C, \{(\hat{E}_i, \hat{E}'_i)\}_{i=\ell+1}^{t-1}, \{\delta_i\}_{i=\ell+1}^t)$ 
   $\hat{T}_M \leftarrow \text{fast-merge}(\sum_{i=s}^{\ell} \delta_i, \hat{T}_L, \sum_{i=\ell+1}^t \delta_i, \hat{T}_R)$ 
  return  $\hat{T}_M \cup (T \setminus \bigcup_{i=s}^{t-1} (E_i \cup E'_i))$ 
    
```

■ **Figure 4** A fast implementation of randomized swap rounding from [20].

518 **Proof.** One can immediately see `fast-swap` is an implementation of `divide-and-conquer-swap`.  
 519 There are some bookkeeping details that are left out of the implementation, such as main-  
 520 taining the set of edges returned by `fast-merge`, but these are easily handled. Lemma 16  
 521 shows that `divide-and-conquer-swap`( $\delta_1, T_1, \dots, \delta_h, T_h$ )  $\stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ , im-  
 522 plying `fast-swap`( $T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h$ )  $\stackrel{d}{=} \text{swap-round}(\delta_1, T_1, \dots, \delta_h, T_h)$ .

523 Now we prove the running time bound. Let  $R(n_T, \gamma)$  denote the running time of the  
 524 call to `fast-swap`( $T, \{(E_i, E'_i)\}_{i=1}^{h-1}, \{\delta_i\}_{i=1}^h$ ). By Lemma 17, the running time of the call to  
 525 `shrink-intersection` is  $\tilde{O}(n_T + \gamma)$ . Let  $(\hat{T}, \{(\hat{E}_i, \hat{E}'_i)\}_{i=1}^{h-1})$  be the output of `shrink-intersection`.  
 526 Lemma 17 also guarantees that  $\gamma = \sum_{i=1}^{h-1} |\hat{E}_i| + |\hat{E}'_i|$  and  $|\hat{T}| \leq \min\{n_T, \gamma\}$ . Then by  
 527 Lemma 11, constructing  $\hat{T}_C$  requires  $\tilde{O}(n_{\hat{T}} + \gamma)$  time. As the size of  $\hat{T}_L$  and  $\hat{T}_R$  is the same  
 528 as  $\hat{T}$  and  $\hat{T}_C$ , the call to `fast-merge` runs in  $\tilde{O}(n_T + \gamma)$  time by Theorem 14. The time it  
 529 takes to compute the returned tree is  $\tilde{O}(n_T + \gamma)$  as we have enough time to scan all of  $T$   
 530 and  $\bigcup_{i=1}^{h-1} (E_i \cup E'_i)$ . So the total time excluding the recursive calls is  $(n_T + \gamma) \cdot \alpha$  for where  
 531  $\alpha = O(\log^c(n_T + \gamma))$  for some fixed integer  $c$ .

532 As for the recursive calls, first define  $\gamma(s, t) := \sum_{i=s}^t (|E_i| + |E'_i|)$ . Then the running time of  
 533 the first recursive call is  $R(n_{\hat{T}}, \gamma(1, \ell-1))$  and the second recursive call is  $R(n_{\hat{T}_C}, \gamma(\ell+1, h-1))$ .  
 534 By choice of  $\ell$ , we always have that  $\gamma(1, \ell-1) \leq \frac{\gamma}{2}$ . As  $\ell$  is the largest integer such that  
 535  $\gamma(1, \ell-1) \leq \frac{\gamma}{2}$ , then  $\gamma(1, \ell) > \frac{\gamma}{2}$ . Therefore, we have  $\gamma(\ell+1, h-1) = \gamma - \gamma(1, \ell) < \frac{\gamma}{2}$ .  
 536 Combining this with the fact that `shrink-intersection` guarantees that  $|\hat{T}| \leq \min\{n_T, \gamma\}$  and  
 537  $|\hat{T}_C| \leq \min\{n_T, \gamma\}$ , we have

$$538 \quad R(n_T, \gamma) \leq 2R(\min\{n_T, \gamma\}, \gamma/2) + \alpha \cdot (n_T + \gamma).$$

539 Note that  $R(n_T, \gamma) = O(1)$  when  $n_T = O(1)$  and  $\gamma = O(1)$ .

540 We claim that  $R(a, b) \leq \alpha\beta \cdot (a + 8b \log b)$  is a valid solution to this recurrence for some  
 541 sufficiently large but fixed constant  $\beta \geq 1$ . By choosing  $\beta$  sufficiently large it is clear that it  
 542 holds for the base case. To prove the inductive step we see the following:

$$543 \quad R(a, b) \leq 2R(\min\{a, b\}, b/2) + \alpha \cdot (a + b) \leq 2[\alpha\beta \cdot (\min\{a, b\} + 4b \log(b/2))] + \alpha \cdot (a + b).$$

544 Hence we need to verify that

$$545 \quad 2[\alpha\beta(\min\{a, b\} + 4b \log(b/2))] + \alpha \cdot (a + b) \leq \alpha\beta \cdot (a + 8b \log b). \quad (1)$$

546 Since  $\beta \geq 1$ , rearranging, it suffices to verify that

$$547 \quad 2 \min\{a, b\} + 8b \log(b/2) + b \leq 8b \log b.$$

548 As  $8b \log b - 8b \log(b/2) = 8b$  and  $2 \min\{a, b\} + b \leq 3b$ , this proves (1) and therefore  
 549  $R(n_T, \gamma) \leq \alpha\beta(n_T + 8\gamma \log \gamma) = \tilde{O}(n_T + \gamma)$ . This concludes the proof. ◀

550 The proof of Theorem 7 then follows by combining Theorem 12 (and remarks after the  
 551 theorem statement) and Theorem 15.

## 552 **4 Sparsification via the LP Solution**

553 Let  $G = (V, E)$  be a graph on  $n$  nodes and  $m$  edges and let  $x$  be a point in  $\text{ST}(G)$ . In  
 554 this section we discuss Theorem 4, which shows that, via random sampling, one can obtain  
 555 a sparse point  $x' \in \text{ST}(G)$  from  $x$ . The random sampling approximately preserves linear  
 556 constraints and thus one can use this technique to obtain sparse LP solutions to the packing  
 557 problems involving spanning tree (and more generally matroid) constraints. The sampling  
 558 and analysis rely on Karger's well-known work on random sampling for packing disjoint bases  
 559 in matroids. We paraphrase the relevant theorem.

560 ▶ **Theorem 19** (Corollary 4.12 from [30]). *Let  $\mathcal{M}$  be a matroid with  $m$  elements and non-*  
 561 *negative integer capacities on elements such that  $\mathcal{M}$  has  $k$  disjoint bases. Suppose each*  
 562 *copy of a capacitated element  $e$  is sampled independently with probability  $p \geq 18(\ln m)/(k\epsilon^2)$*   
 563 *yielding a matroid  $\mathcal{M}(p)$ . Then with high probability the number of disjoint bases in  $\mathcal{M}(p)$  is*  
 564 *in  $[(1 - \epsilon)pk, (1 + \epsilon)pk]$ .*

565 We restate Theorem 4 for the reader's convenience. We leave the details of the proof to  
 566 the full version.

567 ▶ **Theorem 20.** *Let  $x \in \text{ST}(G)$  be a rational vector such that  $Ax \leq b$  for a matrix  $A \in$*   
 568  *$[0, 1]^{r \times m}$  and  $b \in [1, \infty)^r$ . Consider a random subgraph  $G' = (V, E')$  of  $G$  obtained by picking*  
 569 *each edge  $e \in G$  with probability  $\alpha_e := \min\{1, \frac{36 \log(r+m)}{\epsilon^2} \cdot x_e\}$ . Then with high probability the*  
 570 *following hold: (i)  $|E'| = O(n \ln(r+m)/\epsilon^2)$  (ii) there exists a fractional solution  $z \in \text{ST}(G)$*   
 571 *in the support of  $G'$  such that  $Az \leq (1 + 3\epsilon)b$ .*

572 ▶ **Remark 21.** For problems that also involve costs, we have a fractional solution  $\mathbf{x}$  and an  
 573 objective  $\sum_e c_e x_e$ . Without loss of generality we can assume that  $c_e \in [0, 1]$  for all  $e$ . The  
 574 preceding proof shows that the sparse graph obtained by sampling supports a fractional  
 575 solution  $\mathbf{z}$  such that  $\mathbb{E}[\sum_e c_e z_e] \leq (1 + \epsilon) \sum_e c_e x_e$ . Further,  $\sum_e c_e z_e \leq (1 + 3\epsilon) \sum_e c_e x_e$  holds  
 576 with high probability as long as  $\max_e c_e \leq \sum_e c_e x_e$ . This condition may not hold in general  
 577 but can be typically guaranteed in the overall algorithm by guessing the largest cost edge in  
 578 an optimum integer solution.

579 ▶ **Remark 22.** The proof in the preceding theorem also shows that with high probability  
 580 the graph  $G'$  is connected and satisfies the property that  $A\mathbb{1}_{E'} \leq O(\log n/\epsilon^2)b$ . Thus any  
 581 spanning tree of  $G'$  satisfies the constraints to a multiplicative  $O(\log n)$ -factor by fixing  $\epsilon$  to  
 582 a small constant. This is weaker than the guarantee provided by swap-rounding.

583 **5 Fast approximation scheme for solving the LP relaxation**

584 In this section, we discuss Theorem 1, which gives a fast approximation scheme to solve the  
 585 LP relaxation for CROSSING-ST. We recall the LP for CROSSING-ST.

$$\begin{aligned}
 & \min && \sum_{e \in E} c_e y_e \\
 & \text{subject to} && Ay \leq b \\
 & && y \in \text{ST}(G)
 \end{aligned} \tag{P}$$

587 Note that even the feasibility problem (whether there exists  $y \in \text{ST}(G)$  such that  $Ay \leq b$ )  
 588 is interesting and important. We will mainly focus on the feasibility LP and show how we  
 589 can incorporate costs in the full version of the paper. We recast the feasibility LP as a pure  
 590 packing LP using an implicit formulation with an exponential number of variables. This is  
 591 for technical convenience and to more directly apply the framework from [18]. For each tree  
 592  $T \in \mathcal{T}(G)$  we have a variable  $x_T$  and we consider the problem of packing spanning trees.

$$\begin{aligned}
 & \text{maximize} && \sum_{T \in \mathcal{T}} x_T \\
 & \text{subject to} && \sum_{T \in \mathcal{T}} (A\mathbb{1}_T)_i \cdot x_T \leq b_i, \quad \forall i \in [k] \\
 & && x_T \geq 0, \quad \forall T \in \mathcal{T}
 \end{aligned} \tag{C}$$

594 The following is easy to establish from the fact that  $y \in \text{ST}(G)$  iff  $y$  can be written as a  
 595 convex combination of the characteristic vectors of spanning trees of  $G$ .

596 ► **Observation 23.** *There exists a feasible solution  $y$  to  $\mathcal{P}$  iff there exists a feasible solution*  
 597  *$x$  to  $\mathcal{C}$  with value at least 1. Further, if  $x$  is a feasible solution to  $\mathcal{C}$  with value  $(1 - \epsilon)$  there*  
 598 *exists a solution  $y$  such that  $y \in \text{ST}(G)$  and  $Ay \leq \frac{1}{1-\epsilon}b$ .*

599  $\mathcal{C}$  is a pure packing LP, albeit in *implicit* form. In the full version of the paper, we show  
 600 how to approximately solve the LP using MWU techniques and in particular we use the  
 601 randomized MWU framework from [18] for positive LPs. The full version reviews the MWU  
 602 framework and shows how to apply it along with other implementation details to achieve the  
 603 concrete run times that we claim in Theorem 1.

604 **References**

- 
- 605 1 Alexander A. Ageev and Maxim Sviridenko. Pipeage rounding: A new method of constructing  
 606 algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*,  
 607 8:307–328, 2004.
- 608 2 N. Anari and S. O. Gharan. Effective-resistance-reducing flows, spectrally thin trees, and  
 609 asymmetric TSP. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*,  
 610 pages 20–39, 2015.
- 611 3 Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials, entropy,  
 612 and a deterministic approximation algorithm for counting bases of matroids. In *2018 IEEE*  
 613 *59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 35–46. IEEE,  
 614 2018.
- 615 4 Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials  
 616 IV: exchange properties, tight mixing times, and faster sampling of spanning trees, 2020.  
 617 [arXiv:2004.07220](https://arxiv.org/abs/2004.07220).



- 618 5 Arash Asadpour, Michel X Goemans, Aleksander Mádry, Shayan Oveis Gharan, and Amin  
619 Saberi. An  $O(\log n / \log \log n)$ -approximation algorithm for the asymmetric traveling salesman  
620 problem. *Operations Research*, 65(4):1043–1061, 2017.
- 621 6 Nikhil Bansal. On a generalization of iterated and randomized rounding. In *Proceedings of the*  
622 *51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1125–1135, 2019.
- 623 7 Nikhil Bansal, Rohit Khandekar, Jochen Könemann, Viswanath Nagarajan, and Britta Peis.  
624 On generalizations of network design problems with degree bounds. *Mathematical Programming*,  
625 141(1-2):479–506, 2013.
- 626 8 Nikhil Bansal, Rohit Khandekar, and Viswanath Nagarajan. Additive guarantees for degree-  
627 bounded directed network design. *SIAM Journal on Computing*, 39(4):1413–1431, Jan 2010.  
628 URL: <http://dx.doi.org/10.1137/080734340>, doi:10.1137/080734340.
- 629 9 Vittorio Bilo, Vineet Goyal, Ramamoorthi Ravi, and Mohit Singh. On the crossing spanning  
630 tree problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms*  
631 *and Techniques*, pages 51–60. Springer, 2004.
- 632 10 Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone  
633 submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–  
634 1766, 2011.
- 635 11 Deeparnab Chakrabarty, Yin Tat Lee, Aaron Sidford, Sahil Singla, and Sam Chiu-wai Wong.  
636 Faster matroid intersection. In *2019 IEEE 60th Annual Symposium on Foundations of*  
637 *Computer Science (FOCS)*, pages 1146–1168. IEEE, 2019.
- 638 12 Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for  
639 minimizing discrepancy. In *Proceedings of the twenty-second annual ACM-SIAM symposium*  
640 *on Discrete Algorithms*, pages 1607–1614. SIAM, 2011.
- 641 13 Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would  
642 Edmonds do? augmenting paths and witnesses for degree-bounded MSTs. *Algorithmica*,  
643 55(1):157–189, Nov 2007. URL: <http://dx.doi.org/10.1007/s00453-007-9115-5>, doi:10.  
644 1007/s00453-007-9115-5.
- 645 14 Chandra Chekuri, Sarel Har-Peled, and Kent Quanrud. Fast LP-based approximations for  
646 geometric packing and covering problems. In *Proceedings of the Fourteenth Annual ACM-SIAM*  
647 *Symposium on Discrete Algorithms*, pages 1019–1038. SIAM, 2020.
- 648 15 Chandra Chekuri and Kent Quanrud. Approximating the Held-Karp bound for metric TSP  
649 in nearly-linear time. In *2017 IEEE 58th Annual Symposium on Foundations of Computer*  
650 *Science (FOCS)*, pages 789–800. IEEE, 2017.
- 651 16 Chandra Chekuri and Kent Quanrud. Near-linear time approximation schemes for some  
652 implicit fractional packing problems. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM*  
653 *Symposium on Discrete Algorithms*, pages 801–820. SIAM, 2017.
- 654 17 Chandra Chekuri and Kent Quanrud. Fast approximations for Metric-TSP via linear program-  
655 ming. *arXiv preprint arXiv:1802.01242*, 2018.
- 656 18 Chandra Chekuri and Kent Quanrud. Randomized MWU for positive LPs. In *Proceedings*  
657 *of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 358–377.  
658 SIAM, 2018.
- 659 19 Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. Fast approximation algorithms for  
660 bounded degree and crossing spanning tree problems. *CoRR*, abs/2011.03194, 2020. URL:  
661 <https://arxiv.org/abs/2011.03194>, arXiv:2011.03194.
- 662 20 Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via  
663 exchange properties of combinatorial structures. In *2010 IEEE 51st Annual Symposium on*  
664 *Foundations of Computer Science*, pages 575–584. IEEE, 2010.
- 665 21 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid  
666 intersection via dependent rounding. In *Proceedings of the twenty-second annual ACM-SIAM*  
667 *symposium on Discrete Algorithms*, pages 1080–1097. SIAM, 2011.
- 668 22 Ran Duan, Haoqing He, and Tianyi Zhang. Near-linear time algorithms for approximate  
669 minimum degree spanning trees. *ArXiv*, abs/1712.09166, 2017.

- 670 23 Alina Ene and Huy L Nguyen. Towards nearly-linear time algorithms for submodular maxi-  
671 mization with a matroid constraint. In *International Colloquium on Automata, Languages,*  
672 *and Programming*, volume 132, 2019.
- 673 24 M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within  
674 one of optimal. *Journal of Algorithms*, 17(3):409–423, Nov 1994. URL: [http://dx.doi.org/](http://dx.doi.org/10.1006/jagm.1994.1042)  
675 [10.1006/jagm.1994.1042](http://dx.doi.org/10.1006/jagm.1994.1042), doi:10.1006/jagm.1994.1042.
- 676 25 Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent  
677 rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*,  
678 53(3):324–360, 2006.
- 679 26 Kyle Genova and David P Williamson. An experimental evaluation of the best-of-many  
680 Christofides’ algorithm for the traveling salesman problem. *Algorithmica*, 78(4):1109–1130,  
681 2017.
- 682 27 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to  
683 the traveling salesman problem. In *2011 IEEE 52nd Annual Symposium on Foundations of*  
684 *Computer Science*, pages 550–559. IEEE, 2011.
- 685 28 Michel Goemans. Minimum bounded degree spanning trees. *2006 47th Annual IEEE Symposium*  
686 *on Foundations of Computer Science (FOCS '06)*, 2006. URL: [http://dx.doi.org/10.1109/](http://dx.doi.org/10.1109/FOCS.2006.48)  
687 [FOCS.2006.48](http://dx.doi.org/10.1109/FOCS.2006.48), doi:10.1109/focs.2006.48.
- 688 29 Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic  
689 fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity.  
690 *Journal of the ACM (JACM)*, 48(4):723–760, 2001.
- 691 30 David R Karger. Random sampling and greedy sparsification for matroid optimization problems.  
692 *Mathematical Programming*, 82(1-2):41–81, 1998.
- 693 31 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation  
694 algorithm for TSP in the half integral case. In *Proceedings of the 52nd Annual ACM SIGACT*  
695 *Symposium on Theory of Computing*, pages 28–39, 2020.
- 696 32 Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation  
697 algorithm for metric TSP. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on*  
698 *Theory of Computing*, pages 32–45, 2021.
- 699 33 Tamás Király, Lap Chi Lau, and Mohit Singh. Degree bounded matroids and submodular  
700 flows. *Combinatorica*, 32(6):703–720, 2012.
- 701 34 Jochen Könemann and R Ravi. Primal-dual meets local search: approximating MST’s with  
702 nonuniform degree bounds. In *Proceedings of the thirty-fifth annual ACM symposium on*  
703 *Theory of computing*, pages 389–395, 2003.
- 704 35 Jochen Könemann and Ramamoorthi Ravi. Primal-dual meets local search: approximating  
705 MSTs with nonuniform degree bounds. *SIAM Journal on Computing*, 34(3):763–773, 2005.
- 706 36 André Linhares and Chaitanya Swamy. Approximating min-cost chain-constrained spanning  
707 trees: a reduction from weighted to unweighted problems. *Mathematical Programming*, 172(1-  
708 2):17–34, 2018.
- 709 37 Neil Olver and Rico Zenklusen. Chain-constrained spanning trees. *Mathematical Programming*,  
710 167(2):293–314, 2018.
- 711 38 Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an  
712 extension of the Chernoff-Hoeffding bounds. *SIAM J. Comput.*, 26:350–368, 1997.
- 713 39 Kent Quanrud. Fast and deterministic approximations for  $k$ -cut. *arXiv preprint*  
714 *arXiv:1807.07143*, 2018.
- 715 40 Kent Quanrud. *Fast approximations for combinatorial optimization via multiplicative weight*  
716 *updates*. PhD thesis, University of Illinois, Urbana-Champaign, 2019. URL: [https://www.](https://www.ideals.illinois.edu/handle/2142/106153)  
717 [ideals.illinois.edu/handle/2142/106153](https://www.ideals.illinois.edu/handle/2142/106153).
- 718 41 Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation.  
719 In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages  
720 214–227, 2018.

- 721 42 Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer  
722 Science & Business Media, 2003.
- 723 43 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to  
724 within one of optimal. *Journal of the ACM (JACM)*, 62(1):1–19, 2015.
- 725 44 Mohit Singh and Nisheeth K Vishnoi. Entropy, optimization and counting. In *Proceedings of*  
726 *the forty-sixth annual ACM symposium on Theory of computing*, pages 50–59, 2014.
- 727 45 Aravind Srinivasan. Distributions on level-sets with applications to approximation algorithms.  
728 In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 588–597.  
729 IEEE, 2001.
- 730 46 Ola Svensson, Jakub Tarnawski, and László A Végh. A constant-factor approximation  
731 algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual*  
732 *ACM SIGACT Symposium on Theory of Computing*, pages 204–213, 2018.
- 733 47 Vera Traub and Jens Vygen. An improved approximation algorithm for ATSP. In *Proceedings*  
734 *of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1–13, 2020.
- 735 48 Di Wang. *Fast Approximation Algorithms for Positive Linear Programs*. PhD thesis, EECS  
736 Department, University of California, Berkeley, Jul 2017. URL: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-126.html>.
- 737
- 738 49 Sein Win. On a connection between the existence of  $k$ -trees and the toughness of a graph.  
739 *Graphs and Combinatorics*, 5(1):201–205, 1989.

## 740 A Putting things together and extensions

741 In the main body of the paper we discussed the main technical results corresponding to  
742 Theorems 1, 4, and 7. In this section we give formal proofs of the corollaries stated in  
743 Section 1. In addition, we also describe some extensions and other related results that follow  
744 from the ideas in the preceding sections.

### 745 A.1 Proofs of corollaries

746 We start with Corollary 3.

747 **Proof of Corollary 3.** Let  $G = (V, E)$  be the input graph with  $m$  edges and  $n$  vertices and  
748 let  $\epsilon > 0$ . Consider the LP relaxation to test whether  $G$  has a spanning tree with degree at  
749 most a given parameter  $B'$ . Theorem 1 implies that there exists a randomized algorithm that,  
750 with high probability, either correctly determines that there is no feasible solution to the LP,  
751 or outputs a fractional spanning tree  $y \in \text{ST}(G)$  such that  $\sum_{e \in \delta(v)} y_e \leq (1 + \epsilon)B'$  for all  $v$ .  
752 Using the algorithm, we can do binary search over the integers from 2 to  $n - 1$  to find the  
753 smallest value  $B$  for which the algorithm outputs a solution. We will focus on the scenario  
754 where the algorithm from Theorem 1 is correct in each of the  $O(\log n)$  calls in the binary  
755 search procedure; this happens with high probability. For the value of  $B$  found in the binary  
756 search, let  $y \in \text{ST}(G)$  be the solution that is output; we have  $\sum_{e \in \delta(v)} y_e \leq (1 + \epsilon)B$  for all  
757  $v$ . Since the approximate LP solver correctly reported infeasibility for all  $B' < B$ , we have  
758  $B - 1 < B^*$ , which implies  $B \leq B^*$ . As there is a feasible fractional spanning tree  $y$  such  
759 that  $\sum_{e \in \delta(v)} y_e \leq \lceil (1 + \epsilon)B \rceil$  for all  $v$ , the result of [43] implies that  $B^* \leq \lceil (1 + \epsilon)B \rceil + 1$ .

760 Regarding the run time, each call to the algorithm in Theorem 1 takes  $\tilde{O}(m/\epsilon^2)$  time since  
761  $N = O(m)$  in the setting of BD-ST. Binary search adds only an  $O(\log n)$  multiplicative-factor  
762 overhead, leading to the claimed running time. ◀

763 We next prove Corollary 5. The algorithm described in the corollary takes advantage of  
764 Theorem 4 to sparsify the input graph, then runs the Fürer-Raghavachari algorithm [24].

765 **Proof of Corollary 5.** We will assume that the input is a graph  $G = (V, E)$   $m$  edges and  
766  $n$  vertices. Let  $\epsilon > 0$ . As in the proof of Corollary 3, we can use Theorem 1 and binary  
767 search to find in  $\tilde{O}(\frac{m}{\epsilon^2})$  time, with high probability, a fractional spanning tree  $x \in \text{ST}(G)$   
768 such that  $\sum_{e \in \delta(v)} x_e \leq (1 + \epsilon)B^*$  for all  $v \in V$ . By Theorem 4, we can use random  
769 sampling based on  $x$  to obtain a subgraph  $G' = (V, E')$  of  $G$  such that with high probability  
770 we have  $|E'| = O(\frac{n \log(n+m)}{\epsilon^2})$  and there exists a fractional solution  $z \in \text{ST}(G)$  in the  
771 support of  $G'$  such that  $\sum_{e \in \delta(v)} z_e \leq (1 + 3\epsilon)(1 + \epsilon)B^* \leq (1 + 7\epsilon)B^*$  for all  $v \in V$  (for  
772 sufficiently small  $\epsilon$ ). The result of [43] implies there exists a spanning tree  $T$  in  $G'$  such  
773 that  $\max_{v \in V} \deg_T(v) \leq \lceil (1 + 7\epsilon)B^* \rceil + 1$ . For a graph  $H$  on  $n$  vertices and  $m$  edges, the  
774 algorithm of [24] runs in  $\tilde{O}(mn)$  time and outputs a spanning tree  $T$  such that the maximum  
775 degree of  $T$  is at most  $\text{OPT}(H) + 1$ , where  $\text{OPT}(H) = \min_{T \in \mathcal{T}(H)} \max_{v \in V} \deg_T(v)$ . Thus,  
776 the algorithm of [24] when applied to  $G'$ , outputs a spanning tree with degree at most  
777  $\lceil (1 + 7\epsilon)B^* \rceil + 2$ , and runs in  $\tilde{O}(\frac{n^2}{\epsilon^2})$  time. ◀

778 We now prove Corollary 8. This corollary combines the LP solver of Theorem 1 and the  
779 fast implementation of randomized swap rounding of Theorem 7 to give a fast algorithm for  
780 CROSSING-MST.

781 **Proof of Corollary 8.** Let  $G = (V, E)$  be the input graph and  $\epsilon > 0$ . We want to solve  
782  $\min\{c^T x : Ax \leq b, x \in \text{ST}(G)\}$ . By Theorem 1, there exists a randomized algorithm that  
783 runs in  $\tilde{O}(N/\epsilon^2)$  time and with high probability certifies the LP is infeasible or outputs  
784  $y \in \text{ST}(G)$  such that  $c^T y \leq (1 + \epsilon)\text{OPT}$  and  $Ay \leq (1 + \epsilon)b$ . We then apply the fast  
785 implementation of randomized swap rounding of Theorem 7 to  $y$ , which runs in  $\tilde{O}(m/\epsilon^2)$   
786 time and outputs a spanning tree  $T$ . If we only considered the feasibility version (i.e. find  
787  $x \in \text{ST}(G)$  such that  $Ax \leq b$ ), then the existing results on swap rounding [20] imply that  
788  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  with high probability. In the cost  
789 version, [20] implies that  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  with high  
790 probability, and  $\mathbb{E}[c^T \mathbf{1}_T] \leq \sum_e c^T x$ . Thus, the cost is preserved only in expectation. We can,  
791 however, apply Markov's inequality and conclude that  $\Pr[c^T \mathbf{1}_T \geq (1 + \epsilon)c^T x] \leq \frac{1}{1 + \epsilon} \leq 1 - \frac{\epsilon}{2}$   
792 (for  $\epsilon$  sufficiently small). For a suitable choice of the high probability bound, we have  
793 that  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$  and  $c^T \mathbf{1}_T \leq (1 + \epsilon)c^T x$  with  
794 probability at least  $\frac{\epsilon}{2} - \frac{1}{2n^2}$ . We can assume that  $\epsilon > \frac{1}{n^2}$ , for otherwise the  $\frac{1}{\epsilon^2}$  dependence  
795 in the run time of the approximate LP algorithm is not meaningful; one can use other  
796 techniques including an exact LP solver. Thus, with probability at least  $\frac{\epsilon}{4}$ , we have  $c^T \mathbf{1}_T \leq$   
797  $(1 + O(\epsilon))c^T x \leq (1 + O(\epsilon))\text{OPT}$  and  $A\mathbf{1}_T \leq \min\{O(\log k / \log \log k)b, (1 + \epsilon)b + O(\log k)/\epsilon^2\}$ .  
798 To boost the  $\frac{\epsilon}{4}$  probability of success, we can repeat the rounding algorithm  $O(\frac{\log n}{\epsilon})$   
799 times independently; with high probability, one of the trees will yield the desired bicriteria  
800 approximation. ◀

## 801 Non-uniform degree bounds

802 We briefly sketch some details regarding Remark 6. First, we note that the algorithm for  
803 solving the LP relaxation handles the non-uniform degree bound case in  $\tilde{O}(m/\epsilon^2)$  time. It  
804 either certifies that the given bounds are infeasible or outputs a fractional solution with  
805 degree at most  $(1 + \epsilon)B_v$  for each  $v$ . We can then apply the result in [43] to know that  
806 there exists a spanning tree  $T$  in which the degree of each  $v$  is at most  $\lceil (1 + \epsilon)B_v \rceil + 1$ .  
807 We can apply sparsification from Theorem 4 to the fractional solution to obtain a sparse  
808 subgraph that contains a near-optimal fractional solution. It remains to observe that the  
809 Fürer-Raghavachari algorithm can be used even in the non-uniform setting via a simple  
810 reduction to the uniform setting. This was noted in prior work [34, 43] and we provide the

811 details in the full version of the paper. This results in an  $\tilde{O}(n^2/\epsilon^2)$  time algorithm that either  
 812 decides that the given non-uniform degree bounds are infeasible or outputs a spanning tree  
 813 in which the degree of each node  $v$  is at most  $\lceil(1 + \epsilon)B_v\rceil + 2$ .

## 814 A.2 Extensions and related problems

815 We focused mainly on BD-ST, BD-MST and CROSSING-MST. Here we briefly discuss  
 816 related problems that have also been studied in the literature to which some of the ideas in  
 817 this paper apply.

### 818 Estimation of value for special cases of CROSSING-MST

819 As we remarked in Section 1, various special cases of CROSSING-MST have been studied. For  
 820 some of these special cases one can obtain a constant factor violation in the constraint [37, 36].  
 821 We highlight one setting. One can view BD-MST as a special case of CROSSING-MST  
 822 where the matrix  $A$  is a  $\{0, 1\}$ -matrix with at most 2 non-zeroes per column (since an edge  
 823 participates in only two degree constraints); the result in [43] has been extended in [33] (see  
 824 also [8]) to show that if  $A$  is a  $\{0, 1\}$ -matrix with at most  $\Delta$  non-zeroes per column, then the  
 825 fractional solution can be rounded such that the cost is not violated and each constraint is  
 826 violated by at most an additive bound of  $(\Delta - 1)$ . Theorem 1 allows us to obtain a near-linear  
 827 time algorithm to approximate the LP. Combined with the known rounding results, this  
 828 gives estimates of the integer optimum solution in near-linear time. Thus, the bottleneck in  
 829 obtaining a solution, in addition to the value, is the rounding step. Finding faster iterated  
 830 rounding algorithms is an interesting open problem even in restricted settings.

### 831 Multiple cost vectors

832 In some applications one has multiple different cost vectors on the edges, and it is advantageous  
 833 to find a spanning tree that simultaneously optimizes these costs. Such multi-criteria problems  
 834 have been studied in several contexts. Let  $c_1, c_2, \dots, c_r$  be  $r$  different cost vectors on the edges  
 835 (which we assume are all non-negative). In this setting it is typical to assume that we are  
 836 given bounds  $B_1, B_2, \dots, B_r$  and the goal is to find a spanning tree  $T \in \mathcal{T}(G)$  satisfying the  
 837 packing constraints such that  $c_j(T) \leq B_j$  for  $j \in [r]$ . We can easily incorporate these multiple  
 838 cost bounds as packing constraints and solve the resulting LP relaxation via techniques  
 839 outlined in Section 5. Once we have the LP solution we note that swap-rounding is oblivious  
 840 to the objective, and hence preserves each cost in expectation. With additional standard  
 841 tricks one can guarantee that the costs can be preserved to within an  $O(\log r)$  factor while  
 842 ensuring that the constraints are satisfied to within the same factor guaranteed in Corollary 8.

### 843 Lower bounds

844 BD-MST has been generalized to the setting where there can also be lower bounds on  
 845 the degree constraints of each vertex. [43] and [33] showed that the additive degree bound  
 846 guarantees for BD-MST can be extended to the setting with lower bounds in addition to  
 847 upper bounds. One can also consider such a generalization in the context of CROSSING-MST.  
 848 The natural LP relaxation for such a problem with both lower and upper bounds is of the form  
 849  $\min\{c^T x : Ax \leq b, A'x \geq b', x \in \text{ST}(G)\}$  where  $A, A' \in [0, 1]^{k \times m}$ ,  $b, b' \in [1, \infty)^k$ ,  $c \in [0, \infty)^m$ .  
 850 Here  $A$  corresponds to upper bounds (packing constraints) and  $A'$  corresponds to lower  
 851 bounds (covering constraints). This mixed packing and covering LP can also be solved  
 852 approximately in near-linear time by generalizing the ideas in Section 5. Sparsification as

## 24:22 Fast Bounded-Degree and Crossing Spanning Trees

853 well as swap-rounding can also be applied since they are oblivious to the constraints once the  
854 LP is solved. The guarantees one obtains via swap rounding are based on negative correlation  
855 and concentration bounds. They behave slightly differently for lower bounds. One obtains  
856 a tree  $T$  such that  $A\mathbf{1}_T \leq (1 + \epsilon)b + O(\log k)/\epsilon^2$  and  $A'\mathbf{1}_T \geq (1 - \epsilon)b' - O(\log k)/\epsilon^2$  with  
857 high probability. As in the other cases, the LP solution proves the existence of good integer  
858 solutions based on the known rounding results.