Branch-and-bound trees, integrality gaps and online optimization

A tale of algorithms and randomness

Sander Borst

Printed by Gildeprint

ISBN: 978-90-393-7700-0
DOI: https://doi.org/10.33540/2424

Branch-and-bound trees, integrality gaps and online optimization

A tale of algorithms and randomness

Branch-and-bound bomen, geheeltalligheidskloven en online optimalisatie
Een verhaal van algoritmen en willekeurigheid
(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof. dr. H.R.B.M. Kummeling, ingevolge het besluit van het College voor Promoties in het openbaar te verdedigen op

dinsdag 27 augustus 2024 des middags te 4.15 uur

door

Sander Johannes Borst
geboren op 29 januari 1996 te Leidschendam

The research in this thesis was carried out at Centrum Wiskunde & Informatica (CWI) in Amsterdam, part of Stichting Nederlandse Wetenschappelijk Onderzoek Instituten (NWO-I). The research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement QIP-805241).

Voor mijn ouders.

Contents

Chapter 1

Introduction

Optimization problems are everywhere: from creating high school timetables to designing computer chips. Nowadays, optimization algorithms are widely used in industry to increase the efficiency of a broad range of processes. Solving such applied optimization problems using mathematics is part of the field of *operations research*. This field arose in the early twentieth century. An important development was the idea to formulate problems as *integer linear programs* (IPs) and *mixed integer programs* (MIPs). This made it possible to solve many problems using a single type of solver, called a MIP solver.

Various MIP solvers are available, both commercial and academic, and they are widely used in industry. Examples include CPLEX, GUROBI, HiGHS, SCIP and XPRESS. All of these solvers at their core use the same algorithm: the *branch-and-bound algorithm*. Despite the popularity of this algorithm, the theoretical understanding of its behavior is still limited. In this thesis, we aim to provide a better understanding of the branch-and-bound algorithm, through both theoretical and experimental analysis.

In the rest of this introduction, we will introduce integer linear programming and the branch-and-bound algorithm. Let us first start by giving a brief overview of the contents of this thesis. Our analysis of the branch-and-bound algorithm consists of three parts:

- In Chapter 3 we do a theoretical study of the behavior of branch-and-bound on problem instances that are sampled from several classes of probability distributions. This can be seen as an average-case analysis of the performance of the branch-and-bound algorithm. It relies on a result on the discrepancy of random matrices, that we prove in Chapter 4. This part is based on joint work with Daniel Dadush, Sophie Huiberts and Samarth Tiwari [BDHT22] and joint work with Daniel Dadush and Dan Mikulincer [BDM23].

- In Chapter 5 we give a theoretical analysis of *node selection rules*, which determine the order in which the search space is explored. We provide a new node selection rule that has an improved running time in the *explorable heap model*. This part is based on joint work with Daniel Dadush, Sophie Huiberts and Danish Kashaev [BDHK23].

- Chapter 7 contains an experimental analysis of the impact of implementing *propagation* and *conflict analysis* in an MIP solver, that uses exact rational representations of the involved numbers. This part is based on joint work with Leon Eifler and Ambros Gleixner [BEG24].

The branch-and-bound algorithm is typically used to solve optimization problems that are entirely known before the algorithm starts. In Chapter 6 we study a different type of optimization problem, in which the algorithm continuously receives new information and has to make decisions based on this information. Such problems are known as *online problems*. The problem we study is *online hypergraph matching*. This part is based on joint work with Danish Kashaev and Zhuan Khye Koh [BKK24].

1.1 Integer linear programming

Linear optimization is a technique that allows us to solve a wide range of problems. In a linear optimization problem, known as a *linear program* (LP), one wants to find a variable $x \in \mathbb{R}^n$ that minimizes or maximizes a linear function while satisfying a set of linear inequalities. Such a problem can be written in the following general form:

$$\begin{aligned} &\min_x c^\mathsf{T} x \\ &Ax \leq b \\ &x \in \mathbb{R}^n \end{aligned}$$

for some given $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. Here $Ax \leq b$ refers to componentwise inequality, i.e. $(Ax)_i \leq b_i$ for all i. The above problem is a minimization problem. In other parts of this thesis we will also consider maximization problems, in which the objective function needs to be maximized. The two types of problems are equivalent, since a maximization problem can be turned into a minimization problem by multiplying the objective function by -1. Similarly, instead of using smaller-or-equal constraints, one can equivalently use greater-or-equal constraints by multiplying the corresponding row of A by -1.

A classic example of a linear program is that of Stigler's diet problem, which is considered in the following example.

Example 1.1.1 (A diet problem)**.** In Stigler's diet problem, we are given a list of foods. Nutritional information about each food is given and the cost of the food is also known. The problem asks for the cheapest combination of the foods that provides enough nutrients to live on [Sti45].

	Tomato	Potato	Pasta	Zucchini	Pesto
Price	0.2	0.1	0.5	1.0	1.0
Protein (g)	4.0	0.0	6.0	1.0	6.0
Fat (g)	0.0	0.0	1.0	0.0	25.0
Carbohydrates (g)	4.0	25.0	36.0	3.0	6.0

Suppose that the above (made-up) info is given and that we want to find the cheapest combination of foods that contains at least 50 grams of fat, 60 grams of protein and 300 grams of carbohydrates, given the info in the table above. We can formulate this as the following linear program:

$$\begin{aligned} \min_x \; & 0.2x_1 + 0.1x_2 + 0.5x_3 + 1.0x_4 + 1.0x_5 \\ \begin{pmatrix} 4.0 & 0.0 & 6.0 & 1.0 & 1.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 25.0 \\ 4.0 & 25.0 & 36.0 & 3.0 & 6.0 \end{pmatrix} & x \geq \begin{pmatrix} 50 \\ 60 \\ 300 \end{pmatrix} \\ & x \geq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

By solving this linear program, we find that the optimal solution is $(12, 9.6, 0, 0, 2.0)$. That is, the optimal solution is to buy 12 tomatoes, 9.6 potatoes and 2 jars of pesto. This will have a cost of 5.36.

The study of linear programs began in the 1940s, in part motivated by planning problems in the military that arose during the second world war [Sch11; Dan91]. In 1947 Dantzig invented the *simplex algorithm* for solving linear programs. This algorithm works by walking along edges of the polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$ in a direction that improves the objective value. In 1979 Khachiyan proposed *the ellipsoid algorithm*, which solves linear programs in polynomial time [Kha79]. This was a major theoretical breakthrough, as the simplex algorithm is known to have exponential worst-case running time. However, even today the simplex algorithm is still widely used because of its simplicity and the fact that it often performs better in practice than other known algorithms.

Despite the usefulness of linear programming, many real-world problem cannot directly be modeled as an LP. One obstacle is the fact that linear programs do not allow for the modeling of discrete decisions. For instance, consider Example 1.1.1. According to the optimal solution, we should buy 9.6 potatoes, but in practice we might only be able to buy entire potatoes. To account for this restriction, we can impose the additional constraint that a solution must consist of only integers. This turns our linear program into an *integer linear program* (IP). IPs can be stated in the following general form.

$$\begin{aligned} &\min_x c^{\mathsf{T}} x \\ &Ax \leq b \\ &x \in \mathbb{Z}^n \end{aligned}$$

It is also common to have problems with both integer variables and continuous variables. Such problems are called *mixed integer programs* (MIPs). Mixed integer programs are able to capture a wide range of combinatorial problems, arising from both theory and practice.

Integer programming has been shown to belong to the class of NP-complete problems, a class of notoriously hard problems, even when restricting to only $\{0, 1\}$-variables [Kar72]. Hence, it is unlikely that it admits a polynomial time algorithm. By a classical result due to Lenstra [Len83] and Kannan [Kan87] ILPs with n variables are solvable in $n^{O(n)}$ times a polynomial factor in the bit complexity of the problem. This was improved upon by Dadush [Dad12] and later by Reis and Rothvoss, who improved the running time to $(\log(2n))^{O(n)}$ [RR23].

1.2 The branch-and-bound algorithm

The most popular algorithm for solving ILPs is the branch-and-bound algorithm, which was introduced in 1960 by Land and Doig [LD60] and further developed by Dakin [Dak65]. The algorithm can be seen as a divide-and-conquer algorithm that recursively splits up the feasible region of the problem into smaller regions, and creates subproblems for optimizing over these regions. The algorithm starts with just one subproblem, which is the original problem. During its execution, the algorithm maintains an upper bound on the optimal value of the original problem, coming from the best integral solution found so far. The algorithm repeatedly executes the following steps.

1. Pick an unprocessed subproblem.

2. Find an optimal solution x to the LP relaxation of this problem, which is obtained by dropping the integrality constraints from the problem.
   (a) If the LP relaxation is infeasible, conclude that the subproblem is infeasible.
   (b) If the value of the LP relaxation is not smaller than the best integral solution found so far (the global upper bound), prune this branch of the search tree.
   (c) If x satisfies all integrality constraints, the solution is optimal for this subproblem, and the subproblem has been solved. If its objective value is smaller than the global upper bound, update the upper bound.
3. If none of the previous cases apply, split up the feasible region into two polyhedra P_1 and P_2, such that their union contains all integer points in the original polytope, but neither of the polyhedra contains x. This operation is called *branching*.
4. Add the subproblems corresponding to these polyhedra to the list of unsolved subproblems (that is $\min\{c^\mathsf{T}x : x \in P_2 \cap \mathbb{Z}^n\}$ and $\min\{c^\mathsf{T}x : x \in P_1 \cap \mathbb{Z}^n\}$). Wait till these are solved. If both subproblems are infeasible, conclude that the problem is infeasible. Otherwise, the optimal solution to the current subproblem is the best of the optimal solutions to the two subproblems.

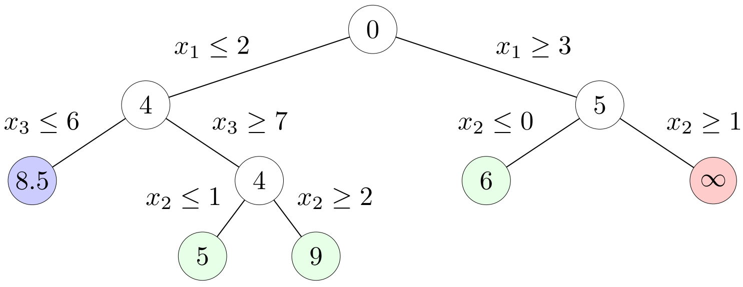
Note that when branching on a subproblem, the LP value of this problem provides a lower bound on the optimal value of the newly created subproblems. Hence, we have a lower bound for each unprocessed subproblem. The minimum of these will be a lower bound on the value of any unfound solution. As soon as this lower bound hits the upper bound, an optimal solution has been found and the algorithm finishes.

Note that in step 3, it is not specified how to split up the feasible polyhedron. This is the responsibility of the *branching rule*, which is a crucial component of the algorithm. A common way is to take a variable x_i that is not integral in x, and to create two polytopes by adding the constraints $x_i \leq \lfloor x_i \rfloor$ and $x_i \geq \lceil x_i \rceil$ to the two subproblems. This is called *branching on variables*. Another way is to branch using *general disjunctions* by adding respectively $a^\mathsf{T}x \leq \delta$ and $a^\mathsf{T}x \geq \delta + 1$ for some $a \in \mathbb{Z}^n$ and some $\delta \in \mathbb{Z}$. However, other ways of branching are possible as well. Substantial research has been done on branching rules, see, e.g., [LS99; AKM05; LZ17; BDSV18]. Another important component of the algorithm is the *node selection rule*, which decides which subproblem to solve next. We will elaborate on this in Section 1.5.

The search tree, in which the nodes represent subproblems and the edges represent the branching operation, is known as the *branch-and-bound tree*. In this tree, the root node represents the original problem, and for each node, the children represent the subproblems created by branching. What makes the algorithm more efficient than a naive enumeration of all possible solutions is that it uses the LP relaxation to prune branches in three ways.

- If the LP relaxation is infeasible, the subproblem is *pruned by infeasibility.*
- If the LP relaxation is greater than the best integral solution found so far, the subproblem is *pruned by bound.*
- If the LP relaxation has an integral optimal solution, the subproblem is *pruned by optimality.*

An illustration of a possible branch-and-bound tree is given in Fig. 1.1.



$$
\begin{aligned}
\min \quad & x_4 \\
& \begin{pmatrix} 2 & 0 & 0 & -1/5 \\ -2 & 0 & 0 & -1/4 \\ 4 & 2 & 0 & 0 \\ 0 & -2 & 0 & -1/6 \\ -16 & 0 & -2 & -2/17 \\ -40 & -2 & 40 & -1/5 \\ -40 & 2 & 40 & -1/9 \end{pmatrix} x \leq \begin{pmatrix} 5 \\ -5 \\ 27/2 \\ -1 \\ -45 \\ 197 \\ 203 \end{pmatrix} \\
& x \geq 0 \\
& x \in \mathbb{Z}^4
\end{aligned}
$$

Figure 1.1: An example of an integer program and its corresponding branch-and-bound tree. The node labels represent the value of the LP relaxation of the corresponding subproblem. Observe that they are always non-decreasing along paths down the tree. The edge labels represent the branching decisions. The green subproblems admit integral optimal solutions, the red subproblem is infeasible and the blue subproblem is pruned by bound.

Now consider the running time of the algorithm. At each processed node of the branch-and-bound tree, an LP is solved. As discussed earlier, LPs can be solved efficiently. However, the number of nodes in the branch-and-bound tree can be exponential in the size of the problem, leading to an exponential worst-case running time. Hence, the number of nodes in the branch-and-bound tree is an important measure of the performance of the algorithm.

Solvers that use the branch-and-bound algorithm typically rely on additional techniques to speed up the solving process. These include *cutting planes*, which are derived inequalities that are added to the LP relaxation to make it tighter, *primal heuristics*, that allow the solver to quickly find good integral solutions [Ach07b], and *symmetry reduction*, which reduces the size of the search space by exploiting symmetries [Lib12; Ost09]. Other techniques are *constraint propagation*, which can tighten variable bounds without solving an LP and *conflict analysis*, which allows the solver to learn from infeasible subproblems [Ach07b]. In Chapter 7 we will elaborate on the latter two techniques. For more background on the branch-and-bound algorithm we refer to [Wol21; NW88; Ach07b; CCZ14].

1.3 Branch-and-bound beyond the worst case

As IP is an NP-complete problem, we should not expect the branch-and-bound algorithm to run in polynomial time. Indeed, there exist instances for which any branch-and-bound tree will be of exponential size, when only branching on variables is allowed, leading to an exponential solving time (see Example 1.3.1). Even when branching on general disjunctions is allowed, there are instances that need a $2^{L^{\Omega(1)}}$ size tree, where L is the encoding length of the problem [GP24]. This shows that the worst-case time complexity of the branch-and-bound algorithm is indeed superpolynomial.

Example 1.3.1. Consider the following IP for some odd $n \in \mathbb{N}$:

$$\begin{aligned} \max \sum_{i=1}^{n} x_i& \\ \sum_{i=1}^{n} x_i &\leq \frac{n}{2} \\ x_i &\in \{0,1\} \end{aligned}$$

It is clear that the optimal solution has objective value $\lfloor n/2 \rfloor$, while the LP relaxation has value $n/2$. Any branch-and-bound tree for this problem that only branches on variables will have size at least $2^{n/2}$, which is exponential in n.

However, there exist many classes of IPs that are solvable in polynomial time. For instance, IPs for which the feasible region is an integral polytope can simply be solved by solving the LP relaxation. Most real-world IPs do not necessarily belong to these classes. Yet, many of them can still be solved using the branch-and-bound algorithm in a reasonable amount of time. This suggests that the worst-case time complexity of the algorithm is not always a good measure of its performance. Instead, we could look at the *average-case time complexity*, by analyzing the time that the algorithm needs to solve a problem instance sampled from a given probability distribution. The study of this quantity is called *average-case analysis*.

A wide range of algorithms has been studied using average-case analysis. A notable example is the aforementioned simplex algorithm for solving LPs, which was shown to run in polynomial time on large classes of LPs [Bor82]. For that algorithm, polynomial time complexity was also shown in the stronger *smoothed analysis* model in which one considers adversarially chosen instances with a small amount of random noise added to the instance data [ST04; HLZ23].

The average-case complexity of integer linear programming has been studied in several settings [Kuz96; BV04a; EW19; JR23; Pap81; DF89; DF92]. A strong result in the context of smoothed analysis is that, under mild conditions, the class of IPs whose smoothed complexity is polynomial corresponds exactly to the class of IPs solvable in pseudopolynomial time [RV07; BV04b].

In each of these results, problems are shown to have polynomial time complexity by showing that they can be efficiently solved using specially constructed algorithms. However, these algorithms are seldom used in practice. Real-world problems are typically solved using the generic branch-and-bound algorithm, which is very effective in practice. This suggests that this algorithm is efficient on large classes of problems and that using specialized algorithms is not necessary. It therefore motivates the study of the average-case complexity of the branch-and-bound algorithm itself.

For the branch-and-bound algorithm, little had been known about its average-case time complexity before the publication of recent work by Dey, Dubey and Molinaro [DDM23]. They studied the average-case complexity of the branch-and-bound algorithm on binary packing IPs for which the coefficients of the cost vector and the constraint matrix are sampled independently from a uniform distribution. When the number of constraints is constant, they show that the average-case time complexity of the algorithm is only polynomial in the number of variables. In this thesis we extend their results to a wide range of probability distributions.

An important step in the analysis in [DDM23] is relating the size of the branch-and-bound tree to the number of integer points in a specific knapsack polytope, whose capacity is equal to integrality gap of the LP relaxation of the IP. This integrality gap, which we denote by IPGAP, is the difference in objective value between the optimal value of the LP relaxation and the optimal integral solution. They then bound the size of this knapsack. In Chapter 3 we give a refined version of this argument, which works for general distributions of IPs with cost coefficients from continuous distributions of bounded density. This shows that with high probability the tree size is at most $e^{O(\sqrt{n \cdot \mathsf{IPGAP}})}$ for fixed m.

By providing a high probability bound on the integrality gap for several classes of IPs, we show that for these IPs the branch-and-bound tree is of polynomial size with high probability. Our first result applies to the following two classes of IPs:

- IPs with entries of c that are all independent Gaussian $\mathcal{N}(0,1)$ distributed, and columns of A that are independently sampled from a logconcave isotropic distribution.

- IPs for which b is integral with entries of c that are all independent Gaussian $\mathcal{N}(0,1)$ distributed, and entries of A that are independently sampled from a uniform distribution on $\{-k,\dots,k\}$ for $k \geq 1$.

We will refer to these as *Centered IPs*. Note that IPs from the second class have discretely distributed coefficients. This makes them especially interesting to study, as many real-world IPs are of a combinatorial nature and have discrete coefficients.

In Chapter 3 we prove that the average-case time complexity of branch-and-bound on these classes of IPs is only polynomial in the number of variables. Here we state a simplified version of the result.

Theorem A (Theorem 3.1.5)**.** *For randomly sampled IPs from a probability distribution of Centered IPs for which $\sum_{i=1}^n \max(-b_i, 0)^2 \leq O(n^2)$, with probability $1 - 1/n$ any branch-and-bound tree has size at most $n^{\mathrm{poly}(m)}$.*

The third class of IPs that we consider is the class of IPs where the entries c come from an exponential distribution and the entries of A are uniformly sampled from $\{1,\dots,k\}$. We will refer to these as *Discrete Packing IPs*. For these IPs we prove the following result. Note that in this setting, we require stronger conditions on the right-hand side b.

Theorem B (Theorem 3.1.5)**.** *For randomly sampled IPs from a probability distribution of Discrete Packing IPs where $b \in ((kn\beta, kn(1/2-\beta)) \cap \mathbb{Z})^m$ for some $\beta \in (0, \frac{1}{4})$, with probability $1 - 1/n$ any branch-and-bound tree has size at most $n^{\exp(1/\beta)\,\mathrm{poly}(m)}$.*

1.4 Integrality gaps and the discrepancy of random matrices

A key ingredient for proving Theorems A and B is to derive high probability bounds on the integrality gap for Centered IPs and Discrete Packing IPs. Integrality gaps for random IPs have been studied for the 0-1 knapsack problem [Lue82], for the multidimensional knapsack problem [DF89], and for the generalized assignment problem [DF92]. In each of these cases, the distribution of the coefficients of the IPs was assumed to be uniform, and an integrality gap of $O(\log(n)^2/n)$ was shown with high probability.

To prove gap bounds for Centered IPs and Discrete Packing IPs, we build on the approach of Dyer and Frieze from [DF89], in which they show that with high probability, a solution to the LP relaxation can be rounded to an integral solution at small cost to the objective value.

To adapt this approach to Centered IPs and Discrete Packing IPs there are some changes that need to be made.An important step in the work of Dyer and Frieze is to show that with high probability there exists a subset of columns of the constraint matrix A, whose sum is very close to some carefully constructed vector t. They prove this using the second moment method. In [BDHT22] we have shown that this method can be used to generalize the argument to the setting of Centered IPs with Gaussian entries in the constraint matrix. However, the same method cannot easily be applied for Centered IPs from discrete distributions, or even natural extensions like logconcave distributions.

Instead, we devote Chapter 4 to a different approach which uses Fourier analysis for proving such bounds. We will show that with high probability over A, for every suitable vector t there is a subset $T \subseteq [n]$ such that $\| \sum_{i \in T} A_i - t\|$ is small. For the discrete distributions that we consider, this amounts to showing the high probability existence of a subset T such that $\sum_{i \in T} A_i = t$.

We note that our result is closely related to the concept of linear discrepancy, introduced by Lovász, Spencer and Vesztergombi [LSV86]. The linear discrepancy of a matrix $A \in \mathbb{R}^{m \times n}$ is defined to be $\mathsf{lindisc}(A) := \max_{\lambda \in [0,1]^n} \min_{x \in \{0,1\}^n} \|A(x - \lambda)\|_\infty$. It can be seen as the "rounding error" one must incur to round a $[0, 1]$ combination of the columns to a $\{0, 1\}$ combination. There is a rich history of work on various notions of discrepancy [Cha01; CST14]. This theory has found many applications in algorithm design and complexity theory [BRS22; HR17].

Our result for the existence of a suitable set T, Theorem 4.3.4, can be interpreted as bounding the linear discrepancy of the random matrix A for combinations $\lambda \in [0, 1]$ which are very close to $p\mathbf{1}_n$, where $p \in (0, 1)$ for appropriate p. We now state a simplified version of our results. For logconcave isotropic distributions, we have the following result.

Theorem C (Theorem 4.3.5). *Suppose the columns of $A \in \mathbb{R}^{m \times n}$ are independently sampled from a logconcave isotropic distribution with mean $\mu \in \mathbb{R}^m$. Let $p \in [0,1]$ with $\frac{\text{poly}(m)\log(n)}{n} \leq p \leq \frac{1}{\text{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every t with $\|t - pn\mu\| \leq O\left(\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $\|A\mathbf{1}_S - t\| \leq \exp\left(-\Omega(\frac{pn}{m})\right)$.

For the discrete distributions that we consider, we prove the following theorem.

Theorem D (Theorem 4.3.7). *Suppose the entries of $A \in \mathbb{R}^{m \times n}$ are uniformly sampled from $\{0, 1, \dots, k\}$. Let $p \in [0,1]$ with $\frac{\text{poly}(m)\log(n)\log(k)}{n} \leq p \leq \frac{1}{\text{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every $t \in \mathbb{Z}^n$ with $\|t - pnk/2\| \leq O\left(k\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $A\mathbf{1}_S = t$.

1.5 Node selection rules

In our description of the algorithm we did not specify how the algorithm decides which subproblem to pick in step 1. This is the responsibility of the *node selection rule*. It is an important component of the algorithm, as it determines the order in which the search space is explored.

One way to do node selection is to always select the unprocessed subproblem whose parent has the smallest LP value. This is known as the *best-first* rule. This rule is theoretically optimal, because it minimizes the number of nodes that need to be processed.

However, a disadvantage of this rule is that consecutively processed subproblems can be far apart in the search tree. For practical reasons this is often less efficient. A simple depth-first search strategy does not suffer from this problem, but this can lead to an unnecessarily large branch-and-bound tree. In practice, often a hybrid strategy is used. While these strategies perform well in practice, they do not come with good theoretical guarantees.

We study the node selection problem from a theoretical perspective in Chapter 5. We do so by studying the closely related problem of explorable heap selection. This problem consists of locating the nth smallest value in a min-heap. The values of the heap can only be accessed by the algorithm by moving the 'cursor' to the corresponding node. The running time of the algorithm is measured by the number of edges in the search tree that it traverses. We will show that this model captures the essence of the node selection problem.

The model was originally introduced by Karp, Saks and Wigderson [KSW86] to design node selection rules with low space complexity. The randomized algorithm that they introduce has an expected running time of $n \cdot e^{O(\sqrt{\log(n)})}$ and space complexity $O(\sqrt{\log n})$. For comparison, the best-first strategy has a running time of $O(n^2)$ and space complexity $O(1)$.

In Chapter 5 we improve on the running time by giving a randomized algorithm with running time $O(n \log(n)^3)$ and space complexity $O(\log n)$.

Theorem E (Theorem 5.3.1). *There exists a randomized algorithm that solves the explorable heap selection problem, with expected running time $O(n \log(n)^3)$ and $O(\log(n))$ space.*

Note that the running time of our algorithm is significantly better than the algorithm from [KSW86], while the space complexity is slightly worse. This suggests that there is a trade-off between running time and space complexity of algorithms for this problem. Indeed, we provide the following lower bound on the running time of space-efficient algorithms.

Theorem F (Theorem 5.4.4). *Any randomized algorithm for explorable heap selection with space complexity s has a running time complexity of $\Omega(\log_{s+1}(n)n)$.*

1.6 Online hypergraph matching

In the integer linear programming problem, both the objective function and the constraints are known in advance. Such a problem is known as an *offline* problem. Many optimization problems arising in practice are *online* problems, for which not all information is known in advance. An example is scheduling deliveries for a food delivery company. The company receives orders throughout the day and needs to decide which driver is going to deliver which orders. The orders are not known up front, so that scheduling decisions need to be made using incomplete information. The aforementioned explorable heap selection problem can also be seen as an online problem, as new information is revealed as the algorithm progresses.

A well-known online problem is *online bipartite matching*, which has applications in online advertising. In this problem, a bipartite graph is given. The goal is to find a matching, a set of disjoint edges, of maximum size. Then, the vertices of one side of the graph are revealed one by one, along with their incident edges. After the arrival of each vertex, it needs to be decided whether to match it to one of its neighbors or not. Added edges cannot be removed from the matching afterwards.

The online bipartite matching problem has been studied using competitive analysis, which compares the objective value that an online algorithm achieves to the optimal offline solution. In seminal work, Karp, Vazirani and Vazirani [KVV90] showed that there exists a randomized algorithm that achieves a competitive ratio of $1 - \frac{1}{e}$.

In Chapter 6 we study a natural generalization of this problem, where instead of a bipartite graph, a k-uniform hypergraph is given. A k-uniform hypergraph, has edges that consist of k vertices. While the problem has been studied in the random-order setting [MSV23; KTRV14], there are still many open questions about the adversarial-order setting, in which the order of the vertices is chosen by an adversary. In particular, no algorithm better than the trivial $1/k$-competitive greedy algorithm is known. We study the fractional version of the problem for the case $k = 3$. For this case, we provide an $(e-1)/(e+1)$ competitive algorithm. Our main contribution is a matching upper bound, which shows that every algorithm is at most $(e-1)/(e+1)$-competitive.

Theorem G (Theorem 6.1.1)**.** *For the online fractional matching problem on* 3*-uniform hypergraphs, there is a deterministic* $(e-1)/(e+1)$*-competitive algorithm. Furthermore, every algorithm is at most* $(e-1)/(e+1)$*-competitive.*

We also provide an algorithm for the integral version of the problem for general k, that achieves a better competitive ratio than the greedy algorithm on graphs with bounded degree.

Theorem H (Theorem 6.1.2)**.** *For the online matching problem on* k*-uniform hypergraphs where online vertices have maximum degree* d*, there exists an algorithm whose competitive ratio is*

$$\min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right).$$

1.7 Solving rational MIPs exactly

In the last chapter of this thesis, we study the branch-and-bound algorithm experimentally. One of the many use cases of MIP solvers is proving results in computational mathematics. There have been recent examples in logic [BMVV19], topology [BO12], (extremal) combinatorics [EGP22; KS18; Pul20] and graph theory [LPR20]. In these applications, it can be important to be certain that the found solution is indeed optimal, as even a small numerical error could invalidate the result. In some other applications numerical errors can also have severe consequences, such as in chip design [Ach07b], where they can lead to malfunctioning chips.

There are no theoretical barriers to implementing the branch-and-bound algorithm exactly. It is a well known fact that MIPs with only rational coefficients have an optimal solution that is also rational [Sch03; Sch11, Theorem 16.1]. So by storing all numbers in the branch-and-bound algorithm and its LP solves as rational numbers, one can solve MIPs exactly. However, in practice, most MIP solvers use floating point arithmetic, as this greatly reduces the running time, especially for the LP solves. This can lead to numerical errors, which can cause the algorithm to return slightly suboptimal or even incorrect solutions. For the applications mentioned above that can be problematic.

To avoid numerical errors for such problems, one can solve all the LPs using rational arithmetic. However, this is prohibitively slow. To circumvent this issue, a hybrid approach has been proposed recently, which uses both floating-point and rational arithmetic [CKSW13]. This approach has been revised and was further improved by [EG22]. Here all LPs are initially solved using floating point arithmetic in a safe way that guarantees that feasible solutions are not lost. Only when the outcome of a floating point solve is inconclusive, the LP is solved using rational arithmetic. This approach has been implemented in the academic MIP solver SCIP, and was shown to be highly effective in practice [EG22]. Still, the approach is significantly slower than the floating point version of SCIP.

The main reason for this is that our previous description of the branch-and-bound algorithm is not all that a MIP solver does. In practice, MIP solvers use additional techniques to speed up the solving process. Two important techniques are *constraint propagation* and *conflict analysis*. These techniques were not implemented in the exact version of SCIP. In Chapter 7 we implement them and analyze the impact on the performance of the solver. In experiments on a wide range of MIPs, we show that this leads to a significant speedup.

Result I. *By implementing constraint propagation and conflict analysis in the exact MIP solver SCIP, we decrease the running time by 23% and 11% more instances are solved to optimality on a subset of the MIPLIB 2017 benchmark set within the time limit of two hours.*

1.8 Organization

In Chapter 2 we introduce notation that is used throughout the thesis and state some basic results that we need. In Chapter 3 we provide bounds on the integrality gap of IPs with random coefficients, and use these bounds to provide an average-case analysis of the branch-and-bound algorithm. In Chapter 4 we prove a result on the discrepancy of random matrices, which is used in the analysis of the integrality gap in Chapter 3. In Chapter 5 we study the node selection problem in the explorable heap model. In Chapter 6 we study the online hypergraph matching problem. In Chapter 7 we study the impact of implementing propagation and conflict analysis in an exact MIP solver.

Chapter 2

Preliminaries

In this chapter, notation and definitions are introduced that will be used throughout the thesis. Furthermore, some well-known results are stated.

2.1 Basic notation

We denote the reals and non-negative reals by $\mathbb{R}, \mathbb{R}^+$ respectively, and the integers and positive integers by $\mathbb{Z}, \mathbb{N}$ respectively. For $k \geq 1$ an integer, we let $[k] := \{1, \dots, k\}$. If $(x_1, \dots, x_m) \in \mathbb{R}^m$ and $p \geq 1$, the p-norm is defined by,

$$\|x\|_p := \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}}.$$

When $p = 2$, i.e. the Euclidean norm, we will sometimes omit the subscript, so $\|x\| := \|x\|_2$. We interpret $p = \infty$ in the limiting sense:

$$\|x\|_\infty := \max_{1 \leq i \leq n} |x_i|.$$

For a matrix A, we use $\|A\|_{\text{op}}$ to denote the operator norm.

For $s \in \mathbb{R}$, we let $s^+ := \max\{s, 0\}$ and $s^- := \min\{s, 0\}$ denote the positive and negative part of s. We extend this to a vector $x \in \mathbb{R}^n$ by letting $x^{+(-)}$ correspond to applying the positive (negative) part operator coordinate-wise.

We use $\log x$ to denote the base 2 logarithm and $\ln x$ to denote the base e natural logarithm. We use $0_m, 1_m \in \mathbb{R}^m$ to denote the all zeros and all ones vector respectively, and $e_1, \dots, e_m \in \mathbb{R}^m$ denote the standard coordinate basis. The identity matrix in $\mathbb{R}^m$ is denoted by I_m. We write $\mathbb{R}^m_+ := [0, \infty)^m$.

If a and b are quantities that depend on the problem's parameters we will write $a = O(b)$ (resp. $a = \Omega(b)$) to mean $a \leq Cb + D$, (resp. $a \geq Cb - D$) for some numerical constants $C, D > 0$. We also write $a \ll b$ or $a = o(b)$ (resp. $a \gg b$ or $a = \omega(b)$) to mean $\lim\limits_{a \to \infty} \frac{a}{b} = 0$ (resp. $\lim\limits_{a \to \infty} \frac{b}{a} = 0$). We will write $\text{poly}(n)$ to refer to some polynomial in n.

For matrices $A, B \in \mathbb{R}^{n \times n}$ we will write $A \succeq B$ to mean that $A - B$ is positive semidefinite. Let I_m denote the identity matrix of size m. When the size of the matrix is clear from the context, we will omit the subscript and simply write I.

2.2 Linear programming

We will often use the following standard form for a linear program:

$$\begin{aligned} \max_x \, & c^\mathsf{T} x \\ Ax & \leq b \\ x & \geq 0 \\ x & \in \mathbb{R}^n. \end{aligned}$$

We will call this the *primal program*. The *dual program* is now given by:

$$\begin{aligned} \min_y \, & b^\mathsf{T} x \\ A^\mathsf{T} y & \geq c \\ y & \geq 0 \\ y & \in \mathbb{R}^m. \end{aligned}$$

By a classical result, the value of the primal program is equal to the value of the dual program [Sch03, Theorem 5.4]. This is known as *strong duality*. Another important result is the *complementary slackness* condition, which states that x and y are optimal solutions to the primal and dual programs respectively if and only if for all $i \in [n]$ and $j \in [m]$, we have $(A^\mathsf{T} y - c)_i \cdot x_i = 0$ and $(Ax - b)_j \cdot y_j = 0$ [Sch03, Section 5.5]. Given an optimal dual solution y, the components of the vector $\bar{c} = c - A^\mathsf{T} y$ are called the *reduced costs*. We note that $\bar{c}_i$ is the rate at which the objective function increases when increasing x_i at rate one while reoptimizing to stay within the feasible region.

2.3 Stochastics

Let $X \in \mathbb{R}^m$ be a random vector distributed according to a probability measure ν on $\mathbb{R}^m$. We will use f_X to refer to the probability density function of X. We denote the expectation by $\mathbb{E}[X]$ and covariance matrix by $\mathrm{Cov}[\nu] := \mathrm{Cov}[X] := \mathbb{E}[XX^\mathsf{T}] - \mathbb{E}[X]\mathbb{E}[X]^\mathsf{T} \succeq 0$. If $X \in \mathbb{R}$ is a real random variable, we use the notation $\mathrm{Var}[X]$ to write the variance instead of $\mathrm{Cov}(X)$. For any $u \in \mathbb{R}^d$, we note that

$\mathrm{Var}[u^\mathsf{T} X] = \mathbb{E}[(u^\mathsf{T} X)^2] - \mathbb{E}[u^\mathsf{T} X]^2 = u^\mathsf{T}\,\mathrm{Cov}[X]u$. We say that X, or its law ν, is isotropic if and $\mathbb{E}[X] = 0$ and $\mathrm{Cov}[X] = I_m$.

$X \in \mathbb{R}^d$ is a continuous random vector if it admits a probability density $f : \mathbb{R}^d \to \mathbb{R}_+$ satisfying $\Pr[X \in A] = \int_A f(x)dx$, for all measurable $A \subseteq \mathbb{R}^d$. We will say that a continuous random vector has maximum density at most $M > 0$ if its probability density f satisfies $\sup_{x \in \mathbb{R}^d} f(x) \leq M$.

Proposition 2.3.1. *Let $X \in \mathbb{R}^n$ satisfy $\mathbb{E}[X] = 0$. Then $\mathbb{E}[X^+] = \mathbb{E}[|X|]/2$.*

Proof. Note that $0 = \mathbb{E}[X] = \mathbb{E}[X^+ - X^-] \Rightarrow \mathbb{E}[X^+] = \mathbb{E}[-X^-]$. Thus, $\mathbb{E}[|X|] = \mathbb{E}[X^+] + \mathbb{E}[-X^-] = 2\mathbb{E}[X^+]$, as needed. □

Lemma 2.3.2. *If the density function f_X of X is bounded from above by M, then:*

$$\mathrm{Var}(X) \geq \frac{1}{12M^2}.$$

Proof. If we want to minimize $\mathrm{Var}(X) = \int_{-\infty}^{\infty} t^2 f_X(t)dt$ under the conditions $f_X \leq M$ and $\mathbb{E}[X] = 0$, then the the unique minimizer is $f_x = M \cdot \mathbf{1}_{[-\frac{1}{2M}, \frac{1}{2M}]}$. We then see,

$$\mathrm{Var}(X) = \int_{-\frac{1}{2M}}^{\frac{1}{2M}} t^2 \cdot M dt = \left[\frac{1}{3} M \cdot t^3\right]_{\frac{1}{-2M}}^{\frac{1}{2M}} = \frac{1}{12M^2}.$$

□

Chernoff bounds and binomial sums

Let $X_1, \ldots, X_n$ be independent $\{0,1\}$ random variables with $\mu = \mathbb{E}[\sum_{i=1}^n X_i]$. Then, the Chernoff bound gives [Doe11, Corollary 1.10]

$$\Pr\left[\sum_{i=1}^n X_i \leq \mu(1-\varepsilon)\right] \leq e^{-\frac{\varepsilon^2 \mu}{2}}, \varepsilon \in [0,1]. \tag{2.1}$$

$$\Pr\left[\sum_{i=1}^n X_i \geq \mu(1+\varepsilon)\right] \leq e^{-\frac{\varepsilon^2 \mu}{3}}, \varepsilon \in [0,1].$$

A more refined version is given by Azuma's inequality which allows the random variables to admit some mild dependencies. Let $X_1, \ldots, X_n$ be $\{0,1\}$ random variables with $\mu = \sum_{i=1}^n \mathbb{E}[X_i | X_1, \ldots, X_{i-1}]$. Then,

$$\Pr[\sum_{i=1}^n X_i \geq (1+\varepsilon)\mu] \leq e^{-\frac{\varepsilon^2 \mu^2}{2n}}, \varepsilon \in [0,1]. \tag{2.2}$$

To see this bound, apply [Doe11, Theorem 1.10.30] to the martingale $S_i := \sum_{j=1}^{i} X_j - \mathbb{E}[X_j|X_1, \dots, X_{j-1}]$.

Gaussian and sub-Gaussian random variables

If $\mu \in \mathbb{R}^m$ and Σ is an $m \times m$ positive-definite matrix, we denote by $\mathcal{N}(\mu, \Sigma)$, the law of the Gaussian with mean μ and covariance Σ. The probability density function of $\mathcal{N}(\mu, \Sigma)$ is given by $\frac{1}{\sqrt{2\pi}^m \det(\Sigma)^{1/2}} e^{-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)}$, $\forall x \in \mathbb{R}^n$.

The following is a basic concentration fact for the norm of the standard Gaussian that we will use in Chapter 4.

Lemma 2.3.3 ([LM00, Lemma 1]). *Let $G \sim \mathcal{N}(0, I_m)$ and let $x \geq 7m$. Then,*

$$\Pr\left(\|G\|^2 \geq x\right) \leq e^{-\frac{x}{3}}.$$

In Chapter 3 we will use the concept of sub-Gaussian random variables. A random variable $Y \in \mathbb{R}$ is *σ-sub-Gaussian* if for all $\lambda \in \mathbb{R}$, we have

$$\mathbb{E}[e^{\lambda Y}] \leq e^{\sigma^2 \lambda^2 / 2}. \tag{2.3}$$

A standard normal random variable $X \sim \mathcal{N}(0,1)$ is 1-sub-Gaussian. If variables $Y_1, \dots, Y_k \in \mathbb{R}$ are independent and respectively σ_i-sub-Gaussian, $i \in [k]$, then $\sum_{i=1}^k Y_i$ is $\sqrt{\sum_{i=1}^k \sigma_i^2}$-sub-Gaussian.
For a σ-sub-Gaussian random variable $Y \in \mathbb{R}$ we have the following standard tail bound [Ver18, Proposition 2.5.2]:

$$\max\{\Pr[Y \leq -\sigma s], \Pr[Y \geq \sigma s]\} \leq e^{-\frac{s^2}{2}}, s \geq 0. \tag{2.4}$$

The following standard lemma shows that bounded random variables are sub-Gaussian.

Lemma 2.3.4. *Let $X \in [-1,1]$ be a mean-zero random variable. Then X is 1-sub-Gaussian.*

Proof. Let $\varphi(x) := e^{\lambda x}$ for $\lambda \in \mathbb{R}$. By convexity of φ, note that for $x \in [-1,1]$, $\varphi(x) \leq \frac{1-x}{2}\varphi(-1) + \frac{1+x}{2}\varphi(1)$. Therefore,

$$\begin{aligned}\mathbb{E}[\varphi(X)] &\leq \mathbb{E}[\frac{1-X}{2}\varphi(-1) + \frac{1+X}{2}\varphi(1)] = \frac{1}{2}(\varphi(-1) + \varphi(1)) = \frac{1}{2}(e^{-\lambda} + e^{\lambda}) \\ &= \sum_{i=0}^{\infty} \frac{\lambda^{2i}}{(2i)!} \leq \sum_{i=0}^{\infty} \frac{(\lambda^2/2)^i}{i!} = e^{\lambda^2/2}, \text{ as needed.}\end{aligned}$$

□

We also need the following fact about truncated sub-Gaussian random variables, which is a slight generalization of [BDHT22, Lemma 7]:

Lemma 2.3.5. *Let $X \in \mathbb{R}$ be 1-sub-Gaussian. Then $\mathbb{E}[X^+] \leq 1/2$ and $X^+ - \mathbb{E}[X^+]$ is $\sqrt{2}$-sub-Gaussian.*

Proof. Since X is 1-sub-Gaussian, note that $\mathbb{E}[X] = 0$ and that $\mathbb{E}[X^2] \leq 1$. Therefore, by Proposition 2.3.1, we have $\mu := \mathbb{E}[X^+] = \mathbb{E}[|X|]/2 \leq \mathbb{E}[X^2]^{1/2}/2 \leq 1/2$ by Hölder. $\sqrt{2}$-sub-Gaussianity of $X^+ - \mu$ now follows verbatim from the proof of [BDHT22, Lemma 5] using that $\mu^2 \leq 1/3$ and replacing Gaussian by sub-Gaussian. □

Logconcave measures

If a measure ν has a density that is a logconcave function, we call ν logconcave. Logconcave distributions have many useful analytical properties. In particular, the marginals of logconcave random vectors are also logconcave.

Theorem 2.3.6 ([Pré71])**.** *Let $X \in \mathbb{R}^d$ be a logconcave random vector. Then, for any surjective linear transformation $T : \mathbb{R}^d \to \mathbb{R}^k$, TX is a logconcave random vector.*

The following gives a (essentially tight) bound on the maximum density of any one dimensional logconcave variable in terms of the variance.

Lemma 2.3.7 ([LV07, Lemma 5.5])**.** *Let $X \in \mathbb{R}$ be a logconcave variable. Then its density function is upper bounded by $\frac{1}{\sqrt{\mathrm{Var}[X]}}$.*

The above has an important consequence. If $X \in \mathbb{R}^n$ is logconcave and isotropic, then for any vector $v \in \mathbb{R}^n \setminus \{0\}$, the random variable $v^\mathsf{T} X$ has maximum density at most $1/\sqrt{\mathrm{Var}[v^\mathsf{T} X]} = 1/\|v\|_2$, where we have used $v^\mathsf{T} X$ is logconcave.

By a result of Grünbaum, the centroid of a convex set is also an approximate median [Grü60]. We will use the following generalization of this result to logconcave measures.

Lemma 2.3.8 ([BV04c])**.** *Let $X \in \mathbb{R}^n$ be a logconcave measure with mean $\mathbb{E}[X] = \mu$. Then for any $\theta \in \mathbb{S}^{n-1}$ and $t \in \mathbb{R}$, $\Pr[\theta^\mathsf{T} X \geq \theta^\mathsf{T}\mu - t] \geq 1/e - |t|$.*

We shall require the fact that logconcave random variables satisfy the following comparison inequality.

Lemma 2.3.9 ([Fra99])**.** *Let $X \in \mathbb{R}_+$ logconcave with $\mathbb{E}[X] = \mu$ and let Z have density e^{-x} (exponential distribution), $x \geq 0$. Then, for any convex function $\varphi : \mathbb{R}_+ \to \mathbb{R}$, $\mathbb{E}[\varphi(X)] \leq \mathbb{E}[\varphi(\mu Z)]$. In particular,*

1. $\mathbb{E}[X^2] \leq 2\mu^2$.
2. $\mathbb{E}[e^{\lambda X}] \leq \frac{1}{1-\lambda\mu}$, $\lambda < 1/\mu$.

Lemma 2.3.10. *For $X \in \mathbb{R}$ mean-zero and logconcave, we have $\mathbb{E}[X^2] \leq e\mathbb{E}[|X|]^2$.*

Proof. Let $X_l := -X \mid X \leq 0$, $p_l = \Pr[X \leq 0]$ and $X_r := X_r \mid X \geq 0$, $p_r = \Pr[X \geq 0]$. Note that X_l, X_r are both non-negative logconcave random variables and that $p_l, p_r \geq 1/e$ by Lemma 2.3.8. By Proposition 2.3.1, $p_l\mathbb{E}[X_l] = \mathbb{E}[X^-] = \mathbb{E}[|X|]/2 = \mathbb{E}[X^+] = p_r\mathbb{E}[X_r]$. We now see that

$$\mathbb{E}[X^2] = p_l\mathbb{E}[X_l^2] + p_r\mathbb{E}[X_r^2] \underbrace{\leq}_{Lemma\ 2.3.9} 2(p_l\mathbb{E}[X_l]^2 + p_r\mathbb{E}[X_r]^2)$$
$$= \mathbb{E}[|X|](\mathbb{E}[X_l] + \mathbb{E}[X_r]) \leq e\mathbb{E}[|X|]^2.$$

□

We will also require the following concentration inequality for sums of non-negative logconcave random variables.

Lemma 2.3.11. *Let $X_1, \ldots, X_n \in \mathbb{R}_+$ be independent non-negative logconcave random variables with mean μ. Then, the following holds:*

1. $\Pr[\sum_{i=1}^n X_i \geq (1+\varepsilon)\mu n] \leq e^{-n(\varepsilon - \ln(1+\varepsilon))} \leq e^{-n(\frac{\varepsilon^2}{2(1+\varepsilon)^2})}$, $\varepsilon > 0$.
2. $\Pr[\sum_{i=1}^n X_i \leq (1-\varepsilon)\mu n] \leq e^{-n(-\ln(1-\varepsilon)+\varepsilon)} = e^{-n(\sum_{j=2}^\infty \varepsilon^j/j)}$, $\varepsilon \in [0,1]$.

Proof. By homogeneity, we assume wlog that $\mu = 1$.
Proof of 1. Let $\lambda := \frac{\varepsilon}{1+\varepsilon}$. Then,

$$\Pr[\sum_{i=1}^n X_i \geq (1+\varepsilon)n] \underbrace{\leq}_{\text{Markov}} \mathbb{E}[e^{\lambda\sum_{i=1}^n X_i}]e^{-\lambda(1+\varepsilon)n}$$
$$\underbrace{\leq}_{Lemma\ 2.3.9} \left(\frac{1}{1-\lambda}\right)^n e^{-\lambda(1+\varepsilon)n} = e^{-n(\varepsilon - \ln(1+\varepsilon))}.$$

Proof of 2. Let $\lambda := \frac{\varepsilon}{1-\varepsilon}$. Then,

$$\Pr[\sum_{i=1}^n X_i \leq (1-\varepsilon)n] \underbrace{\leq}_{\text{Markov}} \mathbb{E}[e^{-\lambda\sum_{i=1}^n X_i}]e^{\lambda(1-\varepsilon)n}$$
$$\underbrace{\leq}_{Lemma\ 2.3.9} \left(\frac{1}{1+\lambda}\right)^n e^{-\lambda(1+\varepsilon)n} = e^{-n(-\ln(1-\varepsilon)-\varepsilon)}.$$

□

Finally, we will require concentration of truncated sums.

Lemma 2.3.12. *Let $X_1, \dots, X_n \in \mathbb{R}$ be i.i.d. mean zero logconcave random variables with $\mathbb{E}[X_1^+] = \alpha$. Then, for $\varepsilon \in [0, 1/2]$, we have that*

1. $\Pr[\sum_{i=1}^n X_i^+ \geq (1+\varepsilon)^2 n\alpha] \leq e^{-\frac{n\varepsilon^2}{3e}} + e^{-\frac{n\varepsilon^2}{2e(1+\varepsilon)}}$.
2. $\Pr[\sum_{i=1}^n X_i^+ \leq (1-\varepsilon)^2 n\alpha] \leq e^{-\frac{n\varepsilon^2}{2e}} + e^{-\frac{n(1-\varepsilon)\varepsilon^2}{2e}}$.

Proof. Define $X_1', \dots, X_n' \in \mathbb{R}_+$ to be i.i.d. copies of $X_1 \mid X_1 \geq 0$ and let $p = \Pr[X_1 \geq 0]$, where $1 - 1/e \geq p \geq 1/e$ (Lemma 2.3.8). Let $C = \sum_{i=1}^n 1[X_i \geq 0]$, which a binomial distribution with parameters n and p. Since $X_1, \dots, X_n$ are i.i.d., $\sum_{i=1}^n X_i^+$ has the same law as $\sum_{i=1}^C X_i'$. Therefore, by (2.1) and Lemma 2.3.11, we have that

$$\begin{aligned}\Pr[\sum_{i=1}^C X_i' \geq (1+\varepsilon)^2 n\alpha] &\leq \Pr[C \geq \lceil (1+\varepsilon)pn \rceil] + \Pr[\sum_{i=1}^{\lfloor (1+\varepsilon)pn \rfloor} X_i' \geq (1+\varepsilon)^2 n\alpha] \\ &\leq e^{-\frac{np\varepsilon^2}{3}} + e^{-\frac{(1+\varepsilon)np\varepsilon^2}{2(1+\varepsilon)^2}} \leq e^{-\frac{n\varepsilon^2}{3e}} + e^{-\frac{n\varepsilon^2}{e(1+\varepsilon)}},\end{aligned}$$

and

$$\begin{aligned}\Pr[\sum_{i=1}^C X_i' \leq (1-\varepsilon)^2 n\alpha] &\leq \Pr[C \leq \lfloor (1-\varepsilon)pn \rfloor] + \Pr[\sum_{i=1}^{\lceil (1-\varepsilon)pn \rceil} X_i' \leq (1-\varepsilon)^2 n\alpha] \\ &\leq e^{-\frac{np\varepsilon^2}{2}} + e^{-\frac{np(1-\varepsilon)\varepsilon^2}{2}} \leq e^{-\frac{n\varepsilon^2}{2e}} + e^{-\frac{n(1-\varepsilon)\varepsilon^2}{2e}}.\end{aligned}$$

□

Khintchine inequality

The Khintchine inequality provides bounds on the moments of weighted sums of independent ± 1 random variables [Khi23; Haa81]. In Chapter 3, we will use the following generalization to the case of random variables with bounded fourth moments.

Lemma 2.3.13. *Let $X_1, \dots, X_n \in \mathbb{R}$ be independent mean zero random variables satisfying $\mathbb{E}[X_i^4] \leq 3\mathbb{E}[X_i^2]^2 < \infty, \forall i$. Then, for any scalars $a_1, \dots, a_n \in \mathbb{R}$, we have that*

$$\sqrt{\frac{1}{3}\mathbb{E}[|\sum_i a_i X_i|^4]} \leq \mathbb{E}[|\sum_i a_i X_i|^2] \leq 3\mathbb{E}[|\sum_i a_i X_i|]^2.$$

Proof. Letting $Z := |\sum_i a_i X_i| \geq 0$, we wish to show $\sqrt{\mathbb{E}[Z^4]} \leq \mathbb{E}[Z^2] \leq 3\mathbb{E}[Z]^2$. We first show that $\mathbb{E}[Z^4] \leq 3\mathbb{E}[Z^2]^2$, proving the first part of our claim:

$$\begin{aligned}\mathbb{E}[Z^4] &= \sum_{i,j,k,l} a_i a_j a_k a_l \mathbb{E}[X_i X_j X_k X_l] = \sum_i a_i^4 \mathbb{E}[X_i^4] + \sum_{i \neq j} 3a_i^2 a_j^2 \mathbb{E}[X_i^2]\mathbb{E}[X_j^2] \\ &= \sum_i a_i^4 (\underbrace{\mathbb{E}[X_i^4] - 3\mathbb{E}[X_i^2]^2}_{\leq 0}) + \sum_{i,j} 3a_i^2 a_j^2 \mathbb{E}[X_i^2]\mathbb{E}[X_j^2] \\ &\leq 3(\sum_i a_i^2 \mathbb{E}[X_i^2])^2 = 3\mathbb{E}[Z^2]^2.\end{aligned}$$

As a consequence, we have

$$\mathbb{E}[Z^2] = \mathbb{E}[Z^{2/3}(Z^4)^{1/3}] \underbrace{\leq}_{\text{Hölder}} \mathbb{E}[Z]^{2/3}\mathbb{E}[Z^4]^{1/3} \underbrace{\leq}_{\mathbb{E}[Z^4] \leq 3\mathbb{E}[Z^2]^2} 3^{1/3}\mathbb{E}[Z]^{2/3}\mathbb{E}[Z^2]^{2/3},$$

which, after rearranging, gives $\mathbb{E}[|\sum_i a_i X_i|^2] \leq 3\mathbb{E}[|\sum_i a_i X_i|]^2$. □

Rejection sampling

In the proofs of Theorems 3.1.1 and 3.1.2 we make use of a tool called rejection sampling. This tool allows us to change the distribution of a random variable $X \sim \mathcal{D}$. It works by 'accepting' an evaluation of the variable with a probability that depends on its value. Let us formally define a rejection sampling procedure first.

Definition 2.3.14. A rejection sampling procedure on a random variable X is a randomized algorithm ψ that, given a realization of X, outputs either reject or accept.

Now the variable X conditioned on $\psi(X) = \text{accept}$, will be distributed according to a probability $\mathcal{D}'$. We can choose the distribution $\mathcal{D}'$ by appropriately setting $\Pr[\psi(X) = \text{accept}|X]$. This is formalized in the following lemma.

Lemma 2.3.15. *Let X, Y be a random variable on some set S, with probability density functions f_X and f_Y, and* $\text{Support}(Y) \subseteq \text{Support}(X)$. *Suppose that $\frac{f_Y(s)}{f_X(s)} \leq K$ for all $s \in S$. Then there exists a rejection sampling procedure ψ with* $\text{Law}(X|\psi(X) = \text{accept}) = \text{Law}(Y)$ *and* $\Pr[\psi(X) = \text{accept}] = \frac{1}{K}$.

Proof. Define ψ such that:

$$\Pr[\psi(X) = \text{accept}|X = x] = \begin{cases} \frac{f_Y(x)}{K f_X(x)} & x \in \text{Support}(X) \\ 0 & \text{else} \end{cases}.$$

This probability is well defined as $\frac{f_Y(x)}{Kf_X(x)} \le \frac{K}{K} = 1$ for all $x \in \text{Support}(X)$. Now $\Pr[\psi(X) = \text{accept}] = \int_{x\in\text{Support}(X)} \frac{f_Y(x)}{Kf_X(x)} f_X(x)dx = \frac{1}{K}$. Call the variable X conditioned $(\psi(X) = \text{accept})$, Z. Now:

$$f_Z(z) = \frac{\frac{f_Y(x)}{Kf_X(x)} f_X(x)}{\Pr[\psi(X) = \text{accept}]} = \frac{f_Y(x)}{K \cdot 1/K} = f_Y(x).$$

This proves the statement. □

2.4 Nets

Let $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d : \|x\|_2 = 1\}$ denote the unit sphere in $\mathbb{R}^d$. We say that $N \subseteq \mathbb{S}^{d-1}$ is an ε-net if for every $x \in \mathbb{S}^{d-1}$ there exists $y \in N$ such that $\|x - y\| \le \varepsilon$. A classic result we will need is that $\mathbb{S}^{d-1}$ admits an ε-net N_ε of size $|N_\varepsilon| \le (1 + \frac{2}{\varepsilon})^d$. See, for example, [Ver18, Chapter 5]. We note that the same bound holds if we wish to construct a net of some subset $A \subseteq \mathbb{S}^{d-1}$ and we wish to have $N_\varepsilon \subseteq A$.

Chapter 3

An average-case analysis of branch-and-bound via integrality gaps

In this chapter we analyze the 'average-case complexity' of the branch-and-bound algorithm, by studying its performance on randomly generated integer programs. Consider the following integer program with n variables and m constraints:

$$\begin{aligned} \mathrm{val}_{\mathsf{IP}} := \max_x \ & \mathrm{val}(x) = c^\mathsf{T} x \\ \text{s.t. } & Ax \leq b \\ & x \in \{0,1\}^n \end{aligned} \tag{3.1}$$

For several classes of distributions we show that with high probability any branch-and-bound tree for such an IP is of size only polynomial in the number of variables of the IP. We will cover high-probability bounds on the performance of branch-and-bound for the following distributions of random IPs:

- Theorem 3.1.4: Both A and c having independent entries uniformly sampled from $[0, 1]$, which was originally studied by [DDM23].
- Theorem 3.1.5: A having independent columns, coming from logconcave isotropic distribution and c having independent entries from the standard normal distribution, studied [BDM23]. This contains the special case where A and c have Gaussian entries, studied in [BDHT22].
- Theorem 3.1.5: A having independent entries sampled uniformly from the set $\{-k, \ldots, k\}$ and c having independent entries from the standard normal distribution, studied in [BDM23].
- Theorem 3.1.5: A having independent entries sampled uniformly from the set $\{1, \ldots, k\}$ and c having independent entries from the standard normal distribution, studied in [BDM23].

The contents of this chapter are based on joint work with Daniel Dadush and Dan Mikulincer [BDM23], and joint work with Daniel Dadush, Sophie Huiberts, and Samarth Tiwari [BDHT22; BDHT21].

The proofs of these results all make use of the framework introduced by [DDM23], relating the size of the branch-and-bound tree to the additive integrality gap. To prove the results, we show that the integrality gap for IPs sampled from distributions from the latter three classes is at most $O(\log(n)^2/n)$ with probability at least $1-1/\operatorname{poly}(n)$ for constant m. The proof relies on results on the discrepancy of random set systems that are covered in Chapter 4.

3.1 Introduction

Bounds on the integrality gap

An important factor controlling our ability to solve IPs is the tightness of the linear programming (LP) relaxation. A natural way to measure tightness is the size of the gap

$$\mathsf{IPGAP} := \mathrm{val}_{\mathsf{LP}} - \mathrm{val}_{\mathsf{IP}},$$

where $\mathrm{val}_{\mathsf{LP}}$ relaxes $x \in \{0,1\}^n$ to $x \in [0,1]^n$.

The integrality gap of integer linear programs forms an important measure for the complexity of solving said problem in a number of works on the average-case complexity of integer programming [BV04a; DDM23; DF92; DF89; GM84; Lue82].

In practice, automated methods for tightening LP relaxations such as presolving and cutting planes are crucial to the performance of modern IP solvers [BR07; AW13]. Presolving refers to simple inference rules applied to constraints in sequence, which among other things are used to find implied variable fixings, tighten variable bounds and strengthen the coefficients of inequalities [Sav94]. Cutting planes refers to additional valid linear inequalities for the IP, which are often generated from the optimal simplex tableau [CCZ10]. The effectiveness of cutting planes is very often measured in terms of the fraction of the integrality gap they close, which helps justify the integrality gap as a key metric in practice [BCC96; FS13; Fis+16].

In the framework of Dey, Dubey and Molinaro [DDM23] the size of branch-and-bound trees is related to the integrality gap. For models directly captured by their result, suitable bounds on the integrality gap have been proven for random packing [DF89; GM84; Lue82], where the entries of (A,c) are independent uniform $[0,1]$. For random packing IPs, when $b \in (n/4, n/2)^m$, Dyer and Frieze [DF89] proved that $\mathsf{IPGAP} \le 2^{O(m)}\log^2(n)/n$ with probability at least $1 - 1/\operatorname{poly}(n) - 2^{-\operatorname{poly}(m)}$.

In this chapter, we analyze the integrality gap of (3.1) for three different cases. Theorem 3.1.1 gives a high probability bound on the integrality gap for IPs with entries of c that are all independent Gaussian $\mathcal{N}(0,1)$ distributed, and columns of A

that are independently sampled from either a logconcave isotropic distribution, or uniformly from the set of integer vectors with entries between $-k$ and k. We refer to these as *Centered IPs.*

We will show that in all these models the integrality gap will be bounded by $g = O_m(\log^2 n/n)$. This implies that the complexity of branch-and-bound grows as $e^{O_m(\sqrt{ng})} = e^{O_m(\log n)} = n^{O_m(1)}$.

Theorem 3.1.1 (Gap Bound for Centered IPs). *For $m \geq 1$, $n \geq \mathrm{poly}(m)$, $b \in \mathbb{R}^m$ with $\|b^-\|_2 \leq O(n)$, if c has i.i.d. $\mathcal{N}(0,1)$ entries and the columns of A are independent isotropic, logconcave random vectors whose support is contained in a ball of radius $O(\sqrt{\log n} + \sqrt{m})$, then*

$$\Pr\left(\mathsf{IPGAP} \geq \frac{\mathrm{poly}(m)(\log n)^2}{n}\right) \leq n^{-\mathrm{poly}(m)}.$$

Furthermore, the same result holds if the entries of A are distributed independently and uniformly in $\{0, \pm 1, \dots, \pm k\}$ and $b \in \mathbb{Z}^m$ with $\|b^-\|_2 \leq O(kn)$, for any $1 \leq k \leq n$.

We recall that a random vector $X \in \mathbb{R}^m$ is isotropic if $\mathbb{E}[X] = 0$ (mean zero) and $\mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])^\mathsf{T}] = I_m$ (identity covariance). Since one can apply an affine transformation to any full dimensional random variable to make it isotropic, this condition may be regarded as a useful normalization. When A has i.i.d. $\mathcal{N}(0,1)$ entries, we remark that the columns of A have norm bounded by $O(\sqrt{m} + \sqrt{\log n})$ with probability $1 - 1/\mathrm{poly}(n)$ by standard tail bounds. Thus, this Gaussian setting is captured by the theorem.

Theorem 3.1.2 provides a similar bound for when the entries of c come from an exponential distribution and the entries of A are uniformly sampled from $\{1, \dots, k\}$. It can be seen as a discrete variant of the random packing IP gap bound of [DF89].

Theorem 3.1.2 (Discrete Packing IPs). *For $m \geq 1$, $k \geq 3$, $\beta \in (0, 1/4)$, $n \geq \mathrm{poly}(m,k)\exp(\Omega(1/\beta))$, $b \in ((kn\beta, kn(1/2 - \beta)) \cap \mathbb{Z})^m$, if c has i.i.d. exponential entries and the entries of A are independent and uniform in $\{1, \dots, k\}$, then*

$$\Pr\left(\mathsf{IPGAP} \geq \frac{\exp(O(1/\beta))\,\mathrm{poly}(m)(\log n)^2}{n}\right) \leq n^{-\mathrm{poly}(m)}.$$

Compared to [DF89] we require exponentially distributed objective coefficients, i.e., with density e^{-x}, $x \geq 0$, instead of uniform $[0,1]$. The additional smoothness of the distribution makes the arguments much cleaner while preserving the qualitative nature of the results. We remark that extending the above gap bound to the setting of non-negative logconcave random vectors is also possible, though we do not pursue this here.

The above results extend our understanding of gap bounds in two significant ways. Firstly, they show that strong gap bounds are obtainable when the entries of the constraint matrix take only small integer values. This is well motivated by the fact that the constraints for practical IPs are often combinatorial and thus are expressed using small integer coefficients. Secondly, in the case of centered IPs, we show that non-trivial correlations between the variables in a column of the constraint (induced by logconcavity) can be handled. Thus, we establish a limited form of universality for these gap bounds.

We would like to stress that neither of these extensions is a priori obvious. To highlight the subtleties in the discrete setting, we note that even for $m = 1$, the gap bound does not hold when the right hand side b is non-integral. As a simple example, it is easy to see that for n odd, the integrality gap of the knapsack program $\max \sum_{i=1}^n x_i$ subject to $\sum_{i=1}^n x_i \leq n/2,\ x \in \{0,1\}^n$ is $1/2$. It is not hard to check that a constant gap is preserved with overwhelming probability even if the profit vector c is exponentially distributed and weights $a_1, \ldots, a_n$ are drawn uniformly from $\{1, \ldots, k\}$, for k fixed.

To circumvent the above issue, a crucial step in our proof in the discrete setting is to show that we can round the optimal LP solution x^* to an integer solution x' that is tight on the same set of constraints. That is, $(Ax^*)_i = b_i \Rightarrow (Ax')_i = b_i, \forall i \in [m]$ (which is only possible if b is integral). Indeed, one of our main technical contributions, described in the next section, is to give general conditions under which such exact roundings are possible, using Fourier analytic techniques from discrepancy theory. We remark that Dyer and Frieze [DF92] believed that a discrete extension of their gap bounds should be possible, provided the range of the discrete distribution (i.e., k above) grows with n, which we confirm here without this assumption[1].

Beyond generalizing previous work, our results also improve the probabilistic guarantees. Compared to prior works, our gap bounds hold with high probability $1 - 1/\operatorname{poly}(n)$ instead of $1 - 1/\operatorname{poly}(n) - 2^{-\operatorname{poly}(m)}$ (which does not converge to 1 as n grows for fixed m). Achieving a high probability bound via prior estimates increases the gap by an $O(\log n)$ factor, which makes the corresponding branch-and-bound complexity quasi-polynomial instead of polynomial.

Is it natural to ask whether there are natural limitations to extending these gap bounds to much wider classes of IPs. For logconcave random IPs, one may observe that any worst-case instance can be encoded as the mean $\mathbb{E}[(A, b, c)]$ of the random instance. In this way, one can view the distribution (A, b, c) as a smoothed version of worst-case IP instance $\mathbb{E}[(A, b, c)]$. By appropriately scaling up the means,

[1]We note that our gap bounds seemingly require that n be somewhat larger than the range k. This is an artificial restriction. For (very) large k, one must treat the discrete distribution as if it were continuous, which requires a slight adaptation of the proofs.

or equivalently scaling down the variance of the entries of (A, b, c), the instance (A, b, c) converges to the worst-case instance $\mathbb{E}[(A, b, c)]$. It is not hard to check that for many hard combinatorial problems such as SET COVER or MAX 3-SAT, adding small random perturbations (of appropriate sign) to the instance data does not change the optimal solution, and thus we cannot expect strong integrality gap results in these settings.

We note that smoothed analysis of integer programming has indeed been studied by various works [BV04a; RV07]. In particular, Röglin and Vöcking [RV07] showed that under mild conditions, the class of IPs that are easy to solve on average over suitable random perturbations of the instance data (i.e., that can be solved in polynomial time with high probability over the perturbations) correspond exactly to the class of IPs solvable in pseudopolynomial time.

Bounds on the tree size

In recent breakthrough work, Dey, Dubey and Molinaro [DDM23] provided a framework for deriving upper bounds on the size of branch-and-bound trees for random IPs with small integrality gaps. Their framework consists of two parts. In the first part, one deterministically relates the size of any branch-and-bound tree using best-first search node selection to the size of knapsack polytopes whose weights are induced by reduced costs and whose capacity is equal to the integrality gap. We recall that in the best-first node selection rule, the next node to be processed is always the node whose LP relaxation value is the largest.

In the second part of the framework, one leverages the randomness in the coefficients of A, c to upper bound the maximum size of the knapsack in (3.6). The authors of [DDM23], give such an upper bound for the specific packing instances studied by Dyer and Frieze [DF89]. In this chapter, we generalize their probabilistic framework to all IPs with independent continuously distributed entries of c. We now state our main meta-theorem, which we prove in Section 3.2.

Theorem 3.1.3. *Let $A \in \mathbb{R}^{m \times n}$, b be random variables. Let $c_1, \dots, c_n$ be independent random variables, each having probability density at most 1. Let $P \subseteq [0,1]^n$ be an integral polytope. Then, for $g \geq 0$, $\delta \in (0,1)$, with probability at least*

$$1 - \Pr_{A,b,c}[\mathsf{IPGAP} \geq g] - \delta,$$

the branch-and-bound algorithm equipped with the best-first search rule applied to the

ILP

$$\begin{aligned} &\max c^{\mathsf{T}} x \\ &\textit{s.t. } Ax \leq b \\ &\quad x \in P \cap \{0,1\}^n. \end{aligned}$$

produces a tree of size at most

$$\frac{n^{O(m)} \exp(\sqrt{12ng})}{\delta} \tag{3.2}$$

The above statement is more general than its equivalent in [BDM23, Theorem 3] in two ways. Firstly, the result also holds when P is not equal to the standard $[0,1]^n$ hypercube. This allows us to obtain a bound on the tree size for instances of the generalized assignment problem, which is not covered by the previous result. Secondly, the above theorem does not pose any restrictions on the distribution of the entries of A.

When A, c have i.i.d. uniform $[0,1]$ coefficients and $b = \beta n$, $\beta \in (0,1/2)^m$ and $\beta_{\text{min}} := \min_{i \in [n]} \beta_i$, Dyer and Frieze [DF89] proved that for n large enough

$$\Pr_{A,c}[\mathsf{IPGAP} \geq \alpha a_1 \log^2 n/n] \leq 2^{-\alpha/a_2} + 1/(2n), \forall \alpha \geq 1,$$

where $a_1 = \Theta(1/\beta_{\text{min}})^m$ and $a_2 = 2^{\Theta(m)}$. In [DDM23], Dey, Dubey and Molinaro use this integrality gap result combined with a probabilistic analysis of the bound in Theorem 3.2.1 to show that the tree size is at most

$$n^{O(ma_1 \log a_1 + \alpha a_1 \log m)}$$

with probability $1 - 2^{-\alpha/a_2} - 1/n$.

A stronger bound can be obtained from Theorem 3.1.3. Plugging the integrality gap $g = \alpha a_1 \log^2 n/n$ into Theorem 3.1.3 with $\delta = 2^{\alpha/a_2}$, we immediately get the following improved tree size bound.

Theorem 3.1.4. *With probability* $1 - 2^{-\alpha/a_2} - 1/n$ *the branch-and-bound algorithm, equipped with the best-first node selection rule, applied to* (3.1) *produces a tree of size at most* $n^{O(m)+\sqrt{3\alpha a_1}}$ *when* A, c *have i.i.d. uniform* $[0,1]$ *coefficients and* $b = \beta n$, $\beta \in (0,1/2)^m$, $\beta_{\text{min}} := \min_{i \in [n]} \beta_i$ *and* $a_1 = \Theta(1/\beta_{\text{min}})^m$ *and* $a_2 = 2^{\Theta(m)}$

Proceeding in a similar fashion, we can easily derive a tree-size bound for Gaussian IPs by combining the previous theorem with Theorems 3.1.1 and 3.1.2.

Theorem 3.1.5. *With probability* $1 - 1/\operatorname{poly}(n)$, *the branch-and-bound algorithm, equipped with the best-first node selection rule, applied to* (3.1) *produces a tree of size at most* $n^{\operatorname{poly}(m)}$ *in the Centered IP model of Theorem 3.1.1 and of size* $n^{\exp(O(1/\beta)) \operatorname{poly}(m)}$ *in the Discrete Packing IP model from Theorem 3.1.2.*

Proof overview

To bound the integrality gap, our proof strategy follows along similar lines to that of Dyer and Frieze [DF89], which we now describe. In their strategy, one first solves an auxiliary LP $\max c^{\mathsf{T}}x, Ax \le b - \varepsilon 1_m$, for $\varepsilon > 0$ small, to get its optimal solution x^*, which is both feasible and nearly optimal for the starting LP (proved by a simple scaling argument), together with its optimal dual solution $u^* \ge 0$ (see Section 3.3 for the formulation of the dual). From here, they round down the fractional components of x^* to get a feasible IP solution $x' := \lfloor x^* \rfloor$. We note that the feasibility of x' depends crucially on the packing structure of the LPs they work with, i.e., that A has non-negative entries (which does not hold in the Gaussian setting).

Finally, they construct a nearly optimal integer solution x'', by carefully choosing a subset of coordinates $T \subseteq \{i \in [n] : x'_i = 0\}$ of size $O(\mathrm{poly}(m) \log n)$, where they flip the coordinates of x' in T from 0 to 1 to get x''. The coordinates of T are chosen according to the following criteria. Firstly, the coordinates should be *very cheap* to flip, which is measured by the absolute value of their *reduced costs*. Namely, they enforce that $|c_i - A_i^{\mathsf{T}} u^*| = O(\log n/n), \forall i \in T$. Secondly, T is chosen to make the *excess slack* $\|A(x^* - x'')\|_\infty \le 1/\mathrm{poly}(n)$, i.e., negligible. We note that guaranteeing the existence of T is highly non-trivial. Crucial to the analysis is that after conditioning on the exact value of x^* and u^*, the columns of $W := \begin{bmatrix} c^{\mathsf{T}} & A \end{bmatrix}^{\mathsf{T}} \in \mathbb{R}^{(m+1)\times n}$ (the objective extended constraint matrix) that are indexed by $N_0 := \{i \in [n] : x_i^* = 0\}$ are independently distributed subject to having negative reduced cost, i.e., subject to $c_i - A_i^{\mathsf{T}} u^* < 0$ for $i \in N_0$ (see Lemma 3.3.2). It is the large amount of left-over randomness in these columns that allowed Dyer and Frieze to show the existence of the subset T via a discrepancy argument (more on this below). Given a suitable T, a simple sensitivity analysis is used to show the bound on the gap between $c^{\mathsf{T}} x''$ and the (Primal LP) value. This analysis uses the basic formula for the optimality gap between primal and dual solutions (see (Gap Formula) in subsection 3.3), and relies upon bounds on the size of the reduced costs of the flipped variables, the total excess slack and the norm of the dual optimal solution u^*.

The discrepancy theorem

To apply this approach for centered IPs and discrete packing IPs, the main challenge is to find an analogue to the discrepancy lemma of Dyer and Frieze [DF89, Lemma 3.4]. This lemma posits that for any large enough set of "suitably random" columns in $\mathbb{R}^m$ and any not too big target vector $D \in \mathbb{R}^m$, with non-negligible probability there exists a set containing half the columns whose sum is very close to D. This is the main lemma used to show the existence of the subset T, chosen from a suitably filtered subset of the columns of A in N_0, used to reduce the excess slack. In

Chapter 4 we prove an improved version of the lemma (Theorem 4.3.4) that works for centered IPs and discrete packing IPs. The results hold for approximately symmetric, anti-concentrated probability distributions, as defined below.

Definition 4.3.1 (approximately symmetric distributions)**.** A probability distribution $\mathcal{D}$ on $\mathbb{R}^m$, with mean μ, is called *approximately symmetric* if, for any $\nu \in \mathbb{R}^m$,

$$\Pr_{X\sim\mathcal{D}}\left(\langle X, \nu\rangle \geq \langle \mu, \nu\rangle\right) \geq \frac{1}{4e^2}.$$

Observe that all symmetric distributions are approximately symmetric.

Definition 4.3.3 (anti-concentration)**.** Let $\sigma \geq 0$ and $\kappa \in (0,1)$. We say the measure $\mathcal{D}$ is (σ,κ)*-anti-concentrated* if $\mathrm{Cov}(\mathcal{D}) \preceq \sigma^2 I_m$, and for any $\nu \in \mathbb{R}^m$ and any $\theta \in V$,

$$\Pr_{X\sim\mathcal{D}}\left[d(\theta^\mathsf{T} X, \mathbb{Z}) \geq \kappa \min\left(1, \|\theta\|_\infty \sigma\right) \mid \langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right] \geq \kappa, \tag{AC}$$

where $d(\theta^\mathsf{T} X, \mathbb{Z}) := \inf_{z\in\mathbb{Z}} |\theta^\mathsf{T} X - z|$. When σ is clear from the context, we will sometimes omit the dependence on σ from the definition.

We use the following corollaries to lemma to prove our bounds on the integrality gap.

Theorem 4.3.5. *Suppose the columns of $A \in \mathbb{R}^{m\times n}$ are continuously distributed, independent with a common mean $\mu \in \mathbb{R}^m$, are approximately symmetric around their mean, and $(\Theta(1), \Omega(1))$-anti-concentrated. Let $p \in [0,1]$ with $\frac{\mathrm{poly}(m)\log(n)}{n} \leq p \leq \frac{1}{\mathrm{poly}(m)}$.*

Then, with probability $1-e^{-\Omega(pn)}$ for every t with $\|t-pn\mu\| \leq O\left(\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $\|A\mathbf{1}_S - t\| \leq \exp\left(-\Omega(\frac{pn}{m})\right)$.

The above corollary includes the case where the columns of A are logconcave, isotropic, by Lemma 4.4.1.

Theorem 4.3.7. *Suppose the entries of A are uniformly sampled from $\{i, i+1, \ldots, i+k\}$ for $k \geq j \geq \max(2, |i|)$. Let $p \in [0,1]$ with $\frac{\mathrm{poly}(m)\log(n)\log(k)}{n} \leq p \leq \frac{1}{\mathrm{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every vector $t \in \mathbb{Z}^n$ with $\|t - pn\mu\| \leq O\left(k\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $A\mathbf{1}_S = t$.

In addition to generalizing the discrepancy lemma to work for both logconcave and discrete probability distributions, these corollaries also provide a quantitative improvement. Namely, for each of the statements the failure probability decreases exponentially in n, whereas in [DF89], the probability of failure is constant.

When applying the above to find the slack repairing set S, n will roughly be $O(\text{poly}(m)\log n)$ and $p = 1/\text{poly}(m)$. The corresponding success probability will now be $1 - n^{-\text{poly}(m)}$ as opposed to $\Theta(1)$, which is what allows us to prove much better tail bounds for the integrality gap.

Gap bounds via discrepancy

Given the discrepancy theorem, the proof of the gap bounds mirrors the proofs in [DF89; BDHT22] as described above, though with non-trivial technical adaptations as well as some simplifications.

As in prior work, the relevant properties of the optimal primal x^* and dual solution u^* must be established in these generalized settings (see Lemma 3.3.4 and Lemma 3.3.12). In particular, one must show that the norm of u^* is suitably bounded, and that there are sufficiently many columns of A with small reduced costs indexed by the zeros of x^*. These properties can be derived using standard tools for concentration and anti-concentration of logconcave and uniform discrete random variables.

Providing "nice enough" columns for our discrepancy theorem can however be challenging. For centered IP where A has independent logconcave columns, conditioning on the size of the reduced costs can significantly perturb the mean of each column (possibly in different directions when the columns are not i.i.d.). We overcome this problem using sophisticated forms of rejection sampling, which allows to "virtually recenter" each logconcave column (see Lemma 3.3.9). Interestingly, the rejection sampling procedure does not even preserve logconcavity, however the properties required for our more general discrepancy theorem persist (see Theorem 4.3.4).

In terms of simplifications, compared to earlier proofs we no longer require repeated trials on disjoint subsets of columns of A to find a suitable slack repairing set T. In particular, due to our new discrepancy theorem, using all the small reduced cost columns together both exponentially decreases the probability of failure and increases the size of the targets one can hit. Furthermore, since the discrepancy theorem directly applies to columns with non-zero means, one can avoid ad-hoc reductions to the mean-zero case as in the proof of [DF89] for the packing case.

Improvements for centered IPs

We can see that there is an important difference between the result for discrete packing instances in Theorem 3.1.2 and the one for centered IPs in Theorem 3.1.1: in the former, we require the entries of b to be contained in $(kn\beta, O(kn(1/2-\beta)))$ for some $\beta \in (0, 1/4)$ which occurs in the probability bound, whereas in the latter, we only require that $\|b^-\|_2 \leq O(n)$. The strict requirements from the discrete packing case also appear in the bound for IPs with uniformly random coefficients of Dyer and Frieze [DF89].

This difference results from the fact that the proof involves flipping the with columns of A having reduced costs in $[-\delta, 0]$ for some small $\delta > 0$. The number of such suitable columns increases as δ grows larger. In the centered case, when $\|b^-\|_2 \leq O(n)$ we get enough suitable columns by setting $\delta = \frac{\text{poly}(m)\log(n)}{n}$.

In the packing setting, setting $\delta = \frac{\text{poly}(m)\log(n)}{n}$ would also result in a sufficient number of suitable columns, as long as the entries of b are $O(kn)$. However, the distribution of these columns does not satisfy the anti-concentration property in Eq. (AC). By using a step of rejection sampling with success probability $\exp(O(1/\beta))$, we can make the distribution satisfy the property. To compensate for the loss in success probability, δ needs to be increased to $\frac{\exp(O(1/\beta))\,\text{poly}(m)\log(n)}{n}$, which results in an extra factor of $\exp(O(1/\beta))$ in the integrality gap bound.

Related work

The worst-case complexity of solving $\max\{c^\mathsf{T}x : Ax = b, x \geq 0, x \in \mathbb{Z}^n\}$ scales as $n^{O(n)}$ times a polynomial factor in the bit complexity of the problem. This is a classical result due to Lenstra [Len83] and Kannan [Kan87] which is based on lattice basis reduction techniques.

Beyond these worst-case bounds, the performance of basis reduction techniques for determining the feasibility of random integer programs has been analyzed. In this context, basis reduction is used to reformulate $Ax \leq b, x \in \mathbb{Z}^n$ as $AUw \leq b, w \in \mathbb{Z}^n$ for some unimodular matrix $U \in \mathbb{Z}^{n\times n}$, after which a simple variable branching scheme is applied (i.e., branching on integer hyperplanes in the original space). Furst and Kannan [FK89] showed that subset-sum instances of the form $\sum_{i=1}^n x_i a_i = b$, $x \in \{0,1\}^n$, where each a_i, $i \in [n]$, is chosen uniformly from $\{1, \dots, M\}$ and $b \in \mathbb{Z}_+$, can be solved in polynomial time with high probability in this way if $M = 2^{\Omega(n^2)}$. Pataki, Tural and Wong [PTW10] proved generalizations of this result for IPs of the form $f \leq Ax \leq g, l \leq x \leq u, x \in \mathbb{Z}^n$, where the coefficients of A are uniform in $\{1, \dots, M\}$ and M is "large" compared to $\|(g-f, u-l)\|$. Apart from the different type of branching, compared to the present work, we note that the IPs analyzed in these models are either infeasible or have a unique feasible solution with high probability.

Another line of works has analyzed dynamic programming algorithm solving IPs with integer data [EW19; JR23; Pap81]. For $A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^m$, [JR23] proved that $\max\{c^\mathsf{T} x : Ax = b, x \geq 0, x \in \mathbb{Z}^n\}$ can be solved in $O(\sqrt{m}\Delta)^{2m} \log(\|b\|_\infty) + O(nm)$ time, where Δ is the largest absolute value of entries in the input matrix A. Integer programs of the form $\max\{c^\mathsf{T} x : Ax = b, 0 \leq x \leq u, x \in \mathbb{Z}^n\}$ can similarly be solved in time

$$n \cdot O(m)^{(m+1)^2} \cdot O(\Delta)^{m\cdot(m+1)} \log^2(m \cdot \Delta),$$

which was proved in [EW19]. Note that integer programs of the form $\max\{c^\mathsf{T} x : Ax \leq b, x \in \{0,1\}^n\}$ can be rewritten in this latter form by adding m slack variables.

The complexity of integer programming has also been studied from the perspective of *smoothed analysis*. In this context, Röglin and Vöcking [RV07] proved that a class of IPs satisfying some minor conditions has polynomial smoothed complexity if and only if that class admits a pseudopolynomial time algorithm. An algorithm has polynomial smoothed complexity if its running time is polynomial with high probability when its input has been perturbed by adding random noise, where the polynomial may depend on the inverse magnitude φ^{-1} of the noise as well as the dimensions n, m of the problem. An algorithm runs in pseudopolynomial time if the running time is polynomial when the numbers are written in unary, i.e., when the input data consists of integers of absolute value at most Δ and the running time is bounded by a polynomial $p(n, m, \Delta)$. In particular, they prove that solving the randomly perturbed problem requires only polynomially many calls to the pseudopolynomial time algorithm with numbers of size $(nm\varphi)^{O(1)}$ and considering only the first $O(\log(nm\varphi))$ bits of each of the perturbed entries.

Few results are known on the complexity of the branch-and-bound algorithm for specific problem classes. For a small number of combinatorial problems such as vertex cover [Lok+14] and multiway cut [Lok+14] the algorithm has been shown to be FPT when parameterized by the integrality gap of the natural LP relaxation. Since then, multiple general frameworks for capturing such problems have been developed [IWY16; Wah17]. These results crucially rely on the use of half-integral and persistent LP relaxations, which are known only for very structured problems. The results presented here thus mainly provide complementary evidence for the effectiveness of branch-and-bound in the unstructured setting.

While branching on variables is most common in practice, there is a theoretical line of work examining the effectiveness of branching schemes for solving random integer programs based on general integer disjunctions. These schemes rely on sophisticated lattice basis reduction methods [LLL82] and follow the template of Lenstra's celebrated polynomial time algorithm for integer programming in fixed dimension [Len83]. In particular, Furst and Kannan [FK89] showed that certain random subset-sum instances can be solved in polynomial time via basis reduction, and Pataki, Tural and Wong [PTW10] extended these results to certain IPs with multiple constraints. Apart from the different type of branching, compared to the present work, the IPs analyzed in these models are either infeasible or have a unique feasible solution with high probability.

Bounds on the integrality gap. In prior work, strong gap bounds have also been proven for random instances of combinatorial optimization problems, though this has not translated into good upper bounds on the size of branch-and-bound trees. Frieze and Sorkin [FS07] showed that the cycle cover relaxation for the asymmetric TSP has an expected $O(\log^2 n/n)$ additive integrality gap, where the edge weights are chosen uniformly from $[0, 1]$ for a complete digraph on n vertices and gave a $2^{\tilde{O}(\sqrt{n})}$ algorithm which solves these instances with high probability. It is an interesting open question to understand if LP-based branch-and-bound can recover the same running times as the specialized algorithms above for these problems.

Organization

In Section 3.2 we prove the meta-theorem for the size of the branch-and-bound tree. Our main contribution, the gap bounds for centered IPs and discrete packing IPs, are proven in Section 3.3. In Section 3.4 we conclude the chapter with some open problems.

3.2 Bounding the tree size

In this chapter, we will prove bounds on the size of the branch-and-bound tree for integer programs of the following form.

$$\begin{aligned} \mathsf{val}_{\mathsf{IP}} = \max\ & c^{\mathsf{T}} x \\ \text{s.t. }\ & Ax \leq b \\ & P \cap \{0,1\}^n. \end{aligned} \tag{3.3}$$

Here P is a polytope contained in $[0, 1]^n$. The integrality gap of this ILP is defined as $\mathsf{IPGAP} := \mathsf{val}_{\mathsf{LP}} - \mathsf{val}_{\mathsf{IP}}$, where

$$\begin{aligned} \mathsf{val}_{\mathsf{LP}} := \max_x \quad & \mathsf{val}_{\mathsf{LP}}(x) = c^{\mathsf{T}} x \\ \text{s.t. }\ & Ax \leq b, x \in P. \end{aligned} \tag{3.4}$$

The bounds will make use of the Lagrangian dual of (3.4):

$$\mathsf{val}_{\mathsf{lag}} := \min_{u \geq 0} \max_{x \in P} \mathsf{val}_{\mathsf{lag}}(u, x) := \langle c, x \rangle - u^{\mathsf{T}}(Ax - b). \qquad \text{(Lagrangian Dual)}$$

By strong duality, assuming (3.4) is feasible, we have that $\mathsf{val}_{\mathsf{LP}} = \mathsf{val}_{\mathsf{lag}}$.

To use the integrality gap bounds to provide bounds on the size of the branch-and-bound tree, Dey, Dubey and Molinaro relate the size of the branch-and-bound tree to the number of solutions to some knapsack problem [DDM23, Theorem 3]. Their result is stated for the case where $P = [0, 1]^n$, but we note that it holds for any polytope $P \subseteq [0, 1]^n$. We state this slightly generalized result below.

Theorem 3.2.1. *Suppose $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. Let $P \subseteq [0, 1]^n$ be an integral polytope. Consider a binary integer program of the form*

$$\begin{aligned} &\max c^{\mathsf{T}} x \\ &\text{s.t. } Ax \leq b \\ &\quad\ x \in P \cap \{0, 1\}^n. \end{aligned} \tag{3.5}$$

Then, the branch-and-bound algorithm, equipped with the best-first node selection rule, produces a tree of size

$$2n \cdot |S| + 1, \tag{3.6}$$

where

$$S = \{x \in P \cap \{0, 1\}^n : (c^{\mathsf{T}} - u^{\mathsf{T}} A)(x^* - x) \leq \mathsf{IPGAP}\}.$$

Here we use u^* for an optimal solution to (Lagrangian Dual).

Proof. Let (x^*, u^*) be an optimal solution pair for (Lagrangian Dual). Observe that we have $\mathsf{val}_{\mathsf{lag}}(u^*, x^*) = \mathsf{val}_{\mathsf{LP}}$. Now consider the nodes of the branch-and-bound tree. For every node N in the tree let $F_N \subseteq P$ be the set of points that is feasible with respect to all the branching constraints that apply to this node. That is, for the root node R, we have $F_R = P$.

Because the best-first node selection rule is used, we know that for any node N in the tree that is branched on, we have:

$$c^{\mathsf{T}} x^* - \mathsf{IPGAP} \leq \max\{c^{\mathsf{T}} x : Ax \leq b, x \in F_N\}.$$

Now we can show:

$$\begin{aligned}
\mathsf{val}_{\mathsf{lag}}(u^*, x^*) - \mathsf{IPGAP} &= c^{\mathsf{T}} x^* - \mathsf{IPGAP} \\
&\leq \max\{c^{\mathsf{T}} x : Ax \leq b, x \in F_N\} \\
&= \min_{u \geq 0} \max_{x \in F_N} \mathsf{val}_{\mathsf{lag}}(u, x) \\
&\leq \max_{x \in F_N} \mathsf{val}_{\mathsf{lag}}(u^*, x) \\
&= \max_{x \in F_N \cap \mathbb{Z}^n} \mathsf{val}_{\mathsf{lag}}(u^*, x).
\end{aligned}$$

Here the first two equalities follow from the fact that the value of an LP is equal to that of its Lagrangian relaxation. The last equality follows because all vertices of F_N are integral, since F_N is an integral polytope. The fact that F_N is an integral polytope follows from the fact that it is the intersection of a polytope spanned by 0-1 vectors and hyperplanes of the form $x_i = 0$ and $x_i = 1$.

Rewriting the above, we have:

$$\begin{aligned}
\max_{x \in F_N \cap \mathbb{Z}^n} (c^{\mathsf{T}} - u^{\mathsf{T}} A)(x^* - x) &= \mathsf{val}_{\mathsf{lag}}(u^*, x^*) - \max_{x \in F_N \cap \mathbb{Z}^n} \mathsf{val}_{\mathsf{lag}}(u^*, x) \\
&\leq \mathsf{IPGAP}.
\end{aligned}$$

For each node N that is branched on, choose $x_N \in \arg\max_{x \in F_N \cap \mathbb{Z}^n} (c^{\mathsf{T}} - u^{\mathsf{T}} A)(x^* - x)$. From the above it is immediately clear that $x_N \in S$. Furthermore, a vector x can only be contained in $F_N \cap F_{N'}$ if N' is either an ancestor or a descendant of N. Since the depth of a branch-and-bound tree is at most n, this implies that a point x is contained in at most n sets F_N. Hence, we have:

$$|\text{nodes branched on in b\&b-tree}| \leq n|S|.$$

Every internal node of the tree is branched on. Since the branch-and-bound tree is binary, we have:

$$|\text{nodes in b\&b-tree}| \leq 2|\text{internal nodes in b\&b-tree}| + 1 \leq 2n|S| + 1.$$

□

As in [DDM23], we relate the set S to the solutions of a knapsack problem.

Lemma 3.2.2. *The set S defined in Theorem 3.2.1 has size at most $|K(A, c, u^*, \mathsf{IPGAP})|$, where*

$$K(A, c, u^*, \mathsf{IPGAP}) = \{x \in \{0,1\}^n : \sum_{i=1}^{n} x_i |(A^{\mathsf{T}} u^* - c)_i| \leq \mathsf{IPGAP}\}.$$

Proof. Let $S' = \{x \in \{0,1\} : (c^\mathsf{T} - u^\mathsf{T}A)(x^* - x) \leq \mathsf{IPGAP}\}$. Note that $S \subseteq S'$ and that equality holds whenever $P = [0,1]^n$. Consider the following function f:

$$f(x)_i = \begin{cases} 1 - x_i & \text{if } x_i^* = 1 \\ x_i & \text{otherwise} \end{cases}.$$

It is clear that f is injective. Now we will show that $f(S') \subseteq K(A, c, u^*, \mathsf{IPGAP})$, which will prove the lemma.

Consider an arbitrary $x \in S'$. By optimality of x^*, we have $x_i^* = 1$ for all i with $(c - A^\mathsf{T}u^*)_i > 0$ and $x_i^* = 0$ for all i with $(c - A^\mathsf{T}u^*)_i < 0$:

$$\begin{aligned}(c^\mathsf{T} - u^{*\mathsf{T}}A)(x^* - x) &= \sum_{i:x_i^*=1} (c - A^\mathsf{T}u^*)_i \mathbf{1}_{x_i \neq x_i^*} - \sum_{i:x_i^*=0} (c - A^\mathsf{T}u^*)_i \mathbf{1}_{x_i \neq x_i^*} \\ &= \sum_{i=1}^n |(c - A^\mathsf{T}u^*)_i| \mathbf{1}_{x_i \neq x_i^*} = \sum_{i=1}^n |(c - A^\mathsf{T}u^*)_i| f(x)_i.\end{aligned}$$

So $\sum_{i=1}^n |(c - A^\mathsf{T}u^*)_i| f(x_N) \leq \mathsf{IPGAP}$. □

The next step is to bound the size of the set $K(A, c, u^*, \mathsf{IPGAP})$. The bound given by [DDM23] only holds for uniformly distributed objective coefficients. We prove a bound that holds whenever the objective coefficients are continuously distributed with a bounded density. We make use of the following auxiliary lemma.

Lemma 3.2.3. *Let $A \in \mathbb{R}^{m \times n}$, $\gamma \geq 0$. Let $c_1, \ldots, c_n$ be independent random variables, each having probability density at most 1. Now:*

$$\Pr\left[\max_{u \in \mathbb{R}^m} \prod_{i=1}^n \left(1 + e^{-\gamma|A_i^\mathsf{T}u - c_i|}\right) \geq \delta^{-1} (3n\gamma)^{m+1} e^{3n/\gamma}\right] \leq \delta.$$

Proof. Let $N = \lceil 2\gamma \rceil$. Consider any $i \in [n]$. Choose $a_1^{(i)} < \ldots < a_{N-1}^{(i)}$, such that for $I_1^{(i)} = (-\infty, a_1^{(i)}), I_2^{(i)} = [a_1^{(i)}, a_2^{(i)}), \ldots, I_N^{(i)} = [a_{N-1}^{(i)}, \infty)$ we have:

$$\Pr[c_1 \in I_j^{(i)}] = \frac{1}{N} \quad \forall j \in [N].$$

Observe that by the bound on the probability density, we have $a_j^{(i)} - a_{j-1}^{(i)} \geq \frac{1}{N}$ for any j. Now define $\eta : \mathbb{R}^m \to [N]^n$ such that $(A_i^\mathsf{T}u) \in I_{\eta(u)_i}^{(i)}$ for all $i \in [n]$. Let $S = \{\eta(u) : u \in \mathbb{R}^n\}$. In the subspace $\{Ay : y \in \mathbb{R}^m\}$ of dimension at most m, each element of S corresponds to a cell bounded by hyperplanes of the form $A_i^\mathsf{T}u = a_j^{(i)}$

for some $i \in [n]$ and some $j \in [N]$. It is well-known that for any hyperplane arrangement in an m-dimensional space that consists of at most $n \cdot N$ hyperplanes, there are at most $\sum_{i=0}^{m} \binom{nN}{m} \leq (nN)^{m+1}$ cells. Hence, $|S| \leq (nN)^{m+1}$.

Take an arbitrary $y \in S$. We will now bound $\prod_{i=1}^{n} \left(1 + e^{-\gamma|A_i^\intercal u - c_i|}\right)$ simultaneously for all $u \in \eta^{-1}(y)$. Observe that:

$$\begin{aligned}
\mathbb{E}\left[\max_{u \in \eta^{-1}(y)} \left(e^{-\gamma|A_i^\intercal u - c_i|}\right)\right] &\leq \Pr[c_i \in I_{y_i}^{(i)}] + \sum_{j=y_i+1}^{N} e^{\gamma(a_{y_i}^{(i)} - a_{j-1}^{(i)})} \Pr[c_i \in I_j^{(i)}] \\
&\quad + \sum_{j=1}^{y_i-1} e^{\gamma(a_j^{(i)} - a_{y_i-1}^{(i)})} \Pr[c_i \in I_j^{(i)}] \\
&= \frac{1}{N} + \frac{1}{N} \sum_{j=y_i+1}^{N} e^{\gamma(a_{y_i}^{(i)} - a_{j-1}^{(i)})} + \frac{1}{N} \sum_{j=1}^{y_i-1} e^{\gamma(a_j^{(i)} - a_{y_i-1}^{(i)})} \\
&\leq \frac{1}{N} + \frac{2}{N} \sum_{j=0}^{\infty} e^{-j\gamma/N} = \frac{1}{N}\left(1 + \frac{2}{1 - e^{-\gamma/N}}\right) \leq \frac{5}{N}.
\end{aligned}$$

This implies that:

$$\begin{aligned}
\mathbb{E}\left[\max_{u \in \eta^{-1}(y)} \prod_{i=1}^{n} \left(1 + e^{-\gamma|A_i^\intercal u - c_i|}\right)\right] &\leq \prod_{i=1}^{n} \mathbb{E}\left[\max_{u \in \eta^{-1}(y)} \left(1 + e^{-\gamma|A_i^\intercal u - c_i|}\right)\right] \\
&\leq \left(1 + \frac{5}{N}\right)^n \leq \exp(5n/N) \leq \exp(3n/\gamma).
\end{aligned}$$

By the Markov bound we have:

$$\Pr[\max_{u \in \eta^{-1}(y)} \prod_{i=1}^{n} \left(1 + e^{-\gamma|A_i^\intercal u - c_i|}\right) \geq \frac{1}{\delta'} \exp(3n/\gamma)] \leq \delta'$$

By taking the union bound over all $y \in S$ we get:

$$\begin{aligned}
\Pr\left[\max_u \prod_{i=1}^{n} \left(1 + e^{-\gamma|A_i^\intercal u - c_i|}\right) \geq \frac{1}{\delta'} \exp(3n/\gamma)\right] &\leq |S|\delta' \leq (nN)^{m+1}\delta' \\
&\leq (3n\gamma)^{m+1}\delta'.
\end{aligned}$$

Setting $\delta' = \delta \left(\frac{1}{3n\gamma}\right)^{m+1}$ we obtain the lemma. $\square$

Now we will use the previous lemma to bound the size of the knapsack.

Lemma 3.2.4. *Let $A \in \mathbb{R}^{m\times n}, \gamma \geq 0$. Let $c_1, \ldots, c_n$ be independent random variables, each having probability density at most 1. Let $P \subseteq [0,1]^n$. Now:*

$$\Pr\left[\max_{u\in\mathbb{R}^m} |K(A,c,u,g)| \geq \delta^{-1}(3n^2)^{m+1}\exp(\sqrt{12ng}+4)\right] \leq \delta.$$

Proof. For any $\gamma \geq 0$, we first note that

$$\max_{u\in\mathbb{R}^m} |K(A,c,u,g)| \leq \sum_{x\in\{0,1\}^n} e^{\gamma(g-\sum_{i=1}^n x_i|(A^\mathsf{T}u-c)_i|)} = e^{\gamma g}\prod_{i=1}^n (1+e^{-\gamma|(A^\mathsf{T}u-c)_i|}). \tag{3.7}$$

To see this, note that each term $e^{\gamma(g-\sum_{i=1}^n x_i|(A^\mathsf{T}u-c)_i|)}$ with $x \in K$ contributes at least 1. By Lemma 3.2.3, we have:

$$\Pr\left[\max_{u\in\mathbb{R}^m} |K(A,c,u,g)| \geq \delta^{-1}\,(3n\gamma)^{m+1}\,e^{3n/\gamma+\gamma g}\right] \leq \delta. \tag{3.8}$$

Setting $\gamma = \min(n, \sqrt{\frac{3n}{g}})$, we conclude that

$$\Pr\left[\max_{u\in\mathbb{R}^m} |K(A,c,u,g)| \geq \delta^{-1}(3n^2)^{m+1}\exp(\sqrt{12ng}+4)\right] \leq \delta.$$

□

The previous bound allows us to prove the main result of this section.

Theorem 3.1.3. *Let $A \in \mathbb{R}^{m\times n}$, b be random variables. Let $c_1, \ldots, c_n$ be independent random variables, each having probability density at most 1. Let $P \subseteq [0,1]^n$ be an integral polytope. Then, for $g \geq 0$, $\delta \in (0,1)$, with probability at least*

$$1 - \Pr_{A,b,c}[\mathsf{IPGAP} \geq g] - \delta,$$

the branch-and-bound algorithm equipped with the best-first search rule applied to the ILP

$$\begin{aligned} &\max c^\mathsf{T}x \\ &\textit{s.t. } Ax \leq b \\ &\quad\; x \in P \cap \{0,1\}^n. \end{aligned}$$

produces a tree of size at most

$$\frac{n^{O(m)}\exp(\sqrt{12ng})}{\delta} \tag{3.2}$$

Proof. Suppose that IPGAP $\leq g$, and call this event E_1. Let $S = \{x \in \{0,1\}^n : \sum_{i=1}^n x_i |(A^\mathsf{T} u^* - c)_i| \leq g\}$. Let E_2 be the event that

$$|S| \leq \delta^{-1}(3n^2)^{m+1} \exp(\sqrt{12ng} + 4).$$

If E_2 occurs, by Theorem 3.2.1 the size of the branch-and-bound tree is at most $1 + 2n\delta^{-1}(3n^2)^{m+1} \exp(\sqrt{12ng} + 4) \leq n^{O(m)} \exp(\sqrt{12ng})$. By Lemma 3.2.4 and the union bound this occurs with probability at least $1 - \Pr[\neg E_1] - \delta$, proving the lemma. □

Bounds for the generalized assignment problem To highlight the generality of the above results, we will now show their applicability to ILPs for which the polytope P is not equal to $[0,1]^n$. We will consider the generalized assignment problem, as studied in [DF92]. Instances of this problem are of the form:

$$\begin{aligned} \max \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} & \\ \text{s.t.} \sum_{i=1}^n a_{ij} x_{ij} \leq \beta_i n & \quad \forall j \in [m] \\ \sum_{j=1}^m x_{ij} = 1 & \quad \forall i \in [n] \\ x_{ij} \in \{0,1\} & \quad \forall i \in [n], j \in [m]. \end{aligned}$$

In [DF92] it is shown that the integrality gap of such a problem with c_{ij} and a_{ij} independently and uniformly distributed on $[0,1]$ is $O(\log(n)^2/n)$ with probability at least $1/3$, when m is fixed and $\beta_i \geq \frac{2}{m(m+1)}$. Note that the above problem can be rewritten in the form of (3.3) where the number of variables is equal to nm in the above formulation. For convenience, we will keep indexing x by pairs from $[n] \times [m]$. The corresponding constraint matrix A' is defined as $A'_{(i,j),k} = a_{ij} \mathbf{1}_{j=k}$. In that case P is defined as $P = \{x : \sum_{j=1}^m x_{ij} = 1 \; \forall i \in [n]; x \geq 0\}$. Note that this polytope is indeed integral. By applying Theorem 3.1.3 with $g = O(\log(n)^2/n)$ we see that the branch-and-bound tree will have size at most $n^{O(1)}$ with probability at least $1/2$ for fixed m.

3.3 Proving bounds on the IPGAP

Now, let us prove the bounds on the integrality gap for the centered and packing IPs. We begin with by proving some properties of the linear programs and their duals that we study in this work in Section 3.3. Then in Section 3.3 we will prove bounds on the integrality gap for centered IPs. In Section 3.3 we will prove bounds on the integrality gap for packing IPs.

Linear programs and their duals

We will examine the integrality gap with respect to the LP relaxation. Recall that it is defined as:

$$\begin{aligned} \mathrm{val}_{\mathsf{LP}} := \max_x \quad & \mathrm{val}_{\mathsf{LP}}(x) = c^{\mathsf{T}} x \\ & \text{s.t. } Ax \le b, x \in [0,1]^n, \end{aligned} \tag{Primal LP}$$

We can express the dual LP in the following convenient form:

$$\begin{aligned} \mathrm{val}^*_{\mathsf{LP}} := \min_u \quad & \mathrm{val}^*_{\mathsf{LP}}(u) = b^{\mathsf{T}} u + \left\| \left(c - A^{\mathsf{T}} u\right)^+ \right\|_1 \\ & \text{s.t. } u \ge 0. \end{aligned} \tag{Dual LP}$$

By strong duality, assuming (Primal LP) is bounded and feasible, we have that $\mathrm{val}_{\mathsf{LP}} = \mathrm{val}^*_{\mathsf{LP}}$.

For any primal solution x and dual solution u to the above pair of programs, we will make heavy use of the standard formula for the primal-dual gap:

$$\begin{aligned} \mathrm{val}^*_{\mathsf{LP}}(u) - \mathrm{val}_{\mathsf{LP}}(x) &= b^{\mathsf{T}} u + \left\| \left(c - A^{\mathsf{T}} u\right)^+ \right\|_1 - c^{\mathsf{T}} x \\ &= (b - Ax)^{\mathsf{T}} u + \left(\langle x, (A^{\mathsf{T}} u - c)^+ \rangle + \langle \mathbf{1}_n - x, (c - A^{\mathsf{T}} u)^+ \rangle\right). \end{aligned} \tag{Gap Formula}$$

In the sequel, we will let x^* denote the optimal solution to Primal LP and u^* denote the optimal solution to Dual LP. For all the LP distributions we work with, the objective c is continuously distributed (either Gaussian or exponentially distributed), from which it can be verified that conditioned on the feasibility of Primal LP (which depends only on A and b) both x^* and u^* are uniquely defined almost surely. Moreover, if $i \in [n]$, we shall use A_i to refer to the ith column of A and extend this definition to other matrices as well.

Once the optimal solution is found for Primal LP, one can round its fractional coordinates to an integral vector. While the rounded vector may not be a feasible solution, we shall use the fact that, as long as the A_i are sufficiently bounded, it cannot be very far from a feasible solution.

Lemma 3.3.1. *There exists $x' \in \{0,1\}^n$, such that,*

$$\|A(x^* - x')\| \leq \sqrt{m} \cdot \max_{i \in [n]} \|A_i\|.$$

Proof. Let Y be the random variable in $\{0,1\}^n$ with independent components such that $\mathbb{E}(Y) = x^*$. Note that this implies that $\text{Var}(Y_i) \leq 1/4$ for all i and $\text{Var}(Y_i) = 0$ for $x_i^* \in \{0,1\}$. Since x^* is a unique optimal solution it has at most m fractional components. So:

$$\begin{aligned}\mathbb{E}\left[\|A(y-Y)\|_2\right]^2 \leq \mathbb{E}\left[\|A(y-Y)\|_2^2\right] &= \sum_{i=1}^n \|A_i\|_2^2 \,\text{Var}(Y_i) \\ &\leq \sum_{i=1}^n C^2 \,\text{Var}(Y_i) \leq \frac{C^2 \cdot m}{4}.\end{aligned}$$

So $\mathbb{E}(\|A(y-Y)\|_2) \leq C\sqrt{m}/2$, which directly implies the existence of a value $y' \in \{0,1\}^n$ with $\|A(y-y')\|_2 \leq C\sqrt{m}/2$. □

For the optimal solution x^*, define,

$$N_0 := \{i \in [n] | x_i^* = 0\}, \text{and } N_1 := \{i \in [n] | x_i^* = 1\}. \tag{3.9}$$

Let W be the matrix with columns $W_i = \begin{bmatrix} c_i & A_i \end{bmatrix}^\mathsf{T}$. The distribution of the columns of W with indices in N_0 plays an important role in the proofs of Theorems 3.1.1 and 3.1.2. The following lemma essentially says that conditioning on the set N_0 and on the values of the non-0-columns preserves the mutual independence of the 0-columns. The conditional distribution of the the 0-columns is also identified.

Lemma 3.3.2. *Let $N \subseteq [n]$. Conditional on $N_0 = N$ and on the values of sub-matrix $W_{[n]\setminus N}$, x^* and u^* are almost surely well defined. Moreover, if $i \in N$, then W_i is independent from $W_{N\setminus\{i\}}$ and the conditional law $W_i \mid i \in N$ is the same as $W_i \mid u^{*\mathsf{T}} A_i - c_i > 0$.*

Proof. Knowing N, we solve the following program to obtain its primal and dual optimal feasible solutions $\bar{x}$ and $\bar{u}$.

$$\begin{aligned} &\max c^\mathsf{T} x \\ &\text{s.t.} \sum_{i \in [n]\setminus N} x_i a_{ji} \leq b_j && \forall j \in [m] \\ &\quad x_i = 0 && \forall i \in N \\ &\quad x \in [0,1]^n. \end{aligned}$$

This does not require knowledge of $W_{N_0'}$, and the optimal feasible primal and dual solutions are unique almost surely.

If $N_0 = N$, then $\bar{x} = x^*$ and $\bar{u} = u^*$. Since these solutions satisfy complementary slackness, this is equivalent to the following system of equations.

$$(1-\bar{x}_i)(c_i - \sum_{j=1}^{m} \bar{u}_i a_{ji})^+ = 0, \qquad \forall i \in [n]. \tag{3.10}$$

$$\bar{x}_i(\sum_{j=1}^{m} \bar{u}_i a_{ji} - c_i)^+ = 0, \qquad \forall i \in [n]. \tag{3.11}$$

$$\bar{u}_j(b - A\bar{x})_j = 0, \qquad \forall j \in [m]. \tag{3.12}$$

Note that for $i \in N_0$, Eqs. (3.11) and (3.12) are trivially satisfied. By definition, the distribution of $W_i := (c_i, a_{1i}, \ldots, a_{mi})$ conditioned on Eq. (3.10) for $\bar{x}_i = 0$ has the same law as $Y^{\bar{u}}$. Note that each of these conditions depends on only one $i \in N_0'$, so all columns of $W_{N_0'}$ are independent.

We conditioned only on $N_0 = N_0'$, which has non-zero probability, and we have shown that for every possible realization of $W_{[n]\setminus N_0'}$, the columns of $W_{N_0'}$ are independently distributed as Y^{u^*}, which proves the lemma. □

The gap bound for centered IPs

In this subsection we will prove Theorem 3.1.1. In the setting of Theorem 3.1.1, the objective $c \in \mathbb{R}^m$ has independent standard Gaussian entries, and the $m \times n$ constraint matrix A has independent columns which are distributed as either one of the following two possibilities:

- (LI) Isotropic logconcave distributions with support bounded by $O(\sqrt{\ln n} + \sqrt{m})$.
- (DSU) Vectors with independent entries, uniform on a discrete symmetric interval of size $k \geq 3$.

Here we list some of the moments of (DSU) for reference.

Proposition 3.3.3 (Discrete Symmetric Moments). *For $k \geq 1$, let U be uniformly distributed on $\{0, \pm 1/k, \ldots, \pm 1\}$. Then, $\mathbb{E}[U^2] = \frac{k+1}{3k} \geq 1/3$, $\mathbb{E}[U^4] = \frac{(k+1)(3k^2+3k-1)}{15k^3}$ and $\mathbb{E}[U^4]/\mathbb{E}[U^2]^2 = \frac{9(3k^2+3k-1)}{15(k+1)k} \leq 2$.*

To simplify the notation in the discrete case, we divide the constraint matrix A and the right hand side b by k (which clearly does not restrict generality). Thus, in the discrete case (DSU), we will assume that entries of A are uniformly distributed in $\{0, \pm 1/k, \dots, \pm 1\}$ and that the right hand side $b \in \mathbb{Z}^m/k$ satisfies $\|b^-\|_2 \le O(n)$. In this way, the discrete case is usefully viewed as a discrete approximation of the continuous setting where the entries of A are uniformly distributed in $[-1, 1]$ (note that the covariance matrix of each column here is $I_m/3$, and thus essentially isotropic).

With the above setup, our goal is to show that $\mathsf{IPGAP} = O(\frac{\operatorname{poly}(m)(\ln n)^2}{n})$ with probability $1 - n^{-\operatorname{poly}(m)}$.

Properties of the optimal solutions

To obtain the gap bound, we will need to show $|N_0| = \Omega(n)$ and that u^*, the optimal dual solution, has small norm. This is given by the following lemma, which is a technical adaptation of [BDHT22, Lemma 8].

Lemma 3.3.4. *For $A \in \mathbb{R}^{m\times n}$, $n \ge 10^5 m$, distributed as (LI) or (DSU), $c \sim \mathcal{N}(0, I_m)$, $\|b^-\| \le \frac{n}{12\sqrt{2}}$ with probability at least $1 - e^{-\Omega(n)}$, we have $\|u^*\| \le 32$ and $|N_0| \ge \frac{n}{10^5}$.*

To prove this, we need two key lemmas. The first lemma will provide a good approximation for the value of any dual solution.

Lemma 3.3.5. *Let $W^\mathsf{T} := (c, A^\mathsf{T})$ where $c \sim \mathcal{N}(0, I_n)$ and $A \in \mathbb{R}^{m\times n}$ is distributed as (LI) or (DSU). Then, for $n = \Omega(m)$, we have that*

$$\Pr\left[\exists v \in \mathbb{S}^m : \|(v^\mathsf{T} W)^+\|_1 \notin \left[\frac{n}{12}, \frac{3n}{4}\right]\right] \le e^{-\Omega(n)}.$$

Proof. Fix $v \in \mathbb{S}^m$ and consider $\Pr\left[\|(v^\mathsf{T} W)^+\|_1 \notin [\frac{n}{8}, \frac{5n}{8}]\right]$. Let $i \in [n]$, we first claim

$$\frac{1}{6} \le \mathbb{E}\left[(v^\mathsf{T} W)_i^+\right] \le \frac{1}{2}. \tag{3.13}$$

To see the right inequality, by Proposition 2.3.1, $\mathbb{E}\left[(v^\mathsf{T} W)_i^+\right] = \frac{1}{2}\mathbb{E}\left[|v^\mathsf{T} W|_i^+\right]$. Now observe that every entry has variance at most 1, so with Jensen's inequality

$$\mathbb{E}\left[(v^\mathsf{T} W)_i^+\right] = \frac{1}{2}\mathbb{E}\left[|v^\mathsf{T} W|_i^+\right] \le \frac{1}{2}\sqrt{\operatorname{Var}(|v^\mathsf{T} W|_i^+)} \le \frac{1}{2}.$$

For the left inequality, if the columns of W are isotropic log-concave (recall that the standard Gaussian is also log-concave) Lemma 2.3.10 to get,

$$\frac{1}{6} \le \frac{1}{2\sqrt{e}} \le \frac{1}{2}\mathbb{E}\left[|v^\mathsf{T} W|_i^+\right] = \mathbb{E}\left[(v^\mathsf{T} W)_i^+\right].$$

If the columns of W are discrete, by Proposition 3.3.3 every entry satisfies, $\mathrm{Var}\left(W_{ij}\right) \geq \frac{1}{3}$. Hence, $\mathrm{Var}\left((v^\mathsf{T} W)_i\right) \geq \frac{1}{3}$. Moreover, Proposition 3.3.3 also implies, $\mathbb{E}[W_{ji}^4] \leq 3\,\mathbb{E}[W_{ji}^2]^2$. So, by the Khintchine inequality in Lemma 2.3.13,

$$\frac{1}{6} \leq \frac{\sqrt{\mathrm{Var}\left((v^\mathsf{T} W)_i\right)}}{2\sqrt{3}} \leq \frac{1}{2}\,\mathbb{E}\left[|v^\mathsf{T} W|_i^+\right] = \mathbb{E}\left[(v^\mathsf{T} W)_i^+\right].$$

Now, having established (3.13), we can bound $\Pr\left[\|(v^\mathsf{T} W)^+\|_1 \notin [\frac{n}{8}, \frac{5n}{8}]\right]$. In the log-concave case, Lemma 2.3.12 immediately gives,

$$\Pr\left[\|(v^\mathsf{T} W)^+\|_1 \notin [\frac{n}{8}, \frac{5n}{8}]\right] \leq e^{-\Omega(n)}.$$

In the discrete case, by Lemma 2.3.4, every entry of W is 1-sub-Gaussian, and Lemma 2.3.5 shows that $(v^\mathsf{T} W)_i^+ - \mathbb{E}\left[(v^\mathsf{T} W)_i^+\right]$ is $\sqrt{2}$-sub-Gaussian. Hence, by summing the coordinates we see that $\|(v^\mathsf{T} W)^+\|_1 - \mathbb{E}\left[\|(v^\mathsf{T} W)^+\|_1\right]$ is $\sqrt{2n}$-sub-Gaussian. Applying (2.4), we can thus conclude a corresponding probability bound, as in the previous display.

We now turn to consider the entire sphere. Fix ε to be a small constant and let $N_\varepsilon \subseteq \mathbb{S}^{m-1}$ be an ε-net. It is standard to show that one may take $|N_\varepsilon| \leq \left(\frac{3}{\varepsilon}\right)^m$. Hence, by applying a union bound,

$$\Pr\left(\exists v \in N_\varepsilon : \|(v^\mathsf{T} W)^+\|_1 \notin \left[\frac{n}{8}, \frac{5n}{8}\right]\right) \leq \left(\frac{3}{\varepsilon}\right)^m e^{-\Omega(n)} \leq e^{-\Omega(n)},$$

where the last inequality holds when $n = \Omega(m)$.

Let us denote by E the event considered above and for $u \in \mathbb{S}^{m-1}$ let $\tilde{u} \in N_\varepsilon$, with $\|u - \tilde{u}\|_2 \leq \varepsilon$. Under E, we have,

$$\begin{aligned}\max_{u \in \mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 &\leq \min_{v \in N_\varepsilon} \|(v^\mathsf{T} W)^+\|_1 + \|((u - \tilde{u})^\mathsf{T} W)^+\|_1 \\ &\leq \frac{5}{8}n + \varepsilon \max_{u \in \mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1,\end{aligned}$$

which is equivalent to,

$$\max_{u \in \mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 \leq \frac{5}{8(1-\varepsilon)}n.$$

On the other hand,

$$\begin{aligned}\min_{u\in\mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 &\geq \min_{v\in N_\varepsilon} \|(v^\mathsf{T} W)^+\|_1 - \|((u-\tilde{u})^\mathsf{T} W)^-\|_1 \\ &\geq \frac{n}{8} - \varepsilon \max_{u\in\mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 \\ &\geq \frac{n}{8} - \varepsilon \frac{5}{8(1-\varepsilon)} n.\end{aligned}$$

Choose now $\varepsilon = \frac{5}{212}$ to conclude,

$$\frac{n}{12} \leq \min_{u\in\mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 \leq \max_{u\in\mathbb{S}^{m-1}} \|(u^\mathsf{T} W)^+\|_1 \leq \frac{3n}{4}.$$

□

The second lemma will imply that any LP solution with large support must have small objective value.

Lemma 3.3.6. *Let $c \sim \mathcal{N}(0, I_n)$. Then, for every $\alpha \in [0, 2\sqrt{\ln(2)}]$,*

$$\Pr\left[\max_{x\in\{0,1\}^n,\ \|x\|_1 \geq \beta n} c^\mathsf{T} x \geq \alpha n\right] \leq e^{\frac{-\alpha^2 n}{2}},$$

where $\beta \in [1/2, 1]$ is such that $H(\beta) \leq \frac{\alpha^2}{4}$, where $H(p) = -p\ln p - (1-p)\ln(1-p)$, $p \in [0,1]$, is base e entropy.

Proof. For any $x \in \{0,1\}^n$, $c^\mathsf{T} x \sim \mathcal{N}(0, \|x\|_2^2)$ and thus, by (2.4),

$$\Pr\left(c^\mathsf{T} x \geq \alpha n\right) \leq e^{-\frac{\alpha^2 n^2}{2\|x\|_2^2}} \leq e^{-\frac{\alpha^2 n}{2}}.$$

We now apply a union bound,

$$\begin{aligned}\Pr\left(\max_{x\in\{0,1\}^n,\ \|x\|_1 \geq \beta n} c^\mathsf{T} x \geq \alpha n\right) &\leq |\{x \in \{0,1\}^n,\ \|x\|_1 \geq \beta n\}|\, e^{-\frac{\alpha^2 n}{2}} \\ &\leq e^{H(\beta)n} e^{-\frac{\alpha^2 n}{2}} \leq e^{-\frac{\alpha^2 n}{4}}.\end{aligned}$$

□

We now have the ingredients to prove the main lemma.

Proof of Lemma 3.3.4. For the proof, we will consider the extended matrix $W^{\mathsf{T}} := (c, A^{\mathsf{T}})$. We begin by showing that, for the optimal solution, $c^{\mathsf{T}}x^*$ is large. Let $u \geq 0$ be any dual solution. Then, under the complement of the event defined in Lemma 3.3.5 for W, using (Dual LP),

$$\begin{aligned}\mathrm{val}^*_{\mathsf{LP}}(u) &= b^{\mathsf{T}}u + \|(c - A^{\mathsf{T}}u)^+\|_1 \geq -\|b^-\|\|u\| + \|((1,-u)^{\mathsf{T}}W)^+\|_1 \\ &\geq -\|b^-\|\|u\| + \sqrt{1+\|u\|^2}\frac{n}{12} \geq \frac{n}{12}\left(-\frac{\|u\|}{\sqrt{2}} + \sqrt{1+\|u\|^2}\right) \geq \frac{n}{12\sqrt{2}}.\end{aligned} \tag{3.14}$$

The second inequality is the lower bound in Lemma 3.3.5 and the last inequality follows since the function $\sqrt{1+t^2} - \frac{t}{\sqrt{2}}$ is minimized at $t = 1$. A lower bound on $c^{\mathsf{T}}x^*$ follows by noting,

$$c^{\mathsf{T}}x^* = \mathrm{val}_{\mathsf{LP}}(x^*) = \mathrm{val}^*{}_{\mathsf{LP}}(u^*).$$

We now prove that $\|u^*\|$ cannot be too large. Again, under the complement of the event in Lemma 3.3.5, but using the upper bound this time,

$$\begin{aligned}\frac{3n}{4} &\geq \|((1,0)^{\mathsf{T}}W)^+\|_1 = \|c^+\|_1 = \mathrm{val}^*{}_{\mathsf{LP}}(0) \geq \mathrm{val}^*{}_{\mathsf{LP}}(u^*) \\ &\geq \frac{n}{12}\left(-\frac{\|u^*\|}{\sqrt{2}} + \sqrt{1+\|u^*\|^2}\right) \geq \frac{n}{12}\left(1 - \frac{1}{\sqrt{2}}\right)\|u^*\|,\end{aligned}$$

where in the third inequality we have applied (3.14) to v^*. Thus, rearranging we get $\|u^*\|_2 \leq \frac{9\sqrt{2}}{\sqrt{2}-1} \leq 32$. Finally, we show that the optimal solution has many 0 coordinates. Since x^* has at most m fractional coordinates,

$$|\{i \in [n] \mid x^*_i = 0\}| \geq n - m - |\{i \in [n] \mid x^*_i = 1\}|.$$

Since, by assumption we know that $n \geq 10^5 m$, to finish the proof it will suffice to show that we have $|\{i \in [n] \mid x^*_i = 1\}| \leq \left(1 - \frac{2}{10^5}\right)n$. Define $\bar{x}$ by,

$$\bar{x}_i := \begin{cases} x^*_i & \text{if } x^*_i \in \{0,1\} \\ 1 & \text{if } x^*_i \notin \{0,1\} \text{ and } c_i \geq 0 \\ 0 & \text{if } x^*_i \notin \{0,1\} \text{ and } c_i < 0 \end{cases}.$$

Letting $\alpha = \frac{1}{12\sqrt{2}}$ and $\beta = 1 - \frac{2}{10^5}$, a calculation reveals that $H(\beta) \leq \frac{1}{4\alpha^2}$. By (3.14), we have

$$c^{\mathsf{T}}\bar{x} \geq c^{\mathsf{T}}x^* \geq \alpha n,$$

and by conditioning on the complement of the event in Lemma 3.3.6 with β and α as above,

$$\beta n \geq |\{i \in [n] \mid \bar{x}_i = 1\}| \geq |\{i \in [n] \mid x_i^* = 1\}| .$$

The proof concludes by applying the union bound to the events in Lemmas 3.3.6 and 3.3.5. □

Conditional distribution of 0*-columns of IP*

Let B be a random variable with the same distribution as the columns of A. By Lemma 4.4.1, B satisfies the anti-concentration inequality (AC) with constant $\kappa \leq 1$. Define $C := \frac{\sqrt{150}\|u^*\|}{\sqrt{\kappa}}$. We first show that the anti-concentration property is unaffected if we condition B on a strip of width $2C$.

Lemma 3.3.7. *Let B' have the law of B, conditioned on $|u^{*\mathsf{T}}B| \leq C$. Then,*

1. $\Pr[|u^{*\mathsf{T}}B| \leq C] \geq 1 - \frac{\kappa}{150}$.
2. *We have* $\frac{1}{10}I_m \preccurlyeq \operatorname{Cov}(B') \preccurlyeq 2I_m$.
3. *If B is (DSU), then B' is symmetric and anti-concentrated with parameter $\kappa/2$.*
4. *If B is (LI), B' is logconcave.*

Proof. Let $E = \{a \in \mathbb{R}^m : |u^{*\mathsf{T}}a| \leq C\}$. From Chebyshev's inequality, and since distributions we consider satisfy $\operatorname{Cov}(B) \preceq I_m$,

$$\Pr(B \in E) \geq 1 - \frac{\mathbb{E}\left[(u^{*\mathsf{T}}B)^2\right]}{C^2} \geq 1 - \frac{\|u^*\|^2}{C^2} \geq 1 - \frac{\kappa}{150},$$

which is the first claim.

If $w \in \mathbb{R}^m$, then

$$\mathbb{E}\left[|w^{\mathsf{T}}B'|^2\right] = \frac{\mathbb{E}\left[|w^{\mathsf{T}}B|^2\mathbf{1}_E\right]}{\Pr(B \in E)} \leq 2\mathbb{E}\left[|w^{\mathsf{T}}B'|^2\right] \leq 2\|w\|^2.$$

In the (DSU) case, to lower bound $\operatorname{Cov}(B')$, we note that by Proposition 3.3.3, $\mathbb{E}[W_{ji}^4] \leq 3\,\mathbb{E}[W_{ji}^2]^2$. As a consequence, Lemma 2.3.13 implies $\sqrt{\mathbb{E}[|w^{\mathsf{T}}B|^4]} \leq \sqrt{3}\,\mathbb{E}[|w^{\mathsf{T}}B|^2]$. By the Cauchy-Schwarz inequality,

$$\begin{aligned}\mathbb{E}[|w^{\mathsf{T}}B \cdot \mathbf{1}_{B\notin E}|^2] &\leq \sqrt{\mathbb{E}[|w^{\mathsf{T}}B|^4] \cdot \mathbb{E}[\mathbf{1}_{B\notin E}^4]} \\ &\leq \sqrt{3\Pr[B \notin E]}\,\mathbb{E}[|w^{\mathsf{T}}B|^2] \leq \sqrt{\frac{\kappa}{50}}\|w\|^2.\end{aligned}$$

By Proposition 3.3.3 we have $\mathbb{E}[|w^\mathsf{T} B|^2] \geq \frac{1}{3}\|w\|^2$. So:

$$\begin{aligned}\mathbb{E}[|w^\mathsf{T} B'|^2] &= \frac{\mathbb{E}[|w^\mathsf{T} B|^2] - \mathbb{E}[|w^\mathsf{T} B|^2 \cdot \mathbf{1}_{B\notin E}]}{\Pr[B \in E]} \\ &\geq \frac{\frac{1}{3}\|w\|^2 - \sqrt{\frac{\kappa}{50}}\|w\|^2 \cdot}{1-\frac{\kappa}{50}} \geq \frac{1}{10}\|w\|^2,\end{aligned}$$

proving the second claim for the (DSU) case.

In the (LI) case, $\langle w, B\rangle$ is logconcave. By Lemma 2.3.7, $f_{\langle B,v\rangle} \leq \frac{1}{\sqrt{\operatorname{Var}\langle B,v\rangle}} = 1$. So, $f_{\langle v,B'\rangle} \leq \frac{1}{1-1/150} f_{\langle B,v\rangle} = \frac{150}{149}$. Now, Lemma 2.3.2 implies that $\operatorname{Var}(\langle B'', v\rangle) \geq \frac{1}{13}$.

Now, if B is (DSU), it is symmetric, and because conditioning on a symmetric set preserves symmetry, so is B'. Consequently, in this case, $\mathbb{E}[B'] = 0$. Now set $\sigma' := \sqrt{\|\operatorname{Cov}(B')\|_{\text{op}}} \leq \sqrt{2}$. For the third claim, let $I(\theta) = \{a \in \mathbb{R}^m : d(\theta^\mathsf{T} a, \mathbb{Z}) \geq \frac{\kappa}{2}\min(1, \sigma'\|\theta\|_\infty)\}$. Choose an arbitrary $\nu \in \mathbb{R}^n$. By the symmetry of B' we have:

$$\begin{aligned}\Pr\left[B' \in I(\theta) \mid \langle B, \nu\rangle \leq 0\right] &= \frac{\Pr\left[B \in I(\theta) \cap E \mid \langle B, \nu\rangle \leq 0\right]}{\Pr[B \in E \mid \langle B, \nu\rangle \leq 0]} \\ &\geq \Pr\left[B \in I(\theta) \mid \langle B, \nu\rangle \leq 0\right] - 2\Pr\left[B \notin E\right] \geq \frac{\kappa}{2}.\end{aligned}$$

The last inequality follows from the anti-concentration equality (AC),

$$\begin{aligned}\Pr\left[B \in I(\theta) \mid \langle B, \nu\rangle \leq 0\right] &\geq \Pr\left[d(\theta^\mathsf{T} B, \mathbb{Z}) \geq \frac{\kappa}{2}\min\left(1, \sigma'\|\theta\|_\infty\right) \mid \langle B, \nu\rangle \leq 0\right] \\ &\geq \kappa.\end{aligned}$$

This shows anti-concentration when B is (DSU).

If B is (LI), then so is B' because it is a restriction to a convex set, proving the last claim.

□

In the proof of Theorem 3.1.1, we work with the columns of A that have negative reduced cost. We show that we can convert their distribution into the distribution of B', by using rejection sampling.

Lemma 3.3.8. *Let $B = A_i$ and let B', c' have* $\operatorname{Law}(B', c_i') = \operatorname{Law}(B, c_i \mid u^{*\mathsf{T}} B - c_i \geq 0)$. *When $\delta \leq C$, there exist a rejection sampling procedure ψ such that:*

- $\operatorname{Law}(B' \mid \psi(B', c_i') = \text{accept}) = \operatorname{Law}(B \mid |u^{*\mathsf{T}} A_i| \leq C)$.
- $\Pr[\psi(B', c_i') = \text{accept}] \geq \delta \exp\left(-\frac{10^4}{\kappa}\right)$.

Proof. Let c'' and B'' be independent random variables with $c'' \sim \mathrm{Unif}(0,\delta)$ and let $\mathrm{Law}(B'') = \mathrm{Law}(B \mid |u^{*\mathsf{T}}B| \leq C)$. Now we will apply Lemma 2.3.15 to transform (B', c') into (B'', c'') using rejection sampling. Observe that for any $\bar{B} \in \mathbb{R}^m$ with $u^{*\mathsf{T}}\bar{B} - \bar{c} \in [0,\delta]$:

$$\frac{f_{(B'',c'')}(\bar{B},\bar{c})}{f_{(B',c'_i)}(\bar{B},\bar{c})} = \frac{f_B(\bar{B})/\Pr[|u^{*\mathsf{T}}B| \leq C]/\delta}{f_B(\bar{B})f_{c_i}(\bar{c})/\Pr[u^{*\mathsf{T}}B - c \geq 0]} = \frac{\Pr[u^{*\mathsf{T}}B - c \geq 0]}{\delta f_{c_i}(\bar{c})\Pr[|u^{*\mathsf{T}}B| \leq C]}$$
$$\leq \frac{1 - \frac{\kappa}{150}}{\delta \exp(-\frac{1}{2}(C+\delta)^2)/\sqrt{2\pi}} \leq \frac{3\exp(2C^2)}{\delta}.$$

For the last inequality we assumed that $\delta \leq C$. By Lemma 2.3.15 we, using rejection sampling we can turn (B', c') into (B'', c'') with success probability $\frac{\delta}{3\exp(\frac{1}{2}(C+\delta)^2)}$. Since $C = \frac{\sqrt{2}\|u^*\|}{\sqrt{\kappa}}$, by Lemma 3.3.4 , the success probability is at least $\frac{\delta}{3\exp(2C^2)} \leq \delta\exp\left(-\frac{10^4}{\kappa}\right)$.

□

When the columns of A are continuously distributed, we have to be more careful, because the distribution that we obtain from rejection sampling is not necessarily symmetric. As a result, B will not necessarily be mean-zero. We apply another step of rejection sampling to handle this case.

Lemma 3.3.9. *Let B' denote the random variable $B \mid |u^{*\mathsf{T}}B| \leq C$. If B' is logconcave, then there exists a rejection sampling procedure ψ, such that $\Pr[\psi(B') = \text{accept}] = \Omega(1)$, and such that the random variable B'' which is defined to have $\mathrm{Law}(B'') = \mathrm{Law}(B' | \psi(B') = \text{accept})$ satisfies:*

1. $\mathbb{E}[B''] = 0$.
2. $\mathrm{Cov}(B'') \succcurlyeq \frac{1}{768} I_m$.
3. *The law of B'' is an approximately symmetric distribution, in the sense of Definition 4.3.1.*
4. *B'' satisfies (AC) with an $\Omega(1)$ constant.*

Proof. Let $\mu' := \mathbb{E}[B']$. By Hölder's inequality, we have any unit vector $v \in \mathbb{R}^m$,

$$\mathbb{E}\left[\langle B', v\rangle\right] = \frac{\mathbb{E}[\langle B, v\rangle \cdot \mathbf{1}_{|u^{*\mathsf{T}}B| \leq C}]}{\Pr[|u^{*\mathsf{T}}B| \leq C]} = -\frac{\mathbb{E}[\langle B, v\rangle \cdot \mathbf{1}_{|u^{*\mathsf{T}}B| > C}]}{\Pr[|u^{*\mathsf{T}}B| \leq C]}$$
$$\leq \frac{\sqrt{\mathbb{E}[\langle B, v\rangle^2]}\sqrt{\Pr[|\langle B, u^*\rangle| > C]}}{\Pr[|u^{*\mathsf{T}}B| \leq C]} \leq \frac{\sqrt{\frac{1}{150}}}{1 - 1/150} \leq \frac{1}{12}$$

where the second equality follows since B is isotropic and from

$$\mathbb{E}[\langle B, v\rangle \cdot \mathbf{1}_{|u^{*\mathsf{T}}B|\le C}] + \mathbb{E}[\langle B, v\rangle \cdot \mathbf{1}_{|u^{*\mathsf{T}}B|>C}] = \mathbb{E}[\langle B, v\rangle] = 0.$$

Recall $C \ge \sqrt{150}\|u^*\|$. Hence, the concentration bound in [LV07, Lemma 5.7], coupled with the fact that B is isotropic, gives,

$$\frac{\sqrt{\mathbb{E}[\langle B, v\rangle^2]}\sqrt{\Pr[|\langle B, u^*\rangle| > C]}}{\Pr[|u^{*\mathsf{T}}B| \le C]} \le \frac{\sqrt{\Pr\left[|\langle B, \frac{u^*}{\|u^*\|}\rangle| > \sqrt{150}\right]}}{\Pr[|u^{*\mathsf{T}}B| \le C]} \le 2e^{-5} \le \frac{1}{12}.$$

So,

$$\|\mu'\| = \sup_{v\in\mathbb{R}^m, \|v\|=1} \mathbb{E}\left[\langle B', b\rangle\right] \le \frac{1}{12}.$$

Define a convex subset of $\mathbb{R}^m$ by

$$M := \left\{\frac{\mathbb{E}[f(B')B']}{1/4} : f \in L_\infty(\mathbb{R}^m, \mathbb{R}),\right.$$
$$\left. 0 \le f(x) \le 1 \text{ for every } x \in \mathbb{R}^m \text{ and } \mathbb{E}[f(B')] = \frac{1}{4}\right\},$$

and consider an arbitrary halfspace H that contains $-\mu'$. By Lemma 2.3.8 we have,

$$\Pr[B' \in H] \ge \Pr[B \in H] - \Pr[|u^{*\mathsf{T}}B| > C] \ge \frac{1}{e} - \frac{1}{12} - \frac{1}{150} = \frac{1}{4}.$$

Therefore, there exists $S \subseteq H$ with $\Pr[B' \in S] = \frac{1}{4}$. Then, $\mathbb{E}[\mathbf{1}_S(B')] = \frac{1}{4}$ and, $\mathbb{E}[B' \mid B' \in S] = \mathbb{E}[\frac{\mathbf{1}_S(B')B'}{1/4}] \in M$, which implies that $M \cap H \ne \emptyset$. Because this holds for any H with $-\mu' \in H$, by the convexity of M we have $-\mu' \in M$. Indeed, suppose not, then there is a hyperplane passing at $-\mu'$ which separates it from M, which cannot happen. We conclude that there exists $f : \mathbb{R}^m \to \mathbb{R}$, with $\|f\|_\infty \le 1$ and $\mathbb{E}[f(B')] = \frac{1}{4}$, such that $\frac{\mathbb{E}[f(B')B']}{1/4} = -\mu'$. Let $g(x) = \frac{f(x)+4}{5}$.

Let B'' be a random variable with $f_{B''}(x) \propto g(x)f_{B'}(x)$. Note that:

$$f_{B''}(x)/f_{B'}(x) = g(x)/\mathbb{E}_{B'}[g(B')] \le 1/\mathbb{E}_{B'}[g(B')] \le 2.$$

Now, by Lemma 2.3.15 there exists a rejection sampling procedure ψ for which $\Pr[\psi(B') = \text{accept}] = \frac{1}{2}$ and $\text{Law}(B' \mid \psi(B') = \text{accept}) = B''$. Now we will show that B'' satisfies the required properties.

Firstly, we have $\mathbb{E}[B''] = \frac{\mathbb{E}[g(B')B']}{\mathbb{E}[g(B')]} = \frac{\mathbb{E}[f(B')B'] + 4\,\mathbb{E}[B']}{5\,\mathbb{E}[g(B')]} = \frac{-4\mu' + 4\mu'}{5\,\mathbb{E}[g(B')]} = 0$, proving the first stated property. Secondly, for every halfspace H containing the origin, by the Grünbaum inequality, $\Pr[B \in H] \ge \frac{1}{e}$. So $\Pr[B' \in H] \ge \Pr[B \in H] -$

$\Pr[|u^{*\mathsf{T}}B| > C] \geq \frac{1}{e} - \frac{1}{150} \geq \frac{1}{4}$. Because $f_{B''} \geq g \cdot f_{B'} \geq \frac{4}{5} f_{B'}$, we have $\Pr[B'' \in H] \geq \frac{4}{5} \cdot \Pr[B' \in H] \geq \frac{1}{5} \geq \frac{1}{4e^2}$. This proves that B'' has an approximately symmetric distribution.

For all unit vectors $v \in \mathbb{R}^m$ we have:

$$f_{\langle v,B''\rangle}(x) \leq \frac{f_{\langle v,B'\rangle}(x)}{\Pr[\psi(B') = \text{accept}]} \leq 2 f_{\langle v,B'\rangle}(x).$$

This implies $\mathrm{Var}(\langle v, B''\rangle) \preccurlyeq 2\,\mathrm{Var}(\langle v, B'\rangle) \preccurlyeq 2I_m$. Because $\mathrm{Cov}(B') \geq \frac{1}{10} I_m$ and the fact that $\langle B', v\rangle$ is logconcave, by Lemma 2.3.7, $f_{\langle B',v\rangle} \leq \frac{1}{\sqrt{\mathrm{Var}\,\langle B',v\rangle}} \leq 4$. Hence, $f_{\langle v,B''\rangle} \leq 2 \cdot f_{\langle v,B'\rangle} \leq 768$. Now, Lemma 2.3.2 implies that $\mathrm{Var}(\langle B'', v\rangle) \geq \frac{1}{768}$. Hence $\frac{1}{768} I_m \preccurlyeq \mathrm{Cov}(B') \preccurlyeq 2I_m$. Now by Lemma 4.5.2 anti-concentration holds with a constant parameter. □

Proof of Theorem 3.1.1

Proof of Theorem 3.1.1. Consider the optimal solutions x^* and u^* to respectively the LPs (Primal LP) and (Dual LP). We condition on the event $|N_0| \geq n/10^5$ and $\|u^*\|_2 \leq 32$, where $N_0 := \{i \in [n] : x_i^* = 0\}$. By Lemma 3.3.4, this event occurs with probability at least $1 - e^{-\Omega(n)}$. Subject to this, we further condition on the exact values of x^*, u^*. We will show that for every such conditioning, the integrality is gap is small with high probability over the randomness of A_{N_0}.

Set $\delta := \frac{\mathrm{poly}(m)\log n}{n}$, where the polynomial factor is the same one as dictated by Theorem 4.3.4. We now show that we can construct a large subset $Z \subseteq N_0$, such that the reduced costs of the variables indexed by Z are small and the columns $A_i, i \in Z$, are independent and satisfy the necessary conditions in order to apply Theorem 4.3.4 to round x^* to a near optimal solution. By Lemma 3.3.2, first note that $(c_i, A_{;i}), i \in N_0$ are independent and distributed according to $u^{*\mathsf{T}}A_i - c_i > 0$.

By Lemma 3.3.8, using rejection sampling we can sample a set $Z \subseteq N_0$, such that $\mathrm{Law}(A_i | i \in Z) = \mathrm{Law}(A_i \mid |u^{*\mathsf{T}}A_i| \leq C)$, $\Pr[i \in Z | i \in N_0] = \Omega(\delta)$ and such that $u^{*\mathsf{T}}A_i - c_i \in [0, \delta]$ for all $i \in Z$. If columns of A have a discrete symmetric distribution (DSU), then $\mathbb{E}[A_i | i \in Z] = 0$. Moreover, by Lemma 3.3.7, the distribution of the columns in Z is symmetric (and therefore satisfies Definition 4.3.1), and satisfies the anti-concentration property with parameter $\kappa = \Omega(1)$. In the logconcave case, (LI), we apply a second round of rejection sampling to Z, as described in Lemma 3.3.9, which achieves that the law of $A_i, i \in Z$ is mean-zero, approximately symmetric, and anti-concentrated with parameter $\kappa = \Omega(1)$. Furthermore, this second step of rejection sampling only decreases the probability that $i \in Z$ by at most a constant factor.

In both cases, we see that $\mathbb{E}[Z] \geq \Omega(\delta|N_0|) = \Omega\left(\mathrm{poly}(m)\log n\right)$. Thus, by the Chernoff bound (2.1), $|Z| \geq \Omega\left(\mathrm{poly}(m)\log n\right)$ with probability at $1 - n^{-\mathrm{poly}(m)}$.

We now condition on the exact set $Z \subseteq N_0$ subject to this size lower bound. Note that $A_i, i \in Z$, are independent approximately symmetric, anti-concentrated with parameter $\kappa = \Omega(1)$ and mean-zero random vectors.

Set $p = \frac{\varepsilon \cdot \kappa^4}{1000 m^5}$, where $\varepsilon > 0$ is chosen small enough to make sure that we have $\kappa^3 \exp\left(\frac{\kappa^3}{3 \cdot 80^2 p m^3}\right) \geq 50000 m^2$. We consider the rounded vector x', from Lemma 3.3.1, and define the target, $t := A(x^* - x') - n^4 \exp(-p\kappa^3 |Z|/m)\mathbf{1}_m$ in (LI) setting and $t := \lfloor kA(x^* - x') \rfloor / k$ in the (DSU) setting. We will now apply Theorem 4.3.4 to obtain a set $T \subseteq Z$ such that $\| \sum_{i \in T} A_i - t \|_2 \leq n^4 \exp(-\kappa^3 p |Z|/m)$ in the (LI) setting or $\sum_{i \in T} A_i = t$ in the (DSU) setting. This will help us both fix the slack introduced by the rounding as well as enforce that the resulting solution is feasible.

We now invoke Lemma 3.3.1, which coupled with $|Z|p = \Omega(\text{poly}(m)\log(n))$ and the fact $\max\limits_{i \in [n]} \|A_i\| = O(\sqrt{\log(n)} + \sqrt{m})$, shows that, as long as the degree of the polynomial in δ is large enough,

$$\begin{aligned} \|t\| &\leq \|A(x^* - x')\| + m(n^4 \exp(-\kappa^3 p|Z|/m) + 1) \\ &\leq O(\sqrt{m \log n} + m) = o(\sqrt{p|Z|m}). \end{aligned}$$

Thus, for large n, Theorem 4.3.4 applies to the matrix A_Z and t in the (LI) setting and the matrix kA_Z, kt in the (DSU) setting. Thus, with probability $1 - e^{-\Omega(p|Z|)} = 1 - n^{-\text{poly}(m)}$ there exists a set $T \subseteq Z$ such that $|T| \leq \frac{3}{2}p|Z|$, and $\| \sum_{i \in T} A_i - t \| \leq 32 n^4 \exp(-\kappa^3 p|Z|/80m)$ in the (LI) setting and $\sum_{i \in T} A_i = t$ in the (DSU) setting.

Now we let $x'' = x' + \mathbf{1}_T$. We now show that

$$Ax'' \leq b \quad \text{and} \quad u^{*\mathsf{T}}(b - Ax'') \leq 1/\text{poly}(n). \tag{3.15}$$

Firstly, in the (DSU) setting, we have

$$Ax'' = Ax' + \lfloor k(Ax^* - Ax') \rfloor / k \leq Ax^* \leq b,$$

so x'' is a feasible integer solution. Take $j \in [m]$ such that $u_j^* > 0$. By complementary slackness we have that $(Ax^*)_j = b_j \in \mathbb{Z}/k$. Since $A \in \mathbb{Z}^{m \times n}/k$ and $x' \in \mathbb{Z}^n$, we have that $Ax' \in \mathbb{Z}^m/k$. In particular,

$$\begin{aligned} (Ax'')_j &= (Ax')_j + \lfloor k(Ax^* - Ax')_j \rfloor / k = (Ax')_j + \lfloor k(b_j - Ax')_j \rfloor / k \\ &= (Ax')_j + k(b_j - Ax')_j / k = b_j. \end{aligned}$$

We conclude that $u^{*\mathsf{T}}(b - Ax'') = 0$ as needed.

In the (LI) setting, we first note that

$$\|Ax'' - (Ax^* - n^4 \exp(-\kappa^3 p|Z|/m)\mathbf{1}_m)\| = \|\sum_{i \in T} A_i - t\| \leq n^4 \exp(-\kappa^3 p|Z|/m).$$

So, we must have $Ax'' \le Ax^* \le b$, and hence x'' is a feasible. Furthermore, by complementary slackness

$$\begin{aligned} u^{*\mathsf{T}}(b - Ax'') &= u^{*\mathsf{T}}(Ax^* - Ax'') \\ &\le \|u^*\|\|Ax^* - Ax''\| \le 32(\|Ax^* - t\| + \|t - Ax''\|) \\ &\le 32(m+1)n^4 \exp(-\kappa^3 p|Z|/m) \le 1/\operatorname{poly}(n). \end{aligned}$$

To conclude, we use x'' to bound the integrality gap with the (Gap Formula) applied to x'' and u^*:

$$\begin{aligned} \mathsf{IPGAP} &= u^{*\mathsf{T}}(b - Ax'') + \left(\sum_{i=1}^{n} x''_i (A^\mathsf{T} u^* - c)_i^+ + (1 - x''_i)(c - A^\mathsf{T} u^*)_i^+ \right) \\ &= u^{*\mathsf{T}}(b - Ax'') + \sum_{i \in T} (A^\mathsf{T} u^* - c)_i \quad \text{(by complementary slackness)} \\ &\le 1/\operatorname{poly}(n) + |T| \cdot \delta \qquad \text{(by (3.15) and } T \subseteq Z) \\ &\le O\left(\frac{\operatorname{poly}(m) \log(n)^2}{n} \right). \end{aligned}$$

□

The gap bound for packing IPs

In this section we will prove Theorem 3.1.2, our bound for Discrete Packing IPs. Here, the objective $c \in \mathbb{R}^m$ has independent entries that are exponentially distributed with parameter $\lambda = 1$. The $m \times n$ constraint matrix A has independent columns which are distributed with discrete uniform (DU) independent entries which are uniform on the interval $\{1, \dots, k\}$, $k \ge 3$.

As in the centered case, we divide the constraint matrix A and the right hand side b by k. So, we will assume that the entries of A are uniformly distributed in $\{\frac{1}{k}, \dots, 1\}$ and that the right hand side b lies in $((n\beta, n(1/2 - \beta)) \cap \frac{\mathbb{Z}}{k})^m$. This way, we can see this setting as a discrete approximation of the continuous setting where the entries of A are uniformly distributed in $[0, 1]$, like in [DF89].

We want to show $\mathsf{IPGAP} \le \frac{\exp(O(1/\beta)) \operatorname{poly}(m) (\log n)^2}{n}$ with probability at least $1 - n^{-\operatorname{poly}(m)}$. We will do this by first solving a slightly modified version of the LP-relaxation. We choose a $b' < b$. Now we let x^* be the minimizer of (Primal LP), where b is replaced by b' and let u^* be the optimal solution to the corresponding (Dual LP). We round down the solution, setting $x'_i := \lfloor x^*_i \rfloor$. Note that $\|A(x^* - x')\| \le \sum_{i : x^*_i \in (0,1)} \|A_i\| \le m\sqrt{m}$.

Similar to the proof of Theorem 3.1.1, our proof proceeds by flipping x'_i to 1 for a subset of indices for which $x^*_i = 0$. By duality, these are columns with $A_i - c_i \ge 0$.

To be able to apply Theorem 4.3.7, we convert the conditional distribution of the columns of A back into their original distribution using rejection sampling:

Lemma 3.3.10. *Let $B = A_i$ and let B', c' have* $\mathrm{Law}(B', c'_i) = \mathrm{Law}(B, c_i \mid u^{*\mathsf{T}}B - c_i \geq 0)$. *When $\delta \leq 1$, here exists a rejection sampling procedure ψ such that:*

- $\Pr[\psi(B', c'_i) = \mathrm{accept}] \geq \frac{1}{11}\delta \exp(-\|u^*\|_1)$.
- $\mathrm{Law}(B' \mid \psi(B', c'_i) = \mathrm{accept}) = \mathrm{unif}((\{\frac{1}{k}, \dots, 1\} \cap [\frac{1}{3m}, 1])^m)$.
- $u^{*\mathsf{T}}B' - c'_i \in [0, \delta]$ *whenever* $\psi(B', c'_i) = \mathrm{accept}$.

Proof. Let c'' and B'' be independent random variables with $c'' \sim \mathrm{Unif}(0, \delta)$ and let $B'' \sim \mathrm{Unif}((\{\frac{1}{k}, \dots, 1\} \cap [\frac{1}{3m}, 1])^m)$. Note that $f_{B''}(x) \leq f_B(x)/(1 - \frac{1}{m})^m$. Now we will apply Lemma 2.3.15 to transform (B', c') into (B'', c'') using rejection sampling. Observe that for any $\bar{B} \in \mathbb{R}^m, \bar{c} \in \mathbb{R}$ with $u^{*\mathsf{T}}B - c \in [0, \delta]$:

$$\begin{aligned}\frac{f_{(B'',c'')}(\bar{B}, \bar{c})}{f_{(B',c')}(\bar{B}, \bar{c})} &= \frac{f_{B''}(\bar{B})/\delta}{f_B(\bar{B})f_c(\bar{c})/\Pr_c[u^{*\mathsf{T}}B - \bar{c} \geq 0]} \leq \frac{f_B(\bar{B})/(1 - \frac{1}{m})^m/\delta}{f_B(\bar{B})f_c(\bar{c})} \\ &\leq \frac{4}{\delta f_c(\bar{c})} \leq \frac{4}{\delta \exp(-u^{*\mathsf{T}}B - \delta)} \leq \frac{11}{\delta \exp(-\|u^*\|_1)}.\end{aligned}$$

The stated result now follows directly by applying Lemma 2.3.15 on (B', c'_i) to get a rejection sampling procedure that satisfies $\mathrm{Law}((B', c'_i) \mid \psi(B', c'_i) = \mathrm{accept}) = \mathrm{Law}(B'', c'')$. □

In the previous lemma, both the acceptance probability and the maximal size of δ depend on $\|u^*\|_1$. To prevent this from affecting the proof, we will show that with high probability $\Omega(\beta^4) \leq \|u^*\|_1 \leq O(\frac{1}{\beta})$. Because our proof of Theorem 3.1.2 will rely on flipping the columns for which $x^*_i = 0$, we will also show that with high probability the number of these columns is at least proportional to n. We make use of the following bound on the binomial coefficient:

Lemma 3.3.11 ([Gal14, Theorem 3.1]). *For all $\alpha \leq \frac{1}{2}$ and all n,*

$$\sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} \leq \exp(H(\alpha)n),$$

where H is the entropy function defined as $H(x) = -x\ln(x) - (1-x)\ln(1-x)$.

Now we can derive the desired result.

Lemma 3.3.12. *Consider the packing setting, with the parameter $\beta \in (0, 1/4)$ and $b' \in ((n\beta/2, n(1/2 - \beta)) \cap \frac{1}{k}\mathbb{Z})^m$. Then, with probability at least $1 - e^{-\Omega(\beta^2 n)}$, we have $\Omega(\beta^4) \le \|u^*\|_1 \le O(\frac{1}{\beta})$ and $|N_0| \ge \Omega(\beta^4 n)$.*

Proof. Note that the distribution of the c_i's is exponential and therefore logconcave with $\mathbb{E}[c_i] = 1$. By Lemma 2.3.11 we now see that with probability $1 - e^{-\Omega(n)}$, we have $3n \ge \sum_{i=1}^n c_i$ and consequently,

$$3n \ge \sum_{i=1}^n c_i = c^\mathsf{T}\mathbf{1}_n \ge \mathrm{val}_{\mathsf{LP}}(x^*) = \mathrm{val}^*(u^*) \ge \sum_{i=1}^m b_i' u_i^* \ge \frac{n \cdot \beta}{2}\|u^*\|_1.$$

Hence, we have $\|u^*\|_1 \le \frac{6}{\beta}$, with high probability.

For the second claim, let $H : (0, \frac{1}{2}] \to (0, -\ln(\frac{1}{2})]$ be defined with $H(x) = -x\ln x - (1-x)\ln(1-x)$. Set $\alpha := \min(\frac{1}{2}\beta, H^{-1}(\frac{1}{8}\beta^2))$. As $H(x) \le 2\sqrt{x}$, we have $H^{-1}(x) \ge \frac{x^2}{4}$ and hence $\alpha \ge \frac{1}{256}\beta^4$. Let $x \in \{0,1\}^n$ and suppose that $K := |\{i : x_i = 1\}| \ge (1-\alpha)n$. By first using $b_1 \le (\frac{1}{2} - \beta)n$, and $\mathbb{E}[(Ax)_1] = \frac{K}{2}$, and then applying Hoeffding's inequality we see,

$$\begin{aligned}
\Pr\left[(Ax)_1 \le b_1'\right] &\le \Pr\left[(Ax)_1 \le \frac{1-\beta}{2}n\right] = \Pr\left[(Ax)_1 - \frac{1-\alpha}{2}n \le -\frac{\beta-\alpha}{2}n\right] \\
&\le \Pr\left[(Ax)_1 - \frac{K}{2} \le -\frac{\beta-\alpha}{2}n\right] \\
&= \Pr\left[(Ax)_1 - \mathbb{E}[(Ax)_1] \le -\frac{\beta-\alpha}{2}n\right] \\
&\le \exp\left(-(\beta-\alpha)^2 n\right) \le \exp\left(-\frac{1}{4}\beta^2 n\right).
\end{aligned}$$

Let $S = \{x \in \{0,1\}^n : |\{i : x_i = 1\}| \ge (1-\alpha)n\}$. Note that by Lemma 3.3.11, $|S| \le \sum_{i=0}^{\lfloor \alpha n \rfloor} \binom{n}{i} \le \exp(H(\alpha)n)$. Taking the union bound over all $x \in S$, we see that

$$\begin{aligned}
\Pr[\exists x \in S : (Ax)_1' \le b_1'] &\le |S|\exp(-\frac{1}{4}\beta^2 n) \le \exp(H(\alpha)n - \frac{1}{4}\beta^2 n) \\
&\le \exp(-\frac{1}{8}\beta^2 n).
\end{aligned}$$

So, with probability at least $1 - e^{-\Omega(\beta^2 n)}$ all feasible values $x \in \{0,1\}^n$ have $|\{i : x_i = 0\}| \ge \alpha n$ and in particular $|N_0| \ge \alpha n \ge \frac{1}{256}\beta^4 n$.

At the same time, observe that when $i \in N_0$, we must have $c_i - u^{*\mathsf{T}} A_i \le 0$, so in particular $\|u^*\|_1 \ge c_i$. We have $\Pr[c_i \le \ln(\frac{1}{1-\alpha/2})] \le 1 - \exp(-\ln(\frac{1}{1-\alpha/2})) = \frac{1}{2}\alpha$. By the Chernoff bound (2.1) this implies that with probability at least $1 - \exp(-\Omega(n))$

we have $|\{i \in [n] : c_i \leq \ln(\frac{1}{1-\alpha/2})\}| \leq \frac{3}{4}\alpha n$. If this event holds and at the same time we have $|N_0| \geq \alpha n$, then this implies $\|u^*\|_1 \geq \ln(\frac{1}{1-\alpha/2})$ because otherwise $N_0 \subseteq \{i \in [n] : c_i \leq \ln(\frac{1}{1-\alpha/2})\}$, contradicting the bounds on their size. So, we can conclude that with high probability we have $\|u^*\|_1 \geq \ln(\frac{1}{1-\alpha/2}) \geq -\ln(1 - 2^{-8}\beta^4) \geq 2^{-8}\beta^4$.

□

Proof of Theorem 3.1.2. Let $r = \lceil \frac{10^6 m^{12} \log(n)}{s^2} \rceil$, where s is a constant that we will choose later. Let $\mu = \frac{k + \lceil \frac{k}{3m} \rceil}{2k} = \mathbb{E}[U]$, where $U \sim \text{Uniform}(\{\frac{1}{k}, \ldots, 1\} \cap [\frac{1}{3m}, 1])$. Now we define

$$\gamma = \frac{r}{1000m^5}\mu \text{ and } b' = b - \gamma \mathbf{1}. \tag{3.16}$$

Let x^* and u^* be the optimal solutions of (Primal LP) and (Dual LP) where b is replaced by b'. We will assume that $\frac{\beta^4}{C_1} \leq \|u^*\|_1 \leq \frac{C_1}{\beta}$ and $|N_0| \geq C_2 \cdot \beta^4 \cdot n$, for some constants C_1, C_2. By Lemma 3.3.12 this happens probability $1 - e^{-\Omega(\beta^2 n)}$. Subject to this, we condition on the exact values of x^*, u^*.

Let $\delta := \frac{11 \exp(C_1/\beta) r}{C_2 \beta^4 n}$. By our assumption that $n \geq \text{poly}(m) \exp(\Omega(1/\beta))$, we may assume that $\delta \leq \beta^4/(C_1 m) \leq \frac{\|u^*\|_1}{3m}$. Thus, by Lemma 3.3.10 we can sample a set $Z \subseteq N_0$ such that for $\text{Law}(A_i | i \in Z) = \text{Uniform}((\{\frac{1}{k}, \ldots, 1\} \cap [\frac{1}{3m}, 1])^m)$ and that $\Pr[i \in Z | i \in N_0] = \frac{1}{11}\delta \exp(-\|u^*\|_1)$. Noting that $\mathbb{E}[|Z|] = 2r$, by Chernoff's inequality (2.1), with probability at least $1 - n^{-\text{poly}(m)}$, we have $|Z| \geq r$. Now we restrict Z to its first s elements, to get $|Z| = r$. Observe that $\mathbb{E}[A_i | i \in Z] = \mu \mathbf{1}$.

We consider the target vector $t \in \mathbb{R}^m$, defined by:

$$t_i := \begin{cases} b_i - (Ax')_i : & u_i^* > 0 \\ \lfloor \gamma \rfloor : & \text{otherwise} \end{cases},$$

which satisfies $(b - Ax' - t)^\mathsf{T} u^* = 0$. Our next step will be to apply Theorem 4.3.7 on $kA_Z \in \mathbb{Z}^{m \times r}$ and $kt \in \mathbb{Z}^m$ with parameter $p = \frac{\gamma}{\mu r} = \frac{1}{1000m^5}$, to get a set $T \subseteq Z$ such that $\sum_{i \in T} A_i = t$. Note that we have chosen γ and r to have $p^4 = \omega(\frac{m^3}{r})$.

To verify that t is indeed covered by Theorem 4.3.7, note that by Lemma 3.3.2 the columns kA_i for $i \in Z$ are independent with entries uniformly distributed in

$\{\lceil k/m\rceil, \dots, k\}$. We now show that t is sufficiently close to the mean $|Z|p\mu$:

$$\begin{aligned}\|t - |Z|p\mu\| = \|t-\gamma\mathbf{1}\| &= \sqrt{\sum_{j:u_j^*>0} (b_j - (Ax')_j)^2 + |\{j : u_j^* = 0\}|(\gamma - \lfloor\gamma\rfloor)^2} \\ &\leq \sqrt{\sum_{j:u_j^*>0} (A(x^*-x')_j)^2 + |\{j : u_j^* = 0\}|} \\ &\leq \sqrt{\sum_{j:u_j^*>0} \|x^*-x'\|_1^2 + |\{j : u_j^* = 0\}|} \\ &\leq m^{1.5} \leq \frac{s}{1000m^5}\sqrt{rm} = sp\sqrt{|Z|m} \leq s\sqrt{p|Z|m}.\end{aligned}$$

Now choose the constant s such that the previous inequality implies the condition from Theorem 4.3.7. As a result, with probability $1-\exp(-p|Z|) \geq 1 - n^{-\operatorname{poly}(m)}$ there exists a set $T \subseteq Z$, such that $\sum_{i\in T} A_i = t$.

Let $x'' = x' + \mathbf{1}_T$. Noting that x' was obtained from x^*, for i with $u_i^* = 0$ we have

$$(Ax'')_i = (Ax')_i + t_i \leq b_i' + \gamma = b_i.$$

For, i with $u_i^* > 0$ we have:

$$(Ax'')_i = (Ax')_i + t_i = b_i,$$

which means that x'' is a feasible solution to the integer program.

Using (Gap Formula) for x'' and u^*, we now get:

$$\begin{aligned}\mathsf{IPGAP} = \mathrm{val}_{\mathsf{LP}} - \mathrm{val}_{\mathsf{IP}} &\leq \mathrm{val}^*_{\mathsf{LP}}(u^*) - \mathrm{val}_{\mathsf{LP}}(x'') \\ &= b^{\mathsf{T}}u^* + \sum_{i=1}^n (c - A^{\mathsf{T}}u^*)_i^+ - c^{\mathsf{T}}x'' \\ &= (b - Ax'')^{\mathsf{T}}u^* + \left(\sum_{i=1}^n x_i''(A^{\mathsf{T}}u^* - c)_i^+ + (1 - x_i'')(c - A^{\mathsf{T}}u^*)_i^+\right) \\ &= (b - Ax' - t)^{\mathsf{T}}u^* + \sum_{i\in T}(A^{\mathsf{T}}u^* - c)_i \quad \text{(by complementary slackness)} \\ &= \sum_{i\in T}(A^{\mathsf{T}}u^* - c)_i \qquad \text{(since } (b - Ax' - t)^{\mathsf{T}}u^* = 0) \\ &\leq \delta|T| \leq \frac{(\exp(C_1/\beta)\operatorname{poly}(m)\log n)^2}{n}. \qquad \text{(by Lemma 3.3.10)}\end{aligned}$$

□

3.4 Conclusion

We have provided high-probability upper bounds on the size of branch-and-bound trees for randomly sampled integer programs. The bounds we have shown apply to instances of the $\{0, 1\}$-knapsack problem and the generalized assignment problem.

As we mentioned before, real-world LPs often have a large number of combinatorial constraints. Such constraints generally have discrete coefficients, which motivated us to study discretely distributed constraint matrices. However, the IPs that we studied do not have an inherent combinatorial structure. For future work, it would be particularly interesting to study the size of branch-and-bound trees for random IPs with more combinatorial structure.

Theorem 3.1.3, our bound on the tree size holds for quite general polytopes P, which means that it can also be applied to other types of problems, such as the matching problem with additional knapsack constraints. In order to derive appropriate bounds for such problems, one would need bounds on the integrality gap of the polytope P, along with a bound on the number of elements of corresponding knapsack polytope.

Chapter 4

Discrepancy bounds for random matrices

In Chapter 3, we have shown a high probability bound on the integrality gap of random IPs from several classes of probability distributions. This proof consists of rounding LP solutions using a new "discrepancy theorem". In this chapter, we will prove this theorem.

4.1 Introduction

Consider the following question: Let $t \in \mathbb{R}^m$ be a target vector and let $A \in \mathbb{R}^{m \times \bar{n}}$ be a "nice" random matrix with independent columns. When can we ensure with high probability that t is equal to or very close to a $\{0, 1\}$ combination of the columns of A?

A general answer to this question was given in [BDHT22, Lemma 1], improving upon [DF89, Lemma 3.4]. However, it enforced very strict conditions on the entries of A. Specifically, the entries of A needed to be independent, mean zero with unit variance, absolutely continuous random variables of bounded density which "converge quickly enough" to a Gaussian when averaged. Furthermore, for the targets t in the "range" of A, the probability of successfully hitting t was only $\Theta(1)$.

In this chapter we give much more general and powerful discrepancy theorems. We state them below, restricted to special cases relevant for our applications (see Theorem 4.3.4 for the general result).

Theorem 4.3.5. *Suppose the columns of $A \in \mathbb{R}^{m \times n}$ are continuously distributed, independent with a common mean $\mu \in \mathbb{R}^m$, are approximately symmetric around their mean, and $(\Theta(1), \Omega(1))$-anti-concentrated. Let $p \in [0, 1]$ with $\frac{\mathrm{poly}(m)\log(n)}{n} \leq p \leq \frac{1}{\mathrm{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every t with $\|t - pn\mu\| \leq O\left(\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $\|A\mathbf{1}_S - t\| \leq \exp\left(-\Omega(\frac{pn}{m})\right)$.

The contents of this chapter are based on joint work with Daniel Dadush and Dan Mikulincer [BDM23].

The conditions of *anti-correlation* and *approximate symmetry* are technical and will be defined in Section 4.3. The above corollary includes the case where the columns of A are logconcave, isotropic, by Lemma 4.4.1.

Theorem 4.3.7. *Suppose the entries of A are uniformly sampled from $\{i, i+1, \dots, i+k\}$ for $k \geq j \geq \max(2, |i|)$. Let $p \in [0,1]$ with $\frac{\mathrm{poly}(m)\log(n)\log(k)}{n} \leq p \leq \frac{1}{\mathrm{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every vector $t \in \mathbb{Z}^n$ with $\|t - pn\mu\| \leq O\left(k\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $A\mathbf{1}_S = t$.

Relations to discrepancy theory

We first explain the connection to linear discrepancy. As defined by Lovász, Spencer and Vesztergombi [LSV86], the linear discrepancy of a matrix $A \in \mathbb{R}^{m\times n}$ is equal to $\mathsf{lindisc}(A) := \max_{\lambda\in[0,1]^n} \min_{x\in\{0,1\}^n} \|A(x-\lambda)\|_\infty$. That is, it is the maximum "rounding error" one must incur to round a $[0,1]$ combination of the columns to a $\{0,1\}$ combination (the specific choice of ℓ_∞ vs ℓ_2 norm is not important in our context). The discrepancy $\mathsf{disc}(A)$ of A is linear discrepancy restricted to $\lambda = \mathbf{1}_n/2$ (the all $1/2$ vector). It is more common to expressed discrepancy by $\min_{x\in\{-1,1\}^n} \|Ax\|_\infty = 2\mathsf{disc}(A)$, in which case $x \in \{-1,1\}^n$ is interpreted as a 2-coloring of the columns of A. While linear discrepancy is always larger than discrepancy, [LSV86] showed the maximum discrepancy of any subset of the columns of A, known as hereditary discrepancy, upper bounds $\mathsf{lindisc}(A)$ up to a factor of 2.

Bounds on the discrepancy of various matrix classes, often induced by the incidence matrix of set system, have found many applications in computational geometry and complexity (see [Mat99; Cha01]). Over the last decade or so, efficient algorithms for producing low-discrepancy colorings have been developed [Ban10; LM15; Rot17; ES18; BDGL18] and have found many applications in the context of approximation algorithms [LRS11; HR17; BT69; BRS22].

With the above perspective, Theorem 4.3.4 can be interpreted as bounding the linear discrepancy of the random matrix A for combinations $\lambda \in [0,1]$ which are very close to $p\mathbf{1}_n$, where $p \in (0,1)$ is as above. As the columns of A are independent with mean μ, one can expect that $A(p\mathbf{1}_n) \approx pn\mu$. Perhaps slightly less clear is that every $t \in \mathbb{R}^m$, which is close to $pn\mu$, will in fact be exactly expressible as $t = A\lambda$, where $\lambda \approx p\mathbf{1}_n$ with high probability (this requires an analysis of the singular values of A). Theorem 4.3.4 now shows that λ can be replaced by $x \in \{0,1\}^n$ with $\|x\|_1 = \Theta(pn)$ incurring either no error in the discrete case (assuming $t \in \mathbb{Z}^n$) or exponentially small error in the continuous case, thereby bounding the linear discrepancy of the combination λ. At least in the continuous case, we note that one can in fact adapt the proof of Theorem 4.3.4 to directly bound the linear discrepancy of any "reasonable" combination λ (i.e., without the detour through the mean μ).

This would be somewhat less useful for the integrality gap application we consider here, since it would be give significantly less precise guarantees on the targets we can expect to hit. Interestingly, while the application pursued here is algorithmic, i.e, bounding the complexity of branch-and-bound, we are not aware of any efficient algorithm to compute the rounding x guaranteed by Theorem 4.3.4.

Crucial to the linear discrepancy bounds we achieve in Theorem 4.3.4, i.e., either exponential small or zero, is that the matrix A has many more columns than the dimension m. In particular, reducing to bounds on hereditary discrepancy becomes useless in this setting. The study of discrepancy in the "over complete" setting has become very active more recently, with works focusing on the discrepancy of large random matrices and set systems [Cos09; KLP12; EL19; HR19; FS20; Pot18; BM19]. Many of these works were motivated by the Beck-Fiala conjecture, which posits that discrepancy of any $\{0, 1\}$ matrix A in which every column has at most t ones is bounded by $O(\sqrt{t})$ (here A can be interpreted as the incidence matrix of a set system). Variants of this conjecture for random set systems and matrices where established in [Cos09; EL19; HR19; FS20; Pot18; MMPP23; BM19], where in the case $n \gg m$, it was shown that discrepancy quickly drops to 1 [HR19; FS20; Pot18; MMPP23] or even exponentially close to zero [Cos09; FS20] depending on whether the columns of the matrix A are discretely or continuously distributed.

Discrepancy via Fourier Analysis

To prove our linear discrepancy theorem, we rely on a Fourier analytic approach, which is entirely different from the second-moment counting based proofs in [DF89; BDHT22]. The high level approach was pioneered by Kuperberg, Peled and Lovett in [KLP12], where they applied Fourier analytic techniques to show the existence of rigid combinatorial objects, such as orthogonal arrays, Steiner systems and regular hypergraphs. At a technical level, for a matrix $A \in \{0, 1\}^{m \times n}$, they were interested in understanding the minimum $p \in (0, 1]$ such that $pA\mathbf{1}_n = Ax$, where $x \in \{0, 1\}^n$ and $\|x\|_1 = pn$. They examined this question for highly symmetric deterministic matrices coming from the above applications. The Fourier analytic approach was later applied by Hoberg and Rothvoss [HR19] and independently by Frank and Saks [FS20] to show very strong upper bounds on the discrepancy $\min_{x \in \{-1,1\}^n} \|Ax\|_\infty$ of a random matrix A when $n \gg m$, where the columns of A were drawn i.i.d. from various distributions.

From a comparative perspective, our Theorem 4.3.4 sits in between the work of [KLP12] and [HR19; FS20]. We work with random matrices A as in [HR19; FS20], though the question we attack is more in the spirit of [KLP12]. We note that unlike [KLP12], it is not sufficient for us to show the existence of a rounding $x \in \{0, 1\}^n$ such that $A(x - p\mathbf{1}_n)$ is small or zero. For our applications, we require this to be true with $Ap\mathbf{1}_n$ replaced by $pn\mu = \mathbb{E}[Ap\mathbf{1}_n]$. We further require the existence of x to hold for any target t close enough to $pn\mu$. This last requirement however generally comes for "free" with Fourier analytic techniques (the moment you can hit $pn\mu$ you can hit everything close to it as well). A more significant difficultly is that the concentration of $Ap\mathbf{1}_n$ around its mean $pn\mu$ is rather weak. That is, the probability that $Ap\mathbf{1}_n$ is close to its mean scales relative to the ambient dimension m (which is constant) instead of n. This makes achieving the $1 - e^{-\Omega(pn)}$ success probability more challenging.

We now explain the high-level approach. Let $A \in \mathbb{R}^{m\times n}$ be our random matrix with columns having mean $\mu \in \mathbb{R}^m$ and covariance matrix $\sigma^2 I_m$, and let $p \in (0, 1)$ be our parameter. The strategy is to directly analyze the probability mass function of the random variable $Y = AX$, where $X_1, \ldots, X_n$ are i.i.d. Bernoulli's with probability p. [KLP12] also use this distribution, whereas [HR19; FS20] choose $X_1, \ldots, X_n$ to be uniform $\{-1, 1\}$ random variables. Restricting attention to the discrete case, where $Y \in \mathbb{Z}^n$, we show that $\Pr[Y = t] \gg 0$ for $t \in \mathbb{Z}^n$, when $\|np\mu - t\| = O_m(\sigma\sqrt{pn})$, where $O_m(\sigma^2 pn)$ roughly measures the "available variance" of AX in all directions. Obtaining a bound for $\Pr[Y = t]$ is done by applying the Fourier inversion formula, showing that the Fourier coefficients are close to those of a Gaussian and integrating (see Section 4.4 for an overview).

For $\theta \in [-1/2, 1/2]^m$, the corresponding Fourier coefficient of AX is expressed by $\mathbb{E}_X[e^{2\pi i\langle\theta,AX\rangle}]$. Similar to [HR19; FS20], we control the magnitude of these coefficients using the anti-concentration properties of the columns of A. Specifically, for our choice of column distributions, we need to show the the probability that $\langle\theta, A_i\rangle$, $i \in [n]$, is "close" to an integer decays predictably as a function of $\sigma\|\theta\|$. We note that the exact expression for the Fourier coefficients indeed differs depending on whether the entries of X are Rademacher or Bernoulli distributed, where the former is more prone to parity issues (e.g., if $A \in \{0, 1\}^{m\times n}$ and $x \in \{-1, 1\}^n$, the parity of Ax is fixed). Such parity issues do not arise in our setting. As mentioned above, we face a different difficulty, which is being able to pinpoint the exact targets whose probabilities we can accurately estimate due to the poor concentration of $\mathbb{E}_X[AX] = A(p\mathbf{1}_n)$ around $\mathbb{E}_{X,A}[AX] = np\mu$.

To deal with this issue, we first carefully subsample a set $S \subseteq [n]$ of columns from A, whose sum is close to the mean, and then generate Y from these subsampled columns. To construct S, we iterate through the columns one by one, adding A_i to S if $\langle\sum_{j\in S}(A_j - \mu), A_i - \mu\rangle \leq 0$ and $\|A_i - \mu\| \leq 2\mathbb{E}[\|A_i - \mu\|_2^2]^{1/2}$. This subsampling deterministically ensures that $\|\sum_{i\in S}(A_i - \mu)\|_2 \leq 2\sqrt{\sum_{i\in S}\mathbb{E}[\|A_i - \mu\|_2^2]}$, suitably biasing the sum towards the mean. For the distributions we work with, it is easy to show that $|S| = \Omega(n)$ with probability $1 - e^{-\Omega(n)}$, so we always have a constant fraction of the columns left over. Note that this subsampling crucially uses our flexibility to drop columns of A, a distinguishing feature of $\{0, 1\}$ combinations versus $\{-1, 1\}$ combinations. This subsampling process, however, causes non-trivial dependencies among the columns of A. That is, the submatrix we use to generate Y no longer has independent columns. Fortunately, we show that even with the conditioning induced by subsampling, the columns still retain enough of their anti-concentration properties to allow the Fourier analytic estimates to go through.

Organization

In Section 4.2 we provide some preliminaries on Fourier analysis. In Section 4.3 we state the general version of our discrepancy theorem and in Section 4.4 we provide the proof. In Section 4.5 we prove that the anti-concentration property that is required in our main discrepancy theorem holds for the distributions we study.

4.2 Preliminaries on Fourier analysis

Our main tool for proving the discrepancy result is Fourier analysis and we review here the necessary details. Fix $X \sim \mathcal{D}$, a random vector in $\mathbb{R}^m$. The Fourier transform of X (sometimes also called the characteristic function) is the complex-valued function defined by $\hat{X}(\theta) := \mathbb{E}[\exp(2\pi i\langle X, \theta\rangle)]$.

To understand the natural domain for θ, we first define the (dual) period of $\mathcal{D}$. For this, choose an arbitrary $a \in \text{support}(\mathcal{D})$ and denote,

$$\text{period}(\mathcal{D}) := \{v \in \mathbb{R}^m : \langle v, w - a\rangle \in \mathbb{Z},\ \forall w \in \text{support}(\mathcal{D})\}. \tag{4.1}$$

The definition of $\text{period}(\mathcal{D})$ does not depend on the choice of a. It is readily seen that when $\mathcal{D}$ is absolutely continuous with respect to the Lebesgue measure $\text{period}(\mathcal{D}) = \{0\}$, while $\text{period}(\mathcal{D}) = \mathbb{Z}^m$, when $\text{support}(\mathcal{D}) \subseteq \mathbb{Z}^m$. These are the cases on which we focus. Using the period, we define the fundamental domain of $\mathcal{D}$, in Fourier space (where we suppress the dependence on $\mathcal{D}$),

$$V := \{\theta \in \mathbb{R}^m : \|\theta\| \leq \inf_{0 \neq w \in \text{period}(\mathcal{D})} \|\theta - w\|\}. \tag{4.2}$$

Observe that if $\mathcal{D}$ is absolutely continuous with respect to the Lebesgue measure, then $V = \mathbb{R}^m$ and when $\text{support}(\mathcal{D}) \subseteq \mathbb{Z}^m$, $V = [-\frac{1}{2}, \frac{1}{2}]^m$.

The connection between X and its Fourier transform comes from the Fourier inversion formula [SW72, Theorem 1.20]:

Theorem 4.2.1 (Fourier inversion formula)**.** *Suppose that either $\mathcal{D}$ is absolutely continuous, or* $\mathrm{support}(\mathcal{D}) \subseteq \mathbb{Z}^m$*. Then, for $t \in \mathrm{support}(\mathcal{D})$:*

$$\Pr[X = t] = \int_{\theta \in V} \hat{X}(\theta) \exp(-2\pi i \langle \theta, t \rangle) d\theta.$$

If X is absolutely continuous, we interpret $\Pr[X = t]$ *as the density of X at t.*

Another desirable property of the Fourier transform is that it is particularly amenable to convolutions (this is clear from the exponential representation but see [SW72, Theorem 3.18]).

Theorem 4.2.2 (Multiplication-convolution theorem)**.** *Let X and Y be two independent random vectors. Then,*

$$(\widehat{X + Y})(\theta) = \hat{X}(\theta)\hat{Y}(\theta).$$

4.3 The discrepancy theorem

Let $A \in \mathbb{R}^{n \times m}$ be a random matrix with independent columns. Throughout the chapter we assume that the columns of A have the same mean, denoted as μ. We also make the assumption that all columns of A have the same period, as defined in (4.1), and denote it by $\mathrm{period}(A)$. We will deal with two different cases: either $\mathrm{period}(A) = \{0\}$, which means that the columns of A are absolutely continuous, or $\mathrm{period}(A) = \mathbb{Z}^m$, in which case the columns of A are supported in the lattice $\mathbb{Z}^m$.

Before stating the result, let us introduce some definitions. We first describe the distributions captured by our result; distributions with a non-negligible mass on every half-space passing through their mean.

Definition 4.3.1 (approximately symmetric distributions)**.** A probability distribution $\mathcal{D}$ on $\mathbb{R}^m$, with mean μ, is called *approximately symmetric* if, for any $\nu \in \mathbb{R}^m$,

$$\Pr_{X \sim \mathcal{D}} (\langle X, \nu \rangle \geq \langle \mu, \nu \rangle) \geq \frac{1}{4e^2}.$$

Remark 4.3.2. The constant $\frac{1}{4e^2}$ in the definition is somewhat arbitrary and could be relaxed to any smaller constant. It is immediately clear that any distribution which is symmetric around its mean is also approximately symmetric. Moreover, Grünbaum's inequality in Lemma 2.3.8 shows that logconcave measures are approximately symmetric.

Suppose that the columns of A are approximately symmetric. As a preprocessing step in our proof, we apply a subsampling procedure to the columns of A that removes approximately half the columns. This step will change the distribution and effective size of the random matrix A. After this step, the matrix A will satisfy appropriate concentration bounds that improve as n increases, irregardless of the value of m (Lemma 4.4.2).

Then, the main idea will be to choose $S \subseteq [n]$ randomly with $\Pr[i \in S] = p$ for $p \leq \frac{1}{\text{poly}(m)}$, independently for all $i \in [n]$. We then show that, with high probability over A and positive probability over S, $A\mathbf{1}_S$ is close to a target vector t. Thus, let us define the random vector $D := A\mathbf{1}_S$.

To understand the distribution of D we will consider its Fourier transform, denoted $\hat{D}(\theta) := \mathbb{E}[\exp(2\pi i\langle D, \theta\rangle)]$. Note, that under the assumptions above, we have $\text{period}(A) = \text{period}(D)$, and so we shall use V to denote the fundamental domain of D, as in (4.2).

The next definition quantifies an appropriate notion of anti-concentration properties for the columns of A.

Definition 4.3.3 (anti-concentration). Let $\sigma \geq 0$ and $\kappa \in (0, 1)$. We say the measure $\mathcal{D}$ is (σ, κ)*-anti-concentrated* if $\text{Cov}(\mathcal{D}) \preceq \sigma^2 I_m$, and for any $\nu \in \mathbb{R}^m$ and any $\theta \in V$,

$$\Pr_{X\sim\mathcal{D}} \left[d(\theta^{\mathsf{T}}X, \mathbb{Z}) \geq \kappa \min\left(1, \|\theta\|_\infty \sigma\right) \mid \langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right] \geq \kappa, \tag{AC}$$

where $d(\theta^{\mathsf{T}}X, \mathbb{Z}) := \inf_{z\in\mathbb{Z}} |\theta^{\mathsf{T}}X - z|$. When σ is clear from the context, we will sometimes omit the dependence on σ from the definition.

Without further details, the definition might seem opaque. Below we explain the rationale for considering this notion of anti-concentration and demonstrate some examples of distributions that satisfy Definition 4.3.3. For now, it shall suffice to say that the (AC) property appears naturally when trying to establish bounds on Fourier transforms.

It will be natural to measure the size of μ in relation to σ. Hence we define $\zeta = \|\mu\|^2/\sigma^2$. With the above notation, the main result of this section is:

Theorem 4.3.4. *Suppose that the columns of A are independent with a common period and a common mean $\mu \in \mathbb{R}^m$, are approximately symmetric, and (σ, κ)-anti-concentrated, with constants $\sigma, \kappa > 0$. Further, let $p \in [0, 1]$ and assume that the following technical condition is met:*

$$p \leq \frac{\kappa^3}{2^{16}m^2\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)} \tag{4.3}$$

$$p \geq \frac{10^{13}m^6(\zeta+1)^3\ln(n)}{\kappa^{12}n}\left(\ln(16\sqrt{\zeta+1}) + m\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)^3. \tag{4.4}$$

for $\zeta = \|\mu\|^2/\sigma^2$. *Then, when* $\mathrm{period}(A) = \{0\}$, *with probability* $1 - e^{-\Omega(\kappa pn)}$ *we have: For all* $t \in \mathbb{R}^m$ *such that*

$$\|t - pn\mu\| \leq \sqrt{\frac{\kappa^3 pn\sigma^2}{2^{15}m\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)}},$$

there is a set S *of size* $|S| \in [\frac{1}{400}pn, \frac{3}{400}pn]$ *such that*

$$\|A\mathbf{1}_S - t\| \leq \exp\left(-\frac{\kappa^3 pn}{80m}\right)\sigma n^3.$$

By getting rid of the dependency on σ and κ, we get the following simplified corollary.

Theorem 4.3.5. *Suppose the columns of* $A \in \mathbb{R}^{m\times n}$ *are continuously distributed, independent with a common mean* $\mu \in \mathbb{R}^m$, *are approximately symmetric around their mean, and* $(\Theta(1), \Omega(1))$*-anti-concentrated. Let* $p \in [0,1]$ *with* $\frac{\mathrm{poly}(m)\log(n)}{n} \leq p \leq \frac{1}{\mathrm{poly}(m)}$.

Then, with probability $1 - e^{-\Omega(pn)}$ *for every* t *with* $\|t - pn\mu\| \leq O\left(\frac{\sqrt{pn}}{\log(m)m}\right)$ *there exists a set* S *of size* $|S| \in [\Omega(pn), O(pn)]$ *such that* $\|A\mathbf{1}_S - t\| \leq \exp\left(-\Omega(\frac{pn}{m})\right)$.

Theorem 4.3.4, as stated, only deals with distributions which are absolutely continuous with respect to the Lebesgue measure. However, the argument also applies to measures with singularities. In particular, the result also holds for distributions supported on the lattice $\mathbb{Z}^m$, the case which is most relevant to our work. Moreover, in the lattice case, if one finds a subset $S \subseteq [n]$, such that $A\mathbf{1}_S$ is very close to some target vector $t \in \mathbb{R}^m$, then, since $A\mathbf{1}_S$ belongs to the lattice as well, one can actually deduce $A\mathbf{1}_S = t$. We prove this statement as a part of the proof of Theorem 4.3.4.

Theorem 4.3.6. *Suppose that* $\mathrm{period}(A) = \mathbb{Z}^m$. *Then, under the same conditions of Theorem 4.3.4 together with* $\exp\left(\frac{\kappa^3 pn}{80m}\right) \geq 2\sigma^2 n^2$, *with probability* $1 - e^{-\Omega(\kappa pn)}$, *there is a set* S *of size* $|S| \in [\frac{1}{400}pn, \frac{3}{400}pn]$ *such that* $A\mathbf{1}_S = t$, *provided that* $t \in \mathbb{Z}^m$ *satisfying the distance bound from Theorem 4.3.4.*

Note that since, by (4.4), $\kappa^3 pn \geq \mathrm{poly}(m)\log(n)$, the condition $\exp\left(\frac{\kappa^3 pn}{80m}\right) \geq 2\sigma^2 n^2$ almost does not restrict generality. Again, we state a simplified version of the result, which does not depend on σ and κ.

Theorem 4.3.7. *Suppose the entries of A are uniformly sampled from $\{i, i+1, \dots, i+k\}$ for $k \geq j \geq \max(2, |i|)$. Let $p \in [0,1]$ with $\frac{\text{poly}(m)\log(n)\log(k)}{n} \leq p \leq \frac{1}{\text{poly}(m)}$.*

Then, with probability $1 - e^{-\Omega(pn)}$ for every vector $t \in \mathbb{Z}^n$ with $\|t - pn\mu\| \leq O\left(k\frac{\sqrt{pn}}{\log(m)m}\right)$ there exists a set S of size $|S| \in [\Omega(pn), O(pn)]$ such that $A\mathbf{1}_S = t$.

4.4 Proving the theorem

We start with a high-level overview of the proof, in which we will limit ourselves to the discrete case. Recall that by Fourier's inversion formula (Theorem 4.2.1):

$$\Pr[D = t] = \int_{\theta \in V} \hat{D}(\theta) \exp(-2\pi i \langle \theta, t \rangle) d\theta.$$

In light of this formula, it will be enough to show that this integral is positive for appropriate t, for most choices of A. In this case we will get that $\Pr[D = t] > 0$, which implies the existence of an appropriate subset of columns. We will do so by showing that most of the mass lies close to the origin and that the integrand has an exponential decay far from the origin. Our proof consists of the following steps:

1. In Lemma 4.4.5 and Lemma 4.4.6 we show that $|\hat{D}(\theta)|$ is roughly proportional to the density function of the probability distribution $\mathcal{N}(0, \frac{1}{\sigma^2 npr^2} I_m)$.

2. In Lemma 4.4.3 we show that for small θ, the argument of $\hat{D}(\theta)$ is close to $2\pi np\langle \theta, \mu \rangle$. This allows us to show in Corollary 4.4.4 that as a consequence $\hat{D}(\theta)\exp(-2\pi i \langle \theta, t \rangle) \geq \frac{1}{\sqrt{2}} |\hat{D}(\theta)|$ for $\|\theta\|_2 \leq \hat{r}$, where:

$$\hat{r} = \Omega\left(\min\left(\frac{1}{p\sqrt{nm}\sigma}, \frac{1}{\sqrt[3]{pn}\left(\sigma\sqrt{m/\kappa} + \|\mu\|\right)}, \frac{1}{\|t - pn\mu\|}\right)\right).$$

Note that a constant fraction of the mass of $\mathcal{N}(0, \frac{1}{\sigma^2 npr^2} I_m)$ lies in the ball of radius r for some $r = \Theta(\frac{1}{\sigma\sqrt{np}})$. By scaling r we can make the fraction arbitrarily close to 1. If we then set $\mathcal{A}_1 = B(0, r)$, then the integral of $|\hat{D}(\theta)|$ over $\mathcal{A}_1$ will be large compared to the same integral over $V \setminus \mathcal{A}_1$. Now observe that for large n and appropriate p, $r < \hat{r}$. Hence, $\int_{\theta \in \mathcal{A}_1} \hat{D}(\theta) \exp(-2\pi i \langle \theta, t \rangle) \geq \frac{1}{\sqrt{2}} \int_{\theta \in \mathcal{A}_1} |\hat{D}(\theta)| d\theta$ is much larger than $\int_{\theta \in V \setminus \mathcal{A}_1} |\hat{D}(\theta)| d\theta$. This allows us to show:

$$\Pr[D = t] = \int_{\theta \in V} \hat{D}(\theta) \exp(-2\pi i \langle \theta, t \rangle) d\theta - \int_{\theta \in V \setminus \mathcal{A}_1} |\hat{D}(\theta)| d\theta > 0.$$

In fact, the proof is a bit more involved, as the upper bound that we provide for $|\hat{D}(\theta)|$ becomes weaker as $\|\theta\|$ gets larger. For this reason we partition $V \setminus \mathcal{A}_1$ into three sets, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$, and bound the integral over each of these sets separately. We set $\mathcal{A}_2 = [-\frac{1}{\sigma}, \frac{1}{\sigma}]^m \setminus \mathcal{A}_1$, $\mathcal{A}_3 = [-R, R]^m \setminus (\mathcal{A}_1 \cup \mathcal{A}_2)$ and $\mathcal{A}_4 = V \setminus (\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3)$ for $R = e^{\frac{\kappa^3 pn}{80m}} \frac{1}{4\sigma n^2}$.

To establish a rapid enough decay of the Fourier spectrum, we require that the columns of A satisfy the (AC) property from Definition 4.3.3. To gain a bit of intuition about Definition 4.3.3, recall that we are working in the Fourier domain. If X is a column of A, it is natural to require that $\langle \theta, X \rangle$ be bounded away from integer points. Otherwise, $\langle \theta, D \rangle$ could be close to an integer point with high probability, making $|\hat{D}(\theta)|$ large. An extra component in the definition says that the anti-concentration continues to hold after conditioning on an arbitrary half-space, passing through the mean. As will become apparent, this is a consequence of our subsampling step.

Let us just note that the (AC) property is not vacuous. In fact Theorems 4.3.5 and 4.3.7 are a direct consequence of the following lemma (see the proof in Section 4.5) and Theorems 4.3.4 and 4.3.6 (note that by Remark 4.3.2 both cases are approximately symmetric).

Lemma 4.4.1. *Suppose that for $X = (X_1, \ldots, X_m) \sim \mathcal{D}$.*

1. *If X is logconcave and isotropic, then X is $(1, \frac{1}{50})$-anti-concentrated.*
2. *If X_i are* i.i.d. *uniformly on an integer interval $\{a, a+1, \ldots, a+k\}$, with $k > 1$, then X is $(k, \frac{1}{50})$-anti-concentrated.*

One may wonder about the necessity of the condition $k > 1$ in Case 2 of Lemma 4.4.1. A moment of reflection reveals that, if X is uniform on $\{0, 1\}$, then X is not anti-concentrated, for any $\kappa > 0$, and thus our framework does not directly apply to this case. However, by taking account of the possible bad cases, our analysis can be refined to also handle such distributions. We do not pursue this direction here.

Preprocessing step: Subsampling

We will generate a sub-matrix of A by selecting a subset of the columns. This will ensure that the norm of the columns is bounded, as well as that the norm of their sum is small. Suppose that the columns of A satisfy (AC) with $\kappa, \sigma > 0$, for $i = 1, \ldots, n$

we define random variables $Y_i \in \{0,1\}$:

$$\Pr[Y_{k+1} = 1 | A_1, Y_1, \dots A_k, Y_k] = \begin{cases} 1 & \text{if } \langle \sum_{j=1}^{k} Y_j \left(A_j - \mu\right), A_{k+1} - \mu \rangle < 0 \\ & \text{and } \|A_{k+1} - \mu\| \leq 10\sigma\sqrt{\frac{m}{\kappa}} \\ \frac{1}{2} & \text{if } \langle \sum_{j=1}^{k} Y_j \left(A_j - \mu\right), A_{k+1} - \mu \rangle = 0 \\ & \text{and } \|A_{k+1} - \mu\| \leq 10\sigma\sqrt{\frac{m}{\kappa}} \\ 0 & \text{else} \end{cases} .$$

We then select all columns of A for which $Y_i = 1$. For now, let A_i' have the law of column A_i, conditional on being selected, and denote the selected set $S_A = \{i \in [n] | Y_i = 1\}$.

Lemma 4.4.2. *Suppose that the columns of A satisfy* (AC) *with parameters $\kappa, \sigma > 0$ and that it is an approximately symmetric distribution, in the sense of Definition 4.3.1. Then, if $n \gg \frac{m^2}{\kappa}$:*

1.
$$\|A_i' - \mu\| \leq 10\sigma\sqrt{\frac{m}{\kappa}}. \qquad \text{(norm concentration)}$$

2. *With probability $1 - e^{-\Omega(n)}$, $|S_A| \geq \frac{n}{200}$,*
$$\sum_{i \in S_A} (A_i - \mu)(A_i - \mu)^\top \preccurlyeq 2n\sigma^2 I_m, \quad \textit{and} \qquad \text{(matrix concentration)}$$
$$\left\| \sum_{i \in S_A} A_i - \mu \right\|^2 \leq 2nm\sigma^2. \qquad \text{(concentration)}$$

3. $\Pr\left(d(\theta^\top A_i', \mathbb{Z}) \geq \kappa \min\left(1, \|\theta\|_\infty \sigma\right) | A_1 Y_1, \dots, A_{i-1} Y_{i-1}\right) \geq \frac{\kappa}{2}$, *for every $\theta \in V$, and $i \in [n']$.*

Proof. The first claim is immediate since we have conditioned the columns on the event $\{\|A_i - \mu\| \leq 10\sigma\sqrt{\frac{m}{\kappa}}\}$. To show that (matrix concentration) holds, let $Z_i \overset{\text{law}}{=} A_i | \left(\|A_i - \mu\| \leq 10\sigma\sqrt{\frac{m}{\kappa}}\right)$ and note,

$$\sum_{i \in S_A} (A_i - \mu)(A_i - \mu)^\top = \sum_{i=1}^{n} Y_i (Z_i - \mu)(Z_i - \mu)^\top \leq \sum_{i=1}^{n} (Z_i - \mu)(Z_i - \mu)^\top.$$

As the random vectors $\{Z_i - \mu\}_{i=1}^{n}$ are mutually independent and $(Z_i - \mu)(Z_i - \mu)^\top \preceq 10\sigma\sqrt{\frac{m}{\kappa}} I_m$ almost surely, (matrix concentration) follows from the matrix Bernstein inequality [Tro15, Theorem 1.6.2].

For (concentration), since $\langle \sum_{j=1}^{i-1} Y_j(A_j-\mu), A_i-\mu\rangle \leq 0$,

$$\left\|\sum_{i\in S_A} A_i-\mu\right\|^2 \leq \sum_{1\in S_A}\|A_i-\mu\|^2 = \mathrm{Tr}\left(\sum_{i\in S_A}^{n}(A_i-\mu)(A_i-\mu)^{\mathsf{T}}\right)$$
$$\leq 2n\mathrm{Tr}\left(\sigma^2 I_m\right),$$

where the last inequality is monotonicity of the trace. Recalling that $\mathrm{Cov}(A_i)\preceq \sigma^2 I_d$, the fact that with high probability $|S_A|\geq \frac{n}{8}$ follows from Azuma's inequality. Indeed, for fixed $i\in[n]$, by Chebyshev's inequality,

$$\Pr\left(\|A_i-\mu\|>10\sigma\sqrt{\frac{m}{\kappa}}\right)\leq \frac{\kappa\,\mathbb{E}\left[\|A_i-\mu\|^2\right]}{100\sigma^2 m} = \frac{\kappa\,\mathrm{Tr}\left(\mathrm{Cov}(A_i)\right)}{100\sigma^2 m}\leq \frac{\kappa}{100}.$$

Since A_i has an approximately symmetric law, then, since A_i is independent from $\{Y_j, A_j\}_{j=1}^{i-1}$, by definition,

$$\Pr\left(\langle\sum_{j=1}^{i-1} Y_j(A_j-\mu), A_i-\mu\rangle \leq 0\right)\geq \frac{1}{4e^2}.$$

Taken together, the above displays imply

$$\Pr\left(Y_i=1|A_1,\dots,A_{i-1}\right)\geq \frac{1}{100}.$$

Applying Azuma's inequality, as in (2.2), we get

$$\Pr\left(|S_A|\geq \frac{n}{200}\right)=1-e^{-\Omega(n)}.$$

Finally, we address the (AC) property. For fixed $i\in[n']$, let us define $\nu=\sum_{j=1}^{i-1}A_jY_j$. So,

$$\begin{aligned}&\Pr\left(d(\theta^{\mathsf{T}}A_i',\mathbb{Z})\geq \kappa\min\left(1,\|\theta\|_\infty\sigma\right)|A_1Y_1,\ldots,A_{i-1}Y_{i-1}\right)\\ &\quad=\Pr\left(d(\theta^{\mathsf{T}}A_i,\mathbb{Z})\geq \kappa\min\left(1,\|\theta\|_\infty\sigma\right)|\langle\nu,A_i-\mu\rangle\leq 0 \text{ and } \|A_i-\mu\|\leq 10\sigma\sqrt{\frac{m}{\kappa}}\right)\\ &\quad\geq \Pr\left(d(\theta^{\mathsf{T}}A_i,\mathbb{Z})\geq \kappa\min\left(1,\|\theta\|_\infty\sigma\right)|\langle\nu,A_i-\mu\rangle\leq 0\right)-\frac{\kappa}{100}\geq\frac{\kappa}{2}.\end{aligned}$$

Here, the last inequality follows from Definition 4.3.3, while the first inequality is a union bound on the anti-concentration event and $\{\|A_i-\mu\|\leq 10\sigma\sqrt{\frac{m}{\kappa}}\}$. □

In light of the lemma, in the sequel, all computations will be made conditioned on the high-probability event defined by Lemma 4.4.2, and we will only consider the first $\lceil\frac{n}{200}\rceil$ selected columns. Thus, with a slight abuse of notation, from now on, the random variables D, $\hat{D}$, A_i, etc., will only be considered with respect to the selected columns. In particular, we will write n for $|S_A|$.

Step I: bounding the argument

We will now show that for small θ, the argument of $\hat{D}(\theta)$ is close to $2\pi np\langle\theta,\mu\rangle$. The main observation in this step, is that, when we fix the columns $\{A_j\}_{j=1}^n$, a Taylor approximation implies the bound,

$$\left|\sum_{j=1}^n \arg\bigl(\mathbb{E}_S[\exp(\mathbf{1}_{i\in S}2\pi i A_j)]\bigr) - 2\pi np\langle\theta,\mu\rangle\right|$$
$$\leq 2\pi p\left|\sum_{j=1}^n\langle\theta,A_j\rangle - n\langle\theta,\mu\rangle\right| + 50p\sum_{j=1}^n|\langle\theta,A_j\rangle|^3 .$$

The term on the LHS controls $\arg(\hat{D}(\theta))$ and the two terms on the RHS can be bounded with (concentration) and (norm concentration) respectively. We then prove:

Lemma 4.4.3. *With probability $1-e^{-\Omega(n)}$ over A, for all $\|\theta\|$ we have,*

$$|\arg(\hat{D}(\theta)) - 2\pi np\langle\theta,\mu\rangle| \leq 4\pi p\|\theta\|\sqrt{nm}\sigma + 50p\|\theta\|^3 n\left(10\sigma\sqrt{m/\kappa}+\|\mu\|\right)^3.$$

Proof. Let $f(x) = \arg(p\cdot\exp(2\pi i\cdot x)+(1-p))$ and observe that $f(x) = \arctan\left(\frac{p\sin(2\pi x)}{p\cos(2\pi x)+(1-p)}\right)$. A calculation shows $f(0)=0$, $f'(0)=2\pi p$ and $f''(0)=0$. Hence, we set $g(x)=f(x)-2\pi px$ and, with a second-order Taylor approximation of $f(x)$ around $x=0$, we see that for every $x\geq 0$ there is some $x'\in[0,x]$ such that

$$g(x) = \frac{d^3f}{dx^3}(x')x'^3.$$

Another calculation shows that, as long as $p\leq 0.1$, we have $\frac{d^3f}{dx^3}(x')\leq 50p$ and hence $|g(x)|\leq 50|x|^3p$. Now, note,

$$\arg(\mathbb{E}[\exp(\mathbf{1}_{j\in S}2\pi ix)]) = \arg(p\cdot\exp(2\pi i\cdot x)+(1-p)) = f(x).$$

Now,

$$\begin{aligned}\left|\arg(\hat{D}(\theta)) - 2\pi np\langle\theta,\mu\rangle\right| &= \left|\arg(\prod_{j=1}^{n}\mathbb{E}[\exp(\mathbf{1}_{j\in S}2\pi i\langle\theta,A_j\rangle)]) - 2\pi np\langle\theta,\mu\rangle\right| \\ &= \left|\sum_{j=1}^{n}\arg(\exp(\mathbf{1}_{j\in S}2\pi i\langle\theta,A_j\rangle)) - 2\pi np\langle\theta,\mu\rangle\right| \\ &= \left|\sum_{j=1}^{n}f(\langle\theta,A_j\rangle) - 2\pi np\langle\theta,\mu\rangle\right|,\end{aligned}$$

where we understand $|\cdot|$ as referring to distance on the circle. The Taylor approximation given above shows that, when $\|\theta\| \le r$, we can bound this distance (where now the bound will be expressed as a distance between real numbers),

$$\begin{aligned}\left|\sum_{j=1}^{n}f(\langle\theta,A_j\rangle) - 2\pi np\langle\theta,\mu\rangle\right| &\le 2\pi p\left|\sum_{j=1}^{n}\langle\theta,A_j\rangle - n\langle\theta,\mu\rangle\right| + 50p\sum_{j=1}^{n}|\langle\theta,A_j\rangle|^3 \\ &\le 2\pi p\left|\left\langle\theta,\sum_{j=1}^{n}(A_j-\mu)\right\rangle\right| + 50p\|\theta\|^3\sum_{j=1}^{n}\|A_j\|_2^3 \\ &\le 2\pi p\|\theta\|\left\|\sum_{j=1}^{n}(A_j-\mu)\right\| + 50p\|\theta\|^3 n\left(10\sigma\sqrt{m/\kappa}+\|\mu\|\right)^3 \\ &\le 4\pi p\|\theta\|\sqrt{nm}\sigma + 50p\|\theta\|^3 n\left(10\sigma\sqrt{m/\kappa}+\|\mu\|\right)^3\end{aligned}$$

where the third inequality follows from (norm concentration) and the final inequality being a consequence of (concentration). □

The following corollary is now immediate.

Corollary 4.4.4. *With probability* $1-e^{-\Omega(n)}$, *we have:*

$$\Re\left[\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle)\right] \ge \frac{1}{\sqrt{2}}|\hat{D}(\theta)|,$$

for all θ *with* $\|\theta\|_2 \le \hat{r}$ *where*

$$\hat{r} = \min\left(\frac{1}{64p\sqrt{nm}\sigma}, \frac{1}{8\sqrt[3]{pn}\left(10\sigma\sqrt{m/\kappa}+\|\mu\|\right)}, \frac{1}{16\|t-pn\mu\|}\right).$$

Proof. From Lemma 4.4.3 we see that for all $\|\theta\| \leq \hat{r}$:

$$|\arg(\hat{D}(\theta)) - 2\pi np\langle\theta,\mu\rangle| \leq 4\pi p\hat{r}\sqrt{nm}\sigma + 50p\hat{r}^3 n\left(10\sigma\sqrt{m/\kappa} + \|\mu\|\right)^3.$$

Plugging in the upper bounds on r and r^3, we get:

$$|\arg(\hat{D}(\theta)) - 2\pi np\langle\theta,\mu\rangle| \leq \frac{1}{16}\pi + \frac{1}{16}\pi \leq \frac{1}{8}\pi.$$

Hence,

$$\begin{aligned}|\arg(\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle))| &\leq |\arg(\hat{D}(\theta)) - 2\pi np\langle\theta,\mu\rangle| \\ &\quad + |\arg(\exp(2\pi\langle\theta, t - pn\mu\rangle i))| \\ &\leq \frac{1}{8}\pi + 2\pi\cdot\hat{r}\cdot\frac{1}{16\hat{r}} \leq \frac{1}{4}\pi.\end{aligned}$$

This implies that:

$$\Re[\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle)] \geq \cos(\pi/4)|\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle)|,$$

which proves the corollary since $\cos\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}}$. □

Step II: bounding the integral from below, near the origin

Next, we prove that the modulus of $\hat{D}$ is bounded from below, near the origin.

Lemma 4.4.5. *The following holds,*

$$\int_{\|\theta\|\leq r} |\hat{D}(\theta)|d\theta \geq \frac{1 - \xi_m(48\pi^2\sigma^2 npr^2)}{(10^2\sqrt{np}\sigma)^m\sqrt{\zeta+1}},$$

where $\xi_m(t) = \Pr[\|G\|^2 \geq t]$ for $G \sim \mathcal{N}(0, I_m)$.

Proof. We have:

$$\begin{aligned}|\hat{D}(\theta)| &= \prod_{j=1}^{n} |\mathbb{E}[\exp(2\pi i\langle\theta,A_j\rangle)]| = \prod_{j=1}^{n} |(1-p)\cdot 1 + p\exp(2\pi i\langle\theta,A_j\rangle)| \\ &= \prod_{j=1}^{n}\sqrt{(1 - p + p\cos(2\pi\langle\theta,A_j\rangle))^2 + p^2\sin(2\pi\langle\theta,A_j\rangle)^2} \\ &\geq \prod_{j=1}^{n}(1 - p + p\cos(2\pi\langle\theta,A_j\rangle)) \geq \exp(-6\pi^2 p\sum_{i=1}^{n}\langle\theta,A_i\rangle^2).\end{aligned}$$

Here the last inequality follows, as long as $p \leq 0.01$, from the elementary inequalities,

$$\begin{aligned}\cos(x) &\geq 1 - x^2 \\ \ln(1-x) &\geq -\frac{3}{2}x \quad \text{when } |x| \leq \frac{1}{2}.\end{aligned}$$

Indeed, it's enough to consider $\langle \theta, A_j \rangle \in [-1, 1]$, for which,

$$\ln(1 - p + p\cos(2\pi\langle\theta, A_j\rangle)) \geq \ln\left(1 - p4\pi^2\langle\theta, A_j\rangle^2\right) \geq -6\pi^2 p\langle\theta, A_j\rangle^2.$$

Hence, by applying the (matrix concentration) property to the obtained bound, we get,

$$\begin{aligned}|\hat{D}(\theta)| &\geq \exp\left(-12\pi^2 p\left(n\langle\theta,\mu\rangle^2 + \sum_{i=1}^{n}\langle\theta, A_i - \mu\rangle^2\right)\right) \\ &\geq \exp\left(-24\pi^2 np\left(\langle\theta,\mu\rangle^2 + \sigma^2\theta\theta^{\mathsf{T}}\right)\right) \\ &\geq \exp\left(-24\pi^2\theta\left(\mu^{\mathsf{T}}\mu + \sigma^2 I_m\right)\theta^{\mathsf{T}} np\right).\end{aligned}$$

Let $Y \sim \mathcal{N}\left(0, \frac{1}{48\pi^2 np}(\mu^{\mathsf{T}}\mu + \sigma^2 I_m)^{-1}\right)$ and $Y' \sim \mathcal{N}\left(0, \frac{1}{48\pi^2\sigma^2 np} I_m\right)$, then,

$$\begin{aligned}\int_{\|\theta\|\leq r}|\hat{D}(\theta)|d\theta &\geq \int_{\|\theta\|\leq r}\exp\left(-24\pi^2\theta\left(\mu^{\mathsf{T}}\mu + \sigma^2 I_m\right)\theta^{\mathsf{T}} np\right)d\theta \\ &= \frac{1}{\sqrt{\det\left(96\pi^3 np\left(\mu^{\mathsf{T}}\mu + \sigma^2 I_m\right)\right)}}\Pr\left[\|Y\| \leq r\right] \\ &= \frac{1}{(96\pi^3 np)^{\frac{m}{2}}\left(\|\mu\|^2 + \sigma^2\right)^{\frac{1}{2}}\sigma^{m-1}}\Pr\left[\|Y\| \leq r\right] \\ &\leq \frac{1}{(96\pi^3 np)^{\frac{m}{2}}\left(\|\mu\|^2 + \sigma^2\right)^{\frac{1}{2}}\sigma^{m-1}}\Pr\left[\|Y'\| \leq r\right] \\ &\leq \frac{\Pr\left[\|Y'\|^2 \leq r^2\right]}{(96\pi^3 np)^{\frac{m}{2}}\left(\|\mu\|^2 + \sigma^2\right)^{\frac{1}{2}}\sigma^{m-1}} \\ &= \frac{1 - \xi_m(48\pi^2\sigma^2 npr^2)}{(96\pi^3 np)^{\frac{m}{2}}\left(\|\mu\|^2 + \sigma^2\right)^{\frac{1}{2}}\sigma^{m-1}}.\end{aligned}$$

□

Step III: Exponential decay of the Fourier spectrum

To show that the Fourier spectrum decays rapidly, we employ an ε-net argument over a very large box. The main difficulty is that $\langle\theta, D\rangle$ can be close to an integer, irregardless of the value $\|\theta\|$. Our anti-concentration assumption allows us to avoid this. Specifically, Item 3 in Lemma 4.4.2 implies that for any given θ in a dense enough net, we can expect many columns to satisfy that $\langle\theta, A_i\rangle$ is far from any integer point. Formally, we prove:

Lemma 4.4.6. *With probability* $1-e^{-\Omega(\kappa n)}$, *we have*

$$|\hat{D}(\theta)| \leq \exp\left(-\frac{1}{80}\kappa^3 n(1-p)p\pi^2 \min\left(1, \|\theta\|_\infty^2\sigma^2\right)\right),$$

for $\theta \in \tilde{V}$, *where* V *is the fundamental domain, as in* (4.2), *and*

$$\tilde{V} := [-R, R]^m \cap V.$$

Proof. We begin with a technical calculation which will help control the size of the net. Observe that (4.4) implies

$$160m \leq \sqrt{\kappa n} \implies \sqrt{\frac{m}{\kappa}} \leq \frac{1}{2}\sqrt{\frac{\kappa n}{80m}} \implies \sqrt{\frac{m}{\kappa}} \leq \frac{1}{2}e^{\frac{\kappa n}{80m}},$$

and similarly,

$$\frac{\|\mu\|}{\sigma} \leq \frac{1}{2}e^{\frac{\kappa n}{80m}}.$$

Now, let N be an ε-net of $\tilde{V}$ for $\varepsilon = \frac{1}{2n^2\left(\sigma\sqrt{m/\kappa}+\|\mu\|\right)}$. Standard arguments show that one can take $|N| \leq \left(\sqrt{\frac{m}{\kappa}} + \frac{\|\mu\|}{\sigma}\right)^m e^{\frac{\kappa}{80}n} \leq e^{\frac{\kappa}{40}n}$, when n is large enough. Here we have used the bound, $\sqrt{\frac{m}{\kappa}} + \frac{\|\mu\|}{\sigma} \leq e^{\frac{\kappa n}{80m}}$, established above.

For $\theta \in N$, define

$$\varphi(\theta) = \frac{\kappa}{4}\min\left(1, \|\theta\|_\infty\sigma\right) \text{ and } E(\theta) = \{j : d(\langle\theta, A_j\rangle, \mathbb{Z}) \geq \varphi(\theta)\}.$$

If we fix θ and set $X_i := \mathbf{1}_{i\in E(\theta)}$ then $|E(\theta)| = \sum\limits_{i=1}^{n} X_i$. The statement of Item 3 in Lemma 4.4.2 is $\mathbb{E}\left[X_i | X_1, \dots, X_{i-1}\right] \geq \frac{\kappa}{2}$. Applying Azuma's inequality (2.2), we get:

$$\Pr\left[|E(\theta)| \leq \frac{\kappa}{4}n\right] \leq \exp\left(-\frac{\kappa}{8}n\right).$$

In this case, by the union bound,

$$\mathbb{P}\left(\exists\theta \in N \ : \ |E(\theta)| \leq \frac{\kappa}{4}n\right) \leq e^{-\frac{\kappa}{16}n}.$$

If $j \in E(\theta)$, since $\varphi(\theta) \leq \frac{1}{4}$, we have:

$$\begin{aligned}
|\mathbb{E}_S[\exp(\mathbf{1}_{j\in S} \cdot 2\pi i\langle\theta, A_j\rangle)]| &= \sqrt{(1-p+p\cos(2\pi\langle\theta, A_j\rangle))^2 + p^2\sin(2\pi\langle\theta, A_j\rangle)^2} \\
&= \sqrt{1+2p^2-2p+2(1-p)p\cos(2\pi\langle\theta, A_j\rangle)} \\
&\leq \sqrt{1+2p^2-2p+2(1-p)p\cos(2\pi\varphi(\theta))} \\
&\leq \sqrt{1-2(1-p)p(2\pi\varphi(\theta))^2/5} \\
&\leq 1-(1-p)p(2\pi\varphi(\theta))^2/5 \\
&\leq 1-\frac{4}{5}(1-p)p\pi^2\frac{\kappa^2}{16}\min\left(1, \|\theta\|_\infty^2\sigma^2\right).
\end{aligned}$$

Observe that $|\mathbb{E}_S[\exp(\mathbf{1}_{j\in S}2\pi i x)]| = \sqrt{(1-p+p\cos(2\pi x))^2 + p^2\sin(2\pi x)^2}$ is $4\pi p$-Lipschitz in x, as long as $p \leq \frac{1}{4}$. Take an arbitrary $\theta \in \tilde{V}$ and let θ' be the closest point in N. Recall that $\max_i \|A_i\| \leq 10\sigma\sqrt{\frac{m}{\kappa}} + \|\mu\|$, which follows from Eq. (norm concentration). So, by our choice of ε and with the Cauchy-Schwartz inequality,

$$|\langle\theta-\theta', A_j\rangle| \leq \frac{1}{2n^2}.$$

Thus:

$$\begin{aligned}
|\hat{D}(\theta)| &= \prod_{j=1}^{n} |\mathbb{E}_S[\exp(\mathbf{1}_{j\in S} 2\pi i\langle\theta, A_j\rangle)]| \\
&\leq \prod_{j\in E(\theta')} \left(\left|\mathbb{E}_S[\exp(\mathbf{1}_{j\in S} 2\pi i\langle\theta', A_j\rangle)]\right| + 4\pi p|\langle\theta-\theta', A_j\rangle|\right) \\
&\leq \prod_{j\in E(\theta')} \left(1 - \frac{4}{5}(1-p)p\pi^2\frac{\kappa^2}{16}\min\left(1, \|\theta'\|_\infty^2\sigma^2\right) + 4\pi p|\langle\theta-\theta', A_j\rangle|\right) \\
&\leq \exp\left(-\frac{4}{5}(1-p)p\pi^2\frac{\kappa^2}{16}|E(\theta')|\min\left(1, \|\theta'\|_\infty^2\sigma^2\right) + 4\pi p\sum_{j=1}^{n}|\langle\theta-\theta', A_j\rangle|\right) \\
&\leq e^{\frac{2\pi}{n}}\exp\left(-\frac{1}{5}\frac{\kappa^3}{16}n(1-p)p\pi^2\min\left(1, \|\theta'\|_\infty^2\sigma^2\right)\right) \\
&\leq e^{\frac{2\pi}{n}}\exp\left(-\frac{1}{5}\frac{\kappa^3}{16}n(1-p)p\pi^2\min\left(1, (1-\varepsilon)^2\|\theta\|_\infty^2\sigma^2\right)\right) \\
&\leq \exp\left(-\frac{1}{80}\kappa^3 n(1-p)p\pi^2\min\left(1, \|\theta\|_\infty^2\sigma^2\right)\right),
\end{aligned}$$

where the last inequality holds, since by (4.4), $\kappa^3 pn \geq 10^{20}$. □

By properly integrating the inequality, we have thus obtained:

Lemma 4.4.7. *With probability* $1 - e^{-\Omega(\kappa n)}$*, the following inequality holds:*

$$\int\limits_{\mathcal{A}_2\cap V} |\hat{D}(\theta)|d\theta \leq \left(\frac{5m}{\kappa^3 pn\sigma^2}\right)^{\frac{m}{2}} \xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right)$$

where

$$\mathcal{A}_2 = \left\{\|\theta\|_\infty < \min\left(e^{\frac{\kappa^3 pn}{80m}}\frac{1}{4\sigma n^2}, \frac{1}{\sigma}\right)\right\} \cap \{\|\theta\| \geq r\}$$

and $\xi_m(t) = \Pr[\|G\|^2 \geq t]$ *for* $G \sim \mathcal{N}(0, I_m)$.

Proof. From Lemma 4.4.6, when $p < 0.1$, and $\sigma\|\theta\|_\infty \leq 1$, we have

$$|D(\theta)| \leq \exp\left(-\frac{\kappa^3 pn\sigma^2\|\theta\|^2}{10m}\right).$$

Now, if $Y \sim \mathcal{N}(0, I_m)$:

$$\begin{aligned}\int\limits_{\mathcal{A}_2} |D(\theta)| d\theta &\le \int\limits_{\|\theta\|\ge r} \exp\left(-\frac{\kappa^3 pn\sigma^2\|\theta\|^2}{10m}\right) d\theta \\ &= \left(\frac{5m}{\kappa^3 pn\sigma^2}\right)^{\frac{m}{2}} \xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right).\end{aligned}$$

□

Lemma 4.4.8. *With probability* $1 - e^{-\Omega(\kappa n)}$, *the following inequality holds:*

$$\int\limits_{\mathcal{A}_3\cap V} |\hat{D}(\theta)| d\theta \le \frac{1}{\sigma^m n^{2m}} \exp\left(-\frac{\kappa^3 pn}{80}\right),$$

where $\mathcal{A}_3 \subseteq [-R, R]^m \setminus [\frac{1}{\sigma}, \frac{1}{\sigma}]^m$ *for* $R = e^{\frac{\kappa^3 pn}{80m}} \frac{1}{4\sigma n^2}$.

Proof. For all $\theta \in \left[-\frac{e^{\frac{\kappa^3 pn}{80m}}}{4\sigma n^2}, \frac{e^{\frac{\kappa^3 pn}{80m}}}{4\sigma n^2}\right]^m$ with $\|\theta\|_\infty \sigma \ge 1$, Lemma 4.4.6 implies,

$$|D(\theta)| \le \exp\left(-\frac{\kappa^3 pn}{40m}\right).$$

So:

$$\begin{aligned}\int\limits_{\mathcal{A}_3} |D(\theta)| d\theta &\le \int\limits_{\frac{e^{\frac{\kappa^3 pn}{80m}}}{2\sigma n^2} \ge \|\theta\|_\infty} \exp\left(-\frac{\kappa^3 pn}{40m}\right) d\theta \\ &= \frac{1}{\sigma^m n^{2m}} \exp\left(\frac{\kappa^3 pn}{80m}\right)^m \cdot \exp\left(-\frac{\kappa^3 pn}{40}\right) \\ &= \frac{1}{\sigma^m n^{2m}} \exp\left(-\frac{\kappa^3 pn}{80}\right).\end{aligned}$$

□

Proving Theorem 4.3.4

Now we will finally prove Theorems 4.3.4 and 4.3.6. First we prove the following lemma that provides a lower bound on the integral from the Fourier Inversion Theorem (Theorem 4.2.1).

For the continuous case we require one extra step to control the Fourier transform outside the domain of Lemma 4.4.7. To deal with continuous distributions, we define $G \sim \mathcal{N}(0, \gamma I_m)$ for γ to be determined later and $H = D + G$. For the discrete case we simply define $G = 0$.

Note that we have $\hat{R}(\theta) = g(\|\theta\|)$, where in the continuous case we have $g(\|\theta\|) = e^{-\frac{\|\theta\|^2\gamma}{2}}$, whereas in the discrete case we have $g(\|\theta\|) = 1$. In both cases g is a decreasing function on $\mathbb{R}_{\geq 0}$. By the multiplication-convolution theorem (Theorem 4.2.2), we have $\hat{H}(\theta) = \hat{D}(\theta) \cdot \hat{R}(\theta) = g(\|\theta\|)\hat{D}(\theta)$.

We will now split up the domain of integration into four sets $\mathcal{A}_1 = B_m(0, r)$, $\mathcal{A}_2 = [-\frac{1}{\sigma}, \frac{1}{\sigma}]^m \backslash \mathcal{A}_1$, $\mathcal{A}_3 = [R, R]^m \backslash (\mathcal{A}_1 \cup \mathcal{A}_2)$, $\mathcal{A}_4 = \{\theta : \|\theta\|_\infty \geq R\} \backslash (\mathcal{A}_1 \cup \mathcal{A}_2)$, for $R = e^{\frac{\kappa^3 pn}{80m}} \frac{1}{4\sigma n^2}$. If we define $\operatorname{val}(S) = \int_{V \cap S} \hat{D}(\theta) \exp(-2\pi i \langle \theta, t \rangle) d\theta$, we have:

$$\Pr[H = t] \geq g(r)(\Re[\operatorname{val}(\mathcal{A}_1)] - |\operatorname{val}(\mathcal{A}_2)| - |\operatorname{val}(\mathcal{A}_3)|) - \int_{\mathcal{A}_4} |\hat{R}(\theta)| d\theta.$$

Choosing r To be able to provide an appropriate lower bound we will choose r such that $\operatorname{val}(\mathcal{A}_2) \leq \frac{1}{8} \operatorname{val}(\mathcal{A}_1)$.

Lemma 4.4.9. *When*

$$r = \sqrt{\frac{15m}{\kappa^3 pn\sigma^2} \left(\ln(16\sqrt{\zeta + 1}) + \frac{m}{2} \ln\left(\frac{10^5 m}{\kappa^3}\right)\right)} \quad \textit{and} \tag{4.5}$$

$$pn \geq \frac{60m}{\kappa^3 \sigma} \left(\ln(16\sqrt{\zeta + 1}) + \frac{m}{2} \ln\left(\frac{10^5 m}{\kappa^3}\right)\right). \tag{4.6}$$

We have:

$$\int_{\mathcal{A}_1} |\hat{D}(\theta)| d\theta = \int_{\|\theta\| \leq r} |\hat{D}(\theta)| d\theta \geq \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta + 1}} \tag{4.7}$$

and in particular $|\operatorname{val}(\mathcal{A}_2)| \leq \frac{1}{8} |\operatorname{val}(\mathcal{A}_1)|$.

Proof. Equation (4.6) implies that $r \leq \frac{1}{2}$, so that $B_m(0, r) \subseteq V$. In Lemma 4.4.5, we have shown that in this case

$$\int_{\mathcal{A}_1} |\hat{D}(\theta)| d\theta = \int_{\|\theta\| \leq r} |\hat{D}(\theta)| d\theta \geq \frac{1 - \xi_m(48\pi^2\sigma^2 npr^2)}{(10^2\sqrt{np}\sigma)^m \sqrt{\zeta + 1}}, \tag{4.8}$$

where $\xi_m(t) = \Pr[\|G\|^2 \geq t]$. for $G \sim \mathcal{N}(0, I_m)$. In Lemma 4.4.7, we have shown that

$$\operatorname{val}(\mathcal{A}_2) \leq \left(\frac{5m}{\kappa^3 pn\sigma^2}\right)^{\frac{m}{2}} \xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right).$$

We will set r such that $\int_{\mathcal{A}_1} |\hat{D}(\theta)|d\theta \geq 8|\operatorname{val}(\mathcal{A}_2)|$. Plugging in the previous bounds, we see that it is enough to set r to satisfy:

$$\frac{\int_{\mathcal{A}_1} |\hat{D}(\theta)|d\theta}{|\operatorname{val}(\mathcal{A}_2)|} \geq \left(\frac{\kappa^3}{10^4 \cdot 5m}\right)^{\frac{m}{2}} \frac{1-\xi_m(48\pi^2\sigma^2 npr^2)}{\sqrt{\zeta+1}\xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right)}$$
$$\geq \left(\frac{\kappa^3}{10^5 m}\right)^{\frac{m}{2}} \frac{1-\xi_m(48\pi^2\sigma^2 npr^2)}{\sqrt{\zeta+1}\xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right)} \geq 8.$$

Rewriting the last inequality we see that we need to pick r to satisfy:

$$\frac{1-\xi_m(48\pi^2\sigma^2 npr^2)}{\xi_m\left(\frac{r^2\kappa^3 pn\sigma^2}{5m}\right)} \geq 8\sqrt{\zeta+1}\left(\frac{10^5 m}{\kappa^3}\right)^{\frac{m}{2}}. \tag{4.9}$$

Observe that by the Markov bound we have $\xi_m(2m) \leq \frac{\mathbb{E}[G]}{2m} = \frac{m}{2m} = \frac{1}{2}$. So, in order to keep the numerator of the left hand side at least $\frac{1}{2}$:

$$r \geq \frac{m}{8\pi\sigma\sqrt{np}}. \tag{4.10}$$

To lower bound the denominator, we note that by Lemma 2.3.3 $\xi_m(x) \leq \exp(-x/3)$ as long as $x \geq 7m$. This implies that to satisfy Eq. (4.9), we can set

$$r = \sqrt{\frac{15m}{\kappa^3 pn\sigma^2}\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)}. \tag{4.11}$$

Observe that this choice of r does indeed satisfy Eq. (4.10). □

Lemma 4.4.10. *We have* $\Re[\operatorname{val}(\mathcal{A}_1)] \geq \frac{1}{\sqrt{2}}|\operatorname{val}(\mathcal{A}_1)|$, *as long as*

$$p \leq \frac{\kappa^3}{2^{16}m^2\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)}$$
$$pn \geq \frac{10^{10}m^6(\zeta+1)^3}{\kappa^{12}}\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)^3$$
$$\|t - pn\mu\| \leq \sqrt{\frac{\kappa^3 pn\sigma^2}{2^{12}m\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)}}.$$

Proof. As in Corollary 4.4.4, let:

$$\hat{r} = \min\left(\frac{1}{64p\sqrt{nm}\sigma}, \frac{1}{8\sqrt[3]{pn}\sigma\left(10\sqrt{m/\kappa}+\sqrt{\zeta}\right)}, \frac{1}{16\|t-pn\mu\|}\right).$$

When all assumptions are satisfied, we have $r \le \hat{r}$ and hence $\mathcal{A}_1 \subseteq B(0,\hat{r})$. Then by Corollary 4.4.4 we have:

$$\begin{aligned}\Re[\mathrm{val}(\mathcal{A}_1)] &= \int_{\{\|\theta\|\le r\}\cap S} \Re\left[\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle)\right]d\theta \\ &\ge \frac{1}{\sqrt{2}}\int_{\{\|\theta\|\le r\}\cap S}|\hat{D}(\theta)\exp(-2\pi i\langle\theta,t\rangle)|d\theta \ge \frac{1}{\sqrt{2}}|\mathrm{val}(\mathcal{A}_1)|.\end{aligned}$$

□

Lemma 4.4.11. *We have* $|\mathrm{val}(\mathcal{A}_3)| \le \frac{1}{8}|\mathrm{val}(\mathcal{A}_1)|$, *as long as* $n \ge 100$ *and* $pn \ge 8\kappa^{-3}\ln(\zeta+1)$.

Proof. By Lemma 4.4.8 we have:

$$|\mathrm{val}(\mathcal{A}_3)| \le \left(\frac{1}{\sigma^2 n^4}\right)^{\frac{m}{2}}\exp\left(-\frac{\kappa^3 pn}{80}\right).$$

By Eq. (4.8), we have $|\mathrm{val}(\mathcal{A}_1)| \ge \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}}\frac{1}{2\sqrt{\zeta+1}}$. So if $n \ge 100$ and $pn \ge 8\kappa^{-3}\ln(\zeta+1)$, we have:

$$|\mathrm{val}(\mathcal{A}_3)| \le \left(\frac{1}{10^6\sigma^2 n}\right)^{\frac{m}{2}}\frac{1}{2\sqrt{\zeta+1}}.$$

This proves the result. □

Bounding the integral over $\mathcal{A}_4$ In the discrete setting we will bound the integral over $\mathcal{A}_4$ by showing that $\mathcal{A}_4 \cap V = \emptyset$. To bound $\mathrm{val}(\mathcal{A}_4)$ in the continuous setting we put $\gamma = n/R^2$.

Lemma 4.4.12. *In the continuous setting, we have* $\int_{\mathcal{A}_4}|\hat{R}(\theta)|d\theta \le \frac{1}{8}g(r)|\mathrm{val}(\mathcal{A}_1)|$ *and* $g(r) \ge \frac{1}{2}$, *as long as* $pn \ge \frac{40m}{\kappa^3}\ln(\frac{10^6 n^4 m^3\sqrt{\zeta+1}}{\kappa^6 p})$ *and* $n \ge \max(7m, 10^3\ln(2(\zeta+1)))$.

Proof. We have:

$$\begin{aligned}
\int\limits_{\mathcal{A}_4} |\hat{R}(\theta)|d\theta &= \int\limits_{\mathcal{A}_4} |g(\|\theta\|)\hat{D}(\theta)|d\theta \leq \int\limits_{\mathcal{A}_4} |\hat{R}(\theta)|d\theta = \int\limits_{\mathcal{A}_4} g(\|\theta\|)d\theta \\
&\leq \int\limits_{\|\theta\|\geq R} e^{\frac{-\gamma\|\theta\|^2}{2}} d\theta = \left(\frac{2\pi}{\gamma}\right)^{\frac{m}{2}} \psi_m\left(\gamma R^2\right) = (2\pi R^2)^{m/2}\psi_m(n) \\
&\leq (2\pi R^2)^{m/2} e^{-n/3} \qquad \text{(by Lemma 2.3.3)} \\
&= \left(\frac{2\pi}{16\sigma^2 n^5}\right)^{\frac{m}{2}} e^{\frac{\kappa^3 pn}{80} - \frac{n}{3}} \\
&\leq \left(\frac{1}{2\sigma^2 n^5}\right)^{\frac{m}{2}} e^{-\frac{1}{6}n}.
\end{aligned}$$

By Eq. (4.8) we have:

$$|\operatorname{val}(\mathcal{A}_1)| \geq \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.$$

This implies that for $n \geq 10^3 \ln(2(\zeta+1))$, $\int_{\mathcal{A}_4} |\hat{R}(\theta)|d\theta \leq \frac{1}{16}|\operatorname{val}(\mathcal{A}_1)|$. Note that $g(r) = \exp(-n \cdot \frac{r^2}{R^2})$. When, $pn \geq \frac{40m}{\kappa^3} \ln(\frac{10^6 n^4 m^3 \sqrt{\zeta+1}}{\kappa^6 p})$ we have:

$$\begin{aligned}
n \cdot \frac{r^2}{R^2} &\leq \exp(-\frac{\kappa^3 pn}{40m})\frac{10^3 n^4 m}{\kappa^3 p}\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right) \\
&\leq \exp(-\frac{\kappa^3 pn}{40m})\frac{10^5 n^4 m^3 \sqrt{\zeta+1}}{\kappa^6 p} \leq \frac{1}{2}.
\end{aligned}$$

Hence, $g(r) \geq \exp(-\frac{1}{2}) \geq \frac{1}{2}$, which proves the lemma. □

Putting it all together

Lemma 4.4.13. *We have:*

$$\Pr\left[|S| \notin \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right] \leq 2\exp(-pn/12).$$

Proof. Using the multiplicative Chernoff bound we can see that:

$$\Pr\left[|S| \notin \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right] = \Pr\left[||S| - \mathbb{E}[|S|]| \geq \frac{1}{2}\,\mathbb{E}[|S|]\right] \leq 2\exp(-pn/12).$$

□

Proof of Theorem 4.3.4. Since the columns of A are now absolutely continuous we have $V = \mathbb{R}^m$. From Eqs. (4.3) and (4.4) it follows that the conditions of Lemmas 4.4.9 to 4.4.13 are satisfied. Hence, we have:

$$\begin{aligned}\Re[\operatorname{val}(\mathcal{A}_1)] - |\operatorname{val}(\mathcal{A}_2)| - |\operatorname{val}(\mathcal{A}_3)| &\geq \frac{1}{\sqrt{2}}|\operatorname{val}(\mathcal{A}_1)| - \frac{1}{8}|\operatorname{val}(\mathcal{A}_1)| - \frac{1}{8}|\operatorname{val}(\mathcal{A}_1)| \\ &\geq \frac{1}{4}|\operatorname{val}(\mathcal{A}_1)|.\end{aligned}$$

By the Fourier inversion formula (Theorem 4.2.1) this shows:

$$\begin{aligned}f_H(t) &\geq g(r)(\Re[\operatorname{val}(\mathcal{A}_1)] - |\operatorname{val}(\mathcal{A}_2)| - |\operatorname{val}(\mathcal{A}_3)|) - \int_{\mathcal{A}_4} |\hat{R}(\theta)| d\theta \\ &\geq \frac{1}{8} g(r)|\operatorname{val}(\mathcal{A}_1)| \geq \frac{1}{16}|\operatorname{val}(\mathcal{A}_1)|.\end{aligned}$$

The previous inequality holds for all t with $\|t - p\mu\| \leq \Delta$ for

$$\Delta := \sqrt{\frac{\kappa^3 p n \sigma^2}{m\left(\ln(16\sqrt{\zeta+1}) + \frac{m}{2}\ln\left(\frac{10^5 m}{\kappa^3}\right)\right)}}.$$

By the lower bound on p in (4.4), one can see that $\sqrt{\gamma n} = \exp\left(-\frac{\kappa^3 pn}{80m}\right)\sigma n^2 \leq \frac{1}{2}\Delta$. In particular, if we choose t such that $\|t - p\mu\| \leq \sqrt{\gamma n}$, then the previous bound holds for all t' with $\|t' - t\| \leq \frac{1}{2}\Delta$, as $\|t' - p\mu\| \leq \Delta$. Hence:

$$\begin{aligned}\Pr\left[\|H - t\| \leq \sqrt{\gamma n}\right] &\geq \int\limits_{B(t,\sqrt{\gamma n})} f_H(t')dt' \geq \frac{1}{16}|\operatorname{val}(\mathcal{A}_1)| \int\limits_{B(t,\sqrt{\gamma n})} 1dt' \\ &\geq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{16\Gamma(\frac{m}{2}+1)}|\operatorname{val}(\mathcal{A}_1)| \geq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{16\Gamma(\frac{m}{2}+1)}\left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.\end{aligned}$$

Now, recall that $H = D + G$, where $G \sim \mathcal{N}(0, \gamma I_m)$. By (4.4), $n > \frac{10^6 m^3}{\kappa^9}$. Hence, by applying Lemma 2.3.3 again,

$$\begin{aligned}Pr\left(\|H - D\| \geq \sqrt{\gamma n}\right) &= \Pr\left(\|G\| \geq \sqrt{\gamma n}\right) \leq \Pr\left(\|G\| \geq \sqrt{\gamma pn}\right) \leq e^{-\frac{pn}{3}} \\ &\leq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{32\Gamma(\frac{m}{2}+1)}\left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.\end{aligned}$$

We conclude that:

$$\begin{aligned}\Pr\left[\|D - t\| \leq 2\sqrt{\gamma n}\right] &\geq \Pr\left[\|H - t\| \leq \sqrt{\gamma n}\right] - \Pr\left(\|H - D\| \geq \sqrt{\gamma n}\right) \\ &\geq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{32\Gamma(\frac{m}{2}+1)}\left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.\end{aligned}$$

By Lemma 4.4.13 we know that $|S| \notin [\frac{1}{2}pn, \frac{3}{2}pn] \leq 2\exp(-pn/12)$. By the lower bound on pn in Eq. (4.4) we can see that:

$$\begin{aligned}
\Pr\left[\|D-t\| \leq \sqrt{\gamma n} \wedge \ |S| \in \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right]& \\
&\geq \Pr\left[\|D-t\| \leq \sqrt{\gamma n}\right] - \Pr\left[|S| \notin \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right] \\
&\geq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{32\Gamma(\frac{m}{2}+1)} \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}} - 2\exp(-pn/12) \\
&\geq \frac{\left(\sqrt{\pi\gamma n}\right)^m}{64\Gamma(\frac{m}{2}+1)} \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.
\end{aligned}$$

As this probability is positive, there must be a choice of $x \in \{0,1\}^n$ such that $\|Ax - t\| \leq 2\sqrt{n}$, such that the support S of x satisfies $|S| \in [\frac{1}{2}pn, \frac{3}{2}pn]$. $\square$

Proof of Theorem 4.3.6. Just like in the proof of Theorem 4.3.4 we have:

$$\begin{aligned}
\Pr[D=t] &\geq \Re[\mathrm{val}(\mathcal{A}_1)] - |\mathrm{val}(\mathcal{A}_2)| - |\mathrm{val}(\mathcal{A}_3)| - |\mathrm{val}(\mathcal{A}_4)| \\
&\geq \frac{1}{4}|\mathrm{val}(\mathcal{A}_1)| - |\mathrm{val}(\mathcal{A}_4)|.
\end{aligned}$$

From Eq. (4.4) it follows that $2\sigma n^2 \leq e^{\frac{\kappa^3 pn}{80m}}$,

$$\left[-\frac{1}{2}, \frac{1}{2}\right]^m = V \subseteq \left\{\|\theta\|_\infty < e^{\frac{\kappa^3 pn}{80m}} \frac{1}{4\sigma n^2}\right\}.$$

Hence $\mathrm{val}(\mathcal{A}_4) = 0$. So, like in the proof of Theorem 4.3.4, we have:

$$\Pr[D=t] \geq \frac{1}{4}|\mathrm{val}(\mathcal{A}_1)| \geq \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.$$

Similarly, we have:

$$\begin{aligned}
\Pr\left[D = t \wedge \ |S| \in \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right]& \\
&\geq \Pr\left[\|D-t\| \leq \sqrt{\gamma n}\right] - \Pr\left[|S| \notin \left[\frac{1}{2}pn, \frac{3}{2}pn\right]\right] \\
&\geq \left(\frac{1}{10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}} - 2\exp(-pn/12) \\
&\geq \left(\frac{1}{2\cdot 10^4 np\sigma^2}\right)^{\frac{m}{2}} \frac{1}{2\sqrt{\zeta+1}}.
\end{aligned}$$

As this probability is positive, there must be a choice of $x \in \{0,1\}^n$ such that $\|Ax - t\| \leq 2\sqrt{n}$, such that the support S of x satisfies $|S| \in [\frac{1}{2}pn, \frac{3}{2}pn]$. □

4.5 Anti-concentration results

Throughout this section we use the notation,

$$d(x, \mathbb{Z}) := \min_{z \in \mathbb{Z}} |x - z|.$$

Our goal in this section is to prove that Definition 4.3.3 is valid for a large family of distributions and prove Lemma 4.4.1.

Distributions with bounded densities

We begin with the following simple 1-dimensional lemma.

Lemma 4.5.1. *Let X be a random variable with $\mathbb{E}[X] = \mu$ and $\mathrm{Var}(X) = \sigma^2$. Suppose that X has a density ρ, which satisfies,*

$$\rho(x) \leq \frac{C}{\sigma},$$

for some $C > 0$. Then, for every $\varepsilon > 0$, then for $\delta = \frac{\varepsilon^2}{12C}$,

$$\Pr\left(d(X, \mathbb{Z}) \geq \delta \min(1, \sigma)\right) \geq 1 - \varepsilon.$$

Proof. By Chebyshev's inequality $\Pr\left(|X - \mu| \geq \frac{2}{\varepsilon}\sigma\right) \leq \frac{\varepsilon^2}{4}$. Define $\sigma' = \min(1, \sigma)$ and note that if $\mathbb{Z} + [-\delta\sigma', \delta\sigma'] := \bigcap\limits_{z \in \mathbb{Z}} [z - \delta\sigma', z + \delta\sigma']$, then, for any $\delta > 0$,

$$\Pr\left(X \in \left[\mu - \frac{2}{\varepsilon}\sigma, \mu + \frac{2}{\varepsilon}\sigma\right] \bigcap \left(\mathbb{Z} + [-\delta\sigma', \delta\sigma']\right)\right) \leq \sum_{z \in \mathbb{Z}, |z - \mu| < \frac{2}{\varepsilon}\sigma} \int\limits_{z - \delta\sigma'}^{z + \delta\sigma'} \rho(x) dx.$$

If $\sigma \geq 1$, since $\rho(x) \leq \frac{C}{\sigma}$,

$$\sum_{z \in \mathbb{Z}, |z - \mu| < \frac{2}{\varepsilon}\sigma} \int\limits_{z - \delta}^{z + \delta} \rho(x) dx \leq \frac{6}{\varepsilon}\sigma \cdot 2\delta \cdot \frac{C}{\sigma}.$$

We now choose $\delta = \frac{\varepsilon^2}{12C}$, so the right hand side becomes smaller than $\frac{\varepsilon}{2}$, and

$$\begin{aligned}
&\Pr\left(d(X,\mathbb{Z}) \geq \delta \min\left(1,\sigma\right)\right)\\
&\geq \Pr\left(|X-\mu| < \frac{\varepsilon}{2}\sigma\right) - \Pr\left(X \in [\mu-\delta,\mu+\delta]\bigcap\left(\mathbb{Z}+[-\delta,\delta]\right)\right)\\
&\geq 1 - \frac{\varepsilon^2}{4} - \frac{\varepsilon}{2} > 1-\varepsilon.
\end{aligned}$$

If $\sigma < 1$, then $\sigma' = \sigma$ and

$$\sum_{z\in\mathbb{Z}, |z-\mu|<\frac{2}{\varepsilon}\sigma} \int_{z-\delta\sigma}^{z+\delta\sigma} \rho(x)dx \leq \frac{6}{\varepsilon}\cdot 2\delta\sigma \cdot \frac{C}{\sigma}.$$

We then arrive at the same conclusion. ☐

We now prove our anti-concentration result measures with an appropriate density bound.

Lemma 4.5.2. *Let X be a random vector in $\mathbb{R}^m$ with $\Sigma := \operatorname{Cov}(X)$. For $\theta \in \mathbb{R}^m$ let ρ_θ stand for the density of $\langle X,\theta\rangle$. Assume that there is a constant $C > 1$, satisfying the following three conditions:*

- *For every $\theta \in \mathbb{R}^m$, $x \in \mathbb{R}$, $\rho_\theta(x) \leq \frac{C}{\sqrt{\operatorname{Var}(\langle X,\theta\rangle)}}$.*
- *For every $\nu \in \mathbb{R}^m$, $\Pr\left(\langle \nu, X\rangle \leq \langle \nu,\mu\rangle\right) \geq \frac{1}{C}$.*
- *$\|\Sigma\|_{\text{op}}\|\Sigma^{-1}\|_{\text{op}} \leq C$.*

Then, for any $\theta,\nu \in \mathbb{R}^m$,

$$\Pr\left[d(\theta^\intercal X,\mathbb{Z}) \geq \frac{1}{48C^3}\min\left(1,\|\theta\|_\infty\sigma\right) \mid \langle\nu,X\rangle \leq \langle\nu,\mu\rangle\right] \geq \frac{1}{2C},$$

where $\sigma := \|\Sigma\|_{\text{op}}$. In other words, X satisfies (AC) *with σ and $\kappa = \frac{1}{48C^3}$.*

Before proving the result, we note that by Lemmas 2.3.8 and 2.3.7, Lemma 4.5.2 applies to isotropic logconcave distributions and hence proves the first half of Lemma 4.4.1.

Proof. Let us denote $\eta^2 := \operatorname{Var}(\theta^\intercal X)$ and observe

$$\frac{\|\theta\|^2}{\|\Sigma^{-1}\|_{\text{op}}} \leq \eta^2 \leq \|\theta\|^2\|\Sigma\|_{\text{op}}.$$

With this in mind, we will actually show the seemingly stronger result,

$$\Pr\left[d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{\kappa}{C}\min(1,\sigma) \mid \langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right] \geq \frac{\kappa}{C}.$$

However, since the class of measures considered by the lemma is preserved under rotations this is an actual equivalent statement.

By assumption, $\frac{\eta}{\sigma\|\theta\|} \geq \frac{1}{\sqrt{C}}$, and if we choose $\varepsilon = \frac{1}{2C}$ in Lemma 4.5.1, then

$$\Pr\left[d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{\delta}{\sqrt{C}}\min(1,\|\theta\|\sigma)\right] \geq 1 - \frac{1}{2C},$$

where $\delta = \frac{1}{48C^3}$. To complete the proof, by assumption ν,

$$\Pr\left(\langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right) \geq \frac{1}{C}.$$

Thus, with a union bound

$$\begin{aligned}
&\Pr\left[d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{\delta}{\sqrt{C}}\min(1,\|\theta\|_\infty\sigma) \mid \langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right] \\
&= \frac{\Pr\left[d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{\delta}{\sqrt{C}}\min(1,\|\theta\|_\infty\sigma) \text{ and } \langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right]}{\Pr\left(\langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right)} \\
&\geq \Pr\left[d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{\delta}{\sqrt{C}}\min(1,\|\theta\|_\infty\sigma)\right] + \Pr\left(\langle \nu, X\rangle \leq \langle \nu, \mu\rangle\right) - 1 \\
&\geq \left(\frac{1}{2C} + 1 - \frac{1}{C}\right) + \frac{1}{C} - 1 \geq \frac{1}{2C}.
\end{aligned}$$

□

Discrete distributions

We now prove anti-concentration results for discrete distributions supported on $\mathbb{Z}^m$. Our result pertains to random variables which are uniform on intervals of length at least 3.

Lemma 4.5.3. *Let $X = (X_1, \ldots, X_m)$ be a random vector in $\mathbb{R}^m$, such that $\{X_i\}_{i=1}^m$ are* i.i.d. *uniformly on $\{a, a+1, \ldots, a+k\}$, for some $a, k \in \mathbb{N}$, with $k > 1$. Set $\mu = \mathbb{E}[X_1]$ and $\sigma = k$. Then* $\mathrm{Var}[X_1] \preccurlyeq \sigma^2 I_m$*, and for every $\theta \in [-\frac{1}{2}, \frac{1}{2}]^m$, and every $\nu \in \mathbb{R}^m$,*

$$\Pr\left(d(\theta^{\mathsf{T}}X,\mathbb{Z}) \geq \frac{1}{20}\min(\|\theta\|_\infty\sigma, 1) \mid \langle \nu, X\rangle \leq \langle \nu, \mu\mathbf{1}\rangle\right) \geq \frac{1}{40}.$$

In other words, X is $(k, \frac{1}{40})$-anti-concentrated.

Proof. Observe that, as X is symmetric around its mean,

$$\begin{aligned}\Pr&\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right)\mid\langle\nu,X\rangle\leq\langle\nu,\mu\mathbf{1}\rangle\right)\\&=\frac{\Pr\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right)\text{ and }\langle\nu,X\rangle\leq\langle\nu,\mu\mathbf{1}\rangle\right)}{\Pr\left(\langle\nu,X\rangle\leq\langle\nu,\mu\mathbf{1}\rangle\right)}\\&\geq\Pr\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right)\text{ and }\langle\nu,X\rangle\leq\langle\nu,\mu\mathbf{1}\rangle\right),\end{aligned}$$

With no loss of generality, let us assume $|\theta_m|=\|\theta\|_\infty$ and consider the event,

$$E=\left\{\sum_{i=1}^{m-1}\nu_iX_i\leq\mu\sum_{i=1}^{m-1}\nu_i\text{ and }\nu_mX_m\leq\mu\nu_m\right\}.$$

Clearly, $E\subseteq\{\langle\nu,X\rangle\leq\langle\nu,\mu\mathbf{1}\rangle\}$, and by symmetry and independence, $\Pr(E)\geq\frac{1}{4}$. With the previous display,

$$\begin{aligned}\Pr&\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right)\mid\langle\nu,X\rangle\leq\langle\nu,\mu\rangle\right)\\&\geq\Pr\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right),\ X\in E\right)\\&=\frac{1}{4}\Pr\left(d(\theta^{\mathsf{T}}X,\mathbb{Z})\geq\frac{1}{20}\min\left(\|\theta\|_\infty\sigma,1\right)\mid X\in E\right).\end{aligned}$$

Now, denote $r:=\sum\limits_{i=1}^{m-1}\theta_iX_i$, and rewrite,

$$d(\theta^{\mathsf{T}}X,\mathbb{Z})=d(\theta_mX_m,(\mathbb{Z}-r)).$$

We observe that under the conditioning on E, depending on $\text{sign}(\nu_m)$, θ_mX_m is either uniform on $\{\theta_ma,\theta_m(a+1)\dots\theta_m\lfloor\mu\rfloor\}$, or on $\{\theta_m\lceil\mu\rceil\dots\theta_m(a+k)\}$. In both cases this set has size $\lfloor\frac{k}{2}\rfloor+1$. We are then interested in the size of the set

$$F=\{\theta_m\cdot x:d(\theta_mx,(\mathbb{Z}-r))\geq\frac{1}{20}\min\left(\theta_m\sigma,1\right)\text{ and }x\in\text{support}(X_m|E)\}.$$

Since $|\theta_m|\leq\frac{1}{2}$, we can show that, as long as $k\geq 2$,

$$\frac{|F|}{|\text{Support}(X_m|E)|}\geq\frac{1}{10}.\tag{4.12}$$

This can be done by considering an arbitrary integer interval S of size $\lceil \frac{k}{2} \rceil + 1$ and noting that at most a $\frac{9}{10}$ fraction of the set $\theta_m S \mod 1$ can occupy *any* interval I of length $\frac{1}{10} \min(\theta_m \sigma, 1)$.

Indeed, let I be such an interval. If $\theta_m > \frac{1}{10} \min(\theta_m \sigma, 1)$, then it cannot be the case that for some j, both $j\theta_m, (j+1)\theta_m \in I + \mathbb{Z}$. On the other hand, if $\theta_m \leq \frac{1}{10} \min(\theta_m \sigma, 1)$, then let $v = \lceil \frac{1}{10} \min(\sigma, \frac{1}{\theta_m}) \rceil$ and consider the set S' consisting of the v smallest elements of S. If, for some $j \in \theta_m S'$, $j\theta_m \in I + \mathbb{Z}$, necessarily, $(j+v)\theta_m \notin I + \mathbb{Z}$.

Note that S' is disjoint from $S' + v$. We have $v \leq \lceil \frac{1}{10}\sigma \rceil \leq \lceil \frac{1}{10} k \rceil$, so that $|S'| + v = 2v \leq 2\lceil \frac{1}{10} k \rceil \leq \lceil k/2 \rceil = |S|$ for $k \geq 2$. So $S' + v \subseteq S$.

Hence, for every $x \in S'$ with $\theta_m x \in (I + \mathbb{Z})$ we have shown that $\theta_m(x + v) \notin (I + \mathbb{Z})$ and that $x + v \in S \setminus S'$. This shows that $|\theta_m S \setminus (I + \mathbb{Z})| \geq |S'| \geq \frac{k}{10}$, proving (4.12).

Thus, by invoking the law of total probability on all possible values of r,

$$\Pr\left(d(\theta^{\mathsf{T}} X, \mathbb{Z}) \geq \frac{1}{20} \min(\theta_m \sigma, 1) \mid X \in E \right) \geq \frac{|F|}{|\mathrm{Support}(X_m | E)|} \geq \frac{1}{10}.$$

Further, note that $\mathrm{Var}[X] = \frac{k^2 + 2k}{12} I_m \preccurlyeq \sigma^2 I$. $\square$

Chapter 5

Node selection in the explorable heap model

An important component of the branch-and-bound algorithm is the node selection rule, which determines the order in which the solution space is explored. Node selection rules can be studied theoretically by considering an abstract problem called *explorable heap selection*. It was originally proposed by Karp, Saks and Wigderson to model node selection for branch-and-bound with low space-complexity [KSW86][1]. As we will explain, the problem remains practically relevant to branch-and-bound even in the full space setting. We provide a randomized algorithm for the problem with an expected running time of $O(n \log(n)^3)$ and space complexity of $O(\log(n))$. This running time is significantly better than the $n \exp(O(\sqrt{\log(n)}))$ running time of the previous best algorithm [KSW86]. We also give a lower bound on the running time of space-restricted algorithms for the problem.

5.1 Introduction

The explorable heap selection problem is an online graph exploration problem for an agent on a rooted (possibly infinite) binary tree. The nodes of the tree are labeled by distinct real numbers (the key values) that increase along every path starting from the root. The tree can thus be thought of as a min-heap. Starting at the root, the agent's objective is to select the nth smallest value in the tree while minimizing the distance traveled, where each edge of the tree has unit travel cost. The key value of a node is only revealed when the agent visits it, and thus the problem has an online nature. When the agent learns the key value of a node, it still does not know the rank of this value.

We note that $\Omega(n)$ is a natural lower bound on the running time. This is because verifying that a value $\mathcal{L}$ is the nth minimum requires $\Theta(n)$ time – one must at least inspect the n nodes with value at most $\mathcal{L}$ – which can be done via straightforward depth-first search.

The contents of this chapter are based on joint work with Daniel Dadush, Sophie Huiberts and Danish Kashaev [BDHK23].

[1] [KSW86] did not give the problem a name, so we have attempted to give a descriptive one here.

A simple selection strategy is to use the best-first rule, which repeatedly explores the unexplored node whose parent has the smallest key value. While this rule is optimal in terms of the number of nodes that it explores, namely $\Theta(n)$, the distance traveled by the agent can be far from optimal. In the worst-case, an agent using this rule will need to travel a distance of $\Theta(n^2)$ to find the nth smallest value. A simple bad example for this rule is to consider a rooted tree consisting of two paths (which one can extend to a binary tree), where the two paths are consecutively labeled by all positive even and odd integers respectively. Moreover, the space complexity becomes $\Omega(n)$ in general when using the best-first rule, because essentially all the explored nodes might need to be kept in memory. We note that irrespective of computational considerations on the agent, either in terms of working memory or running time restrictions, minimizing the total travel distance in explorable heap selection remains a challenging online problem.

Improving on the best-first strategy, Karp, Saks and Wigderson [KSW86] gave a randomized algorithm with expected cost $n \cdot \exp(O(\sqrt{\log(n)}))$ using $O(\sqrt{\log(n)})$ working space. They also showed how to make the algorithm deterministic using $O(\log(n)^{2.5})$ space. In this work, our main contribution is an improved randomized algorithm with expected cost $O(n \log(n)^3)$ using $O(\log(n))$ space. Given the $\Omega(n)$ lower bound, our travel cost is optimal up to logarithmic factors. Furthermore, we show that any algorithm for explorable heap selection that uses only s units of memory, must take at least $n \cdot \log_s(n)$ time in expectation. An interesting open problem is the question whether a superlinear lower bound also holds without any restriction on the memory usage.

To clarify the memory model, it is assumed that any key value and $O(\log n)$ bit integer can be stored using $O(1)$ space. We also assume that maintaining the current position in the tree does not take up memory. Furthermore, we assume that key value comparisons and moving across an edge of the tree require $O(1)$ time. Under these assumptions, the running times of the above algorithms happen to be proportional to their travel cost. Throughout the chapter, we will thus use travel cost and running time interchangeably.

Motivation

The study of the explorable heap selection problem is motivated by the branch-and-bound algorithm. At the core, the branch-and-bound algorithm consists of two important components: the branching rule and the node selection rule. The branching rule determines how to split up a problem into subproblems, by choosing a variable to branch on. Substantial research has been done on branching rules, see, e.g., [LS99; AKM05; LZ17; BDSV18].

The node selection rule decides which subproblem to solve next. Not much theoretical research has been done on the choice of the node selection rule. Traditionally, the best-first strategy is thought to be optimal from a theoretical perspective because this rule minimizes the number of nodes that need to be visited. However, a disadvantage of this rule is that searches using it might use space proportional to the number of explored nodes, because all of them need to be kept in memory. In contrast to this, a simple strategy like depth-first search only needs to store the current solution. Unfortunately, performing a depth-first search can lead to an arbitrarily bad running time. This was the original motivation for introducing the explorable heap selection problem [KSW86]. By guessing the number N of branch-and-bound nodes whose LP values are at most that of the optimal IP solution (which can be done via successive doubling), a search strategy for this problem can be directly interpreted as a node selection rule. The algorithm that they introduced can therefore be used to implement branch-and-bound efficiently in only $O\left(\sqrt{\log(N)}\right)$ space.

Nowadays, computers have a lot of memory available. This usually makes it feasible to store all explored nodes of the branch-and-bound tree in memory. However, many MIP solvers still make use of a hybrid method that consists of both depth-first and best-first searches. This is not only done because depth-first search uses less memory, but also because it is often faster. Experimental studies have confirmed that the depth-first strategy is in many cases faster than best-first one [CP99]. This seems contradictory, because the running time of best-first search is often thought to be theoretically optimal.

In part, this contradiction can be explained by the fact that actual IP-solvers often employ complementary techniques and heuristics on top of branch-and-bound, which might benefit from depth-first searches. Additionally, a best-first search can hop between different parts of the tree, while a depth first search subsequently explores nodes that are very close to each other. In the latter case, the LP-solver can start from a very similar state, which is known as warm starting. This is faster for a variety of technical reasons [Ach07b]. For example, this can be the case when the LP-solver makes use of the LU-factorization of the optimal basis matrix [MJSS16]. Through the use of dynamic algorithms, computing this can be done faster if a factorization for a similar LP-basis is known [SS93]. Because of its large size, MIP solvers will often not store the LU-factorization for all nodes in the tree. This makes it beneficial to move between similar nodes in the branch-and-bound tree. Furthermore, moving from one part of the tree to another means that the solver needs to undo and redo many bound changes, which also takes up time. Hence, the amount of distance traveled between nodes in the tree is a metric that influences the running time. This can also be observed when running the academic MIP solver SCIP[2].

[2]Ambros Gleixner, personal communication, 2023

The explorable heap selection problem captures these benefits of locality by measuring the running time in terms of the amount of travel through the tree. Therefore, we argue that this problem is still relevant for the choice of a node selection rule, even if all nodes can be stored in memory.

Related work

The explorable heap selection problem was first introduced in [KSW86]. Their result was later applied to prove an upper bound on the parallel running time of branch-and-bound [PPSV15].

When random access to the heap is provided at constant cost, selecting the n'th value in the heap can be done by a deterministic algorithm in $O(n)$ time by using an additional $O(n)$ memory for auxiliary data structures [Fre93].

The explorable heap selection problem can be thought of as a *search game* [AG03] and bears some similarity to the *cow path problem.* In the cow path problem, an agent explores an unweighted unlabeled graph in search of a target node. The location of the target node is unknown, but when the agent visits a node they are told whether or not that node is the target. The performance of an algorithm is judged by the ratio of the number of visited nodes to the distance of the target from the agent's starting point. In both the cow path problem and the explorable heap selection problem, the cost of backtracking and retracing paths is an important consideration. The cow path problem on infinite b-ary trees was studied in [DCD95] under the assumption that when present at a node the agent can obtain an estimate on that node's distance to the target.

Other explorable graph problems exist without a target, where typically the graph itself is unknown at the outset. There is an extensive literature on exploration both in graphs and in the plane [Ber98; Kam]. In some of the used models the objective is to minimize the distance traveled [BCGL23; MMS12; KP94]. Other models are about minimizing the amount of used memory [DFKP04]. What distinguises the explorable heap selection problem from these problems is the information that the graph is a heap and that the ordinal of the target is known. This can allow an algorithm to rule out certain locations for the target. Because of this additional information, the techniques used here do not seem to be applicable to these other problems.

Organization

In Section 5.2 we formally introduce the explorable heap selection problem and any notation we will use. In Section 5.3 we introduce a new algorithm for solving this problem and provide a running time analysis. In Section 5.4 we give a lower bound on the complexity of solving explorable heap selection using a limited amount of memory. In Section 5.5 we conclude the chapter with some open problems.

5.2 The explorable heap selection problem

We introduce in this section the formal model for the explorable heap selection problem. The input to the algorithm is an infinite binary tree $T = (V, E)$, where each node $v \in V$ has an associated real value, denoted by $\mathrm{val}(v) \in \mathbb{R}$. We assume that all the values are distinct. Moreover, for each node in the tree, the values of its children are larger than its own value. Hence, for every $v_1, v_2 \in V$ such that v_1 is an ancestor of v_2, we have that $\mathrm{val}(v_2) > \mathrm{val}(v_1)$. The binary tree T is thus a heap.

The algorithmic problem we are interested in is finding the nth smallest value in this tree. This may be seen as an online graph exploration problem where an agent can move in the tree and learns the value of a node each time they explore it. At each time step, the agent resides at a vertex $v \in V$ and may decide to move to either the left child, the right child or the parent of v (if it exists, i.e. if v is not the root of the tree). Each traversal of an edge costs one unit of time, and the complexity of an algorithm for this problem is thus measured by the total traveled distance in the binary tree. The algorithm is also allowed to store values in memory.

We now introduce a few notations used throughout the chapter.

- For a node $v \in V$, also per abuse of notation written $v \in T$, we denote by $T^{(v)}$ the subtree of T rooted at v.
- For a tree T and a value $\mathcal{L} \in \mathbb{R}$, we define the subtree $T_{\mathcal{L}} := \{v \in T \mid \mathrm{val}(v) \leq \mathcal{L}\}$.
- We denote the nth smallest value in T by $\mathsf{SELECT}^T(n)$. This is the quantity that we are interested in finding algorithmically.
- We say that a value $\mathcal{V} \in \mathbb{R}$ is *good* for a tree T if $\mathcal{V} \leq \mathsf{SELECT}^T(n)$ and *bad* otherwise. Similarly, we call a node $v \in T$ *good* if $\mathrm{val}(v) \leq \mathsf{SELECT}^T(n)$ and *bad* otherwise.
- We will use $[k]$ to refer to the set $\{1, \ldots, k\}$.

- When we write $\log(n)$, we assume the base of the logarithm to be 2.

For a given value $\mathcal{V} \in \mathbb{R}$, it is easy to check whether it is good in $O(n)$ time: start a depth first search at the root of the tree, turning back each time a value strictly greater than $\mathcal{V}$ is encountered. In the meantime, count the number of values below $\mathcal{V}$ found so far and stop the search if more than n values are found. If the number of values below $\mathcal{V}$ found at the end of the procedure is at most n, then $\mathcal{V}$ is a good value. This procedure is described in more detail later in the DFS subroutine.

We will often instruct the agent to move to an already discovered good vertex $v \in V$. The way this is done algorithmically is by saving $\text{val}(v)$ in memory and starting a depth first search at the root, turning back every time a value strictly bigger than $\text{val}(v)$ is encountered until finally finding $\text{val}(v)$. This takes at most $O(n)$ time, since we assume v to be a good node. If we instruct the agent to go back to the root from a certain vertex $v \in V$, this is simply done by traveling back in the tree, choosing to go to the parent of the current node at each step.

In later sections, we will often say that a subroutine takes a subtree $T^{(v)}$ as input. This implicitly means that we in fact pass it $\text{val}(v)$ as input, make the agent travel to $v \in T$ using the previously described procedure, call the subroutine from that position in the tree, and travel back to the original position at the end of the execution. Because the subroutine knows the value $\text{val}(v)$ of the root of $T^{(v)}$, it can ensure it never leaves the subtree $T^{(v)}$, thus making it possible to recurse on a subtree as if it were a rooted tree by itself. We write the subtree $T^{(v)}$ as part of the input for simplicity of presentation.

We will sometimes want to pick a value uniformly at random from a set of values $\{\mathcal{V}_1, \dots, \mathcal{V}_k\}$ of unknown size that arrives in a streaming fashion, for instance when we traverse a part of the tree T by doing a depth first search. That is, we see the value $\mathcal{V}_i$ at the ith time step, but do no longer have access to it in memory once we move on to $\mathcal{V}_{i+1}$. This can be done by generating random values $\{X_1, \dots, X_k\}$ where, at the ith time step, $X_i = \mathcal{V}_i$ with probability $1/i$, and $X_i = X_{i-1}$ otherwise. It is easy to check that X_k is a uniformly distributed sample from $\{\mathcal{V}_1, \dots, \mathcal{V}_k\}$.

5.3 A new algorithm

The authors of [KSW86] presented a deterministic algorithm that solves the explorable heap selection problem in $n \cdot \exp(O(\sqrt{\log(n)}))$ time and $O(n\sqrt{\log(n)})$ space. By replacing the binary search that is used in the algorithm by a randomized variant, they are able to decrease the space requirements. This way, they obtain a randomized algorithm with expected running time $n \cdot \exp(O(\sqrt{\log(n)}))$ and space complexity $O(\sqrt{\log(n)})$. Alternatively, the binary search can be implemented using a deterministic routine by [MP80] to achieve the same running time with $O(\log(n)^{2.5})$ space.

We present a randomized algorithm with a running time $O(n\log(n)^3)$ and space complexity $O(\log(n))$. Unlike the algorithms mentioned before, our algorithm fundamentally relies on randomness to bound its running time. This bound only holds when the algorithm is run on a tree with labels that are fixed before the execution of the algorithm. That is, the tree must be generated by an adversary that is oblivious to the choices made by the algorithm. This is a stronger assumption than is needed for the algorithm that is given in [KSW86], which also works against adaptive adversaries. An adaptive adversary is able to defer the decision of the node label to the time that the node is explored. Note that this distinction does not really matter for the application of the algorithm as a node selection rule in branch-and-bound, since there the node labels are fixed because they are derived from the integer program and branching rule.

Theorem 5.3.1. *There exists a randomized algorithm that solves the explorable heap selection problem, with expected running time $O(n\log(n)^3)$ and $O(\log(n))$ space.*

As mentioned above, checking whether a value v is good can be done in $O(n)$ time by doing a depth-first search with cutoff value $\text{val}(v)$ that returns when more than n good nodes are found. For a set of k values, we can determine which of them are good in $O(\log(k)n)$ time by performing a binary search.

The explorable heap selection problem can be seen as the problem of finding all n good nodes. Both our method and that of [KSW86] function by first identifying a subtree consisting of only good nodes. The children of the leaves of this subtree are called "roots" and the subtree is extended by finding a number of new good nodes under these roots in multiple rounds.

In [KSW86] this is done by running $O(c^{\sqrt{2\log(n)}})$ different rounds, for some constant $c > 1$. In each round, the algorithm finds $n/c^{\sqrt{2\log(n)}}$ new good nodes. These nodes are found by recursively exploring each active root and using binary search on the observed values to discover which of these values are good. Which active roots are recursively explored further depends on which values are good. The recursion in the algorithm is at most $O(\sqrt{\log(n)})$ levels deep, which is where the space complexity bound comes from.

In our algorithm, we take a different approach. We will call our algorithm consecutively with $n = 1, 2, 4, 8, \ldots$. Hence, for a call to the algorithm, we can assume that we have already found at least $n/2$ good nodes. These nodes form a subtree of the original tree T. In each round, our algorithm chooses a random root under this subtree and finds every good node under it. It does so by doing recursive subcalls to the main algorithm on this root with values $n = 1, 2, 4, 8, \ldots$. As soon as the recursively obtained node is a bad node, the algorithm stops searching the subtree of this root, since it is guaranteed that all the good nodes there have been found. The largest good value that is found can then be used to find additional good nodes under the other roots without recursive calls, through a simple depth-first search. Assuming that the node values were fixed in advance, we expect this largest good value to be greater than half of the other roots' largest good values. Similarly, we expect its smallest bad value to be smaller than half of the other roots' smallest bad values. By this principle, a sizeable fraction of the roots can, in expectation, be ruled out from getting a recursive call. Each round a new random root is selected until all good nodes have been found.

This algorithm allows us to effectively perform binary search on the list of roots, ordered by the largest good value contained in each of their subtrees in $O(\log n)$ rounds, and the same list ordered by the smallest bad values (Lemma 5.3.4). Bounding the expected number of good nodes found using recursive subcalls requires a subtle induction on two parameters (Lemma 5.3.3): both n and the number of good nodes that have been identified so far.

Importantly, the concept of a good node is always used with respect to the current function call. So, a node might be good in one recursive call, but not good in another.

The algorithm

We present in this subsection the algorithm for solving the explorable heap selection problem, as well as the subroutines used in it. This algorithm is named Select and outputs the nth smallest value in the tree T.

A procedure used in Select is the Extend algorithm, which assumes that at least $n/2$ good nodes have already been found in the tree, and also outputs the nth smallest one.

Algorithm 1 The SELECT procedure

```
Input : $n \in \mathbb{N}$
Output : SELECT$(n)$, the $n$th smallest value in the heap $T$.
procedure SELECT$(n)$
    $k \leftarrow 1$
    $\mathcal{L} \leftarrow \text{val}(v)$  // $v$ is the root of the tree $T$
    while $k < n$ do
        if $k < n/2$ then
            $k' \leftarrow 2k$
        else
            $k' \leftarrow n$
        $\mathcal{L} \leftarrow$ EXTEND$(T, k', k, \mathcal{L})$
        $k \leftarrow k'$
    return $\mathcal{L}$
```

Algorithm 2 The EXTEND procedure

```
Input: $T$: tree which is to be explored.
       $n \in \mathbb{N}$: total number of good values to be found, satisfying $n \geq 2$.
       $k \in \mathbb{N}$: number of good values already found, satisfying $k \geq n/2$.
       $\mathcal{L}_0 \in \mathbb{R}$: value satisfying DFS$(T, \mathcal{L}_0, n) = k$.
Output: the $n$th smallest value in $T$.
procedure EXTEND($T$, $n$, $k$, $\mathcal{L}_0$)
    $\mathcal{L} \leftarrow \mathcal{L}_0$
    $\mathcal{U} \leftarrow \infty$
    while $k < n$ do
        $r \leftarrow$ random element from ROOTS($T$, $\mathcal{L}_0$, $\mathcal{L}, \mathcal{U}$)
        $\mathcal{L}' \leftarrow \max(\mathcal{L}, \mathrm{val}(r))$
        $k' \leftarrow$ DFS($T$, $\mathcal{L}'$, $n$) // count the number of values $\leq \mathcal{L}'$ in $T$
        $c \leftarrow$ DFS($T^{(r)}$, $\mathcal{L}'$, $n$) // counting the number of values $\leq \mathcal{L}'$ in $T^{(r)}$
        $c' \leftarrow \min(n - k' + c, 2c)$ // increase number of values to be found in $T^{(r)}$

        while $k' < n$ do // loop until it is certified that SELECT$^T(n) \leq \mathcal{L}'$
            $\mathcal{L}' \leftarrow$ EXTEND($T^{(r)}$, $c'$, $c$, $\mathcal{L}'$)
            $k' \leftarrow$ DFS($T$, $\mathcal{L}'$, $n$)
            $c \leftarrow c'$
            $c' \leftarrow \min(n - k' + c, 2c)$
        $\tilde{\mathcal{L}}, \tilde{\mathcal{U}} \leftarrow$ GOODVALUES$(T, T^{(r)}, \mathcal{L}', n)$ // find the good values in $T^{(r)}$
        $\mathcal{L} \leftarrow \max(\mathcal{L}, \tilde{\mathcal{L}})$
        $\mathcal{U} \leftarrow \min(\mathcal{U}, \tilde{\mathcal{U}})$
        $k \leftarrow$ DFS($T$, $\mathcal{L}$, $n$) // compute the number of good values found in $T$
    return $\mathcal{L}$
```

Let us describe a few invariants from the Extend procedure.

- $\mathcal{L}$ and $\mathcal{U}$ are respectively lower and upper bounds on $\mathsf{SELECT}^T(n)$ during the whole execution of the procedure. More precisely, $\mathcal{L} \leq \mathsf{SELECT}^T(n)$ and $\mathcal{U} > \mathsf{SELECT}^T(n)$ at any point, and hence $\mathcal{L}$ is good and $\mathcal{U}$ is bad. The integer k counts the number of values $\leq \mathcal{L}$ in the full tree T.
- No root can be randomly selected twice. This is ruled out by the updated values of $\mathcal{L}$ and $\mathcal{U}$, and the proof can be found in Theorem 5.3.2.
- After an iteration of the inner while loop, $\mathcal{L}'$ is set to the cth smallest value in $T^{(r)}$. The variable c' then corresponds to the next value we would like to find in $T^{(r)}$ if we were to continue the search. Note that $c' \leq 2c$, enforcing that the recursive call to Extend satisfies its precondition, and that $c' \leq n - (k' - c)$ implies that $(k' - c) + c' \leq n$, which implies that the recursive subcall will not spend time searching for a value that is known in advance to be bad.
- From the definition of k' and c one can see that $k' \geq k^\star + c$. Combined with the previous invariant, we see that $c' \leq n - k$.
- k' always counts the number of values $\leq \mathcal{L}'$ in the full tree T. It is important to observe that this is a global parameter, and does not only count values below the current root. Moreover, $k' \geq n$ implies that we can stop searching below the current root, since it is guaranteed that all good values in $T^{(r)}$ have been found, i.e., $\mathcal{L}'$ is larger than all the good values in $T^{(r)}$.

We now describe the subroutines used in the Extend procedure.

The procedure DFS

The procedure DFS is a variant of depth first search. The input to the procedure is T, a cutoff value $\mathcal{L} \in \mathbb{R}$ and an integer $n \in \mathbb{N}$. The procedure returns the number of vertices in T whose value is at most $\mathcal{L}$.

It achieves that by exploring the tree T in a depth first search manner, starting at the root and turning back as soon as a node $w \in T$ such that $\mathrm{val}(w) > \mathcal{L}$ is encountered. Moreover, if the number of nodes whose value is at most $\mathcal{L}$ exceeds n during the search, the algorithm stops and returns $n + 1$.

The algorithm output is the following integer.

$$\mathrm{DFS}(T, \mathcal{L}, n) := \min\left\{ \left|T_{\mathcal{L}}\right|, n + 1 \right\}.$$

Observe that the DFS procedure allows us to check whether a node $w \in T$ is a good node, i.e. whether $\mathrm{val}(w) \leq \mathsf{SELECT}^T(n)$. Indeed, w is good if and only if $\mathrm{DFS}(T, \mathrm{val}(w), n) \leq n$.

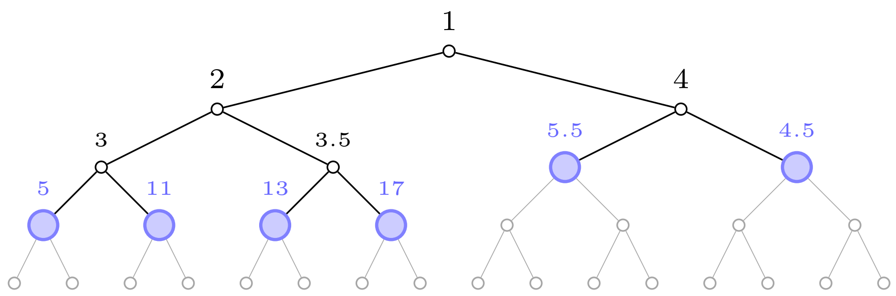


Figure 5.1: An illustration of $R(T, \mathcal{L}_0)$ with $\mathcal{L}_0 = 4$. The number above each vertex is its value, the blue nodes are $R(T, \mathcal{L}_0)$, whereas the subtree above is $T_{\mathcal{L}_0}$.

This algorithm visits only nodes in $T_{\mathcal{L}}$ or its direct descendants and its running time is $O(n)$. The space complexity is $O(1)$, since the only values needed to be stored in memory are $\mathcal{L}$, $\mathsf{val}(v)$, where v is the root of the tree T, and a counter for the number of good values found so far.

The procedure Roots

The procedure Roots takes as input a tree T as well as an initial fixed lower bound $\mathcal{L}_0 \in \mathbb{R}$ on the value of $\mathsf{SELECT}^T(n)$. We assume that the main algorithm has already found all the nodes $w \in T$ satisfying $\mathsf{val}(w) \leq \mathcal{L}_0$. This means that the remaining values the main algorithm needs to find in T are all lying in the subtrees of the following nodes, that we call the *$\mathcal{L}_0$-roots of T*:

$$R(T, \mathcal{L}_0) := \left\{ r \in T \setminus T_{\mathcal{L}_0} \;\middle|\; r \text{ is a child of a node in } T_{\mathcal{L}_0} \right\}$$

In other words, these are all the vertices in T one level deeper in the tree than $T_{\mathcal{L}_0}$, see Figure 5.1 for an illustration. In addition to that, the procedure takes two other parameters $\mathcal{L}, \mathcal{U} \in \mathbb{R}$ as input, which correspond to (another) lower and upper bound on the value of $\mathsf{SELECT}^T(n)$. These bounds $\mathcal{L}$ and $\mathcal{U}$ will be variables being updated during the execution of the main algorithm, where $\mathcal{L}$ will be increasing and $\mathcal{U}$ will be decreasing. More precisely, $\mathcal{L}$ will be the largest value that the main algorithm has certified being at most $\mathsf{SELECT}^T(n)$, whereas $\mathcal{U}$ will be the smallest value that the algorithm has certified being at least that. A key observation is that these lower and upper bounds can allow us to remove certain roots in $R(T, \mathcal{L}_0)$ from consideration, in the sense that all the good values in that root's subtree will be certified to have already been found. The only roots that the main algorithm needs to consider, when $\mathcal{L}$ and $\mathcal{U}$ are given, are thus the following.

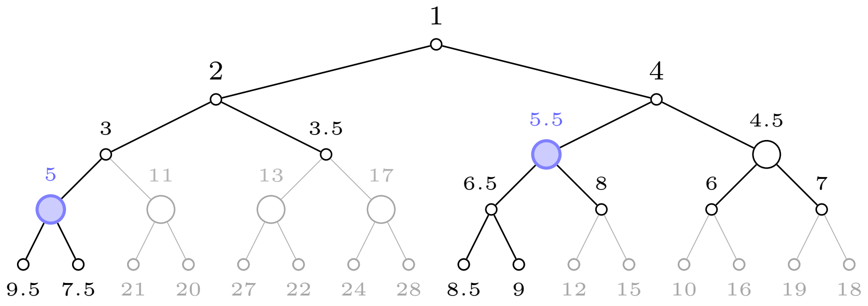


Figure 5.2: An illustration of the Roots procedure with $\mathcal{L}_0 = 4, \mathcal{L} = 7$ and $\mathcal{U} = 10$. Only two active roots remain, and are both colored in blue. The other roots are considered killed since all the good values have been found in their subtrees.

$$\text{Roots}(T, \mathcal{L}_0, \mathcal{L}, \mathcal{U}) := \left\{ r \in R(T, \mathcal{L}_0) \mid \exists w \in T^{(r)} \text{ with } \mathrm{val}(w) \in (\mathcal{L}, \mathcal{U}) \right\}$$

This subroutine can be implemented as follows. Run a depth first search starting at the root of T. Once a node $r \in T$ with $\mathrm{val}(r) > \mathcal{L}_0$ is encountered, the subroutine marks that vertex r as belonging to $R(T, \mathcal{L}_0)$. The depth first search continues deeper in the tree until finding a node $w \in T^{(r)}$ with $\mathrm{val}(w) > \mathcal{L}$. At this point, if $\mathrm{val}(w) < \mathcal{U}$, then the search directly returns to r without exploring any additional nodes in $T^{(r)}$ and adds r to the output. If however $\mathrm{val}(w) \geq \mathcal{U}$, then the search continues exploring $T_{\mathcal{L}}^{(r)}$ (and its direct descendants) by trying to find a node w with $\mathrm{val}(w) \in (\mathcal{L}, \mathcal{U})$. In case the algorithm explores all of $T_{\mathcal{L}}^{(r)}$ with its direct descendants, and it turns out that no such node exists (i.e. every direct descendant w of $T_{\mathcal{L}}^{(r)}$ satisfies $\mathrm{val}(w) \geq \mathcal{U}$), then r is not added to the output.

This procedure takes time $O(|T_{\mathcal{L}}|)$, i.e. proportional to the number of nodes in T with value at most $\mathcal{L}$. Since the procedure is called only on values $\mathcal{L}$ which are known to be good, the time is bounded by $O(|T_{\mathcal{L}}|) = O(n)$.

In the main algorithm, we will only need this procedure in order to select a root from $\text{Roots}(T, \mathcal{L}_0, \mathcal{L}, \mathcal{U})$ uniformly at random, without having to store the whole list in memory. This can then be achieved in $O(1)$ space, since one then only needs to store $\mathrm{val}(v), \mathcal{L}_0, \mathcal{L}$ and $\mathcal{U}$ in memory, where v is the root of the tree T.

The procedure GoodValues

The procedure GoodValues takes as input a tree T, a subtree $T^{(r)}$ for a node $r \in T$, a value $\mathcal{L}' \in \mathbb{R}_{\geq 0}$ and an integer $n \in \mathbb{N}$. The procedure then analyzes the set

$$S := \left\{\mathrm{val}(w) \mid w \in T^{(r)}, \mathrm{val}(w) \leq \mathcal{L}'\right\}$$

and outputs both the largest good value and the smallest bad value in that set, that we respectively call $\mathcal{L}$ and $\mathcal{U}$. If no bad values exist in S, the algorithm sets $\mathcal{U} = \infty$. Notice that this output determines, for each value in S, whether it is good or not. Indeed, any $\mathcal{V} \in S$ is good if and only if $\mathcal{V} \leq \mathcal{L}$, and is bad if and only if $\mathcal{V} \geq \mathcal{U}$.

The implementation is as follows. Initialize the variables $\mathcal{L} = -\infty$ and $\mathcal{U} = \mathcal{L}'$. These variables correspond to lower and upper bounds on $\mathsf{SELECT}^T(n)$. Loop through the values in

$$S' := \left\{\mathrm{val}(w) \mid w \in T^{(r)},\ \mathcal{L} < \mathrm{val}(w) < \mathcal{U}\right\}$$

using a depth first search starting at r and sample one value $\mathcal{V}$ uniformly randomly from that set. Check whether $\mathcal{V}$ is a good value by calling DFS$(T, \mathcal{V}, n)$. If it is good, update $\mathcal{L} = \mathcal{V}$. If it is bad, update $\mathcal{U} = \mathcal{V}$. Continue this procedure until S' is empty, i.e. $|S'| = 0$. If, at the end of the procedure, $\mathcal{L} = \mathcal{L}' = \mathcal{U}$, then set $\mathcal{U} = \infty$. The output is thus:

$$\textsc{GoodValues}(T, T^{(r)}, \mathcal{L}', n) := \{\mathcal{L}, \mathcal{U}\}$$

where

$$\mathcal{L} := \max\left\{\mathcal{V} \in S \mid \mathcal{V} \leq \mathsf{SELECT}^T(n)\right\},$$
$$\mathcal{U} := \min\left\{\mathcal{V} \in S \mid \mathcal{V} > \mathsf{SELECT}^T(n)\right\}.$$

Sampling a value from S' and checking whether the sampled value is good takes $O(n)$ time (under the assumption that $|S'| = O(n)$, which will always be the case when this procedure is called in the main algorithm, since $|\{v \in T \mid \mathrm{val}(v) \leq \mathcal{L}'\}| \leq 2n$). Moreover, in expectation, the number of updates before the set S' is empty is $O(\log(n))$, leading to a total running time of $O(n \log(n))$.

This procedure can be implemented in $O(1)$ space, since the only values needed to be kept in memory are val(v) (where v is the root of the tree T), val(r), $\mathcal{L}, \mathcal{U}$ and $\mathcal{L}'$, as well as the fact that every call to DFS also requires $O(1)$ space.

Proof of correctness

Theorem 5.3.2. *At the end of the execution of Algorithm 1, $\mathcal{L}$ is set to the nth smallest value in T. Moreover, the algorithm is guaranteed to terminate.*

Proof. We show $\mathcal{L} = \mathsf{SELECT}^T(n)$ holds at the end of Algorithm 2, i.e. the Extend procedure. Correctness of Algorithm 1, i.e. the Select procedure, then clearly follows from that. First, notice that $\mathcal{L}$ is always set to the first output of the procedure GoodValues, which is always the value of a good node in T, implying

$$\mathcal{L} \leq \mathsf{SELECT}^T(n)$$

at any point during the execution of the algorithm. Since the outer while loop ends when at least n good nodes in T have value at most $\mathcal{L}$, we get

$$\mathcal{L} \geq \mathsf{SELECT}^T(n),$$

which implies that when the algorithm terminates it does so with the correct value.

It remains to prove that the algorithm terminates. We observe that every recursive call $\mathcal{L}' \leftarrow$ Extend$(T^{(r)}, c', c, \mathcal{L}')$ strictly increases the value of $\mathcal{L}'$, ensuring that at least one extra value in T is under the increased value. This implies that k' strictly increases every iteration of the inner while loop, thus ensuring that this loop terminates.

To see that the outer loop terminates, we observe that after each iteration the set Roots$(T, \mathcal{L}_0, \mathcal{L}, \mathcal{U})$ shrinks by at least one element. As soon as this set is empty, there will be no more roots with unexplored good values in their subtrees, so $k = n$ and the algorithm terminates.

□

Running time analysis

The main challenge in analyzing the running time of the algorithm is dealing with the cost of the recursive subcalls in the Extend procedure. For this we rely on two important ideas.

Firstly, remember that n is the index of the node value that we want to find, while k is the index of the node value that is passed to the procedure. In particular this means that the procedure needs to find only $n - k$ new good nodes. Because of this, our runtime bound for the recursive subcalls that are performed does not just depend on n, but also on $n - k$.

We will show that the amount of travel done in the non-recursive part of a call of Extend with parameters n and k is bounded by $O(n \log(n)^2)$. We will charge this travel to the parent call that makes these recursive calls. Hence, a parent call that does z recursive calls with parameters $(n_1, k_1), \dots, (n_z, k_z)$ will be charged a cost of $\sum_{i=1}^{z} n_i \log(n_i)^2$. In our analysis we will show that this sum can be upper bounded by $4(n-k)\log(n)^2$. So, for every recursive call with parameters n and k, a cost of at most $(n-k)\log(n)^2$ is incurred by the caller.

So, we only have to bound the sum over $(n-k)\log(n)^2$ for all calls with parameters n and k that are done. We do this by first considering a single algorithm call with parameters n and k that makes z recursive subcalls with parameters $(n_1, k_1), \dots, (n_z, k_z)$. For such a subcall we would like to bound the sum $\sum_{i=1}^{z}(n_i - k_i)\log(n_i)^2$ by $(n-k)\log(n)^2$. However, this bound does not hold deterministically. Instead, we show that this bound does hold in expectation.

Now we know that every layer of recursion incurs an expected cost of at most $(n-k)\log(n)^2$. Because the parameter n will decrease by at least a constant factor in each layer of recursion, there can be at most $O(\log(n))$ layers. An upper bound of $O((n-k)\log(n)^3)$ on the expected running time of the Extend then follows for the recursive part.

Combining this with the upper bound of $O(n\log(n)^2)$ on the non-recursive part, we get a total running time of $O(n\log(n)^2) + O((n-k)\log(n)^3)$ for the Extend procedure, which then implies a running time of $O(n\log(n)^3)$ for the Select procedure.

Let us now prove these claims. We first show that the expectation of $\sum_{i=1}^{z}(n_i - k_i)$ is bounded.

Lemma 5.3.3. *Let z be the number of recursive calls that are done in the main loop of Extend($T, n^\star, k^\star, \mathcal{L}$) with parameter $k \geq 1$. For $i \in [z]$, let n_i and k_i be the values of n and k that are given as parameters to the ith such subcall. Then:*

$$\mathbb{E}\left[\sum_{i=1}^{z} n_i - k_i\right] \leq n^\star - k^\star.$$

Proof. Assume we have m roots, whose order is fixed arbitrarily. For $i \in [z]$, let $r_i \in [m]$ be such that the ith recursive subcall is done on the root with index r_i. For $t \in [m]$, let $s_t = \sum_{i=1}^{z} \mathbf{1}_{r_i = t}(n_i - k_i)$. From the algorithm we see that when $r_i = t$, all successive recursive calls will also be on root t, until all good nodes under this root have been found. The updated values of $\mathcal{L}$ and $\mathcal{U}$ ensure this root is never selected again after this, hence all iterations i with $r_i = t$ are consecutive. Now let a_t, b_t be variables that respectively denote the first and last indices i with $r_i = t$. When there is no iteration i with $r_i = t$, then $a_t = b_t = \infty$.

For two calls i and $i+1$ with $r_i = t = r_{i+1}$, observe that after call i already n_i good nodes under root t have been found. Note that on line 16, c' corresponds to n_i and c corresponds to k_i, hence $k_{i+1} = n_i$. Therefore, the definition of s_t is a telescoping series and can be rewritten as $s_t = n_{b_t} - k_{a_t}$, when we define $k_\infty = n_\infty = 0$.

Let $p = n^\star - k^\star$ and let $W = \{w_1, \ldots, w_p\}$ denote the p smallest values under T that are larger than $\mathcal{L}_0$, in increasing order. Now each of these values in W will be part of a subtree generated by one of the roots. For $j \in [p]$, let $d_j \in [m]$ be such that value w_j is part of the subtree of root d_j. Let $S_t = \{j \in [p] : d_j = t\}$.

We will now show that for each root r_t, we have:

$$\mathbb{E}[s_t] \leq |S_t|.$$

This will imply that $\mathbb{E}\left[\sum_{i=1}^{z} n_i - k_i\right] = \sum_{t=1}^{m} \mathbb{E}[s_t] \leq \sum_{t=1}^{m} |S_t| = n^\star - k^\star$.

First, let us consider a root t with $t \neq d_p$. On line 10 each iteration a random root is chosen. Because in every iteration root d_p will be among the active roots, the probability that this root is chosen before root t, is at least a half. So, we have two cases:

- If root d_p is chosen before root t, then DFS(T, $\mathcal{L}$, n) will return n, and the algorithm will terminate. Because no subcalls will be done on root t, in this case $s_t = 0$.
- If instead root t is chosen before root d_p, then consider iteration b_t, the last iteration i that has $r_i = t$. Before this iteration, already k_{b_t} good nodes under the root have been found by the algorithm. It can be seen in the algorithm on lines 14 and 19 that $n_{b_t} \leq 2k_{b_t}$. Hence, $s_t = n_{b_t} - k_{a_t} \leq n_{b_t} \leq 2k_{b_t} \leq 2|S_t|$.

We therefore have:

$$\mathbb{E}[s_t] \leq \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 2|S_t| = |S_t|.$$

Now consider the root d_p. If $S_{d_p} = [p]$, then $s_p = n_{b_{d_p}} - k_{a_{d_p}} \leq n^\star - k^\star = |S_{d_p}|$, because $n_i \leq n^\star - k^\star$ for all i.

If $S_{d_p} \subsetneq [p]$, then there exists a j with $d_j \neq d_p$. Thus, we can define $j^\star = \max\{j \in [p] : d_j \neq d_p\}$. With probability a half, root $d_{j^\star}$ is considered before root d_p. If this happens, $\mathcal{L}$ will be equal to $w_{j^\star}$ when root d_p is selected by the algorithm. In particular, this means that $k_{a_{d_p}}$ will be equal to $j^\star$. Recall the stated invariant that $c' \leq n^\star - k^\star = p$. Now we can see that $s_{d_p} = n_{b_{d_p}} - k_{a_{d_p}} \leq p - j^\star$.

If root d_p is chosen before root $d_{j^\star}$, then consider the last recursive call b_{d_p} to Extend that we do on root d_p. Abbreviate $k' := k_{b_{d_p}}$ and define $A = [k' - k^\star] \cap S_{d_p}$, i.e. the set of all good values under root d_p that have been found so far. We may write $c = |A|$ and $c' = n_{b_{d_p}}$ We distinguish two cases.

If $k' - k^\star \geq j^\star$, i.e., when all good values under $d_{j^\star}$ have been found, then by the definition of $j^\star$ we have $[p] \setminus [k' - k^\star] \subseteq [p] \setminus [j^\star] \subseteq S_{d_p}$. Because A and $[p] \setminus [k' - k^\star]$ are disjoint, we have $|A| + (n^\star - k') = |A| + |[p] \setminus [k' - k^\star]| \leq |S_{d_p}|$. Hence:

$$c' \leq n^\star - k' + c = n^\star - k' + |A| \leq |S_{d_p}|.$$

Therefore, $s_{d_p} \leq n_{b_{d_p}} = c' \leq |S_{d_p}|$.

If $k' - k^\star < j^\star$ at the time of subcall b_{d_p}, then the last good value under $d_{j^\star}$ has yet to be found, implying that $A \subseteq [j^\star]$. From the definition of $j^\star$ we get $[p] \setminus [j^\star] \subseteq S_{d_p}$. Hence, $|A| \leq |S_{d_p}| - |[p] \setminus [j^\star]| = |S_{d_p}| - (p - j^\star)$. Thus, $c' \leq 2c = 2|A| \leq 2(|S_{d_p}| - (p - j^\star))$. So, in this case we have $s_{d_p} \leq n_{b_{d_p}} = c' \leq 2(|S_{d_p}| - (p - j^\star))$.

Collecting all cases above, we find that

$$\begin{aligned}\mathbb{E}[s_{d_p}] &\leq \frac{1}{2} \cdot (p - j^\star) + \frac{1}{2} \cdot \max\left(|S_{d_p}|, 2(|S_{d_p}| - (p - j^\star))\right) \\ &\leq \max\left(\frac{1}{2}|S_{d_p}| + \frac{1}{2}(p - j^\star), |S_{d_p}| - \frac{1}{2}(p - j^\star)\right).\end{aligned}$$

Lastly, by definition of $j^\star$ we have $[p] \setminus [j^\star] \subseteq S_{d_p}$, from which it follows that $p - j^\star \leq |S_{d_p}|$. We finish the proof by observing that this implies

$$\max\left(\frac{1}{2}|S_{d_p}| + \frac{1}{2}(p - j^\star), |S_{d_p}| - \frac{1}{2}(p - j^\star)\right) \leq |S_{d_p}|,$$

which finishes the proof. □

Now we will bound the expected number of iterations of the outermost while-loop.

Lemma 5.3.4. *The expected number of times that the outermost while-loop (at line 9) is executed by the procedure* Extend *is at most* $O(\log(n))$.

Proof. Let $r_1, \ldots, r_m$ denote the roots returned by Roots$(T, \mathcal{L}_0, \mathcal{L}_0, \infty)$. For $j \in [m]$, let ℓ_j and u_j respectively denote the largest good value and the smallest non-good value under root r_j. Let $A_\ell(\mathcal{L}) := \{r_j : \ell_j > \mathcal{L}\}$ and $A_u(\mathcal{U}) := \{r_j : u_j < \mathcal{U}\}$. Observe that Roots$(T, \mathcal{L}_0, \mathcal{L}, \mathcal{U}) = A_\ell(\mathcal{L}) \cup A_u(\mathcal{U})$ for any $\mathcal{L} \leq \mathcal{U}$.

Let $\mathcal{L}_i$ and $\mathcal{U}_i$ denote the values of $\mathcal{L}$ and $\mathcal{U}$ at the start of the ith iteration. After an iteration i in which root r_j was selected, the algorithm updates $\mathcal{L}$ and $\mathcal{U}$ such that $\mathcal{L}_{i+1} = \max(\mathcal{L}, \ell_j)$ and $\mathcal{U}_{i+1} = \min(\mathcal{U}, u_j)$. Observe that $\mathcal{L}_i$ is nondecreasing and that $\mathcal{U}_i$ is nonincreasing.

We will now show that if a root from $A_\ell(\mathcal{L}_i)$ is selected in iteration i, then the expected size of $A_\ell(\mathcal{L}_{i+1})$ is at most half that of $A_\ell(\mathcal{L}_i)$. This will imply that in expectation only $\log(n)$ iterations are needed to make $|A_\ell(\mathcal{L})| = 1$.

Let $\mathcal{F}_i$ be the filtration containing all information up till iteration i. Let X_i denote the value of $|A_\ell(\mathcal{L}_i)|$. Let $(s_k)_{k\geq 1}$ be the subsequence consisting of iteration indices i in which a root from $A_\ell(\mathcal{L}_i)$ is selected. Because roots are selected uniformly at random, we have $\mathbb{E}[X_{s_{k+1}} \mid \mathcal{F}_{s_k}] \leq \frac{1}{2} X_{s_k}$.

Let $Y_i = \max(\log(X_i), 0)$. Note that when $Y_{s_k} \geq 2$, we have $\mathbb{E}[Y_{s_{k+1}} \mid \mathcal{F}_{s_k}] = \mathbb{E}[\log(X_{s_{k+1}}) \mid \mathcal{F}_{s_k}] \leq \log(\mathbb{E}[X_{s_{k+1}} \mid \mathcal{F}_{s_k}]) \leq Y_{s_k} - 1$. Let τ be the smallest k such that $Y_{s_k} = 0$. Note that τ is the number of iterations i in which a root from $A_\ell(\mathcal{L}_i)$ is selected, and hence $\tau \leq n$. The sequence $(Y_{s_k} + k)_{k=1,\dots,\tau}$ is therefore a supermartingale and τ is a stopping time. By the martingale stopping theorem [MU05, Theorem 12.2], we have $\mathbb{E}[\tau] = \mathbb{E}[Y_{s_\tau} + \tau] \leq \mathbb{E}[Y_{s_1} + 1] = \log(m) + 1$.

Now we have shown that in expectation at most $\log(m) + 1$ iterations i are needed in which roots from $A_\ell(\mathcal{L}_i)$ are considered. The same argument can be repeated for $A_u(\mathcal{U})$. Since in every iteration a root from $A_\ell(\mathcal{L})$ or $A_u(\mathcal{U})$ is selected, the expected total number of iterations is at most $2\log(m) + 2$. This directly implies the lemma as $m \leq |T_\mathcal{L}| + 1 \leq n + 1$. $\square$

Finally, we are able to prove the running time bound.

Lemma 5.3.5. *Let $R(T, n, k)$ denote the running time of a call* Extend*(T, n, k, $\mathcal{L}_0$). Then there exists $C > 0$ such that*

$$\mathbb{E}[R(T, n, k)] \leq 5C(n - k)\log(n)^3 + Cn\log(n)^2.$$

Proof. We will prove this with induction on $r := \lceil \log(n) \rceil$. For $r = 1$, we have $n \leq 2$. In this case R is constant, proving our induction base.

Now consider a call Extend(T, n, k, $\mathcal{L}_0$) and assume the induction claim is true when $\lceil \log(n) \rceil \leq r - 1$. Let p be the number of iterations of the outer-most while-loop that are executed.

We will first consider the running time induced by the base call itself, excluding any recursive subcalls. Note that all of this running time is incurred by the calls to the procedures DFS, Roots and GoodValues, plus the cost of moving to the corresponding node before each of these calls. In the base call, the procedure will only move between nodes that are among the ones with the n smallest values, or the nodes directly below them. For this reason, we can upper bound the cost of each movement action by a running time of $O(n)$.

- On line 12, 13, 23 each call DFS incurs a running time of at most $O(n)$. Each of these lines will be executed p times, incurring a total running time of $O(pn)$.

- On line 17 each call DFS(T, $\mathcal{L}'$, n) incurs a running time of at most $O(n)$. The code will be executed $O(p\log(n))$ times since c' doubles after every iteration of the inner loop and never grows larger than n, thus incurring a total running time of $O(pn\log(n))$.
- The call to GoodValues on line 20 takes $O(n\log(n))$ time. The line is executed at most p times, so the total running time incurred is $O(pn\log(n))$.

Adding up all the running times listed before, we see that the total running time incurred by the non-recursive part is $O(pn\log(n))$. By Lemma 5.3.4, $\mathbb{E}[p] \leq \log(n)$. Hence, we can choose C such that the expected running time of the non-recursive part is bounded by

$$Cn\log(n)^2.$$

Now we move on to the recursive part of the algorithm. All calls to Extend(T, n, k, $\mathcal{L}_0$) with $k = 0$ will have $n = 1$, so each of these calls takes only $O(1)$ time. Hence, we can safely ignore these calls.

Let z be the number of recursive calls to Extend(T, n, k, $\mathcal{L}_0$) that are done from the base call with $k \geq 1$. Let T_i, k_i, n_i for $i \in [z]$ be the arguments of these function calls. Note that $n/2 \geq n-k \geq n_i \geq 2$ for all i. By the induction hypothesis, the expectation of the recursive part of the running time is:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{z} R(T_i, n_i, k_i)\right] &\leq \mathbb{E}\left[\sum_{i=1}^{z} 5C(n_i - k_i)\log(n_i)^3 + Cn_i\log(n_i)^2\right] \\
&\leq 5C\log(n/2)^3\,\mathbb{E}\left[\sum_{i=1}^{z} n_i - k_i\right] + C\log(n/2)^2\,\mathbb{E}\left[\sum_{i=1}^{z} n_i\right] \\
&\leq 5C(\log(n)-1)\log(n)^2\,\mathbb{E}\left[\sum_{i=1}^{z} n_i - k_i\right] + C\log(n)^2\,\mathbb{E}\left[\sum_{i=1}^{z} n_i\right] \\
&\leq 5C(\log(n)-1)\log(n)^2(n-k) + 5C\log(n)^2(n-k) \\
&\leq 5C(n-k)\log(n)^3.
\end{aligned}$$

Here we used Lemma 5.3.3 as well as the fact that $\sum_{i=1}^{z} n_i \leq 4(n-k)$. To see why the latter inequality is true, consider an arbitrary root q that has q values under it that are good (with respect to the base call). Now $\sum_{i=1}^{z} \mathbf{1}_{T_i = T^{(q)}} n_i \leq \sum_{i=2}^{\lceil\log(s+1)\rceil} 2^i \leq 2^{\lceil\log(s+1)\rceil+1} \leq 4s$. In total there are $n-k$ good values under the roots, and hence $\sum_{i=1}^{z} n_i \leq 4(n-k)$.

Adding the expected running time of the recursive and the non-recursive part, we see that

$$\mathbb{E}[R(T,n,k)] \leq 5C(n-k)\log(n)^3 + Cn\log(n)^2.$$

□

This now implies the desired running time for the procedure Select.

Theorem 5.3.6. *The procedure Select(n) runs in expected $O(n\log(n)^3)$ time.*

Proof. The key idea is that Select calls Extend$(T,k',k,\mathcal{L})$ at most $\lceil\log(n)\rceil$ times with parameters $(k',k)=(2^i,2^{i-1})$ for $i\in\{1,\dots,\lceil\log(n)\rceil\}$. By Lemma 5.3.5, the running time of Select can thus be upper bounded by

$$\sum_{i=1}^{\lceil\log(n)\rceil}\mathbb{E}[R(T,2^i,2^{i-1})]\le 5C\log(n)^3\sum_{i=1}^{\lceil\log(n)\rceil}(2^i-2^{i-1})+\sum_{i=1}^{\lceil\log(n)\rceil}Cn\log(n)^2$$
$$=O(n\log(n)^3).$$

□

Space complexity analysis

We prove in this section the space complexity of our main algorithm.

Theorem 5.3.7. *The procedure Select(n) runs in $O(\log(n))$ space.*

Proof. Observe that it is enough to prove that the statement holds for Extend$(T,n,k,\mathcal{L})$ with $k\ge n/2$, since the memory can be freed up (only keeping the returned value in memory) after every call to Extend in the Select(n) algorithm.

Moreover, observe that the subroutines DFS, Roots and GoodValues all require $O(1)$ memory, as argued in their respective analyses. Any call Extend$(T,n,k,\mathcal{L})$ only makes recursive calls Extend$(T^{(r)},\hat{n},\hat{k},\hat{\mathcal{L}})$ with $1\le\hat{n}\le n-k\le\frac{1}{2}n$. So the depth of the recursion is at most $\log(n)$, and the space complexity of the algorithm is $O(\log(n))$. □

5.4 Lower bound

No lower bound is known for the running time of the selection problem on explorable heaps. However, we will show that any (randomized) algorithm with space complexity at most s, has a running time of at least $\Omega(n\log_s(n))$. Somewhat surprisingly, the tree that is used for the lower bound construction is very simple: a root with two trails of length $O(n)$ attached to it.

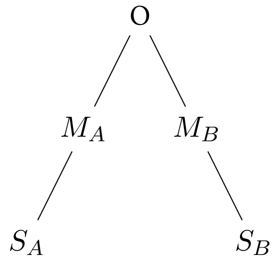
We will make use of a variant of the communication complexity model. In this model a totally ordered set W is given, which is partitioned into (S_A,S_B). There are two agents A and B, that have access to the sets of values in S_A and S_B respectively. We have $|S_A| = n+1$ and $|S_B| = n$. Assume that all values S_A and S_B are different. Now consider the problem where player A wants to compute the median, that is the $(n+1)$th smallest value of W.

Because the players only have access to their own values, they need to communicate. For this purpose they use a protocol, that can consist of multiple rounds. In every odd round, player A can do computations and send units of information to player B. In every even round, player B does computations and sends information to player A. We assume that sending one value from S_A or S_B takes up one *unit of information.* Furthermore, we assume that, except for comparisons, no operations can be performed on the values. For example, the algorithm cannot do addition or multiplication on the values.

We will now reduce the median computation problem to the explorable heap selection problem.

Lemma 5.4.1. *If there is a (randomized) algorithm that solves* SELECT$(3n)$ *in* $f(n)n$ *time and* g *space, then there is a (randomized) protocol for median computation that uses* $f(n)/2$ *rounds in each of which at most* g *units of information are sent.*

Proof. Consider arbitrary sets S_A and S_B with $|S_A| = n+1$ and $|S_B| = n$ and $S_A \cap S_B = \emptyset$. Introduce a new element O, such that $O < x$ for all $x \in S_A \cup S_B$. Let M_A and M_B be two sets with $|M_A| = |M_B| = n$ and $O < y < x$ for all $y \in M_A \cup M_B$ and $x \in S_A \cup S_B$. Write $S_A = \{a_1, \ldots, a_{n+1}\}$. Now consider a subtree for which the root node has value a_1. Let every node that has value a_i have a child with value a_{i+1} and another child with some value that is larger than any value in $S_A \cup S_B \cup M_A \cup M_B$. We will call this a *trail* of S_A.



Now we will construct a labeled tree in the following way: create a tree with a root node of value 0. Attach a trail of M_A as the left child of this root and a trail of M_B as the right child. Attach a trail of S_A as a child of the largest node in M_A and do the same for a trail of M_B under the largest node of S_B. The resulting tree will now look as shown in the above picture.

Observe that the $3n$th smallest value in this tree is the median of $S_A \cup S_B$. Now we can view the selection algorithm as an algorithm for median computation if we consider moving between S_A and S_B in the tree as sending the g units of information that are in memory to the other player. Because moving between the two sets takes at least $2n$ steps, the number of rounds in the corresponding communication protocol is at most $\frac{f(n)n}{2n} = f(n)/2$, proving the statement. $\square$

Lemma 5.4.2. *Let $S \subseteq [n]$ be a randomly distributed subset of $[n]$ with size $|S| \leq k \leq n$. Then for $\ell \leq \frac{n}{8k}$ there exists a length-ℓ interval $\subseteq [n]$ (i.e. $I = \{i, i+1, \ldots, i+\ell-1\}$) such that:* $\Pr[S \cap I \neq \emptyset] \leq \frac{1}{4}$.

Proof. Let $\mathcal{I}_\ell$ be the set of length-ℓ intervals in $[n]$. We have $|\mathcal{I}_\ell| = n - \ell + 1$. Observe that any value in $[n]$ is contained in at most ℓ elements of $\mathcal{I}_\ell$. Hence, for any set S of size at most k, there are at most $k \cdot \ell$ elements of $\mathcal{I}_\ell$ that contain any of the elements of S. That is: $|\{I \in \mathcal{I}_\ell : I \cap S \neq \emptyset\}| \leq k \cdot \ell$. This implies that for a randomly distributed set $S \subseteq [n]$ we also have:

$$\begin{aligned}\sum_{I \in \mathcal{I}_\ell} \Pr_S[I \cap S \neq \emptyset] = \sum_{I \in \mathcal{I}_\ell} \mathbb{E}_S[\mathbf{1}_{I \cap S \neq \emptyset}] &= \mathbb{E}_S\left[\sum_{I \in \mathcal{I}_\ell} \mathbf{1}_{I \cap S \neq \emptyset}\right] \\ &= \mathbb{E}_S[|\{I \in \mathcal{I}_\ell : I \cap S \neq \emptyset\}|] \leq k \cdot \ell.\end{aligned}$$

So there must be an $I \in \mathcal{I}_\ell$ with:

$$\Pr_S[I \cap S \neq \emptyset] \leq \frac{k \cdot \ell}{|\mathcal{I}_\ell|} \leq \frac{k \cdot \ell}{n - \ell + 1} \leq \frac{k \cdot \frac{n}{8k}}{\frac{1}{2}n} = \frac{1}{4}.$$

$\square$

Theorem 5.4.3. *Any randomized protocol for median computation that sends at most g units of info per round, takes at least $\Omega(\log_{g+1}(n))$ rounds in expectation.*

Proof. We will instead prove the following result for a symmetric version of median computation, because this makes the proof a bit easier. In this setting, we have $|S_A| = |S_B| = n$ and the objective is to find both the nth and the $(n+1)$th smallest element of $S_A \cup S_B$. We will call the set consisting of these two values the 2*-median* of $S_A \cup S_B$ and we will denote it by $\mathsf{2median}(S_A \cup S_B)$. Because this problem can be trivially solved by appending two rounds to any median-computation algorithm, proving a lower bound for this case is sufficient.

Let $g' = g + 1$. We can assume that $g \geq 1$, and hence $g' \geq 2$. We will prove with induction on n that the expected number of rounds to compute the median is at least $\frac{1}{10}\log_{g'}(n) - 9$. For $n < 2^8(g')^2$, this is trivial. Now let $n \geq 2^8(g')^2$. Assume that the claim is true for values strictly smaller than n. We will now prove the claim for n.

Consider an arbitrary randomized algorithm. Let V_i be the set of indices in S_{p_i} of the values that are emitted during the round i. Observe that the distribution of the set V_1 does not depend on the input, because player A only has access to his own set of n values that he can compare to each other. Order the values in S_A by their values as $x_1, \ldots, x_n$. Order the values of S_B in decreasing order as $y_1, \ldots, y_n$.

Let $\ell = \lfloor \frac{n}{8g} \rfloor$. From Lemma 5.4.2 it follows that there exists an interval $I = \{a, \ldots, a + \ell - 1\} \subseteq [n]$ such that $\Pr[V_1 \cap I \neq \emptyset] \leq \frac{1}{4}$. Now let $L = \{1, \ldots, a-1\}$ and $U = \{a + \ell, \ldots, n\}$. Observe that $\{L, I, U\}$ forms a partition of $[n]$. Now order the values in the sets such that we have $y_u < x_l < y_i < x_u < y_l$ for all $l \in L, u \in U, i \in I$. Note that this fixes the ordinal index of any element in $S_A \cup S_B$, except for the elements x_i and y_i for $i \in I$.

Condition on the event that $I \cap V_1 = \emptyset$. Observe that in this case, the results of all comparisons that player 2 can do in the second round have been fixed. Hence, V_2 will be a random subset of $[n]$, whose distribution will not depend on the order of the values $x_a, \ldots, x_{a+\ell+1}$ with respect to $y_1, \ldots, y_n$.

Let $\ell' = \lfloor \frac{\ell}{8g} \rfloor$. From Lemma 5.4.2 there exist an interval $I' = \{a', \ldots, a' + \ell' - 1\} \subseteq I$ such that $\Pr[I' \cap V_2 \neq \emptyset \mid I \cap V_1 = \emptyset] \leq \frac{1}{4}$. Define $L' = \{a, \ldots, a' - 1\}$ and $U' = \{a' + \ell', \ldots, a + \ell - 1\}$. Observe that $\{L', I', U'\}$ forms a partition of I. We now order the values in the sets such that we have $y_u < x_l < y_i < x_u < y_l$ for all $l \in L', u \in U', i \in I'$. Note that we have now fixed the ordinal index of any element in $S_A \cup S_B$, except for the elements x_i and y_i for $i \in I'$.

Because $I' \subseteq I$, we have $\Pr[I' \cap (V_1 \cup V_2) \neq \emptyset] \leq \Pr[I \cap V_1 \neq \emptyset] + \Pr[I' \cap V_2 \neq \emptyset \mid S \cap V_1 = \emptyset] \leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. Now define $S'_A = \{x_i : i \in I'\}$ and $S'_B = \{y_i : i \in I'\}$. Observe that $\mathsf{2median}(S_A \cup S_B) = \mathsf{2median}(S'_A \cup S'_B)$. So the algorithm can now be seen as an algorithm to compute the 2-median of $S'_A \cup S'_B$. However, with some probability $\varphi := Pr[I' \cap (V_1 \cup V_2) = \emptyset] \geq \frac{1}{2}$, no information about S'_A and S'_B is transmitted in the first two rounds. So, we can consider the algorithm that leaves out these two first rounds whenever this happens. If R' is the number of rounds that this algorithm takes, then $\mathbb{E}[R'] = \varphi \, \mathbb{E}[R-2] + (1 - \varphi) \, \mathbb{E}[R] = \mathbb{E}[R] - 2\varphi \leq \mathbb{E}[R] - 1$.

By our induction hypothesis it follows that R' will satisfy $\mathbb{E}[R'] \geq \frac{1}{10} \log_{g'}(|S'_B|) - 9 = \frac{1}{8} \log_{g'}(\ell') - 9 = \frac{1}{10}(\log_{g'}(n) - 2\log_{g'}(8g) - 2) - 9 \geq \frac{1}{10} \log_{g'}(n) - 10$. So $\mathbb{E}[R] \geq \mathbb{E}[R'] + 1 \geq \frac{1}{10} \log_{g'}(n) - 9$. □

By Lemma 5.4.1, this directly implies:

Theorem 5.4.4. *The time complexity of any randomized algorithm for* $\mathsf{SELECT}(n)$ *with at most g units of storage is* $\Omega(\log_{g+1}(n)n)$.

5.5 Conclusion

We have introduced a new algorithm for the explorable heap selection problem with running time $O(n \log(n)^3)$ with space complexity $O(\log(n))$. We also show that any algorithm of the same space complexity will need at least $\Omega(n \log(n)/\log\log n)$. It is an interesting open problem to further close this gap.

This question becomes even more interesting in the full-space setting. Here, our algorithm is still the best known algorithm, but best known lower bound is only $\Omega(n)$. This raises the question whether there is a better algorithm for this setting.

Another interesting aspect is the fact that our algorithm is a randomized algorithm. While the algorithm from [KSW86] is also randomized, their algorithm can be derandomized at small cost to the space complexity. Our algorithm, on the other hand, fundamentally relies on randomness. It would be interesting to see if there is a deterministic algorithm with similar running time. Such a deterministic algorithm would also work against adaptive adversaries, while our algorithm only works against oblivious adversaries.

Chapter 6

Online hypergraph matching

The problem of online matching was first studied for bipartite graphs with one-sided vertex arrivals. In this setting it has been shown that the optimal competitive ratio is $1 - 1/e$ for both the integral and the fractional version of the problem. Since then, there has been considerable effort to find optimal competitive ratios for other related settings.

In this chapter, we go beyond the graph case and study the online matching problem on k-uniform hypergraphs. For $k = 3$, we provide an optimal primal-dual fractional algorithm, which achieves a competitive ratio of $(e-1)/(e+1) \approx 0.4621$. We also present a carefully constructed adversarial instance, which shows that this ratio is in fact optimal. For $k \geq 3$, we give a simple integral algorithm which performs better than greedy when the online nodes have bounded degree.

6.1 Introduction

Online matching is a classic problem in the field of online algorithms. It was first introduced in the seminal work of Karp, Vazirani and Vazirani [KVV90], who considered the bipartite version with one-sided vertex arrivals. In this setting, we are given a bipartite graph where vertices on one side are known in advance (offline), and vertices on the other side arrive sequentially (online). When an online vertex arrives, it reveals its incident edges, at which point the algorithm must decide how to match it (or not to match it) irrevocably. The goal is to maximize the cardinality of the resulting matching. Karp et al. [KVV90] gave an elegant randomized algorithm Ranking, which achieves the optimal competitive ratio of $1 - 1/e$.

In certain applications, each offline vertex may be matched more than once. Examples include matching online jobs to servers, or matching online impressions to advertisers. This is the online b-matching model of Kalyanasundaram and Pruhs [KP00], where $b \geq 1$ is the maximum number of times an offline vertex can be matched. As b and the number of online vertices tend to infinity, it in turn captures

The contents of this chapter are based on joint work with Danish Kashaev and Zhuan Khye Koh [BKK24].

the fractional relaxation of the Karp et al. [KVV90] model. This means that the algorithm is allowed to match an online node fractionally to multiple neighbors, as long as the total load on every vertex does not exceed 1. For this problem, it is known that the deterministic algorithm BALANCE (or WATER-FILLING) achieves the optimal competitive ratio of $1 - 1/e$.

Online hypergraph matching

The online bipartite matching can be naturally generalized to hypergraphs as follows. For $k \geq 2$, let $\mathcal{H} = (V, W, H)$ be a k-uniform hypergraph with offline vertices V, online vertices W and hyperedges H. Every hyperedge $h \in H$ contains $k - 1$ elements from V and 1 element from W. Just like before, the online vertices arrive sequentially with their incident hyperedges, and the goal is to select a large matching. The greedy algorithm is $1/k$-competitive. On the other hand, no integral algorithm can be $2/k$-competitive [TU24][1]. A major open question is whether there exists an integral algorithm better than greedy.

For the *fractional* version of the problem, Buchbinder and Naor [BN09] constructed an instance on which any algorithm is at most H_k-competitive, where H_k is the kth harmonic number. In particular, this shows that no algorithm can be $1/\log k$-competitive. They also gave a deterministic algorithm which is $\Omega(1/\log k)$-competitive. In fact, their results apply to the more general setting of online packing linear programs, in which variables arrive sequentially. In the context of hypergraphs, this means that the hyperedges arrive sequentially. Recently, it was shown that for this edge-arrival model on k-uniform hypergraphs, the algorithm can be fine-tuned to achieve a competitive ratio of $(1 - o(1))/\log k$ [TU24], which reduces the gap to the upper bound from multiplicative to additive. Note that for k-uniform hypergraphs, there is a trivial reduction from this *edge-arrival* model to our *vertex-arrival* model on $(k + 1)$-uniform hypergraphs, by adding degree 1 online nodes.

One might wonder whether algorithms for the fractional version of the problem can be converted into integral algorithms. In the b-matching model, in which vertices can be matched to at most b hyperedges, this can be done through randomized rounding whenever $b = \Omega(\log k)$. This way, a randomized integral algorithm can be obtained that is $\Omega(1/\log k)$-competitive in expectation. We provide a proof in Section 6.B. However, this method fails for smaller values of b.

The aforementioned results show that asymptotically, both integral and fractional versions of the online matching problem on k-uniform hypergraphs are essentially settled (up to an additive constant). However, our understanding of the problem

[1]In [TU24], it is shown that no algorithm can be $(2 + f(k))/k$-competitive for some positive function f with $f(k) = o(1)$. Using a similar technique one can show that no algorithm can be $2/k$-competitive. We provide a proof in Section 6.A.

for small values of k (other than $k = 2$) remains poor. Many applications of online hypergraph matching in practice have small values of k. For instance, in ride-sharing and on-demand delivery services [PSST22], $k - 1$ represents the capacity of service vehicles, which is often small. Another example is network revenue management problems [MRST20]. In this setting, given a collection of limited resources, a sequence of product requests arrive over time. When a product request arrives, we have to decide whether to accept it irrevocably. Accepting a product request generates profit, but also consumes a certain amount of each resource. The goal is to devise a policy which maximizes profit. In this context, $k - 1$ represents the maximum number of resources used by a product. As Ma et al. [MRST20] noted, many of these problems have small values of k. In airlines, for example, $k - 1$ corresponds to the maximum number of flight legs included in an itinerary, which usually does not exceed two or three.

Our contributions

Motivated by the importance of online hypergraph matching for small values of k, we focus on 3-uniform hypergraphs, with the goal of obtaining tighter bounds. Our main result is a tight competitive ratio for the fractional version of this problem.

Theorem 6.1.1. *For the online fractional matching problem on* 3*-uniform hypergraphs, there is a deterministic* $(e - 1)/(e + 1)$*-competitive algorithm. Furthermore, every algorithm is at most* $(e - 1)/(e + 1)$*-competitive.*

The deterministic algorithm in Theorem 6.1.1 belongs to the class of Water-Filling algorithms. It uses the function $f(x) := e^x/(e + 1)$ to decide which hyperedges receive load. In particular, for every online vertex w, the incident hyperedges $h = \{u, v, w\} \in \delta(w)$ which minimize $\varphi(h) := f(x(\delta(u))) + f(x(\delta(v)))$ receive load until $\varphi(h) \geq 1$.

The upper bound in Theorem 6.1.1 is technically the most challenging part of the chapter. We construct an instance which is adaptive to the actions of the algorithm. The key idea is to combine two hard instances for online matching on *bipartite graphs* [KVV90; Gam+19]. The instance in [KVV90], designed for one-sided vertex-arrivals, relies on uncertainty about the neighbors of each vertex. On the other hand, the instance in [Gam+19], designed for edge-arrivals, relies on uncertainty about the size of the graph. To make the combination work, a more fine-grained understanding of these two instances is necessary. An important ingredient of our analysis is the (fractional) degree distribution of vertices in the edge-arrival instance [Gam+19]. We remark that our instance is a tripartite hypergraph.

Our next result concerns the online integral matching problem on k-uniform hypergraphs. We show that one can do better than the greedy algorithm if the online nodes have bounded degree. It is achieved by the simple algorithm Random: for every online vertex w, uniformly select a hyperedge among all the hyperedges incident to w which are disjoint from the current matching.

Theorem 6.1.2. *For the online matching problem on k-uniform hypergraphs where online vertices have maximum degree d, the competitive ratio of* Random *is at least*

$$\min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right).$$

Note that in Theorem 6.1.2, the first term is at most the second term if and only if $d \leq k-1$. Moreover, Random always performs better than the greedy algorithm, since the latter is $1/k$-competitive. A nice application of Theorem 6.1.2 is for 3-uniform hypergraphs with $d = 2$. In this setting, Random is $1/2$-competitive and is in fact optimal. This is because the online matching problem on graphs under edge arrivals is a special case of this setting (with $k = 3, d = 1$), for which an upper bound of $1/2$ is known even against fractional algorithms on bipartite graphs [Gam+19].

Since every randomized algorithm for integral matching induces a deterministic algorithm for fractional matching, the upper bound of $(e-1)/(e+1) \approx 0.4621$ in Theorem 6.1.1 also applies to the integral problem on 3-uniform hypergraphs. However, the best known lower bound is 1/3, given by the greedy algorithm. An interesting question for future research is whether there exists an integral algorithm better than greedy on 3-uniform hypergraphs.

Related work

Since the online matching problem was introduced in [KVV90], it has garnered a lot of interest, leading to extensive follow-up work. We refer the reader to the excellent survey by Mehta [Meh13] for navigating this rich literature. The original analysis of Ranking [KVV90] was simplified in a series of papers [BM08; DJK13; GM08; EFFS21]. Many variants of the problem have been studied, such as the online b-matching problem [KP00], and its extension to the AdWords problem [BJN07; DJ12; HZZ20; MSVV07]. Weighted generalizations have been considered, e.g., vertex weights [AGKM11; HTWZ19] and edge weights [FHTZ22]. Weakening the adversary by requiring that online nodes arrive in a random order has also been of interest [KMT11; MY; KRTV13]. Another line of research explored more general arrival models such as two-sided vertex arrival [WW15], general vertex arrival [Gam+19], edge arrival [BST19; Gam+19], and general vertex arrival with departure times [Hua+20; HTWZ20; Ash+23].

In contrast, the literature on the online hypergraph matching problem is relatively sparse. Most work has focused on stochastic models, such as the random-order model. Korula and Pal [KP09] first studied the edge-weighted version under this model. For k-uniform hypergraphs, they gave an $\Omega(1/k^2)$-competitive algorithm. This was subsequently improved to $\Omega(1/k)$ by Kesselheim et al. [KRTV13]. Ma et al. [MRST20] gave a $1/k$-competitive algorithm under 'nonstationary' arrivals. Pavone et al. [PSST22] studied online hypergraph matching with delays under the adversarial model. At each time step, a vertex arrives, and it will depart after d time steps. A hyperedge is revealed once all of its vertices have arrived. Note that their model is incomparable to ours because every vertex has the same delay d.

In the prophet-IID setting, in which for each online node the weight-function is independently sampled from the same distribution, an $O(\log k/k)$ hardness is known [MSV23]. For a good overview of known results for random settings we refer to [MSV23].

Hypergraph matching on k-uniform hypergraphs is a well-studied problem in the offline setting. It is known to be NP-hard to approximate within a factor $\Omega(\log(k)/k)$ [HSS06]. The factor between the value of the natural LP-relaxation and the optimal integral solution is also known to be at least $1/(k-1+1/k)$ for k-uniform hypergraphs [CL12].

A special case that has been studied is the restriction to k-partite graphs, where the vertices are partitioned into k sets and every hyperedge contains exactly one vertex from each set. This setting is known as k-dimensional matching. In this setting, the optimal solution is known to be at least $1/(k-1)$ times the optimal objective to the standard LP-relaxation [HSS06]. For $k = 3$, the best known polynomial time approximation algorithm gives a $(2/3-\varepsilon)$-approximation [CGM13]. Since our construction in Section 6.4 is 3-partite, it also gives an upper bound on the competitive ratio of any fractional algorithm for online 3-dimensional matching.

Organization

In Section 6.2, we give the necessary preliminaries and discuss notation. Section 6.3 presents the optimal primal-dual fractional algorithm for 3-uniform hypergraphs, which shows the first part of Theorem 6.1.1. Section 6.4 complements this with a tight upper bound, proving the second part of Theorem 6.1.1. The proof of our result for hypergraphs with online vertices of bounded degree is in Section 6.5. We conclude with an outlook on future research in Section 6.6.

6.2 Preliminaries

Given a hypergraph $\mathcal{H} = (V, H)$, where H denotes the hyperedges, the maximum matching problem consists of finding a maximum cardinality subset of disjoint hyperedges. The primal and dual linear programming relaxations for this problem are given by:

$$\begin{aligned} \max \sum_{h\in H} x_h & \\ \sum_{h\in\delta(v)} x_h \le 1 & \quad \forall v \in V \\ x_h \ge 0 & \quad \forall h \in H \end{aligned} \qquad\qquad \begin{aligned} \min \sum_{v\in V} y_v & \\ \sum_{v\in h} y_v \ge 1 & \quad \forall h \in H \\ y_v \ge 0 & \quad \forall v \in V. \end{aligned}$$

We denote by $\mathsf{OPT}_{\mathsf{LP}}(\mathcal{H})$ the fractional offline optimal objective value of these two linear programs, which is the same for both programs by strong duality. We denote by OPT$(\mathcal{H})$ the objective value of the optimal integral offline solution for the primal linear program, which clearly satisfies OPT$(\mathcal{H}) \le \mathsf{OPT}_{\mathsf{LP}}(\mathcal{H})$.

Consider the online matching problem on k-uniform hypergraphs under vertex arrivals. Formally, an instance consists of a k-uniform hypergraph $\mathcal{H} = (V, W, H)$, where $W = (w_1, w_2, \ldots)$ is a sequence of online nodes and V is a set of offline nodes. The ordering of W corresponds to the arrival order of the online nodes. Each hyperedge $h \in H$ has exactly one node in W and $k-1$ nodes in V. At the arrival of an online node $w \in W$, a fractional algorithm is allowed to increase x_h for every $h \in \delta(w)$, and an integral algorithm can irrevocably pick one of these hyperedges.

For a given algorithm $\mathcal{A}$ and input instance $\mathcal{H}$, we denote by

$$\text{val}(\mathcal{A}, \mathcal{H}) := \sum_{h\in H} x_h$$

the value of the (fractional) matching obtained by the algorithm on instance $\mathcal{H}$. A fractional algorithm $\mathcal{A}$ is called ρ-competitive if for any instance $\mathcal{H}$, val$(\mathcal{A}, \mathcal{H}) \ge \rho\, \mathsf{OPT}_{\mathsf{LP}}(\mathcal{H})$.

A large part of this chapter is concerned with 3-uniform hypergraphs. For a 3-uniform vertex arrival instance $\mathcal{H} = (V, W, H)$, we denote by $\Gamma(\mathcal{H}) = (V, E)$ the graph on the offline nodes with edge set

$$E := \Big\{(u, v) \in V \times V, \quad \exists w \in W \text{ s.t. } \{u, v, w\} \in H\Big\}. \tag{6.1}$$

We remark that $\Gamma(\mathcal{H})$ is not a multigraph. In particular, an edge $(u, v) \in E$ can have several hyperedges in H containing it. A fractional matching x on the hyperedges H

naturally induces a fractional matching x' on the edges E by setting $x'_e = \sum_{e \subseteq h} x_h$ for every $e \in E$. The value obtained by an algorithm in this model can thus also be counted as $\text{val}(\mathcal{A}, \mathcal{H}) = \sum_{h \in H} x_h = \sum_{e \in E} x'_e$.

6.3 Optimal algorithm for 3-uniform hypergraphs

In this section, we present a primal-dual algorithm for the online fractional matching problem on 3-uniform hypergraphs under vertex arrivals. This algorithm will turn out to be optimal in this setting with a tight competitive ratio of $(e-1)/(e+1) \approx 0.4621$. We define the following distribution function $f : [0,1] \to [0,1]$:

$$f(x) := \frac{e^x}{e+1}. \tag{6.2}$$

When an online node w arrives, our algorithm chooses to continuously increase the fractional primal value on the hyperedges $\{u, v, w\}$ for which $f(x(\delta(u))) + f(x(\delta(v)))$ is minimal. We note that this belongs to the class of *water-filling* algorithms [KP00]. For this reason, we define the *priority* of a hyperedge $h = \{u, v, w\}$ as:

$$\varphi(h) := f(x(\delta(u))) + f(x(\delta(v))). \tag{6.3}$$

Algorithm 3 Water-filling fractional algorithm $\mathcal{A}$ for online vertex arrival

```
Input : 3-uniform hypergraph $\mathcal{H} = (V, W, H)$ with online arrivals of $W$.
Output : Fractional matching $x \in [0,1]^H$

when $w \in W$ arrives with $\delta(w) \subseteq H$:
    set $x_h = 0$ for every $h \in \delta(w)$
    increase $x_h$ for every $h = \{u, v, w\} \in \arg\min_{h \in \delta(w)}\{\varphi(h)\}$ at rate 1
    increase $y_u$ and $y_v$ at rates $f(x(\delta(u)))$ and $f(x(\delta(v)))$
    increase $y_w$ at rate $1 - f(x(\delta(u))) - f(x(\delta(v)))$
        until $x(\delta(w)) = 1$ or $\varphi(h) \geq 1$ for every $h \in \delta(w)$.
return $x$
```

Theorem 6.3.1. *Algorithm 3 is $(e-1)/(e+1)$-competitive for the online fractional matching problem on 3-uniform hypergraphs.*

Proof. We first show that the algorithm produces a feasible primal solution. Note that the fractional value of a hyperedge h is only being increased if $\varphi(h) \leq 1$. If

$x(\delta(v)) = 1$ for some offline node v then for any hyperedge $h \ni v$, we have:

$$\varphi(h) = f(x(\delta(u))) + f(x(\delta(v))) \geq f(1) + f(0) = \frac{e+1}{e+1} = 1,$$

where u denotes the second offline node belonging to h. The value of the hyperedge h will thus not be increased anymore, proving the feasibility of the primal solution.

In order to prove the desired competitive ratio, we show that the primal-dual solutions constructed during the execution of the algorithm satisfy:

$$\text{val}(\mathcal{A}) = \sum_{h \in H} x_h = \sum_{v \in V \cup W} y_v \qquad \text{and} \tag{6.4}$$

$$\sum_{v \in h} y_v \geq \rho \qquad \forall h \in H. \tag{6.5}$$

This is enough to imply the desired competitiveness of our algorithm, since $y/\rho \in \mathbb{R}^V_+$ is then a feasible dual solution, giving:

$$\text{val}(\mathcal{A}) \geq \sum_{v \in V \cup W} y_v \geq \rho \, \mathsf{OPT}_{\mathsf{LP}}.$$

Note that (6.4) holds at the start of the algorithm. Let us fix a hyperedge $h = \{u, v, w\} \in H$. When x_h is continuously being increased at rate one, the duals on the incident nodes y_u, y_v and y_w are being increased at rate $f(x(\delta(u)))$, $f(x(\delta(v)))$ and $1 - f(x(\delta(u))) - f(x(\delta(v)))$ respectively. Observe that these rates sum up to one. Hence, val$(\mathcal{A}) = \sum_{h \in H} x_h$ and $\sum_{v \in V \cup W} y_v$ are increased at the same rate, meaning that (6.4) holds at all times during the execution of the algorithm.

We now show that (6.5) holds at the end of the execution of the algorithm. Let us fix an online node $w \in W$. For a given hyperedge $h \in \delta(w)$, note that the algorithm only stops increasing x_h, as soon as either $\varphi(h) \geq 1$ or $x(\delta(w)) = 1$ is reached. We distinguish these two cases for the analysis.

Let us first focus on the first case, meaning that $\varphi(h) \geq 1$ has been reached for every $h \in \delta(w)$. Consider an arbitrary $h = \{u, v, w\} \in \delta(w)$. For every unit of increase in $x(\delta(u))$, y_u will have been increased by $f(x(\delta(u)))$. If we denote by $\ell_u := x(\delta(u))$ and $\ell_v := x(\delta(v))$ the fractional loads on u and v after the last increase on the hyperedges adjacent to w, then:

$$y_u = \int_0^{\ell_u} f(s)ds = f(\ell_u) - f(0) \quad \text{and} \quad y_v = \int_0^{\ell_v} f(s)ds = f(\ell_v) - f(0), \tag{6.6}$$

where we have used the fact that f is an antiderivative of itself. Therefore,

$$\begin{aligned} y_u + y_v + y_w \geq y_u + y_v &= f(\ell_u) - f(0) + f(\ell_v) - f(0) \\ &= \varphi(h) - 2f(0) \geq 1 - 2f(0) = \frac{e-1}{e+1}. \end{aligned}$$

Suppose now that $x(\delta(w)) = 1$ has been reached. In particular, this means that for each $\{u, v, w\} \in \delta(w)$, the rate at which y_w was increased must have been at least $1 - f(\ell_u) - f(\ell_v)$ at all times, where ℓ_u and ℓ_v denote the fractional loads on u and v after that the algorithm has finished increasing the edges incident to the online node w. Hence, we have:

$$y_w \geq 1 \cdot (1 - f(\ell_u) - f(\ell_v)).$$

By using (6.6) we see that:

$$\begin{aligned} y_u + y_v + y_w &\geq f(\ell_u) + f(\ell_v) - 2f(0) + (1 - f(\ell_u) - f(\ell_v)) \\ &= 1 - 2f(0) = \frac{e-1}{e+1}. \end{aligned}$$

This proves (6.5), and thus completes the proof of the theorem. □

6.4 Tight upper bound for 3-uniform hypergraphs

We now prove the second part of Theorem 6.1.1, restated below.

Theorem 6.4.1. *Every algorithm is at most* $(e-1)/(e+1)$*-competitive for the online fractional matching problem on 3-uniform hypergraphs.*

Overview of the construction

We prove Theorem 6.4.1 by constructing an adversarial instance that is adaptive to the behavior of the algorithm. The main idea is to combine the vertex-arrival instance of Karp et al. [KVV90] and the edge-arrival instance of Gamlath et al. [Gam+19] for bipartite graphs.

We start by giving a high-level overview of the construction. The offline vertices of the hypergraph can be partitioned into m sets $C_1, \dots, C_m$, which we call *components*. Each component can be seen as a bipartite graph with bipartition $C_i = U_i \cup V_i$, where $|U_i| = |V_i| = T$.

The global instance consists of T phases. In each phase $t \in \{1, \dots, T\}$, the adversary first selects a bipartite matching $\mathcal{M}_i^{(t)}$ on each component C_i. Taking the union of all these bipartite matchings gives a larger matching on the offline nodes:

$$\mathcal{M}^{(t)} = \bigcup_{i=1}^{m} \mathcal{M}_i^{(t)} \qquad \forall t \in \{1, \dots, T\}.$$

After selecting the matching $\mathcal{M}^{(t)}$ at phase t, the adversary selects the online nodes, with their incident hyperedges, arriving in that phase. The set of online nodes arriving in phase t is denoted by $W^{(t)}$. Each node $w \in W^{(t)}$ connects to a subset of edges $E(w) \subseteq \mathcal{M}^{(t)}$, meaning that the hyperedges incident to w are $\{\{w\} \cup e : e \in E(w)\}$.

We briefly explain how the matchings $\mathcal{M}_i^{(t)}$ are constructed and how the edges $E(w)$ are picked:

1. On each component C_i, the matching $\mathcal{M}_i^{(t)}$ is constructed based on the behavior of the algorithm in phase $t - 1$. It draws inspiration from the edge-arrival instance in [Gam+19], together with the function $f(x) = e^x/(e+1)$ defined in (6.2). The exact construction is described in Section 6.4 and illustrated in Fig. 6.4.

2. For every online node $w \in W^{(t)}$, the edge set $E(w) \subseteq \mathcal{M}^{(t)}$ is selected based on the behavior of the algorithm during phase t. This part can be seen as incorporating the vertex-arrival instance in [KVV90]. The exact construction is described in Lemma 6.4.4 and illustrated in Figure 6.2.

To summarize, the global instance is a hypergraph $\mathcal{H} = (V, W, H)$ with offline nodes V, online nodes W and hyperedges H given by

$$V := \bigcup_{i=1}^{m} C_i = \bigcup_{i=1}^{m} U_i \cup V_i \quad W := \bigcup_{t=1}^{T} W^{(t)} \quad H := \bigcup_{t=1}^{T} \bigcup_{w \in W^{(t)}} \{\{w\} \cup e : e \in E(w)\}.$$

Overview of the analysis

To simplify the analysis of our instance, we will make two assumptions. First, we need the following definition, which relates the behavior of an algorithm to the priority function φ defined in (6.3).

Definition 6.4.2. Fix $\varepsilon \in \mathbb{R}$. Let x be the fractional solution given by an algorithm $\mathcal{A}$ after the arrival of an online node w. We say that $\mathcal{A}$ is *ε-threshold respecting* on w if $\varphi(h) = \sum_{v \in h \setminus \{w\}} f(x(\delta(v))) \leq 1 + \varepsilon$ for all incident hyperedges $h \in \delta(w)$ with

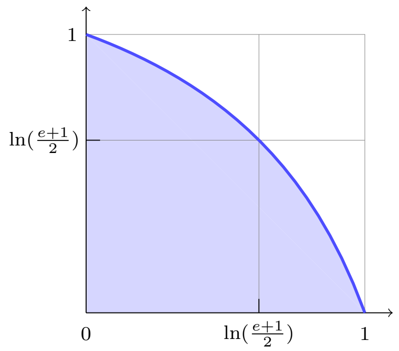


Figure 6.1: An illustration of the region $R = \{(a,b) \in [0,1]^2 : f(a) + f(b) \leq 1\}$. The symmetric point at the boundary of the region has both coordinates $\ln((e+1)/2) \approx 0.62$. When an online node w arrives, a threshold respecting algorithm ensures that the fractional matching x satisfies $(x(\delta(u)), x(\delta(v))) \in R$ for every hyperedge $h = \{u,v,w\} \in \delta(w)$ with $x_h > 0$ at the end of that iteration.

$x_h > 0$. We also call $\mathcal{A}$ *threshold respecting* if $\varepsilon = 0$, and *strictly threshold respecting* if $\varepsilon < 0$.

For a hyperedge $h = \{u,v,w\} \in \delta(w)$, Figure 6.1 shows the possible values of $x(\delta(u))$ and $x(\delta(v))$ such that $\varphi(h) \leq 1$.

Remark 6.4.3. We emphasize that the property in Definition 6.4.2 only needs to hold for the fractional solution x after w has arrived, and before the arrival of the next online node. In particular, it is possible that $\varphi(h) > 1 + \varepsilon$ in later iterations.

The two assumptions that we will make are:

1. The algorithm is *ε-threshold respecting* on all online nodes in the first $T-1$ phases for some arbitrarily small $\varepsilon > 0$.

2. The algorithm is *symmetric* on each component $C_i = U_i \cup V_i$. In particular, for every $t \in \{1,\dots,T\}$, the tth vertex of U_i and V_i have the same fractional degrees throughout the execution of the algorithm.

In Section 6.4, we show that they can be made without loss of generality. Our proof will now consist of two main steps:

1. In Section 6.4, we show that for each phase $t \in \{1,\dots,T-1\}$, the value incurred by the algorithm on that phase can be upper bounded by

$$\frac{e-1}{e+1}m + \varepsilon tm + O\left((tm)^{2/3}\right).$$

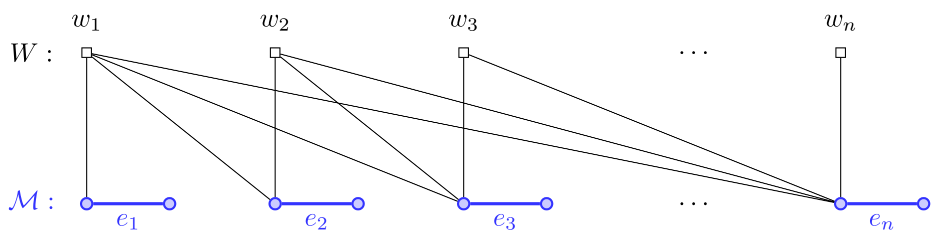


Figure 6.2: An illustration of the instance constructed in Lemma 6.4.4

This means that the total value obtained on the first $T-1$ phases can be upper bounded by

$$\sum_{t=1}^{T-1}\left(\frac{e-1}{e+1}m+\varepsilon tm+O\left((tm)^{2/3}\right)\right)\leq Tm\left(\frac{e-1}{e+1}+\varepsilon T+O\left(\frac{T^{\,2/3}}{m^{\,1/3}}\right)\right)$$

2. In Section 6.4, we show that, for every $i\in\{1,\dots,m\}$, the value gained by the algorithm on the component C_i during the last phase T can be upper bounded by $O(\sqrt{T})+\varepsilon O(T^2)$, meaning that the algorithm gains a value of at most $m\ (O(\sqrt{T})+\varepsilon O(T^2))$ for the last phase.

The offline optimal solution at the end of phase T has size $\mathrm{OPT}=Tm$. Hence, as $T,m\to\infty$ and picking $T=o(\sqrt{m})$ and $\varepsilon=o(1/T)$, we can easily check that the competitive ratio is upper bounded by $(e-1)/(e+1)$.

Bound for the first $T-1$ phases

The following lemma is an adaptation of the vertex-arrival instance for bipartite graphs [KVV90] to tripartite hypergraphs. It gives an upper bound on the value obtained by any fractional algorithm, parametrized by the maximum (fractional) degree $\Delta\in[0,1]$ attained by an offline node.

Lemma 6.4.4. *For any (graph) matching $\mathcal{M}=(V,E)$, there exists an online tripartite hypergraph instance $\mathcal{H}=(V,W,H)$ such that $\Gamma(\mathcal{H})=\mathcal{M}$ and* $\mathrm{OPT}(\mathcal{H})=|\mathcal{M}|$. *Moreover, for any fractional algorithm $\mathcal{A}$ whose returned solution x satisfies $x(\delta(v))\leq\Delta$ for all offline nodes $v\in V$, we have*

$$\mathrm{val}(\mathcal{A},\mathcal{H})\leq(1-e^{-\Delta})|\mathcal{M}|+\frac{3}{2}.$$

Proof. Let us fix a fractional algorithm $\mathcal{A}$ and let us fix a matching $\mathcal{M} = (V, E)$ of size n, meaning that $|E| = n$. The adversarial online 3-uniform hypergraph instance $\mathcal{H}$ consists of n online nodes $W = \{w_1, \dots, w_n\}$ arriving and connecting to a subset of edges of the matching $\mathcal{M}$. For every w_i, we denote by $E(w_i) \subseteq E$ the edges of the matching the online node w_i is connected to, meaning that the 3-hyperedges incident to w_i are $\delta(w_i) = \{w_i \cup e : e \in E(w_i)\}$. Let us denote by $x \in \mathbb{R}^E$ the fractional solution generated online by algorithm $\mathcal{A}$, and note that this is in fact the induced fractional solution on E.

1. The first online node w_1 connects to every edge of the matching, i.e. $E(w_1) = E$. The algorithm $\mathcal{A}$ now assigns fractional value $x(e)$ to every edge $e \in E$, and we denote by $e_1 \in E$ the edge with the lowest fractional value $x(e_1)$. Observe that $x(e_1) \leq 1/n$.

2. The second online node w_2 connects to $E(w_2) = E \setminus \{e_1\}$. The algorithm $\mathcal{A}$ can thus increase the fractional values $x(e)$ for every $e \in E(w_2)$. We then denote by e_2 the edge in $E(w_2)$ with the lowest fractional value after this iteration, and it is easy to check that $x(e_1) + x(e_2) \leq 2/n + 1/(n-1)$.

3. More generally, for every $k \in \{1, \dots, n\}$, the online node w_k connects to $n - k + 1$ edges $E(w_k) = E \setminus \{e_1, \dots, e_{k-1}\}$, and e_k is defined as the edge having the lowest fractional value at the end of the iteration of w_k. We thus get a bound of

$$\sum_{k=1}^{\ell} x(e_k) \leq \sum_{k=1}^{\ell} \sum_{i=1}^{k} \frac{1}{n-i+1} \qquad \forall \ell \in \{1, \dots, n\}. \tag{6.7}$$

The inner sum in (6.7) reaches Δ approximately when $k \approx (1 - e^{-\Delta})n$. For higher values of k, it is thus better to use the bound $x(e_k) \leq \Delta$, which holds by assumption. By defining $p := \lfloor e^{-\Delta} n \rfloor$ and $q := n - p$, we can now compute a precise upper bound on the total value generated by the algorithm using (6.7):

$$\begin{aligned} \text{val}(A, \mathcal{H}) &= \sum_{k=1}^{n} x(e_k) \leq \sum_{k=1}^{q} \sum_{i=1}^{k} \frac{1}{n-i+1} + \sum_{k=q+1}^{n} \Delta = \sum_{i=1}^{q} \sum_{k=i}^{q} \frac{1}{n-i+1} + p\Delta \\ &= \sum_{i=1}^{q} \frac{q-i+1}{n-i+1} + p\Delta = q - (n-q) \sum_{i=1}^{q} \frac{1}{n-i+1} + p\Delta \\ &= p\Delta + (n-p) - p \sum_{i=n-q+1}^{n} \frac{1}{i} = p\Delta + n - p - p(H_n - H_p). \end{aligned} \tag{6.8}$$

In order to get the desired result for every value of $n \geq 1$, we now need to tightly approximate the difference of the harmonic numbers $H_n - H_p$. In particular, the well known bounds $\ln(n) + 1/n \leq H_n \leq \ln(n+1)$ for every $n \in \mathbb{N}$ are not enough in this case. We use the equality

$$H_n = \ln(n) + \gamma + \varepsilon(n) \qquad \text{for some } 0 < \varepsilon(n) < \frac{1}{2n} \tag{6.9}$$

where $\gamma = \lim_{n\to\infty}(H_n - \ln(n)) \approx 0.58$ is Euler's constant. Moreover, recall that

$$e^{-\Delta}n - 1 \leq p \leq e^{-\Delta}n. \tag{6.10}$$

Using (6.9) and (6.10) together gives:

$$H_n - H_p = \ln\left(\frac{n}{p}\right) + \varepsilon(n) - \varepsilon(p) \geq \ln\left(\frac{n}{e^{-\Delta}n}\right) - \frac{1}{2p} = \Delta - \frac{1}{2p}. \tag{6.11}$$

Finally, plugging (6.10) and (6.11) into (6.8) gets us the desired result for every value of $n \in \mathbb{N}$:

$$\operatorname{val}(A, \mathcal{H}) \leq p\Delta + n - p - p\left(\Delta - \frac{1}{2p}\right) = n - p + \frac{1}{2} \leq (1 - e^{-\Delta})n + \frac{3}{2}.$$

□

Given a 3-uniform hypergraph instance $\mathcal{H} = (V, W, H)$, let $\Gamma(\mathcal{H}) = (V, E)$ be the graph on the offline nodes as defined in (6.1). Let $x \in [0,1]^E$ be a fractional matching on $\Gamma(\mathcal{H})$. For an offline node $u \in V$, we denote its *load* (or *fractional degree*) as $\ell_u = x(\delta(u)) \in [0,1]$. For an edge $e = (u, v) \in E$, we overload the priority function φ defined for hyperedges in (6.3) as $\varphi(e) := f(\ell_u) + f(\ell_v)$.

Given an edge $e = (u, v)$ with levels ℓ_u and ℓ_v below the threshold, i.e. satisfying $\varphi(e) = f(\ell_u) + f(\ell_v) < 1$, we can compute how much fractional value is needed to put on that edge in order to make it reach priority one:

$$\begin{aligned}\min\left\{x_e \geq 0 \mid f(\ell_u + x_e) + f(\ell_v + x_e) \geq 1\right\} &= \ln\left(\frac{e+1}{e^{\ell_u} + e^{\ell_v}}\right) \\ &= -\ln\Big(f(\ell_u) + f(\ell_v)\Big) = -\ln\varphi(e).\end{aligned}$$

A similar computation shows that if an edge e is above the threshold, i.e. $\varphi(e) \geq 1$, the fractional value above the threshold is $\ln\varphi(e)$. Note that in both cases, this quantity is non-negative, by definition of the natural logarithm.

Recall from Section 6.4 that our global instance consists of T phases. In order to analyze the value of the algorithm, we split the total value gained by the algorithm into the value gained in each phase. Note that at the beginning of phase t, the algorithm has already generated a certain fractional matching $x^{(t-1)}$, meaning that every offline node v already has some load $\ell_v^{(t-1)}$, with corresponding priorities $\varphi_e^{(t-1)}$ for the edges. We denote the total value gained by an algorithm $\mathcal{A}$ during phase t as:

$$\mathrm{val}^{(t)}(\mathcal{A}) := \sum_{e \in \mathcal{M}^{(t)}} x_e.$$

We first show how, during a fixed phase t, we incorporate the instance of Lemma 6.4.4 in our global instance, and how we are able to analyze the value gained by a threshold respecting algorithm on that phase.

Lemma 6.4.5. *Let $\mathcal{M} = (V, E)$ be a (graph) matching, where every node $v \in V$ has a pre-existing load $\ell_v \in [0,1]$. For every $\delta \in (0,1]$, there exists an online tripartite hypergraph instance $\mathcal{H}$ satisfying $\Gamma(\mathcal{H}) = \mathcal{M}$, $\mathrm{OPT}(\mathcal{H}) = |\mathcal{M}|$ such that*

$$\mathrm{val}(\mathcal{A}, \mathcal{H}) \leq (1+\delta)|\mathcal{M}| - \sum_{v \in V} f(\ell_v) + C\delta^{-2}$$

against any threshold respecting algorithm $\mathcal{A}$, where C is a constant satisfying $C \leq 10$.

Remark 6.4.6. Since E is a matching, one can also rewrite this bound as

$$\mathrm{val}(\mathcal{A}, \mathcal{H}) \leq \sum_{(u,v) \in E} \Big(1 - f(\ell_u) - f(\ell_v)\Big) + \delta|\mathcal{M}| + C\delta^{-2}.$$

Let us give intuition behind this lemma. At the beginning of phase t, the matching $\mathcal{M}^{(t)}$ will have pre-existing loads $\ell_v^{(t-1)}$, obtained after the $t-1$ previous phases. This lemma allows to relate the value on this phase, parametrized by the pre-existing loads, and will then be used in an inductive argument. At the beginning of the first phase, all the loads satisfy $\ell_v = 0$. Plugging this into the above with $|\mathcal{M}| = n$ and $\delta = n^{-1/3}$ gives a competitive ratio of

$$\frac{n - 2\,n\,f(0) + (C+1)n^{2/3}}{n} \xrightarrow{n \to \infty} 1 - 2f(0) = \frac{e-1}{e+1}$$

which is what we are aiming for.

Proof. Let us fix $N := \lceil 2/\delta \rceil$. We partition the edges of the matching $\mathcal{M}$ into N^2 submatchings, by setting

$$\mathcal{M}(i,j) := \left\{ (u,v) \in \mathcal{M} : \ell_u \in \left[\frac{i-1}{N}, \frac{i}{N}\right] \ ,\ \ell_v \in \left[\frac{j-1}{N}, \frac{j}{N}\right] \right\}$$

for every $i, j \in \{1, \ldots, N\}$. The main idea now is to apply Lemma 6.4.4 separately to every submatching $\mathcal{M}(i,j)$. Since we are considering a threshold respecting algorithm, we can compute an upper bound on the amount that the algorithm can put on an edge $e \in \mathcal{M}(i,j)$ while staying below the threshold:

$$\begin{aligned}\Delta(i,j) &:= \max\left\{x : f\left(\frac{i-1}{N}+x\right)+f\left(\frac{j-1}{N}+x\right)\le 1\right\}\\ &= \ln\left(\frac{e+1}{\exp\left(\frac{i-1}{N}\right)+\exp\left(\frac{j-1}{N}\right)}\right).\end{aligned}$$

A simpler way to write this equality is as follows:

$$\exp(-\Delta(i,j)) = f\left(\frac{i-1}{N}\right)+f\left(\frac{j-1}{N}\right).$$

By Lemma 6.4.4, we know that there exists sets of online nodes $W(i,j)$ which, together with the matchings $\mathcal{M}(i,j)$, form online hypergraphs $\mathcal{H}(i,j)$ such that

$$\text{val}\Big(\mathcal{A},\mathcal{H}(i,j)\Big) \le \Big(1-\exp(-\Delta(i,j))\Big)\Big|\mathcal{M}(i,j)\Big| + \frac{3}{2} \qquad \forall i,j \in \{1,\ldots,N\}.$$

Now, observe that for an edge $(u,v) \in \mathcal{M}(i,j)$ with levels ℓ_u and ℓ_v, we have that

$$f\left(\frac{i-1}{N}\right) \ge f\left(\ell_u - \frac{1}{N}\right) \ge f(\ell_u) - \frac{1}{N} \quad \text{and} \quad f\left(\frac{j-1}{N}\right) \ge f(\ell_v) - \frac{1}{N}.$$

The first inequality follows from the fact that f is a non-decreasing function. The second inequality follows from the fact that $f'(x) = f(x) \le 1$ for every $x \in [0,1]$. Hence, the total value gained by the algorithm can be upper bounded as follows:

$$\begin{aligned}\text{val}\Big(\mathcal{A},\mathcal{H}\Big) &= \sum_{i,j=1}^{N} \text{val}\Big(\mathcal{A},\mathcal{H}(i,j)\Big)\\ &\le \sum_{i,j=1}^{N}\left(1-f\left(\frac{i-1}{N}\right)-f\left(\frac{j-1}{N}\right)\right)\Big|\mathcal{M}(i,j)\Big| + \frac{3}{2}N^2\\ &\le \frac{3}{2}N^2 + \sum_{i,j=1}^{N}\sum_{(u,v)\in\mathcal{M}(i,j)}\left(1-f(\ell_u)-f(\ell_v)+\frac{2}{N}\right)\\ &= \frac{3}{2}N^2 + \sum_{(u,v)\in\mathcal{M}}\left(1-f(\ell_u)-f(\ell_v)+\frac{2}{N}\right)\\ &= \left(1+\frac{2}{N}\right)|\mathcal{M}| - \sum_{v\in V} f(\ell_v) + \frac{3}{2}N^2 \le (1+\delta)|\mathcal{M}| - \sum_{v\in V} f(\ell_v) + 9\delta^{-2}.\end{aligned}$$

For the last inequality, since $N = \lceil 2/\delta \rceil$, we have used the bounds $2/N \leq \delta$ and $N^2 \leq (2/\delta + 1)^2 = (4 + 4\delta + \delta^2)/\delta^2 \leq 9/\delta^2$.

□

We now upper bound the value gained by a threshold respecting algorithm for every phase $t \in \{1, \ldots, T-1\}$. The following lemma uses two properties from the construction of the matchings $\mathcal{M}^{(t)}$ that we state now. The exact construction of $\mathcal{M}^{(t)}$ is described and illustrated in detail in the next section, since it is mostly needed to bound the value obtained by the algorithm in the last phase. Let us fix a component C_i for $i \in \{1, \ldots, m\}$.

1. At the end of phase $t-1$, let $\mathcal{E}_i \subseteq \mathcal{M}_i^{(t-1)}$ be the subset of edges which exceed the threshold, i.e., $\mathcal{E}_i := \{(u,v) \in \mathcal{M}_i^{(t-1)} : f(\ell_u^{(t-1)}) + f(\ell_v^{(t-1)}) \geq 1\}$. The vertex set of $\mathcal{M}_i^{(t)}$ consists of the nodes incident to $\mathcal{E}_i$, in addition to two *fresh* nodes, one each from U_i and V_i. By fresh, we mean that they have never appeared in previous matchings $\mathcal{M}_i^{(1)}, \ldots, \mathcal{M}_i^{(t-1)}$.

2. For every phase t, the size of the matching satisfies $1 \leq |\mathcal{M}_i^{(t)}| \leq t$ and thus $m \leq |\mathcal{M}^{(t)}| \leq tm$.

Lemma 6.4.7. *For every phase $t \in \{1, \ldots, T-1\}$, the value obtained by a threshold respecting algorithm is at most*

$$\mathrm{val}^{(t)}(\mathcal{A}) \leq \frac{e-1}{e+1}\, m + C\, t^{2/3}\, m^{2/3}$$

where C is an absolute constant satisfying $C \leq 10$.

Proof. We bound the value of $\mathrm{val}^{(t)}(\mathcal{A})$ using Lemma 6.4.5, applied with $\delta = |\mathcal{M}^{(t)}|^{-1/3}$, which can be written as

$$\begin{aligned} \mathrm{val}^{(t)}(\mathcal{A}) \leq & \sum_{(u,v)\in\mathcal{M}^{(t)}} \Big(1 - f(\ell_u^{(t-1)}) - f(\ell_v^{(t-1)})\Big) + C|\mathcal{M}^{(t)}|^{2/3} \\ = & \sum_{i=1}^{m} \sum_{(u,v)\in\mathcal{M}_i^{(t)}} \Big(1 - f(\ell_u^{(t-1)}) - f(\ell_v^{(t-1)})\Big) + C|\mathcal{M}^{(t)}|^{2/3} \end{aligned} \tag{6.12}$$

by splitting the matching into its m components. Fix a component $i \in \{1, \ldots, m\}$, and let $\mathcal{E}_i := \Big\{e \in \mathcal{M}_i^{(t-1)} \mid f(\ell_u^{(t-1)}) + f(\ell_v^{(t-1)}) \geq 1\Big\}$ be the subset of edges in the matching $\mathcal{M}_i^{(t-1)}$ which exceed the threshold at the end of phase $t-1$. By

Property 1 described above, the nodes of the matching $\mathcal{M}_i^{(t)}$ consist of the nodes incident to $\mathcal{E}_i$, in addition to two new fresh nodes whose fractional degree is 0. This allows to expand the inner sum in (6.12) as:

$$\begin{aligned}\sum_{(u,v)\in\mathcal{M}_i^{(t)}}\left(1-f(\ell_u^{(t-1)})-f(\ell_v^{(t-1)})\right) &= 1-2f(0)+\sum_{(u,v)\in\mathcal{E}_i}\left(1-f(\ell_u^{(t-1)})-f(\ell_v^{(t-1)})\right)\\ &\le 1-2f(0)=\frac{e-1}{e+1}.\end{aligned}$$

Plugging this into (6.12) with the bound $|\mathcal{M}^{(t)}|\le tm$ described in Property 2 yields the desired result. □

Corollary 6.4.8. *The total value obtained by an ε-threshold respecting algorithm on the first $T-1$ phases is at most:*

$$\sum_{t=1}^{T-1}\left(\frac{e-1}{e+1}m+C\,t^{2/3}m^{2/3}+\varepsilon tm\right)\le Tm\left(\frac{e-1}{e+1}+\varepsilon T+C\,T^{2/3}m^{-1/3}\right)$$

Proof. The proof follows from Lemma 6.4.7 and the fact that the value above the threshold is upper bounded by $\varepsilon|\mathcal{M}^{(t)}|\le\varepsilon tm$ for every phase $t\in\{1,\dots,T-1\}$. The latter is due to f being 1-Lipschitz on $[0,1]$, $|\mathcal{M}^{(t)}|\le tm$ by Property 2, and our assumption that the algorithm is ε-threshold respecting. □

Bound for the last phase

It is left to describe the construction of the matchings $\mathcal{M}_i^{(1)},\dots,\mathcal{M}_i^{(T)}$ for every $i\in[m]$. Since for all $i\neq j$ and for all $t,t'\in[T]$, the matchings $\mathcal{M}_i^{(t)}$ and $\mathcal{M}_j^{(t')}$ will be disjoint, it suffices to define the construction of the matchings for a single component C_i. This construction will then allow us to prove that the value gained by the algorithm during the last phase T of our instance is at most $O(\sqrt{T})+\varepsilon\,O(T^2)$ on every component C_i for $i\in\{1,\dots,m\}$.

The matchings $\mathcal{M}_i^{(1)},\dots,\mathcal{M}_i^{(T)}$ of a fixed component C_i can be seen as evolving over the phases. We now describe this construction, which is adaptive to the behavior of the algorithm. It is essentially the instance of [Gam+19] with our threshold function incorporated. The vertex set of these matchings is on a bipartite graph, with T nodes on both sides of the bipartition. Let us denote this bipartition as $U_i=\{1,\dots,T\}$ and $V_i=\{1,\dots,T\}$. We index them the same way due to the symmetry assumption on the algorithm in Section 6.4.

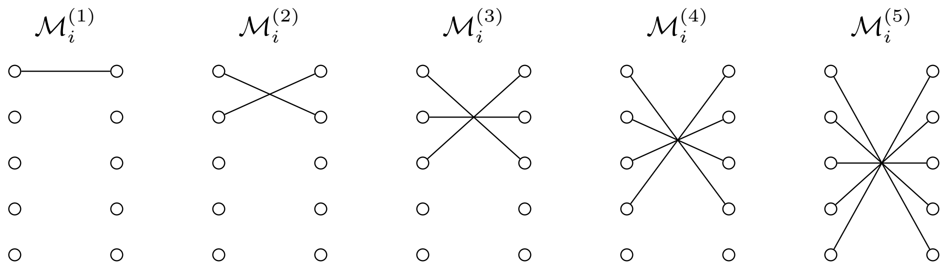


Figure 6.3: The partial matchings $\mathcal{M}_i^{(t)}$ if the fractional algorithm ensures that every edge reaches the threshold at the end of every phase, meaning that $f(\ell_u^{(t)}) + f(\ell_v^{(t)}) \geq 1$ for every $(u, v) \in \mathcal{M}_i^{(t)}$. The optimal solution at the end of phase 5 has size five and consists of $\mathcal{M}_i^{(5)}$.

The matchings of a fixed component C_i can be seen as evolving over time (or over the phases), corresponding to the (partial) matchings $\mathcal{M}_i^{(1)}, \dots, \mathcal{M}_i^{(T)}$. We now describe this construction, which is adaptive to the behavior of the algorithm, and which is essentially the instance of [Gam+19] with our threshold function incorporated. The vertex set of these partial matchings is on a bipartite graph, with T nodes on both sides of the bipartition. Let us denote this bipartition as $U_i = \{1, \dots, T\}$ and $V_i = \{1, \dots, T\}$. We index them the same way because of the symmetry assumption on the algorithm in Section 6.4.

- $\mathcal{M}_i^{(1)}$ is a matching of size one consists of the single edge $(1, 1)$.
- If the algorithm does not increase $e = (1, 1)$ up to the threshold, i.e. $\varphi(e) = 2\, f(\ell_1^{(1)}) < 1$, then both nodes 1 are called *inactive* and $\mathcal{M}_i^{(2)}$ also consists of a single edge $(2, 2)$. If $\varphi(e) \geq 1$, then $\mathcal{M}_i^{(2)} = \{(1, 2), (2, 1)\}$.
- More generally, at the end of phase t, the active nodes will be the ones incident to edges $e = (u, v) \in \mathcal{M}_i^{(t)}$ satisfying $\varphi(e) = f(\ell_u^{(t)}) + f(\ell_v^{(t)}) \geq 1$. The nodes appearing in any of the other edges are called inactive. Inactive nodes will not appear in any of the matchings of later phases.
- Let $\sigma_t(1) < \sigma_t(2) < \ldots < \sigma_t(r_t)$ be the active nodes at the end of phase t, where r_t denotes the number of such active nodes. This means that the edges exceeding the threshold at the end of phase t are:

$$\left\{ \Big(\sigma_t(k), \sigma_t(r_t + 1 - k)\Big), k \in \{1, \dots, r_t\} \right\} \subseteq \mathcal{M}_i^{(t)}.$$

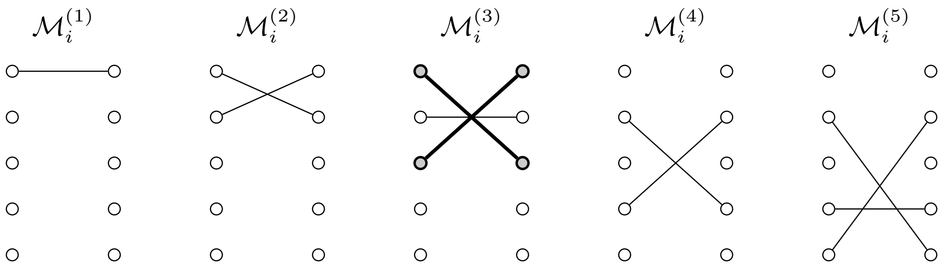


Figure 6.4: In this example, the algorithm does not increase the edge $(1,3)$, and thus by symmetry the edge $(3,1)$, up to the threshold during phase $t=3$. This means that $f(\ell_1^{(3)}) + f(\ell_3^{(3)}) < 1$ and that the nodes 1 and 3 become inactive from that point on. The optimal solution at the end of phase 5 still has size five and consists of $\mathcal{M}_i^{(5)}$, in addition to the two edges $(1,3)$ and $(3,1)$ that are below the threshold.

They satisfy $f(\ell_u^{(t)}) + f(\ell_v^{(t)}) \geq 1$ for every $(u,v) = (\sigma_t(k), \sigma_t(r_t+1-k))$. The matching at phase $t+1$ is then of size r_t+1 and is defined as:

$$\mathcal{M}_i^{(t+1)} := \Big\{ \Big(\sigma_t(k), \sigma_t(r_t+2-k)\Big), k \in \{1,\dots,r_t+1\} \Big\},$$

where we define $\sigma_t(r_t+1) := t+1$ for convenience. In particular, note that $t+1 \in U_i$ and $t+1 \in V_i$ are two fresh nodes, which are always part of the matching $\mathcal{M}_i^{(t+1)}$, but not part of any matching from a previous phase. Figures 6.3 and 6.4 illustrate the construction.

- The size of every matching satisfies $1 \leq \left|\mathcal{M}_i^{(t)}\right| \leq t$ and thus also $m \leq \left|\mathcal{M}^{(t)}\right| \leq tm$ if we consider the matching $\mathcal{M}^{(t)} = \bigcup_{i=1}^m \mathcal{M}_i^{(t)}$ on all the components.

Observe that the nodes $\sigma_t(k) \in U_i$ and $\sigma_t(k) \in V_i$ for every $k \in \{1,\dots,\lceil (r_t+1)/2 \rceil\}$ form a vertex cover of the matching $\mathcal{M}_i^{(t+1)}$, meaning that every edge of the matching at phase $t+1$ is covered by one of these active nodes at phase t. Intuitively, this construction ensures, as t gets large, that these nodes have a high fractional degree, meaning that the algorithm does not have a lot of room to increase the fractional value on any edge of $\mathcal{M}_i^{(t+1)}$, due to the degree constraints. In order to upper bound the value that the algorithm can get in phase $t+1$, we will thus lower bound the fractional degree of the active nodes $\sigma_t(i)$ for $i \in \{1,\dots,\lceil (r_t+1)/2 \rceil\}$. For this reason, we define:

$$\ell(t,i) := x^{(t)}\Big(\delta(\sigma_t(i))\Big) = \sum_{e \in \delta(\sigma_t(i))} x_e^{(t)}.$$

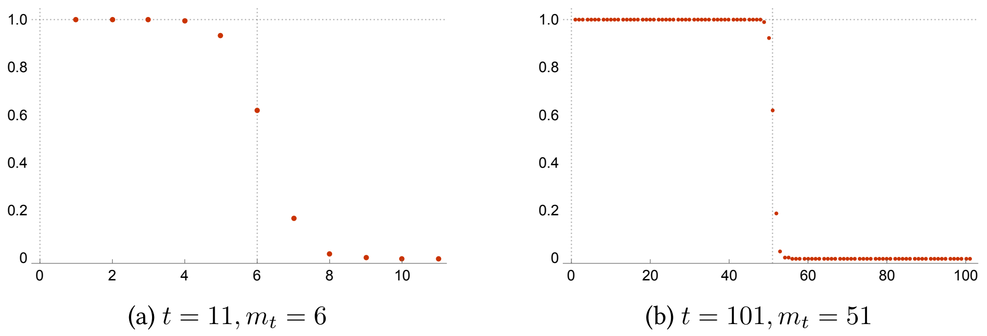


Figure 6.5: Plot of the $\ell(t,i)$ process for two different values of t if the algorithm exactly matches the threshold at every phase. Observe that, since t is odd, $(\sigma_t(m_t), \sigma_t(m_t)) \in \mathcal{M}_i^{(t)}$ with the load of node $\sigma_t(m_t)$ staying at $\ln((e+1)/2) \approx 0.62$.

In words, this is the fractional degree of the ith active node at the end of phase t. One can now see $\{\ell(t,i)\}_{t,i}$ as a process with two parameters, which depends on the behavior of the algorithm. For intuition, Figure 6.5 provides an example of $\ell(t,i)$ if the algorithm exactly reaches the threshold for every edge, i.e. $f(\ell_u^{(t)}) + f(\ell_v^{(t)}) = 1$ for every $(u,v) \in \mathcal{M}_i^{(t)}$.

Let us define $m_t := (r_t+1)/2$. Our final goal of this section is to show that

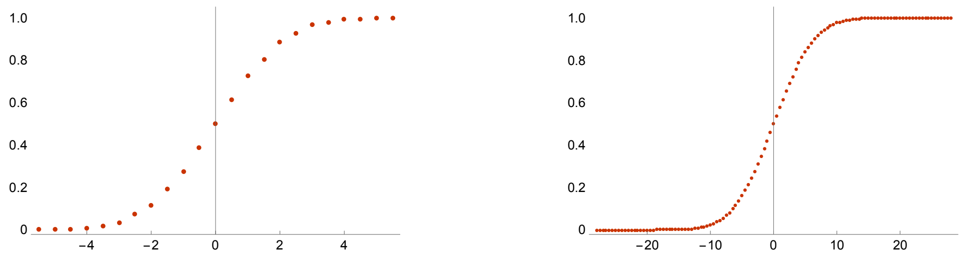
$$\sum_{i=1}^{\lceil m_{T-1} \rceil} 2(1 - \ell(T-1,i)) = O(\sqrt{T}) + \varepsilon\, O(T^2),$$

where T is the total number of phases of our instance. The factor 2 is not necessary, but we write it to make clear that we are lower bounding the loads of the active nodes at the end of phase $T-1$ with active index at most $\lceil m_{T-1} \rceil$ on both sides of the bipartition, since these nodes form a vertex cover of the matching $\mathcal{M}_i^{(T)}$. Clearly, this claim would also imply that the algorithm can get a value of at most $O(\sqrt{T}) + \varepsilon\, O(T^2)$ in component C_i during the last phase by the degree constraints.

In order to be able to get a lower bound on $\ell(t,i)$, we now relate it to a process which is simpler to analyze, defined as follows on $\mathbb{N} \times \mathbb{Z}/2$:

$$\zeta(t,y) = \Pr_{X \sim B(t,\frac{1}{2})}\left[X < \frac{t}{2} + y\right] + \frac{1}{2} \Pr_{X \sim B(t,\frac{1}{2})}\left[X = \frac{t}{2} + y\right],$$

where $B(t, \frac{1}{2})$ is the binomial distribution with parameters t and $\frac{1}{2}$ (see Figure 6.6 for an illustration). An important consequence of this function is the following upper



(a) $t = 11$ (b) $t = 101$

Figure 6.6: Plot of $\zeta(t, y)$ for two different values of t, the horizontal axis represents $y \in \mathbb{Z}/2$.

bound:

$$\sum_{y=0}^{\infty} \left(1 - \zeta\left(t, \frac{y}{2}\right)\right) \leq 1 + \frac{1}{2}\sqrt{t} \qquad \forall t \geq 1. \tag{6.13}$$

We will then relate the process $\ell(t, i)$ to a linear transformation of the process $\zeta(t, i)$, by defining:

$$\xi(t, i) := a\, \zeta\left(t, m_t - i\right) + b - \varepsilon t,$$

where $a := 2 - 2\ln\Big((e+1)/2\Big) \approx 0.76$ and $b := 2\ln\Big((e+1)/2\Big) - 1 \approx 0.24$. Here, a and b are chosen such that whenever the algorithm exactly hits the threshold for every edge, we have $\xi(t, t/2) = \ln((e+1)/2) = \ell(t, t/2)$ for all even t and $\lim_{t\to\infty} \xi(t, x) = 1 = \lim_{t\to\infty} \ell(t, x)$ for all x. As a reminder, the ε parameter comes from the fact that we assume the algorithm to be ε-threshold respecting.

Claim 6.4.9. *The function ζ satisfies:*

1. *For all t, $\zeta(t, \frac{t+1}{2}) = 1$.*
2. *For all t, $\zeta(t, y)$ is nondecreasing in y.*
3. *For all t, we have: $\zeta(t, 0) = \frac{1}{2}$.*
4. *For all t and y, we have: $\zeta(t+1, y) = \frac{1}{2}\zeta(t, y - \frac{1}{2}) + \frac{1}{2}\zeta(t, y + \frac{1}{2})$.*
5. *For all t, we have $\sum_{y=0}^{\infty}(1 - \zeta(t, \frac{1}{2}y)) \leq 1 + \frac{1}{2}\sqrt{t}$.*

Proof. The first two statements follow directly from the definition. The third statement follows from the symmetry of $B(t, \frac{1}{2})$ around $\frac{t}{2}$.

For the fourth statement, let $X \sim B(t, \frac{1}{2})$ and $Y \sim B(1, \frac{1}{2})$ be independent. Then, we have:

$$\begin{aligned}
\zeta(t+1, y) &= \Pr\left[X + Y < \frac{t+1}{2} + y\right] + \frac{1}{2}\Pr\left[X + Y = \frac{t+1}{2} + y\right] \\
&= \Pr[Y = 1]\Pr\left[X < \frac{t}{2} + y - \frac{1}{2}\right] + \Pr[Y = 0]\Pr\left[X < \frac{t}{2} + y + \frac{1}{2}\right] \\
&+ \frac{1}{2}\Pr[Y = 0]\Pr\left[X = \frac{t}{2} + y + \frac{1}{2}\right] + \frac{1}{2}\Pr[Y = 1]\Pr\left[X = \frac{t}{2} + y - \frac{1}{2}\right] \\
&= \frac{1}{2}\zeta\left(t, y - \frac{1}{2}\right) + \frac{1}{2}\zeta\left(t, y + \frac{1}{2}\right).
\end{aligned}$$

Now, let us prove the last statement. Let $X \sim B(t, \frac{1}{2})$ and observe that $1 - \zeta(t, y) \leq \Pr[X \geq \frac{t}{2} + y]$, leading to:

$$\begin{aligned}
\sum_{y=0}^{\infty}(1 - \zeta(t, \frac{1}{2}y)) &\leq \sum_{y=0}^{\infty}\Pr\left[X - \frac{t}{2} \geq \frac{y}{2}\right] \leq \sum_{y=0}^{\infty}\Pr\left[\left|X - \frac{t}{2}\right| \geq \frac{y}{2}\right] \\
&\leq 1 + 2\,\mathbb{E}[|X - \frac{t}{2}|] \\
&\leq 1 + 2\sqrt{\mathbb{E}\left[\left(X - \frac{t}{2}\right)^2\right]} \qquad \text{(by Jensen's inequality)} \\
&= 1 + 2\sqrt{\mathrm{Var}\left[X\right]} = 1 + \frac{1}{2}\sqrt{t}.
\end{aligned}$$

□

We need one additional lemma before being able to get a lower bound for the levels of the nodes. Let us define $d_t(u) = |i - m_t|$ for i when $\sigma_t(i) = u$. For intuition, let us illustrate this definition for the node $u = 2$ in Fig. 6.3 and Fig. 6.4. In the first figure, $m_1 = 1, m_2 = 1.5, m_3 = 2, m_4 = 2.5, m_5 = 3$, whereas $m_1 = 1, m_2 = 1.5, m_3 = 1, m_4 = 1.5, m_5 = 2$ in the second one. In both figures, we have that $d_2(2) = 0.5, d_3(2) = 0, d_4(2) = 0.5, d_5(2) = 1$. The following lemma states how this distance evolves over time. Both bounds stated are tight when every edge reaches the threshold, see Fig. 6.3.

Lemma 6.4.10. *Let $u = \sigma_t(i)$ for $i \leq r_t$. If $i < m_t$, then:*

$$d_t(u) \leq d_{t-1}(u) + \frac{1}{2}.$$

If $i > m_t$, then:

$$d_t(u) \leq d_{t-1}(u) - \frac{1}{2}.$$

Proof. Consider the case that $i < m_t$. Let S be the set of indices $i \leq r_{t-1} + 1$ such that $\sigma_{t-1}(i)$ is not active after phase t. Let j be such that $\sigma_{t-1}(j) = u$. Let $c = |s \in S : i < s < r_{t-1} + 2 - i|$, which denotes the number of nodes from phase $t-1$ that are not active in phase t and that are between u and its neighbor in phase t. We have $d_t(u) = m_t - i = m_{t-1} + \frac{1}{2} - \frac{1}{2}c - i \leq m_{t-1} + \frac{1}{2} - i = d_{t-1}(u) + \frac{1}{2}$. The proof for $i > m_t$ is similar. □

We are now ready to prove the desired lower bound.

Lemma 6.4.11. *For every $t \in \{1, \ldots, T-1\}$ and $i \in \{1, \ldots, \lceil m_t \rceil\}$, we have*

$$\ell(t,i) \geq \xi(t,i),$$

where $m_t = (r_t + 1)/2$ and r_t is the number of active nodes at the end of phase t, when $r_t \geq 1$.

Proof. We will prove this statement by induction on t. For the base case, consider $t = 1$. There are two possibilities, either the edge $(1,1)$ does not make it to the threshold, i.e. $2f(\ell_1^{(1)}) < 1$, in which case $r_t = 0$ and the statement is then trivially satisfied. If the edge $(1,1)$ makes it to the threshold, then $2f(\ell_1^{(1)}) = 2f(\ell(1,1)) \geq 1$, which is equivalent to $\ell(1,1) \geq \ln((e+1)/2)$ by definition of $f(x) = e^x/(e+1)$. Observe that in this case $r_t = m_t = 1$, leading to

$$\ell(1,1) \geq \ln((e+1)/2) = \frac{a}{2} + b = a\,\zeta(1,0) + b = \xi(1,1) + \varepsilon t \geq \xi(1,1),$$

where we have used the fact that $\zeta(1,0) = 1/2$.

Suppose now by induction that the statement holds for $t-1$, let r_t be the active nodes at the end of phase t, let $m_t := (r_t + 1)/2$ and consider an arbitrary $i \in \{1, \ldots, \lceil m_t \rceil\}$. Let us first consider the case where $i = m_t = (r_t + 1)/2$, which can only occur when r_t is odd. Observe that this means the edge $(\sigma_t(i), \sigma_t(i))$ belongs to the matching $\mathcal{M}_i^{(t)}$ and exceeds the threshold, i.e. $\ell(t,i) \geq \ln((e+1)/2)$. Using the fact that $\zeta(t,0) = 1/2$ for all t, $i = m_t$ and the exact same arguments as above, we get

$$\begin{aligned}\ell(t,i) \geq \ln((e+1)/2) &= \frac{a}{2} + b = a\,\zeta(t,0) + b = a\,\zeta(t, m_t - i) + b \\ &= \xi(t,i) + \varepsilon t \geq \xi(t,i).\end{aligned}$$

Consider now the case where $i < m_t$. Let $e = (u, v) = (\sigma_t(i), \sigma_t(r_t+1-i)) \in \mathcal{M}_i^{(t)}$ and observe that e exceeds the threshold, i.e. $f(\ell_u^{(t)})+f(\ell_v^{(t)}) = f(\ell(t,i))+f(\ell(t,r_t+1-i)) \geq 1$. Let us pick indices j, k such that $u = \sigma_{t-1}(j)$ and $v = \sigma_{t-1}(r_{t-1}+1-k)$. Observe that:

$$\ell(t,i) = \ell(t-1,j) + x_e^{(t)} \quad \text{and} \quad \ell(t, r_t+1-i) = \ell(t-1, r_{t-1}+1-k) + x_e^{(t)}.$$

If $k = 0$, then v has not appeared in any of the prior matchings, so $\ell_v^{(t-1)} = 0$. In particular, we have $f(\ell_v^{(t-1)}) = f(0) = 1 - f(1) \leq 1 + \varepsilon - f(\xi(t-1,k))$, because $\xi(t-1,k) \leq 1$.

Otherwise, we have $(\sigma_{t-1}(k), v) \in \mathcal{M}_i^{(t-1)}$ and by using the fact that the algorithm is ε-threshold respecting, we get $f(\ell(t-1,k)) + f(\ell_v^{(t-1)}) \leq 1 + \varepsilon$. By using the inductive hypothesis $\ell(t-1,k) \geq \xi(t-1,k)$, we get

$$f(\ell_v^{(t-1)}) \leq 1 + \varepsilon - f(\xi(t-1,k)).$$

Since edge e exceeds the threshold at the end of phase t, we have $f(\ell_u^{(t-1)} + x_e^{(t)}) + f(\ell_v^{(t-1)} + x_e^{(t)}) \geq 1$, which leads to

$$\begin{aligned} x_e^{(t)} &\geq -\ln\Big(f(\ell_u^{(t-1)}) + f(\ell_v^{(t-1)})\Big) \geq -\ln\Big(1 + \varepsilon - f(\xi(t-1,k)) + f(\ell_u^{(t-1)})\Big) \\ &\geq f(\xi(t-1,k)) - f(\ell_u^{(t-1)}) - \varepsilon = f(\xi(t-1,k)) - f(\ell(t-1,j)) - \varepsilon, \end{aligned}$$

where we have used the fact that $f(x) = e^x/(e+1)$ is an increasing function and $\ln(1+x) \leq x$. Finally, since we have $f'(\ell_u^{(t-1)}) = f(\ell_u^{(t-1)}) \geq f\Big(\ln((e+$

$1)/2)\Big) = 1/2$:

$$\begin{aligned}
\ell(t,i) &= \ell_u^{(t-1)} + x_e^{(t)} \\
&\geq \ell_u^{(t-1)} + f'(\ell_u^{(t-1)})\Big(\xi(t-1,k) - \ell(t-1,j)\Big) - \varepsilon \\
&\quad \text{(by convexity of } f \text{ and } \ell_u^{(t-1)} = \ell(t-1,j)) \\
&= \frac{1}{2}\ell(t-1,j) + \frac{1}{2}\xi(t-1,k) - \varepsilon \geq \frac{1}{2}\xi(t-1,j) + \frac{1}{2}\xi(t-1,k) - \varepsilon \\
&= \frac{a}{2}\Big(\zeta\Big(t-1, d_{t-1}(u)\Big) + \zeta\Big(t-1, d_{t-1}(v)\Big)\Big) + b - \varepsilon t \\
&\geq \frac{a}{2}\Big(\zeta\Big(t-1, d_t(u) + \frac{1}{2}\Big) + \zeta\Big(t-1, d_t(v) - \frac{1}{2}\Big)\Big) + b - \varepsilon t \quad \text{(by Lemma 6.4.10)} \\
&= \frac{a}{2}\Big(\zeta\Big(t-1, d_t(u) + \frac{1}{2}\Big) + \zeta\Big(t-1, d_t(u) - \frac{1}{2}\Big)\Big) + b - \varepsilon t \\
&= a\,\zeta\,(t, m_t - i) + b - \varepsilon t \\
&= \xi(t,i)
\end{aligned}$$

where we have used the inductive hypothesis in the second inequality and the fourth property of the ζ function in the second to last inequality.

□

Theorem 6.4.12. *The value that the algorithm gains on each component C_i in the last phase is at most:*

$$O(\sqrt{T}) + \varepsilon\, O(T^2).$$

Proof. Consider the end of phase $T-1$. We can assume that $r_{T-1} \geq 1$, since otherwise the statement is trivially satisfied. Observe that the nodes $\sigma_{T-1}(k) \in U_i$ and $\sigma_{T-1}(k) \in V_i$ for $k \in \{1, \dots, \lceil m_{T-1} \rceil\}$ form a vertex cover of the final matching $\mathcal{M}_i^{(T)}$. Because of the degree constraints, this means that the value the algorithm can gain on the last phase T is at most twice the following expression:

$$\begin{aligned}
\sum_{k=1}^{\lceil m_{T-1}\rceil} \Big(1-\ell(T-1,k)\Big) &\le \sum_{k=1}^{\lceil m_{T-1}\rceil} \Big(1-\xi(T-1,k)\Big) \\
&= \sum_{k=1}^{\lceil m_{T-1}\rceil} \Big(1-a\,\zeta(T-1,m_{T-1}-k)-b+\varepsilon(T-1)\Big) \\
&= a\sum_{k=1}^{\lceil m_{T-1}\rceil} \Big(1-\ \zeta(T-1,m_{T-1}-k)\Big)+\varepsilon(T-1)\lceil m_{T-1}\rceil \\
&\le \varepsilon\,T^2+a\sum_{y=-1}^{\infty}\Big(1-\ \zeta\Big(T-1,\frac{y}{2}\Big)\Big) \\
&\le \varepsilon\,T^2+a\left(2+\frac{1}{2}\sqrt{T-1}\right) \\
&= O(\sqrt{T})+\varepsilon\,O(T^2).
\end{aligned}$$

The first equality uses the relation $1-b=a$, the second inequality is due to $m_{T-1}=(r_{T-1}+1)/2\le T/2$ and the change of index $y:=2(m_{T-1}-k)$, while the last inequality is by (6.13) and $\xi(t,-1)\le 1$ for all $t\ge 1$. □

Finishing up the proof

In this section, we complete the proof of Theorem 6.4.1.

Lemma 6.4.13. *The optimal solution at the end of phase T of our adversarial instance $\mathcal{H}=(V,W,H)$ satisfies*

$$\mathrm{OPT}(\mathcal{H})=Tm.$$

Proof. We show a stronger statement and prove that the optimal solution at the end of phase t, that we denote by $\mathrm{OPT}^{(t)}(\mathcal{H})$, has size tm for every phase $t\in\{1,\dots,T\}$. Let us focus first on the graph $\Gamma(\mathcal{H})=(V,E)$ and show that $\mathrm{OPT}^{(t)}(\Gamma(\mathcal{H}))=tm$. On each component C_i, we can construct a matching of size t by taking $\mathcal{M}_i^{(t)}$, along with all the edges that did not make it to the threshold in previous phases, see Figure 6.4. In addition, only the nodes $\{1,\dots,t\}$ are incident to edges on each component at phase t, meaning that an upper bound of t on the size of the optimal matching holds. By applying this argument to every component $i\in\{1,\dots,m\}$, we get that $\mathrm{OPT}^{(t)}(\Gamma(\mathcal{H}))=tm$.

Observe that we can complete this matching of size tm on $\Gamma(\mathcal{H})$ to a matching of 3-hyperedges of size tm on $\mathcal{H}$ by adding a distinct online node to every edge.

This is possible since the online nodes are chosen through the construction of Lemma 6.4.5, which itself uses the construction of Lemma 6.4.4 several times. An illustration is provided in Figure 6.2. This construction thus ensures that $\mathrm{OPT}^{(t)}(\mathcal{H}) = \mathrm{OPT}^{(t)}(\Gamma(\mathcal{H}))$ for all phases. □

We now have all the ingredients to prove Theorem 6.4.1.

Proof of Theorem 6.4.1. By Corollary 6.4.8 and Theorem 6.4.12, the total value of the algorithm on the instance $\mathcal{H}$ can be upper bounded by

$$\mathrm{val}(\mathcal{A},\mathcal{H}) \le Tm\left(\frac{e-1}{e+1} + \varepsilon T + O(T^{2/3}m^{-1/3})\right) + m\left(O(\sqrt{T}) + \varepsilon\, O(T^2)\right).$$

By Lemma 6.4.13, the competitive ratio is upper bounded by:

$$\frac{\mathrm{val}(\mathcal{A},\mathcal{H})}{\mathrm{OPT}(\mathcal{H})} \le \frac{e-1}{e+1} + \varepsilon\, O(T) + O(T^{2/3}m^{-1/3}) + O(T^{-1/2}).$$

Picking $T = o(\sqrt{m})$ and $\varepsilon = o(1/T)$, and taking the limit as $T, m \to \infty$ finishes the proof. □

Justification of our assumptions

In this section, we justify the two assumptions made on the algorithm in Section 6.4.

Assumption 1: Symmetry

We start by justifying the symmetry assumption. For a vertex-arrival hypergraph $\mathcal{H}$, we denote $V(\mathcal{H})$ as the set of offline nodes, $W(\mathcal{H})$ as the set of online nodes, and $H(\mathcal{H})$ as the set of hyperedges.

Definition 6.4.14. Given a vertex-arrival hypergraph $\mathcal{H} = (V, W, H)$, an *automorphism* is a bijective map $\sigma : V \to V$ on the offline nodes such that for every $S \subseteq V$ and $w \in W$,

$$S \cup \{w\} \in H \iff \{\sigma(v) : v \in S\} \cup \{w\} \in H.$$

For $S \subseteq V$, we write $\sigma(S) \coloneqq \{\sigma(v) : v \in S\}$ for the sake of brevity. For a hyperedge $h = S \cup \{w\}$ where $S \subseteq V$ and $w \in W$, we denote $\sigma(h) \coloneqq \sigma(S) \cup w$. We also denote the relabelled hypergraph after applying σ to $\mathcal{H}$ as

$$\sigma(\mathcal{H}) \coloneqq (\sigma(V), W, \{\sigma(h) : h \in H\}).$$

Definition 6.4.15. Given a vertex-arrival hypergraph $\mathcal{H} = (V, W, H)$, let Σ be a subset of its automorphisms. A fractional matching x in $\mathcal{H}$ is Σ-*symmetric* if $x_h = x_{\sigma(h)}$ for all $h \in H$ and $\sigma \in \Sigma$. An algorithm $\mathcal{A}$ is Σ-*symmetric* on $\mathcal{H}$ if it outputs a Σ-symmetric fractional matching given $\mathcal{H}$.

Since the construction of our vertex-arrival instance depends on the behavior of the algorithm, we will overload the notation $\mathcal{H}$ as follows. An *(adaptive vertex-arrival) instance* is a function $\mathcal{H}$ which takes as input an algorithm $\mathcal{A}$ and outputs a vertex-arrival hypergraph $\mathcal{H}(\mathcal{A})$. For $i \geq 1$, let $\mathcal{H}_i(\mathcal{A})$ be the subgraph of $\mathcal{H}(\mathcal{A})$ right after the arrival of the ith online node w_i. We make the following assumptions about $\mathcal{H}$:

1. For any pair of algorithms $\mathcal{A}, \mathcal{A}'$, we have $\mathcal{H}_1(\mathcal{A}) = \mathcal{H}_1(\mathcal{A}')$.
2. For any pair of algorithms $\mathcal{A}, \mathcal{A}'$ and indices $i, j \geq 1$, we have $V(\mathcal{H}_i(\mathcal{A})) = V(\mathcal{H}_j(\mathcal{A}'))$.

Definition 6.4.16. We say that σ is a *strong automorphism* of an instance $\mathcal{H}$ if σ is an automorphism of $\mathcal{H}(\mathcal{A})$ for every algorithm $\mathcal{A}$.

Note that the set of strong automorphisms forms a group.

Given an instance $\mathcal{H}$, we now show that for any algorithm $\mathcal{A}$, there exists another algorithm $\mathcal{A}'$ which performs at least as well as $\mathcal{A}$ on the hypergraph $\mathcal{H}(A')$. Furthermore, $\mathcal{A}'$ is symmetric on $\mathcal{H}(A')$ with respect to any subgroup of strong automorphisms of $\mathcal{H}$.

Lemma 6.4.17. *Let Σ be a subgroup of strong automorphisms of an instance $\mathcal{H}$. For every algorithm $\mathcal{A}$, there exists a Σ-symmetric algorithm $\mathcal{A}'$ on $\mathcal{H}(\mathcal{A}')$ such that*

$$\mathrm{val}(\mathcal{A}', \mathcal{H}(\mathcal{A}')) \geq \min_{\sigma \in \Sigma} \mathrm{val}(\mathcal{A}, \sigma(\mathcal{H}(\mathcal{A}'))).$$

Proof. From $\mathcal{A}$, we construct a new algorithm $\mathcal{A}'$ as follows. When the ith online node w_i arrives for $i \geq 1$, run $\mathcal{A}$ on $\sigma(\mathcal{H}_i(\mathcal{A}'))$ for each $\sigma \in \Sigma$. Note that σ is an automorphism of $\mathcal{H}_i(\mathcal{A}')$. Let $y(\sigma, h)$ be the value that $\mathcal{A}$ puts on edge h when it ran on $\sigma(\mathcal{H}_i(\mathcal{A}'))$. Then, $\mathcal{A}'$ sets $x_h = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} y(\sigma, h)$ for all $h \in \delta(w_i)$. This completes the description of $\mathcal{A}'$.

First, we show that x is a fractional matching in $\mathcal{H}(\mathcal{A}')$. For any node v,

$$\sum_{h \in \delta(v)} x_h = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \sum_{h \in \delta(v)} y(\sigma, h) \leq 1.$$

Next, we show that x is Σ-symmetric. Fix an $i \geq 1$ and a strong automorphism $\tau \in \Sigma$. Since Σ is a group, for every edge $h \in \delta(w_i)$ we have

$$x_h = \frac{\sum_{\sigma \in \Sigma} y(\sigma, h)}{|\Sigma|} = \frac{\sum_{\sigma \in \Sigma} y(\sigma, \tau(h))}{|\Sigma|} = x_{\tau(h)}$$

as desired. Finally,

$$\text{val}(\mathcal{A}', \mathcal{H}(\mathcal{A}')) = \sum_h x_h = \frac{1}{|\Sigma|} \sum_{\sigma \in \Sigma} \sum_h y(\sigma, h) \geq \min_{\sigma \in \Sigma} \text{val}(\mathcal{A}, \sigma(\mathcal{H}(\mathcal{A}'))).$$

□

In Section 6.4, we stated that we assume that the algorithm treats the kth vertex in U_i, say $u_{i,k}$, and the kth vertex in V_i, say $v_{i,k}$, symmetrically. If our constructed hypergraph $\mathcal{H}$ would be symmetric with respect to these vertices, i.e. if the permutation σ swapping these two vertices would be a strong automorphism of $\mathcal{H}$, then Lemma 6.4.17 would show that this assumption can be made without loss of generality.

However, one part of the global instance that breaks this symmetry is the construction given in the proof of Lemma 6.4.4 and illustrated in Figure 6.2. As a reminder, this construction is repeatedly applied to submatchings of $\mathcal{M}^{(t)}$ during some phase t in Lemma 6.4.5. Let us fix one such submatching and denote it by $\mathcal{M}$. As described in Section 6.4 and illustrated in Fig. 6.4, if some $u_{i,k} \in \mathcal{M}$, then $v_{i,k} \in \mathcal{M}$ and the submatching is symmetric with respect to this pair, i.e. $\sigma(e) \in \mathcal{M}$ for every edge e in $\mathcal{M}$, where σ is the permutation swapping these two vertices. However, due to the Lemma 6.4.4 construction, $e \cup \{w\}$ might be a hyperedge in $\mathcal{H}$ for some online vertex w, while $\sigma(e \cup \{w\}) = \sigma(e) \cup \{w\}$ might not be a hyperedge in $\mathcal{H}$.

To fix this, the construction can be slightly tweaked in the following way. An important observation is that the horizontal edges in $\mathcal{M}$ (between $u_{i,k}$ and $v_{i,k}$) are not isomorphic to any other edge in the hypergraph, whereas each of the diagonal edges (all non-horizontal edges) are isomorphic to exactly one other edge in $\mathcal{M}$. For this reason, we can first apply the Lemma 6.4.4 construction on just the horizontal edges of $\mathcal{M}$.

We can then can apply a slightly modified construction to the diagonal edges, where the pairs of isomorphic edges are treated in the same way. In the original construction, a newly arriving online vertex w would be connected to all edges in $\mathcal{M}$ that were incident to the previous online vertex, except for the one with the smallest fractional value. In the modified construction, we instead consider the online vertices in groups of two. For every two consecutive online vertices, we connect them to all edges in $\mathcal{M}$ that were incident to the previous online vertex, except for the diagonal pair with the smallest total fractional value. This ensures that the symmetry between the diagonal edges is respected.

This modified construction would slightly worsen the upper bound that we gave in Lemma 6.4.4 to become $(1 - e^{-\Delta})|\mathcal{M}| + 9/2$. However, since it only changes the constant, this does not affect the asymptotic upper bound for large m. Hence, it shows that Theorem 6.4.1 also holds for non-symmetric algorithms.

Assumption 2: ε-Threshold Respectingness

Next, we justify the assumption of ε-threshold respectingness. Let $\mathcal{H}$ be the instance constructed in Section 6.4. Let f be the function given by

$$f(x) := \frac{e^x}{e+1},$$

and recall the definition of ε-threshold respecting with respect to f (Definition 6.4.2).

For any algorithm $\mathcal{A}$ and $\varepsilon > 0$, we now show that there exists an algorithm $\mathcal{A}'$ which is ε-threshold respecting on all online nodes before the last phase. Moreover, there exists an instance $\mathcal{H}'$ such that the performance of $\mathcal{A}$ on $\mathcal{H}'$ matches the performance of $\mathcal{A}'$ on $\mathcal{H}$.

Lemma 6.4.18. *Let $\mathcal{H}$ be the instance constructed in Section 6.4. For any algorithm $\mathcal{A}$ and $\varepsilon > 0$, there exists an algorithm $\mathcal{A}'$ which is ε-threshold respecting on all online nodes before the last phase. Furthermore, there exists an instance $\mathcal{H}'$ such that*

$$\frac{\text{val}(\mathcal{A}, \mathcal{H}'(\mathcal{A}))}{\text{OPT}(\mathcal{H}'(\mathcal{A}))} = \frac{\text{val}(\mathcal{A}', \mathcal{H}(\mathcal{A}'))}{\text{OPT}(\mathcal{H}(\mathcal{A}'))}.$$

Proof. From $\mathcal{H}$, we construct a new instance $\mathcal{H}'$ as follows. Let $N = \lceil 2/\varepsilon \rceil$. For every offline node v in $\mathcal{H}$, create N offline copies in $\mathcal{H}'$, denoted $v'_1, v'_2, \ldots, v'_N$. The new algorithm $\mathcal{A}'$ will be defined based on the behavior of $\mathcal{A}$ on $\mathcal{H}'$. When the ith online node w_i arrives in $\mathcal{H}(\mathcal{A}')$ for $i \geq 1$, at most N copies of w_i arrives sequentially in $\mathcal{H}'(\mathcal{A})$, denoted $w'_{i,1}, w'_{i,2}, \ldots$. When the jth copy $w'_{i,j}$ arrives, for every edge $h = S \cup w_i$ in $\mathcal{H}_i(\mathcal{A}')$, add the edge $h'_j := \{v'_j : v \in S\} \cup w'_{i,j}$ to $\mathcal{H}'(\mathcal{A})$. Now, let $x'_{i,j}$ denote the solution given by $\mathcal{A}$ in $\mathcal{H}'$ after the arrival of $w'_{i,j}$. Consider the following averaged solution

$$x_{i,j}(h) := \frac{1}{N} \sum_k x'_{i,j}(h'_k) \qquad \forall h \in H(\mathcal{H}_i(\mathcal{A}')).$$

If $j = N$, or w_i appeared before the last phase and there exists a hyperedge $h \in \delta(w_i)$ such that

$$\sum_{v \in h \setminus \{w_i\}} f(x_{i,j}(\delta(v))) \geq 1,$$

then $w'_{i,j+1}, \ldots, w'_{i,N}$ will not arrive in $\mathcal{H}'$. In this case, $\mathcal{A}'$ sets $x(h) \leftarrow x_{i,j}(h)$ for all $h \in \delta(w_i)$ in $\mathcal{H}$. Otherwise, we proceed to let the $(j+1)$th copy $w'_{i,j+1}$ arrive in $\mathcal{H}'$. This completes the description of $\mathcal{A}'$.

Clearly, x is a fractional matching in $\mathcal{H}(\mathcal{A}')$. Moreover,

$$\text{val}(\mathcal{A}', \mathcal{H}(\mathcal{A}')) = \sum_h x(h) = \frac{\text{val}(\mathcal{A}, \mathcal{H}'(\mathcal{A}))}{N}.$$

Next, we claim that $N \cdot \text{OPT}(\mathcal{H}(\mathcal{A}')) = \text{OPT}(\mathcal{H}'(\mathcal{A}))$. Based on the construction of $\mathcal{H}$, the offline optimal matching in $\mathcal{H}(\mathcal{A}')$ covers all the online nodes in the last phase, and the online nodes on which $\mathcal{A}'$ is strictly threshold respecting. Let W_{OPT} denote the union of these two sets. For each $w_i \in W_{\text{OPT}}$, observe that $w_{i,j}$ is present in $\mathcal{H}'(\mathcal{A})$ for all $j \in [N]$ by our construction of $\mathcal{H}'$. Hence, the offline optimal matching in $\mathcal{H}'(\mathcal{A})$ covers the following online nodes

$$\{w_{i,j} : w_i \in W_{\text{OPT}}, j \in [N]\}.$$

So, $N \cdot \text{OPT}(\mathcal{H}(\mathcal{A}')) = \text{OPT}(\mathcal{H}'(\mathcal{A}))$ as desired.

It is left to show that $\mathcal{A}'$ is ε-threshold respecting on all online nodes before the last phase. Pick such an online node w_i and let $w_{i,j}$ be its last copy in $\mathcal{H}'(\mathcal{A})$. Note that $x_{i,j}$ is the output of $\mathcal{A}'$ in $\mathcal{H}_i(\mathcal{A}')$. For any $h \in \delta(w_i)$, we have

$$\begin{aligned}
\sum_{v \in h \setminus \{w_i\}} f(x_{i,j}(\delta(v))) &\le \sum_{v \in h \setminus \{w_i\}} f\left(x_{i,j-1}(\delta(v)) + \frac{1}{N}\right) && (x'_{i,j}(\delta(w'_{i,j})) \le 1) \\
&\le \sum_{v \in h \setminus \{w_i\}} \left(f(x_{i,j-1}(\delta(v)) + \frac{1}{N}\right) && (f \text{ is 1-Lipschitz}) \\
&< 1 + \frac{2}{N} && (\text{due to } |h| = 3 \text{ and the construction of } \mathcal{H}') \\
&\le 1 + \varepsilon.
\end{aligned}$$

□

6.5 Integral algorithm for bounded degree hypergraphs

In this section, we show that Random (Algorithm 4) performs better than the greedy algorithm when the online nodes have bounded degree.

We prove Theorem 6.1.2, restated below.

Algorithm 4 Random algorithm for bounded degree hypergraphs

```
Input : $k$-uniform hypergraph $\mathcal{H} = (V, W, H)$ with online arrivals of each $w \in W$ with $|\delta(w)| \leq d$.
Output : Matching $\mathcal{M} \subseteq H$

set $\mathcal{M} \leftarrow \emptyset$
when $w \in W$ arrives with $\delta(w) \subseteq H$:
    pick uniformly at random $h \in \delta(w)$ among the hyperedges that are disjoint from $\mathcal{M}$
    set $y_v = \min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right)$ for all $v \in h \setminus \{w\}$
    set $y_w = \max\left(0, \frac{d-k+1}{(d-1)k+1}\right)$
return $\mathcal{M}$
```

Theorem 6.5.1. *Algorithm 4 is ρ-competitive for k-uniform hypergraphs whose online nodes have degree at most d, where*

$$\rho = \min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right).$$

Proof. Let the algorithm be denoted by $\mathcal{A}$. We prove the result via a primal-dual analysis, where the random primal solution is given by $x_h := \mathbb{1}_{\{h \in \mathcal{M}\}}$ for every $h \in H$ and the random dual solution is the vector $y \in [0,1]^{V \cup W}$ constructed during the execution of the algorithm. Observe that the objective values of both solutions are equal at all times during the execution of the algorithm:

$$\text{val}(\mathcal{A}) = |\mathcal{M}| = \sum_{h \in H} x_h = \sum_{v \in V \cup W} y_v. \tag{6.14}$$

This holds since every time a hyperedge $h \in H$ is matched by the algorithm, increasing the primal value val($\mathcal{A}$) by one, the dual objective increases by $\sum_{v \in h} y_v$. Two easy computations that we omit show that the latter is also equal to one in both cases where $d \leq k-1$ and $d \geq k-1$.

We will now show that, in expectation, the dual constraints are satisfied up to a factor of ρ, i.e.

$$\mathbb{E}\left[\sum_{v \in h} y_v\right] \geq \rho \qquad \forall h \in H. \tag{6.15}$$

This will imply the theorem, since the random vector $\mathbb{E}[y]/\rho$ will then be a feasible dual solution, leading to $\mathbb{E}[\text{val}(\mathcal{A})] = \mathbb{E}\left[\sum_{v \in V \cup W} y_v\right] \geq \rho\, \mathsf{OPT}_{\mathsf{LP}}$ by (6.14) and (6.15).

To show this inequality, let $h \in H$ be an arbitrary hyperedge incident to some online node $w \in W$. We now consider the following probabilistic event upon the arrival of w:

$$\mathcal{E} := \Big\{ \exists v \in h \setminus \{w\} \text{ which is already matched at the arrival of } w \Big\}.$$

We will show (6.15) by conditioning on $\mathcal{E}$ and on its complementary event $\bar{\mathcal{E}}$, which states that all nodes in $h \setminus \{w\}$ are unmatched when w arrives, and that the hyperedge h is thus available and considered in the random choice of the algorithm in this step. In the first case, if $\mathcal{E}$ happens, then some offline node $u \in h \setminus \{w\}$ has already had its dual value set to $y_u = \min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right) = \rho$ in a previous step of the algorithm, leading to

$$\mathbb{E}\left[\sum_{v \in h} y_v \,\middle|\, \mathcal{E}\right] = \sum_{v \in h} \mathbb{E}\left[y_v \mid \mathcal{E}\right] \geq \mathbb{E}\left[y_u \mid \mathcal{E}\right] = \rho.$$

Otherwise, if $\bar{\mathcal{E}}$ happens, we know that with probability at least $1/d$, the algorithm adds h to the matching. Summing the dual values of the offline nodes contained in h gives

$$\sum_{v \in h \setminus \{w\}} \mathbb{E}\left[y_v \mid \bar{\mathcal{E}}\,\right] \geq \frac{1}{d} \cdot (k-1) \cdot \min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right).$$

Furthermore, since the algorithm will always match w to a hyperedge in this case, we have:

$$\mathbb{E}\left[y_w \mid \bar{\mathcal{E}}\,\right] = \max\left(0, \frac{d-k+1}{(d-1)k+1}\right).$$

Adding those terms together, we get:

$$\begin{aligned}
\sum_{v \in h} \mathbb{E}\left[y_v \mid \bar{\mathcal{E}}\,\right] &\geq \frac{1}{d} \cdot (k-1) \cdot \min\left(\frac{1}{k-1}, \frac{d}{(d-1)k+1}\right) + \max\left(0, \frac{d-k+1}{(d-1)k+1}\right) \\
&= \min\left(\frac{1}{d}, \frac{k-1}{(d-1)k+1}\right) + \max\left(0, \frac{d-(k-1)}{(d-1)k+1}\right) \\
&\geq \min\left(\frac{1}{d}, \frac{k-1}{(d-1)k+1} + \frac{d-(k-1)}{(d-1)k+1}\right) \geq \rho.
\end{aligned}$$

This shows that (6.15) holds, and hence proves that the algorithm is ρ-competitive.

$\square$

6.6 Conclusion

We have studied the online matching problem on k-uniform hypergraphs. For the case $k = 3$, we have shown that the best competitive ratio for the fractional version of the problem is $\frac{e-1}{e+1} \approx 0.4621$. This provides an upper bound on the competitive ratio of any online algorithm for the integral version of the problem as well. However, on the lower bound side, we do not know of any (randomized) algorithm that performs better than the greedy algorithm, which is $\frac{1}{3}$-competitive.

A natural strategy for finding such an algorithm would be to round the $(e-1)/(e+1)$-competitive algorithm. However, already in the case of bipartite matching, it was [BSW23] shown that not every fractional matching can be losslessly rounded to an integral matching. In the case of hypergraphs, the situation is more complicated, as even for the offline version of the problem, there is a non-zero integrality gap. The multiplicative integrality gap of 3-uniform matching is 2 [CL12]. As $(e-1)/(e+1)/2 \approx 0.231$ is less than $1/3$, even rounding the online fractional solution afterwards in an offline way might not be enough to get a better competitive ratio.

Another natural strategy would be to try to adapt algorithms for the online bipartite matching problem, such as the ranking algorithm, to the online hypergraph matching problem. Here, the main difficulty is that such algorithms can induce a strong correlation between the nodes that are matched in the hypergraph setting, which makes them hard to analyze.

6.A Integral upper bound for k-uniform hypergraphs

In this section, we prove a strong upper bound against any randomized integral algorithm, showing that the greedy algorithm is almost optimal, since it achieves a competitive ratio of $1/k$.

Theorem 6.A.1. *For the online matching problem on k-uniform hypergraphs, every integral randomized algorithm is at most $2/k$-competitive.*

Proof. We in fact prove a stronger statement and show that any randomized integral algorithm is at most $(2-2^{-k+1})/k$-competitive. To do so, we make use of Yao's principle [Yao77]: it suffices to construct a randomized instance for which any deterministic integral algorithm is at most $(2-2^{-k+1})/k$-competitive in expectation. Let us now describe our randomized construction $\mathcal{H} = (V, W, H)$ for any $k \in \mathbb{N}$.

- The offline nodes are partitioned into $k-1$ blocks: $V = C_1 \cup \cdots \cup C_{k-1}$, where $|C_i| = 2(k-i)$ for each $i \in \{1, \ldots, k-1\}$. So, $|V| = k\,(k-1)$ holds.

- The instance consists of k iterations, with online nodes $w_1, \dots, w_k$ arriving. For every $i \in \{1, \dots, k-1\}$, w_i is incident to 2 hyperedges. They are disjoint on the offline nodes V, each containing $(k-i)$ nodes from C_i, and 1 node from C_j for all $j \in \{1, \dots, i-1\}$. On the other hand, the last online node w_k is incident to only 1 hyperedge. It contains 1 node from C_j for all $j \in \{1, \dots, k-1\}$.

- The offline perfect matching $H^* \subseteq H$ is chosen randomly as follows. When w_i arrives for $i \leq k-1$, select one of its two incident hyperedges uniformly at random and add it to H^*. For the last online node w_k, its unique incident hyperedge is included in H^*. For $i \in \{1, \dots, k\}$, denote $H_i^* := H^* \cap (\cup_{j=1}^{i} \delta(w_j))$. Observe that for H^* to be a matching, the hyperedge(s) in $\delta(w_{i+1})$ must be disjoint from H_i^* for all $i \leq k-1$.

Let $V(H_i^*)$ be the offline nodes covered by H_i^*. To ensure that the above construction is possible, we will show the following invariant: for every element $i \in \{1, \dots, k-1\}$,

$$|C_j \setminus V(H_i^*)| = k - i \qquad \forall j \in \{1, \dots, i\}. \tag{6.16}$$

We prove (6.16) by induction on i. In the first iteration, both hyperedges in $\delta(w_1)$ partition C_1 because $|C_1| = 2(k-1)$. One of them is chosen to enter H^*, meaning that $C_1 \setminus V(H^*) = k-1$ after iteration 1. Let us now fix an iteration $i \leq k-1$ and suppose that (6.16) holds for all previous iterations. By construction, C_i is completely covered by the two hyperedges arriving in iteration i, since each of them contains $k-i$ nodes from C_i and $|C_i| = 2(k-i)$. One of them enters H^* at the end of iteration i, meaning that $|C_i \setminus V(H^*)| = k-i$ indeed holds. For any other C_j with $j < i$, note that $|C_j \setminus V(H^*)| = k-i+1$ at the beginning of iteration i by the induction hypothesis. Both hyperedges in $\delta(w_i)$ intersect C_j at two different nodes, one of which enters $V(H^*)$ by the random choice. Hence, $|C_j \setminus V(H^*)|$ drops by 1 and equals $k-i$, proving (6.16).

Clearly, $\mathrm{OPT}(\mathcal{H}) = |H^*| = k$. We now upper bound the value that any deterministic algorithm can get on this randomized instance. The key observation is that, if the algorithm picks a hyperedge $h \in \delta(w_i)$ which is not placed in H^* for some iteration $i \in \{1, \dots, k-1\}$, then it cannot pick any hyperedge arriving in later iterations. This holds, since in that case, $C_i \setminus V(H^*) \subseteq h$, and any hyperedge arriving in later iterations necessarily intersects $C_i \setminus V(H^*)$ by construction.

Let us denote by val_i the maximum expected value achievable by a deterministic algorithm if it can only select hyperedges from iteration i to iteration k. Clearly, $\mathrm{val}_k = 1$. In an iteration $i \in \{1, \dots, k-1\}$, the algorithm either does nothing, or picks a hyperedge and risks not being able to pick anything later with probability

$1/2$. We thus get the following recurrence relation:

$$\mathrm{val}_i = \max\left\{ \mathrm{val}_{i+1}, \frac{1}{2} + \frac{1}{2}(1 + \mathrm{val}_{i+1})\right\} = \max\left\{ \mathrm{val}_{i+1}, 1 + \frac{1}{2}\mathrm{val}_{i+1} \right\}.$$

The first term is at most the second term if and only if $\mathrm{val}_{i+1} \leq 2$. Thus, the solution to this recurrence is the geometric series $\mathrm{val}_i = \sum_{j=0}^{k-i} 2^{-j}$ and thus $\mathrm{val}_1 = \sum_{j=0}^{k-1} 2^{-j} = 2 - 2^{-k+1}$. We have therefore just shown that any algorithm is at most $(2 - 2^{-k+1})/k$ competitive. $\square$

6.B Rounding algorithm for online hypergraph b-matching

In this section, we will consider the online hypergraph b-matching problem on k-uniform hypergraphs, in which every (offline and online) node v can be matched to at most b hyperedges. We show that, for $b = \Omega(\log k)$, any fractional algorithm can be rounded to a randomized integral algorithm while incurring a small loss in the competitive ratio.

Let $\mathcal{A}$ be a fractional algorithm that is ρ-competitive and let $\mathcal{H} = (V, W, H)$ be an online k-uniform hypergraph instance. We denote by $x \in [0,1]^H$ the fractional solution constructed by $\mathcal{A}$ on the instance $\mathcal{H}$. The rounding algorithm is now quite simple and is similar to the methods used in [EJ12; RT87; SS95].

Fix some small $0 < \varepsilon < \frac{1}{2}$ and initialize two empty sets of hyperedges $S, \mathcal{M} \leftarrow \emptyset$. Upon the arrival of an online vertex $w \in W$ with $\delta(w) \subseteq H$ and $x_h \in [0,1]$ for every $h \in \delta(w)$, the rounding algorithm is as follows:

- For all $h \in \delta(w)$, independently add h to S with probability $x'_h := (1-\varepsilon)x_h$.
- If h was added to S, add it to $\mathcal{M}$ as long as it does not violate the degree constraints.

The solution outputted is $\mathcal{M} \subseteq H$. Let us denote this rounding algorithm by $R(\mathcal{A}, \varepsilon)$.

Lemma 6.B.1. *Let $\mathcal{A}$ be a fractional algorithm which is ρ-competitive. The randomized integral algorithm $R(\mathcal{A}, \varepsilon)$ then achieves a competitive ratio of at least $(1-\varepsilon)(1 - k\exp(-\varepsilon^2 b/3)) \cdot \rho$.*

Proof. Consider an arbitrary node $v \in V \cup W$. To bound the probability that v is matched to more than b hyperedges in S, we use a Chernoff bound [Doe11, Corollary

1.10]. Fix a node v and a hyperedge h, and let $X_{v,h} = \sum_{h' \in \delta(v) \setminus \{h\}} \mathbf{1}_{\{h' \in S\}}$. Note that $\mu := \mathbb{E}[X_{v,h}] \leq (1-\varepsilon)b$. We now have:

$$\Pr\left[X_{v,h} \geq b\right] \leq \exp\left(-\left(\frac{b-\mu}{\mu}\right)^2 \mu/3\right) \leq \exp(-\varepsilon^2 b^2/3\mu) \leq \exp(-\varepsilon^2 b/3),$$

where the second inequality follows from $b - \mu \geq \varepsilon b$ and the last inequality from $b/\mu \geq 1$. We now upper bound the probability that a hyperedge h cannot be included in $\mathcal{M}$ because of the degree constraints:

$$\Pr[h \in S \setminus \mathcal{M} \mid h \in S] \leq \sum_{v \in h} \Pr\left[X_{v,h} \geq b\right] \leq k \exp(-\varepsilon^2 b/3).$$

Hence, we have:

$$\begin{aligned}
\mathbb{E}[|\mathcal{M}|] &= \sum_{h \in H} \Pr[h \in \mathcal{M}] \geq \sum_{h \in H} \Pr[h \in S]\,(1 - \Pr[h \in S \setminus \mathcal{M}]) \\
&\geq \sum_{h \in H} x'_h \left(1 - k\exp(\varepsilon^2 b/3)\right) \geq \left(1 - k\exp(\varepsilon^2 b/3)\right)(1-\varepsilon) \sum_{h \in H} x_h \\
&\geq \left(1 - k\exp(\varepsilon^2 b/3)\right)(1-\varepsilon)\rho\, \mathsf{OPT}_{\mathsf{LP}}.
\end{aligned}$$

□

If $b = C \cdot \log(k)$ for some $C \geq 1$, then by choosing $\varepsilon = \sqrt{3\log(C)/C}$ we get that the competitive ratio is at least $(1 - \sqrt{3\log(C)/C})(1 - 1/C)\rho$. By using the $\Omega(1/\log k)$-competitive fractional algorithm from [BN09], this gives an $\Omega(1/\log k)$-competitive integral algorithm for this setting.

Chapter 7

Propagation and dual proof analysis in an exact MIP solver

In this chapter we approach the branch-and-bound algorithm from a practical point of view. We consider the use case of solving MIPs exactly, and show how to adapt the techniques of constraint propagation and dual proof analysis to the exact setting. Using an experimental analysis, we show that enabling these techniques can lead to significant performance improvements in the context of exact MIP-solving.

7.1 Introduction

A wide variety of solvers is available for solving mixed integer programs. These solvers generally use the branch-and-bound algorithm at their core, but employ many complementary techniques to improve their performance [NW88; Ach07b; CCZ14; Ach+19; HP19]. They typically use double precision floating-point arithmetic together with the careful handling of numerical tolerances to quickly compute accurate solutions. While this approach makes the solving process highly efficient, and is accurate enough in practice for most applications, there exist problems where exact solutions are necessary. This is the case for applications in computational mathematics [BMVV19; BO12; EGP22; KS18; LPR20; Pul20] as well as for several industry applications where exact correctness is critical [Ach07b; WLH00; SBD19]. Furthermore, there exist pathological instances that exhibit such numerical difficulties that floating-point solvers produce large errors. For these reasons, there is a need for roundoff-error-free MIP solvers.

In recent years, a hybrid approach has been proposed that aims to take advantage of both floating-point and exact arithmetic in order to solve MIPs exactly [CKSW13]. This approach has been revised and further improved by [EG22], and more recently in [EG23] with the addition of cutting planes. What can be clearly seen in these works is that for numerical techniques to be beneficial, it is crucial to employ them

The contents of this chapter are based on joint work with Leon Eifler and Ambros Gleixner [BEG24].

as much as possible using floating-point arithmetic in a numerically safe way, as symbolic computation can be prohibitively slow.

An important feature of MIP solvers is *constraint propagation*, which is a technique that uses the problem constraints to tighten variable bounds, which can be used to detect infeasibility of a subproblem before the LP is solved. This saves time, as performing constraint propagation is generally much faster than solving an LP.

Another key feature of MIP solvers that has not yet been adapted to the exact setting is *conflict analysis*, which can be categorized in two distinct types. In this chapter we will study *dual proof analysis*, which was introduced by [WBH17; WBH21]. This type of conflict analysis derives constraints from Farkas certificates of infeasible subproblems and dual solutions of bound-exceeding LPs. These are redundant constraints that do not strengthen the LP relaxation. However, by performing propagation on these constraints, infeasibility of subsequent subproblems can be detected more efficiently. Another form of conflict analysis is *graph-based* conflict analysis [Ach07a], which was inspired by a similar procedure in SAT solving [BS97; MS99; Mos+01]. This form of conflict analysis is numerically simpler, as it only generates disjunctive constraints with ± 1 coefficients. In this chapter we study dual proof analysis, since our focus is on adapting numerical methods and studying the challenges that arise in the exact setting.

An important aspect in the context of roundoff-error-free algorithms is the *certification of their correctness*. A certifying algorithm creates a certificate alongside its solution that can be *independently verified* to be correct [MMNS11; Alk+11]. Such techniques have become standard in SAT solving [WHJ14; Cru+17; CMS17; GN03], and have also been adapted to SMT solving [dMB08; Bar+22], pseudo boolean optimization [Goc19; EGMN20], as well as to MIP solving [CGS17; EG22; EG23].

To summarize, our contribution is to develop numerically safe versions constraint propagation and dual proof analysis, and to show that they can be used to improve the performance of an exact MIP solver in practice. Furthermore, we show how to certify the correctness of the derived bounds in the VIPR [CGS17] certificate format.

Organization

First, we introduce the existing methods of dual proof analysis and propagation in Section 7.2. Then, we describe how to adapt these techniques to the exact setting as well as how to certify their correctness in Section 7.3. We conduct a thorough computational study in Section 7.4, showing that these techniques can be used to improve the running time by 23% on the MIPLIB 2017 benchmark test set [Gle+21] and solve more instances within a time limit of two hours. We conclude with an outlook on future research in Section 7.5.

7.2 Constraint propagation and conflict analysis

In this chapter we consider MIPs of the form

$$\begin{aligned} \text{minimize} \quad & c^{\mathsf{T}} x \\ \text{subject to} \quad & Ax \geq b, \\ & \ell_i \leq x_i \leq u_i \quad \text{for all } i \in [n], \\ & x_i \in \mathbb{Z} \quad \text{for all } i \in S, \end{aligned}$$

for $A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m$. For the sake of simplicity of presentation, we assume that all variables have finite bounds. The techniques are still applicable to MIPs with infinite bounds, but propagation of some constraints involving unbounded variables may become ineffective and is skipped. In the following, we first review the techniques of constraint propagation and dual proof analysis in the floating-point setting.

Constraint propagation

Constraint propagation is a technique that uses the problem constraints to tighten variable bounds, see, e.g. [Ach07b, p. 426]. Consider a constraint $a^{\mathsf{T}} x \geq b$. The maximum activity is defined as $\text{act}^+ := \sum_{i=1}^n \max(a_i \ell_i, a_i u_i)$. Similarly, the minimum activity is defined to be $\text{act}^- := \sum_{i=1}^n \min(a_i \ell_i, a_i u_i)$. We define the maximum activity relative to variable x_k as

$$\text{act}_k^+ := \sum_{i \in [n] \setminus \{k\}} \max(a_i \ell_i, a_i u_i),$$

and the minimum activity act_k^- analogously.

Any feasible vector x will satisfy $a^{\mathsf{T}} x \leq \text{act}^+$. Hence, for any variable x_i with $a_k > 0$, we have

$$x_k \geq \frac{b - \text{act}_k^+}{a_k}, \tag{7.1}$$

and if $a_k < 0$ we have

$$x_k \leq \frac{b - \text{act}_k^+}{a_k}. \tag{7.2}$$

Constraint propagation is the technique of computing the appropriate bounds from Eqs. (7.1) and (7.2) and using them to tighten one of the variable bounds ℓ_k, u_k if possible. It can be performed in an iterated fashion, as long as some of the bounds have been tightened. This is also the case at every node in a branch-and-bound tree, where the local bounds of the subproblem are taken into account after they have been tightened by branching decisions.

In addition, by computing the minimum activity of a constraint, it can sometimes be determined that a subproblem is infeasible. This happens when $\mathrm{act}^- < b$. In this case, the problem is infeasible because the constraint is violated by any solution that respects the variable bounds. Conversely, whenever $\mathrm{act}^+ \geq b$, then constraint is redundant and can be safely removed for the corresponding subproblem.

Dual proof analysis

Dual proof analysis is a technique that allows to learn from previously solved subproblems that are infeasible or exceed the objective bound. In this type of conflict analysis, for any subproblem with an infeasible or bound-exceeding LP relaxation a constraint is added to the global problem from which the infeasibility of the subproblem can be derived by applying constraint propagation to this single constraint instead of solving the LP relaxation over all constraints. Note that the constraint will be redundant, and therefore does not need to be added to the LP. However, it can be used for constraint propagation in other nodes of the tree.

If branching is only performed on variables, then such a subproblem is given as a sequence of bounds ℓ'_i, u'_i that have to be satisfied in addition to the problem constraints. So, the LP relaxation in any node in the tree will be of the form

$$\begin{aligned} \min c^{\mathsf{T}} x & \\ Ax \geq b & \\ \ell'_i \leq x_i \leq u'_i & \quad \text{for all } i \in [n]. \end{aligned}$$

The dual program to this LP is given by

$$\begin{aligned} \max\ b^{\mathsf{T}} y + \ell'^{\mathsf{T}} r^+ - u'^{\mathsf{T}} r^- & \\ r^+ - r^- - A^{\mathsf{T}} y = -c & \\ y \geq 0, r^+ \geq 0, r^- \geq 0. & \end{aligned}$$

If $\ell' \leq u'$, we can assume that for all $i \in [n]$ either r_i^+ or r_i^- is zero, so we will write $r = r^+ - r^-$. If the subproblem is infeasible, then the dual program will be unbounded. This implies that there must exist a ray y, r^+, r^- with

$$\begin{aligned} b^{\mathsf{T}} y + \ell'^{\mathsf{T}} r^+ - u'^{\mathsf{T}} r^- > 0, \\ A^{\mathsf{T}} y + r = 0. \end{aligned}$$

Such a ray is called a Farkas proof of infeasibility [Far02]. Rewriting this gives

$$0 < b^{\mathsf{T}} y + \ell'^{\mathsf{T}} r^{+} - u'^{\mathsf{T}} r^{-} = b^{\mathsf{T}} y - \ell'^{\mathsf{T}} (A^{\mathsf{T}} y)^{-} + u'^{\mathsf{T}} (A^{\mathsf{T}} y)^{+}. \tag{7.3}$$

In dual proof analysis, the MIP solver adds the constraint $y^{\mathsf{T}} A x \leq y^{\mathsf{T}} b$ to the problem. Note that this constraint is globally valid, since it is a linear combination of problem constraints. The minimum activity of this constraint in the current subproblem is $\text{act}^{-} = \ell'^{\mathsf{T}} (A^{\mathsf{T}} y)^{+} - u'^{\mathsf{T}} (A^{\mathsf{T}} y)^{-}$. Substituting this into Eq. (7.3) shows that $\text{act}^{-} > b^{\mathsf{T}} y$. Hence, for this subproblem, infeasibility can immediately be derived using propagation of the newly added constraint.

Now consider the case of a bound-exceeding subproblem. This occurs when the objective value to the LP relaxation of the current problem is higher than the objective value $c^{\mathsf{T}} x^{\bullet}$ of the best IP-solution $x^{\bullet}$ found so far. We can cut off bound-exceeding subproblems by imposing $c^{\mathsf{T}} x \leq c^{\mathsf{T}} x^{\bullet}$. Let y be the optimal dual solution. Dual proof analysis adds the constraint $(y^{\mathsf{T}} A - c)^{\mathsf{T}} x \geq y^{\mathsf{T}} b - c^{\mathsf{T}} x^{\bullet}$. This constraint is valid, as it is a linear combination of the constraints of A and $c^{\mathsf{T}} x \leq c^{\mathsf{T}} x^{\bullet}$. Note that for any feasible x for the subproblem, we have

$$b^{\mathsf{T}} y + \ell'^{\mathsf{T}} (A^{\mathsf{T}} y - c)^{+} - u'^{\mathsf{T}} (A^{\mathsf{T}} y - c)^{-} = b^{\mathsf{T}} y + \ell'^{\mathsf{T}} r^{+} - u'^{\mathsf{T}} r^{-} \geq c^{\mathsf{T}} x^{\bullet}, \tag{7.4}$$

that is

$$\ell'^{\mathsf{T}} (A^{\mathsf{T}} y - c)^{+} - u'^{\mathsf{T}} (A^{\mathsf{T}} y - c)^{-} \geq c^{\mathsf{T}} x^{\bullet} - b^{\mathsf{T}} y. \tag{7.5}$$

Note that the left hand is the minimum activity of the newly added conflict constraint. So just like in the case of an infeasible subproblem, we see that the bound-exceeding subproblem can now be cut of using constraint propagation on the newly added constraint.

The constraints will not be added to the LP, but are only used for propagation. Since dual proof analysis might be applied many times it can get too expensive to store all the derived constraints. Therefore, we employ a method called aging, to dynamically remove constraints that have not been useful for a long time [WBH17]. In the exact setting, we make use of the same method.

7.3 Application in numerically exact MIP solving

To efficiently implement constraint propagation and dual proof analysis in an exact MIP solver, we will perform most computations in floating-point arithmetic. To guarantee correctness, we carefully make use of directed rounding. We start with some definitions.

Let $\mathbb{F} \subseteq \mathbb{Q}$ denote the set of floating-point numbers. In practice, these will be standard IEEE double-precision [IEE08] numbers with 11 bits for the exponent and 52 bits for the mantissa. For all $x \in \mathbb{Q}$, we define:

$$\overline{x} = \min\{y \in \mathbb{F} : y \geq x\} \qquad\qquad \underline{x} = \max\{y \in \mathbb{F} : y \leq x\}.$$

We will write $\overline{a+b+c+\cdots}$ for $\overline{\overline{\overline{a}+b}+\cdots}$. An expression $\underline{a+b+\cdots}$ is defined analogously, as well as multiplication, division, and any combination thereof. We note that this is consistent with how these expressions are computed in practice.

Numerically safe constraint propagation

Constraint propagation is applied at every node in the branch-and-bound tree. Recomputing the minimum and maximum activity each time would be costly. Therefore, a solver will keep track of these activities and update them whenever they change. This happens when the variable bounds change, i.e., when the bounds (ℓ, u) of variable x_i become (ℓ', u'), we update the maximum activity as

$$\text{act}^{+} \leftarrow \begin{cases} (u' - u)a_i & \text{if } a_i \geq 0, \\ (\ell' - \ell)a_i & \text{otherwise.} \end{cases}.$$

In the exact setting, the activities would ideally be computed in exact arithmetic, to give the tightest possible bounds. However, updating the activities using symbolic computations can be prohibitively expensive. So instead our implementation uses floating-point arithmetic. To ensure that no incorrect bounds are derived, we enforce the maximum activity that we maintain to be at least as large as the actual maximum activity. So when bounds (ℓ, u) are updated to (ℓ', u'), the maximum activity is updated to

$$\text{act}^{+} \leftarrow \begin{cases} \overline{\text{act}^{+} + u'a_i - ua_i} & \text{if } a_i \geq 0, \\ \underline{\text{act}^{+} + \ell'a_i - \ell a_i} & \text{otherwise,} \end{cases}$$

using directed rounding. Similarly, for the minimum activity we maintain only a lower bound on the exact value.

In the floating-point setting the maintained activities need to be recomputed regularly to keep the aggregated inaccuracy from accumulating. In the exact setting this is not necessary, since the activities are guaranteed to remain valid. For this reason we do not recompute them.

Numerically safe dual proof analysis

As explained in Section 7.2, the dual ray y is computed and the constraint $y^{\mathsf{T}}Ax \leq y^{\mathsf{T}}b$ is added to the problem. For the sake of efficiency, we let the MIP solver compute y inexactly, i.e., y is obtained by a floating-point LP solve, using the standard error tolerances. This suffices, since slightly negative dual multipliers can be set to zero and any conic combination of the constraints is globally valid.

To aggregate the constraints we also use floating-point arithmetic. We start from the constraint $\hat{a}^{\mathsf{T}}x \leq \hat{b}$ for $\hat{a} = 0$ and $\hat{b} = 0$, which holds trivially. Now we add the constraints $y_j A_{\cdot j}^{\mathsf{T}} x \leq y_j b_j$ (where $A_{\cdot j}$ is the jth column of A) to this constraint one by one: For each j, we set $\hat{b} \leftarrow \overline{\hat{b} + y_j b_j}$ and the components of $\hat{a}$ are updated in the following way:

- If $u_i \leq 0$, then we set $\hat{a}_i \leftarrow \overline{\hat{a}_i + y_j A_{ji}}$.
- Otherwise, if $\ell_i \geq 0$, then we round $\hat{a}_i \leftarrow \underline{\hat{a}_i + y_j A_{ji}}$.
- Otherwise, if $u_i \leq \infty$, then we set $\hat{b} \leftarrow \overline{\hat{b} + (\overline{\overline{\hat{a}_i + y_j A_{ji}} - \underline{(\hat{a}_i + y_j A_{ji})}})u_i}$ and set $\hat{a}_i \leftarrow \overline{\hat{a}_i + y_j A_{ji}}$.
- Otherwise, if $\ell_i \geq -\infty$, then we set $\hat{b} \leftarrow \overline{\hat{b} - (\overline{\overline{\hat{a}_i + y_j A_{ji}} - \underline{(\hat{a}_i + y_j A_{ji})}})\ell_i}$ and set $\hat{a}_i \leftarrow \underline{\hat{a}_i + y_j A_{ji}}$.

Note that this way of rounding guarantees that the constraint stays valid, and yields an approximation of the exact conflict constraint. This approach has been used before to implement Gomory mixed integer cuts [EG23].

Producing certificates of correctness

MIP solvers are complex pieces of software, that might contain bugs. Hence, it can be desirable to verify the correctness of a solution that was found using a MIP solver, especially when finding exact solutions is important. While checking feasibility is easy, verifying optimality is harder, since this requires checking many nodes in the branch-and-bound tree without using the solver. To make it possible to verify optimality, a MIP solver can be extended to emit a certificate of optimality [CGS17]. The certificate contains all derived bounds, plus a reason for why each bound holds. These certificates can be verified using a simple piece of software that does not make use of the solver.

SCIP has been extended to generate certificates that can be verified using the program VIPR [CGS17]. A VIPR certificate contains all the initial problem constraints and any derived constraints. Each derived constraint contains a reason, that justifies why the constraint holds. Previously derived constraints can be referred to by their line number in the certificate. In VIPR certificates there are two deduction rules to derive an inequality:

- **Linear combination.** If a constraint is a linear combination of previously derived constraints, it has to be valid. To certify the derivation, the line numbers of these constraints are printed, along with the corresponding coefficients.
- **Rounding.** If $c^{\mathsf{T}}x \leq b$ has been previously derived such that c is zero for all indices that correspond to non-integral variables, then $\sum_i \lfloor c_i \rfloor x_i$ must be integral for any feasible solution x. Hence, $\sum_i \lfloor c_i \rfloor x_i \leq \lfloor b \rfloor$ holds. To certify the derivation, the line number of the constraint $c^{\mathsf{T}}x \leq b$ is printed to the certificate. This procedure is called a *Chvátal-Gomory cut* in the context of pure integer programming.

Additionally, there are deduction rules that allow to encode a branching proof.

To be able to verify bounds that were derived using constraint propagation or dual proof analysis, these bounds need to be written to the certificate in terms of the above derivation rules. Because Eqs. (7.1) and (7.2) are both linear combinations of valid variable bounds, constraint propagation steps can be added to the certificate using the 'linear combination' deduction rule. If $a_k > 0$ the constraint is

$$x_k \geq \frac{1}{a_k}\left(b_j - \sum_{i\in[n]\setminus\{k\}}^{n} \max(a_i\ell_i, a_iu_i)\right). \tag{7.6}$$

For each $i \neq k$, the line number of $x_i \geq \ell_i$ is printed to the certificate if $a_i\ell_i > a_iu_i$. Otherwise, the line number of $x_i \leq u_i$ is printed. Finally, the line number corresponding to the constraint $\sum_{i=1}^n a_ix_i \geq b_j$ is printed as well. For each of the constraints, the corresponding coefficient is $1/a_k$.

If x_k is an integer variable and for all i with nonzero a_i, both a_i and x_i are integral, a rounding derivation is printed to the certificate, certifying

$$x_k \geq \lceil \frac{1}{a_k}(b_j - \sum_{i\in[n]\setminus\{k\}}^{n} \max(a_i\ell_i, a_iu_i))\rceil. \tag{7.7}$$

In dual proof analysis, only linear combinations of globally valid constraints are added as constraints. However, these constraints are computed using safe rounding methods instead of exact arithmetic. For that reason, the derived constraints cannot be computed directly using the given multipliers and the "linear compbination" deduction rule.

Instead, we let the solver write just the linear combination corresponding to the original constraint to the certificate along with a list of the current bounds for all variables appearing in the constraint. Then afterwards a post-processing tool, called `viprcomplete` is applied to the certificate to repair the linear combination of all of the constraints that have slight inaccuracies due to the safe rounding procedure. One of the advantages of this approach is that the repair step only needs to be done for constraints that will actually be used in the solving process. For details, we refer to [EG23].

7.4 Computational study

To analyze the actual impact of constraint propagation and dual proof analysis in the context of exact MIP solving, we implemented these techniques in the exact variant of SCIP and compared them with the impact of the same techniques in the floating-point version of SCIP. Our runtime experiments do not include the time for certificate generation in order to be as comparable as possible with the floating-point version. We stress that while certificate generation does incur a computational cost, it does not change the solving path and therefore does not affect the solvability of instances. We refer to [EG22; EG23] for a detailed discussion of the computational cost of certificate generation.

Experimental setup and test set

The experiments were all performed on a cluster of Intel Xeon Gold 5122 CPUs with 3.6 GHz and 96 GB main memory. For all symbolic computations, we use the GNU Multiple Precision Library (GMP) 6.1.2 [GT15]. All compared algorithms are implemented within SCIP 8.0.3 [Vig+23], using SoPlex 6.0.3 [Mil+22] as both the floating-point and the exact LP solver. For presolving in exact arithmetic, we use PaPILO 2.0.1 [Hoe22] and disable all other presolving steps. Our proposed algorithms are freely available on GitHub.[1]

We use the MIPLIB 2017 benchmark test set [Gle+21]; in order to save computational effort, we exclude all those that could not be solved by the floating-point default version of SCIP 8.0.3 within two hours. For the remaining 132 instances we use three random seeds, making the size of our test set 396. We report aggregated times in shifted geometric mean with a shift of 1 second, and node numbers in shifted

[1]As part of the development version of exact SCIP mirrored under `https://github.com/scipopt/scip/tree/exact-rational`.

geometric mean with a shift of 100 nodes. For all tests, a time limit of two hours is used.

Experimental results

The experimental results can be found in Table 7.1. From the table we see that enabling propagation leads to an improvement in the running time of 11.6%. Also, 12 more instances are solved to optimality within the time limit. The number of nodes decreases by 13.6%. Enabling dual proof analysis decreases the running time by another 13.4% and the node count by 17.7%. 3 more instances are solved to optimality within the time limit. Together this comes down to a 23.4% decrease in running time, a 28.9% decrease in node count and 15 more instances that can be solved. From the performance profile in Fig. 7.1 it is also clear that enabling constraint propagation and conflict analysis consistently speeds up the solver. We also measured the total time that is spent in propagation. As can be seen from the table, this amount is small in comparison with the total time.

To compare this with the impact that these techniques have in the floating-point setting, we ran the same experiments on floating-point SCIP. In these experiments we disabled the features not present in the exact version of SCIP: all cutting planes, graph-based conflict analysis, restarts, presolving (except for PAPILO), custom propagation rules. As shown in Table 7.2 in the floating-point setting, enabling propagation reduces solving time by 12.7%. The node count increases by 6.2% and 11 more instances are solved to optimality. Dual proof analysis reduces the running time by another 36.5% and the node count by 67.1%. 20 more instances are solved to optimality. Together this comes down to a decrease in the running time of 44.5%, a 65.1% decrease in node count and 31 more instances being solved to optimality.

It is clear that the performance boost is significantly larger in the floating-point setting. The same is true for the increase in the number of instances solved. While the speedup from only enabling propagation is comparable, we observe a much greater reduction in the number of nodes, and a much larger speedup from conflict analysis in the floating-point setting. We see several reasons contributing to this difference.

First, performance variability plays a part and the fact that we look at different subsets of the instances. Second, propagation takes up more time in the exact solving mode. Although the shifted geometric mean of the propagation times shown in Table 7.1 is not large, there exist instances where exact propagation even takes up a major portion of the solving time. This is the case for instances in which propagation leads to a large number of bound changes, since applying these bound changes is computationally more expensive in the exact setting. This is because, while the propagation procedure itself is implemented using floating-point arithmetic,

Settings	# Solved	Time				Nodes	(rel)
		Total	(rel)	CP	DPA		
Baseline	135	759.21	—	—	—	8735.1	—
+ CP	147	671.22	(0.88)	9.11	—	7550.4	(0.86)
+ CP + DPA	150	581.46	(0.77)	9.22	5.44	6211.1	(0.71)

Table 7.1: Experimental results comparing the performance of exact SCIP with and without constraint propagation (CP) and dual proof analysis (DPA) enabled. All times shown are in seconds. For the times and node counts, the shifted geometric with shift 1 seconds or 100 nodes respectively is used. Columns (rel) show times and nodes relative to the baseline. Only the instances that could be solved by at least one of the given configurations are included in these statistics.

Settings	# Solved	Time	(rel)	Nodes	(rel)
Baseline	164	584.84	—	11415.0	—
+ CP	175	510.85	(0.87)	12119.4	(1.06)
+ CP + DPA	195	324.41	(0.55)	6986.1	(0.61)

Table 7.2: Experimental results comparing the performance of floating-point SCIP with and without constraint propagation (CP) and dual proof analysis (DPA) enabled. All times shown are in seconds. For the times and node counts, the shifted geometric with shift 1 seconds or 100 nodes respectively is used. Columns (rel) show times and nodes relative to the baseline. Only the instances that could be solved by at least one of the given configurations are included in these statistics.

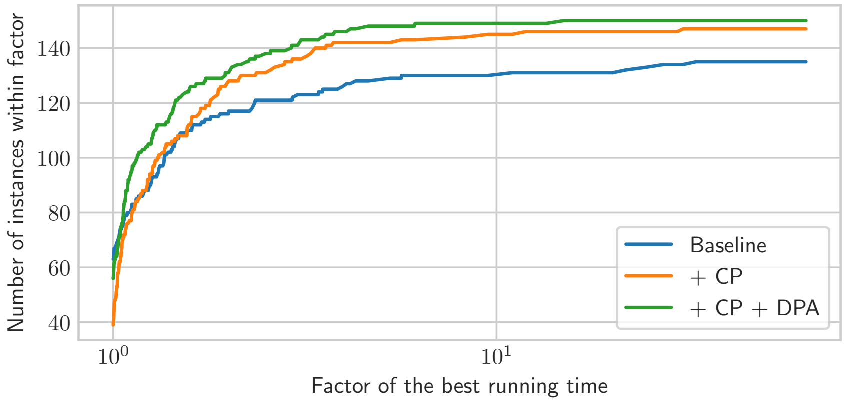


Figure 7.1: Performance profile for different configurations. For each configuration it is shown how many instances achieved a running time within a given factor of the shortest running time for that instances. Only the instances that could be solved by at least one of the given configurations are included in these statistics.

applying a bound change in exact SCIP is currently done using exact arithmetic. So every floating-point bound change needs to be converted to a rational number first, and then that rational number needs to be used to update the exact bound. This can slow down the process significantly. If we only look at the solving time without the time spent in propagation, the speedup from enabling propagation is 17.1%, and the speedup from enabling conflict analysis is 28.3%.

There are more reasons that might lead to a lower impact of dual proof analysis in the exact setting. The most important one is that constraint propagation is performed using floating-point arithmetic, and that we cannot determine infeasibility within tolerances in the exact setting. This might sometimes lead to derived bounds not being precise enough to be useful in the exact settings or conflicts not being able to prove infeasibility. For example, it is possible that in floating-point mode such an inexact bound can be used to decide the infeasibility of a node, whereas in exact mode the corresponding LP still needs to be solved. In numerically difficult cases, this might even lead to incorrect cutoffs in the floating-point setting. It is clear that this is a trade-off that we can never fully avoid, guaranteeing correctness while sacrificing performance.

Another possible reason that the techniques are slightly less effective in exact SCIP could be that the derived variable bounds on non-integral variables can have a large denominator. This can slow down solving LPs exactly, as has been observed for safe cutting plane generation in [EG23]. We experimented with two variants that try to prevent this from happening:

1. Limit the size of the denominators of the derived variable bounds for continuous variables to a fixed upper bound.
2. Disable constraint propagation for continuous variables.

However, we did not observe any positive effects on the solving times. This does not mean that the described negative effects do not occur, but rather that all in all, by enabling propagation for non-integral variables and allowing arbitrary denominators in their bounds, we gain more than we lose.

7.5 Conclusion

In this study, we investigated the feasibility and impact of constraint propagation and dual proof analysis in the exact MIP solver SCIP. We found that enabling both techniques decreases the running time by 23.4% and the number of nodes by 28.9%. This performance boost is significant, but less than the 44.5% decrease in running time and 65.1% decrease in node count that we observed in the equivalent floating-point setting. Such weaker performance is partially the price to be paid for exactness, coming from weaker inequalities due to directed rounding and the fact that we cannot determine infeasibility within tolerances in the exact setting.

Still, we see several future research opportunities to further improve these results. On the implementation side, we believe there is potential to decrease the time spent in propagation by adding support for floating-point bound changes in exact SCIP. Also, due to technical reasons, it is currently not possible to apply propagation within strong branching in the exact solving mode. Enabling this could also be beneficial.

Algorithmically, incorporating the strengthening techniques from [WBH21] for conflict constraints also in the exact setting is likely to yield additional performance improvements. Finally, implementing graph-based conflict analysis [Ach07a] would be straightforward due to its combinatorial nature and is sure to yield positive impact in the exact setting. More open-ended is the question if sporadically recomputing the activities also in the exact setting could lead to tighter bounds and thus better performance.

Bibliography

[Ach+19] Tobias Achterberg, Robert Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. "Presolve Reductions in Mixed Integer Programming". *INFORMS Journal on Computing* 32 (2019), pp. 473–506. DOI: 10.1287/ijoc.2018.0857.

[Ach07a] Tobias Achterberg. "Conflict Analysis in Mixed Integer Programming". *Discrete Optimization* 4.1 (2007), pp. 4–20. DOI: 10.1016/j.disopt.2006.10.006.

[Ach07b] Tobias Achterberg. "Constraint Integer Programming". PhD thesis. Technische Universität Berlin, 2007. DOI: 10.14279/depositonce-1634.

[AG03] Steve Alpern and Shmuel Gal. *The Theory of Search Games and Rendezvous*. International Series in Operations Research & Management Science 55. Boston: Kluwer Academic Publishers, 2003. DOI: 10.1007/b100809.

[AGKM11] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. "Online Vertex-Weighted Bipartite Matching and Single-bid Budgeted Allocations". *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2011, pp. 1253–1264. DOI: 10.1137/1.9781611973082.95.

[AKM05] Tobias Achterberg, Thorsten Koch, and Alexander Martin. "Branching Rules Revisited". *Operations Research Letters* 33.1 (2005), pp. 42–54. DOI: 10.1016/j.orl.2004.04.002.

[Alk+11] Eyad Alkassar, Sascha Böhme, Kurt Mehlhorn, Christine Rizkallah, and Pascal Schweitzer. "An Introduction to Certifying Algorithms". *it - Information Technology* 53.6 (2011), pp. 287–293. DOI: 10.1524/itit.2011.0655.

[Ash+23] Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. "Edge-Weighted Online Windowed Matching". *Mathematics of Operations Research* 48.2 (2023), pp. 999–1016. DOI: 10.1287/moor.2022.1289.

[AW13] Tobias Achterberg and Roland Wunderling. "Mixed Integer Programming: Analyzing 12 Years of Progress". *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*. Berlin: Springer, 2013, pp. 449–481. DOI: 10.1007/978-3-642-38189-8_18.

[Ban10] Nikhil Bansal. "Constructive Algorithms for Discrepancy Minimization". *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. 2010, pp. 3–10. DOI: 10.1109/FOCS.2010.7.

[Bar+22] Haniel Barbosa et al. “Flexible Proof Production in an Industrial-Strength SMT Solver”. *Automated Reasoning*. Cham: Springer, 2022, pp. 15–35. DOI: 10.1007/978-3-031-10769-6_3.

[BCC96] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. “Mixed 0-1 Programming by Lift-and-Project in a Branch-and-Cut Framework”. *Management Science* 42.9 (1996), pp. 1229–1246. DOI: 10.1287/mnsc.42.9.1229.

[BCGL23] Siddhartha Banerjee, Vincent Cohen-Addad, Anupam Gupta, and Zhouzi Li. “Graph Searching with Predictions”. *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*. Schloss-Dagstuhl - Leibniz Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.ITCS.2023.12.

[BDGL18] Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. “The Gram-Schmidt Walk: A Cure for the Banaszczyk Blues”. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. Los Angeles, CA: ACM Press, 2018, pp. 587–597. DOI: 10.1145/3188745.3188850.

[BDHK23] Sander Borst, Daniel Dadush, Sophie Huiberts, and Danish Kashaev. “A Nearly Optimal Randomized Algorithm for Explorable Heap Selection”. *Integer Programming and Combinatorial Optimization (IPCO)*. Lecture Notes in Computer Science 13904. Cham: Springer International Publishing, 2023, pp. 29–43. DOI: 10.1007/978-3-031-32726-1_3.

[BDHT21] Sander Borst, Daniel Dadush, Sophie Huiberts, and Samarth Tiwari. “On the Integrality Gap of Binary Integer Programs with Gaussian Data”. *Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science 12707. Cham: Springer International Publishing, 2021, pp. 427–442. DOI: 10.1007/978-3-030-73879-2_30.

[BDHT22] Sander Borst, Daniel Dadush, Sophie Huiberts, and Samarth Tiwari. “On the Integrality Gap of Binary Integer Programs with Gaussian Data”. *Mathematical Programming* (2022). DOI: 10.1007/s10107-022-01828-1.

[BDM23] Sander Borst, Daniel Dadush, and Dan Mikulincer. “Integrality Gaps for Random Integer Programs via Discrepancy”. *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2023, pp. 1692–1733. DOI: 10.1137/1.9781611977554.ch65.

[BDSV18] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. “Learning to Branch”. *Proceedings of the 35th International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 344–353. URL: https://proceedings.mlr.press/v80/balcan18a.html.

[BEG24] Sander Borst, Leon Eifler, and Ambros Gleixner. *Certified Constraint Propagation and Dual Proof Analysis in a Numerically Exact MIP Solver*. 2024. DOI: 10.48550/arXiv.2403.13567. Pre-published.

[Ber98] Piotr Berman. “On-Line Searching and Navigation”. *Online Algorithms*. Red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Lecture Notes in Computer Science 1442. Berlin: Springer, 1998, pp. 232–241. DOI: 10.1007/BFb0029571.

[BJN07] Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. “Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue”. *Algorithms – ESA 2007*. Lecture Notes In Computer Science 4698. Berlin: Springer, 2007, pp. 253–264. DOI: 10.1007/978-3-540-75520-3_24.

[BKK24] Sander Borst, Danish Kashaev, and Zhuan Khye Koh. *Online Matching on 3-Uniform Hypergraphs.* 2024. DOI: 10.48550/arXiv.2402.13227. Pre-published.

[BM08] Benjamin Birnbaum and Claire Mathieu. "On-Line Bipartite Matching Made Simple". *ACM SIGACT News* 39.1 (2008), pp. 80–87. DOI: 10.1145/1360443.1360462.

[BM19] Nikhil Bansal and Raghu Meka. "On the Discrepancy of Random Low Degree Set Systems". *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2019, pp. 2557–2564. DOI: 10.1137/1.9781611975482.157.

[BMVV19] Miquel Bofill, Felip Manyà, Amanda Vidal, and Mateu Villaret. "New Complexity Results for Łukasiewicz Logic". *Soft Computing* 23 (2019), pp. 2187–2197. DOI: 10.1007/s00500-018-3365-9.

[BN09] Niv Buchbinder and Joseph Naor. "Online primal-dual algorithms for covering and packing". *Mathematics of Operations Research* 34.2 (2009), pp. 270–286.

[BO12] Benjamin A. Burton and Melih Ozlen. "Computing the Crosscap Number of a Knot Using Integer Programming and Normal Surfaces". *ACM Transactions on Mathematical Software* 39.1 (2012). DOI: 10.1145/2382585.2382589.

[Bor82] K. H. Borgwardt. "The Average Number of Pivot Steps Required by the Simplex-Method Is Polynomial". *Zeitschrift für Operations-Research* 26.1 (1982), pp. 157–177. DOI: 10.1007/BF01917108.

[BR07] Robert Bixby and Edward Rothberg. "Progress in Computational Mixed Integer Programming—A Look Back from the Other Side of the Tipping Point". *Annals of Operations Research* 149.1 (2007), pp. 37–41. DOI: 10.1007/s10479-006-0091-y.

[BRS22] Nikhil Bansal, Lars Rohwedder, and Ola Svensson. "Flow Time Scheduling and Prefix Beck-Fiala". *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC).* New York, NY: Association for Computing Machinery, 2022, pp. 331–342. DOI: 10.1145/3519935.3520077.

[BS97] Roberto J. Bayardo Jr and Robert C. Schrag. "Using CSP Look-Back Techniques to Solve Real-World SAT Instances". *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI)* (1997), pp. 203–208.

[BST19] Niv Buchbinder, Danny Segev, and Yevgeny Tkach. "Online Algorithms for Maximum Cardinality Matching with Edge Arrivals". *Algorithmica* 81.5 (2019), pp. 1781–1799. DOI: 10.1007/s00453-018-0505-7.

[BSW23] Niv Buchbinder, Joseph (Seffi) Naor, and David Wajc. "Lossless Online Rounding for Online Bipartite Matching (Despite Its Impossibility)". *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2023, pp. 2030–2068. DOI: 10.1137/1.9781611977554.ch78.

[BT69] E. M. L. Beale and John Tomlin. "Special Facilities in a General Mathematical Programming System for Nonconvex Problems Using Ordered Sets of Variables". *Proceedings of the Fifth International Conference on Operational Research.* 1969, pp. 447–454.

[BV04a] Rene Beier and Berthold Vöcking. "Probabilistic Analysis of Knapsack Core Algorithms". *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2004, pp. 468–477.

[BV04b] Rene Beier and Berthold Vöcking. “Typical Properties of Winners and Losers in Discrete Optimization”. *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC).* New York, NY: Association for Computing Machinery, 2004, pp. 343–352. DOI: 10.1145/1007352.1007409.

[BV04c] Dimitris Bertsimas and Santosh Vempala. “Solving Convex Programs by Random Walks”. *Journal of the ACM* 51.4 (2004), pp. 540–556. DOI: 10.1145/1008731.1008733.

[CCZ10] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. “Polyhedral Approaches to Mixed Integer Linear Programming”. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art.* Berlin: Springer, 2010, pp. 343–385. DOI: 10.1007/978-3-540-68279-0_11.

[CCZ14] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming.* 1st ed. 2014. Graduate Texts in Mathematics 271. Cham: Springer, 2014. DOI: 10.1007/978-3-319-11008-0.

[CGM13] Marek Cygan, Fabrizio Grandoni, and Monaldo Mastrolilli. “How to Sell Hyperedges: The Hypermatching Assignment Problem”. *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2013, pp. 342–351. DOI: 10.1137/1.9781611973105.25.

[CGS17] Kevin K. H. Cheung, Ambros Gleixner, and Daniel E. Steffy. “Verifying Integer Programming Results”. *Integer Programming and Combinatorial Optimization (IPCO).* Lecture Notes in Computer Science 10328. Cham: Springer International Publishing, 2017, pp. 148–160. DOI: 10.1007/978-3-319-59250-3_13.

[Cha01] Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity.* Cambridge: Cambridge Univ. Press, 2001.

[CKSW13] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter. “A Hybrid Branch-and-Bound Approach for Exact Rational Mixed-Integer Programming”. *Mathematical Programming Computation* 5.3 (2013), pp. 305–344. DOI: 10.1007/s12532-013-0055-6.

[CL12] Yuk Hei Chan and Lap Chi Lau. “On Linear and Semidefinite Programming Relaxations for Hypergraph Matching”. *Mathematical Programming* 135.1 (2012), pp. 123–148. DOI: 10.1007/s10107-011-0451-5.

[CMS17] Luís Cruz-Filipe, Joao Marques-Silva, and Peter Schneider-Kamp. “Efficient Certified Resolution Proof Checking”. *Tools and Algorithms for the Construction and Analysis of Systems.* Lecture Notes in Computer Science 10205. Berlin: Springer, 2017, pp. 118–135. DOI: 10.1007/978-3-662-54577-5_7.

[Cos09] Kevin P. Costello. “Balancing Gaussian Vectors”. *Israel Journal of Mathematics* 172.1 (2009), pp. 145–156. DOI: 10.1007/s11856-009-0068-z.

[CP99] J. Clausen and M. Perregaard. “On the Best Search Strategy in Parallel Branch-and-bound: Best-First Search versus Lazy Depth-First Search”. *Annals of Operations Research* 90.0 (1999), pp. 1–17. DOI: 10.1023/A:1018952429396.

[Cru+17] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt, Matt Kaufmann, and Peter Schneider-Kamp. “Efficient Certified RAT Verification”. *Automated Deduction – CADE 26.* Cham: Springer International Publishing, 2017, pp. 220–236.

[CST14] William Chen, Anand Srivastav, and Giancarlo Travaglini, eds. *A Panorama of Discrepancy Theory*. Lecture Notes in Mathematics 2107. Cham: Springer International Publishing, 2014. DOI: 10.1007/978-3-319-04696-9.

[Dad12] Daniel Dadush. "Integer Programming, Lattice Algorithms, and Deterministic Volume Estimation". PhD thesis. Atlanta, GA: Georgia Tech, 2012.

[Dak65] R. J. Dakin. "A Tree-Search Algorithm for Mixed Integer Programming Problems". *The Computer Journal* 8.3 (1965), pp. 250–255. DOI: 10.1093/comjnl/8.3.250.

[Dan91] George B. Dantzig. "Linear Programming". *History of Mathematical Programming: A Collection of Personal Reminiscences*. Ed. by Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and Alexander Schrijver. Amsterdam: CWI, 1991.

[DCD95] Pallab Dasgupta, P. P. Chakrabarti, and S. C. DeSarkar. "A near Optimal Algorithm for the Extended Cow-Path Problem in the Presence of Relative Errors". *Foundations of Software Technology and Theoretical Computer Science*. Red. by Gerhard Goos, Juris Hartmanis, and Jan Leeuwen. Lecture Notes in Computer Science 1026. Berlin: Springer, 1995, pp. 22–36. DOI: 10.1007/3-540-60692-0_38.

[DDM23] Santanu S. Dey, Yatharth Dubey, and Marco Molinaro. "Branch-and-Bound Solves Random Binary IPs in Poly(n)-Time". *Mathematical Programming* 200.1 (2023), pp. 569–587. DOI: 10.1007/s10107-022-01895-4.

[DF89] Martin Dyer and Alan Frieze. "Probabilistic Analysis of the Multidimensional Knapsack Problem". *Mathematics of Operations Research* 14.1 (1989), pp. 162–176. URL: http://www.jstor.org/stable/3689842.

[DF92] Martin Dyer and Alan Frieze. "Probabilistic Analysis of the Generalised Assignment Problem". *Mathematical Programming* 55.1-3 (1992), pp. 169–181. DOI: 10.1007/BF01581197.

[DFKP04] Krzysztof Diks, Pierre Fraigniaud, Evangelos Kranakis, and Andrzej Pelc. "Tree Exploration with Little Memory". *Journal of Algorithms* 51.1 (2004), pp. 38–63. DOI: 10.1016/j.jalgor.2003.10.002.

[DJ12] Nikhil R. Devanur and Kamal Jain. "Online Matching with Concave Returns". *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing (STOC)*. New York, NY: Association for Computing Machinery, 2012, pp. 137–144. DOI: 10.1145/2213977.2213992.

[DJK13] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. "Randomized Primal-Dual Analysis of RANKING for Online Bipartite Matching". *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2013, pp. 101–107. DOI: 10.1137/1.9781611973105.7.

[dMB08] Leonardo Mendonça de Moura and Nikolaj S Bjørner. "Proofs and Refutations, and Z3". *Proceedings of the LPAR 2008 Workshops Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics*. CEUR Workshop Proceedings, 2008, pp. 123–132. URL: https://ceur-ws.org/Vol-418/paper10.pdf.

[Doe11] Benjamin Doerr. "Analyzing Randomized Search Heuristics: Tools from Probability Theory". *Theory of Randomized Search Heuristics*. Series on Theoretical Computer Science 1. World Scientific, 2011, pp. 1–20. DOI: 10.1142/9789814282673_0001.

[EFFS21] Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. “An Economics-Based Analysis of RANKING for Online Bipartite Matching”. *Symposium on Simplicity in Algorithms (SOSA)*. Society for Industrial and Applied Mathematics. 2021, pp. 107–110.

[EG22] Leon Eifler and Ambros Gleixner. “A Computational Status Update for Exact Rational Mixed Integer Programming”. *Mathematical Programming* (2022). DOI: 10.1007/s10107-021-01749-5.

[EG23] Leon Eifler and Ambros Gleixner. “Safe and Verified Gomory Mixed Integer Cuts in a Rational MIP Framework”. *SIAM Journal on Optimization* (2023).

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. “Justifying All Differences Using Pseudo-Boolean Reasoning”. *Proceedings of the AAAI Conference on Artificial Intelligence* 34.02 (2020), pp. 1486–1494. DOI: 10.1609/aaai.v34i02.5507.

[EGP22] Leon Eifler, Ambros Gleixner, and Jonad Pulaj. “A Safe Computational Framework for Integer Programming Applied to Chvátal’s Conjecture”. *ACM Transactions on Mathematical Software* 48.2 (2022). DOI: 10.1145/3485630.

[EJ12] Mourad El Ouali and Gerold Jäger. “The B-Matching Problem in Hypergraphs: Hardness and Approximability”. *Combinatorial Optimization and Applications*. Red. by David Hutchison et al. Lecture Notes in Computer Science 7402. Berlin: Springer, 2012, pp. 200–211. DOI: 10.1007/978-3-642-31770-5_18.

[EL19] Esther Ezra and Shachar Lovett. “On the Beck-Fiala Conjecture for Random Set Systems”. *Random Structures & Algorithms* 54.4 (2019), pp. 665–675. DOI: 10.1002/rsa.20810.

[ES18] Ronen Eldan and Mohit Singh. “Efficient Algorithms for Discrepancy Minimization in Convex Sets”. *Random Structures & Algorithms* 53.2 (2018), pp. 289–307. DOI: 10.1002/rsa.20763.

[EW19] Friedrich Eisenbrand and Robert Weismantel. “Proximity Results and Faster Algorithms for Integer Programming Using the Steinitz Lemma”. *ACM Transactions on Algorithms* 16.1 (2019), 5:1–5:14. DOI: 10.1145/3340322.

[Far02] Julius Farkas. “Theorie Der Einfachen Ungleichungen.” *Journal für die reine und angewandte Mathematik (Crelles Journal)* 1902.124 (1902), pp. 1–27. DOI: 10.1515/crll.1902.124.1.

[FHTZ22] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. “Edge-Weighted Online Bipartite Matching”. *Journal of the ACM* 69.6 (2022), pp. 1–35. DOI: 10.1145/3556971.

[Fis+16] Matteo Fischetti, Andrea Lodi, Michele Monaci, Domenico Salvagnin, and Andrea Tramontani. “Improving Branch-and-Cut Performance by Random Sampling”. *Mathematical Programming Computation* 8.1 (2016), pp. 113–132. DOI: 10.1007/s12532-015-0096-0.

[FK89] Merrick L. Furst and Ravi Kannan. “Succinct Certificates for Almost All Subset Sum Problems”. *SIAM Journal on Computing* 18.3 (1989), pp. 550–558. DOI: 10.1137/0218037.

[Fra99] Matthieu Fradelizi. “Hyperplane Sections of Convex Bodies in Isotropic Position.” *Beiträge zur Algebra und Geometrie* 40.1 (1999), pp. 163–183. URL: https://eudml.org/doc/232153.

[Fre93] Greg N. Frederickson. “An Optimal Algorithm for Selection in a Min-Heap”. *Information and Computation* 104.2 (1993), pp. 197–214. doi: 10.1006/inco.1993.1030.

[FS07] Alan Frieze and Gregory B. Sorkin. “The Probabilistic Relationship Between the Assignment and Asymmetric Traveling Salesman Problems”. *SIAM Journal on Computing* 36.5 (2007), pp. 1435–1452. doi: 10.1137/S0097539701391518.

[FS13] Matteo Fischetti and Domenico Salvagnin. “Approximating the Split Closure”. *INFORMS Journal on Computing* (2013). doi: 10.1287/ijoc.1120.0543.

[FS20] Cole Franks and Michael Saks. “On the Discrepancy of Random Matrices with Many Columns”. *Random Structures & Algorithms* 57.1 (2020), pp. 64–96. doi: 10.1002/rsa.20909.

[Gal14] David Galvin. *Three Tutorial Lectures on Entropy and Counting*. 2014. doi: 10.48550/arXiv.1406.7872. Pre-published.

[Gam+19] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. “Online Matching with General Arrivals”. *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. Baltimore, MD: IEEE, 2019, pp. 26–37. doi: 10.1109/FOCS.2019.00011.

[Gle+21] Ambros Gleixner et al. “MIPLIB 2017: Data-driven Compilation of the 6th Mixed-Integer Programming Library”. *Mathematical Programming Computation* (2021). doi: 10.1007/s12532-020-00194-3.

[GM08] Gagan Goel and Aranyak Mehta. “Online Budgeted Matching in Random Input Models with Applications to Adwords”. *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2008, pp. 982–991.

[GM84] Andrew V. Goldberg and Alberto Marchetti-Spaccamela. “On Finding the Exact Solution of a Zero-One Knapsack Problem”. *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC)*. New York, NY: Association for Computing Machinery, 1984, pp. 359–368. doi: 10.1145/800057.808701.

[GN03] Evgueni Goldberg and Yakov Novikov. “Verification of Proofs of Unsatisfiability for CNF Formulas”. *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. IEEE Computer Society, 2003, pp. 886–891. doi: 10.1109/DATE.2003.1253718.

[Goc19] Stephan Gocht. *VeriPB*. Version 0.1.0. 2019. doi: 10.5281/zenodo.3548582.

[GP24] Max Gläser and Marc E. Pfetsch. “Sub-Exponential Lower Bounds for Branch-and-Bound with General Disjunctions via Interpolation”. *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2024, pp. 3747–3764. doi: 10.1137/1.9781611977912.132.

[Grü60] B. Grünbaum. “Partitions of Mass-Distributions and of Convex Bodies by Hyperplanes.” *Pacific Journal of Mathematics* 10.4 (1960), pp. 1257–1261. doi: 10.2140/pjm.1960.10.1257.

[GT15] Torbjorn Granlund and Gmp Development Team. *GNU MP 6.0 Multiple Precision Arithmetic Library*. London, GBR: Samurai Media Limited, 2015.

[Haa81] Uffe Haagerup. “The Best Constants in the Khintchine Inequality”. *Studia Mathematica* 70.3 (1981), pp. 231–283. URL: https://eudml.org/doc/218383.

[HLZ23] Sophie Huiberts, Yin Tat Lee, and Xinzhi Zhang. “Upper and Lower Bounds on the Smoothed Complexity of the Simplex Method”. *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC).* New York, NY: Association for Computing Machinery, 2023, pp. 1904–1917. DOI: 10.1145/3564246.3585124.

[Hoe22] Alexander Hoen. *Scipopt/Papilo: V2.0.0.* Version v2.0.0. Zenodo, 2022. DOI: 10.5281/zenodo.6414882.

[HP19] Christopher Hojny and Marc E. Pfetsch. “Polytopes Associated with Symmetry Handling”. *Mathematical Programming* 175 (2019), pp. 197–240. DOI: 10.1007/s10107-018-1239-7.

[HR17] Rebecca Hoberg and Thomas Rothvoss. “A Logarithmic Additive Integrality Gap for Bin Packing”. *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2017, pp. 2616–2625. DOI: 10.1137/1.9781611974782.172.

[HR19] Rebecca Hoberg and Thomas Rothvoss. “A Fourier-Analytic Approach for the Discrepancy of Random Set Systems”. *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA).* Society for Industrial and Applied Mathematics, 2019, pp. 2547–2556. DOI: 10.1137/1.9781611975482.156.

[HSS06] Elad Hazan, Shmuel Safra, and Oded Schwartz. “On the Complexity of Approximating K-Set Packing”. *Computational Complexity* 15.1 (2006), pp. 20–39. DOI: 10.1007/s00037-006-0205-6.

[HTWZ19] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. “Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals”. *ACM Transactions on Algorithms (TALG)* 15.3 (2019), pp. 1–15.

[HTWZ20] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. “Fully Online Matching II: Beating Ranking and Water-filling”. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS).* 2020, pp. 1380–1391. DOI: 10.1109/FOCS46700.2020.00130.

[Hua+20] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. “Fully Online Matching”. *J. ACM* 67.3 (2020), 17:1–17:25.

[HZZ20] Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. “Adwords in a panorama”. *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS).* IEEE. 2020, pp. 1416–1426.

[IEE08] *IEEE Standard for Floating-Point Arithmetic.* Standard IEEE Std 754-2008. New York, NY: IEEE Computer Society, 2008. URL: https://web.archive.org/web/20160806053349/http://www.csee.umbc.edu/~tsimo1/CMSC455/IEEE-754-2008.pdf.

[IWY16] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. “Half-Integrality, LP-branching, and FPT Algorithms”. *SIAM Journal on Computing* 45.4 (2016), pp. 1377–1411. DOI: 10.1137/140962838.

[JR23] Klaus Jansen and Lars Rohwedder. “On Integer Programming, Discrepancy, and Convolution”. *Mathematics of Operations Research* 48.3 (2023), pp. 1481–1495. DOI: 10.1287/moor.2022.1308.

[Kam] Tom Kamphans. "Models and Algorithms for Online Exploration and Search". PhD thesis. Rheinische Friedrich-Wilhelms-Universität Bonn. URL: https://hdl.handle.net/20.500.11811/2622.

[Kan87] Ravi Kannan. "Minkowski's Convex Body Theorem and Integer Programming". *Mathematics of Operations Research* 12.3 (1987), pp. 415–440. DOI: 10.1287/moor.12.3.415.

[Kar72] Richard M. Karp. "Reducibility Among Combinatorial Problems". *Proceedings of a Symposium on the Complexity of Computer Computations.* The IBM Research Symposia Series. New York, NY: Plenum Press, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[Kha79] L. G. Khachiyan. "A polynomial algorithm in linear programming". *Doklady Akademii Nauk SSSR* (1979).

[Khi23] A. Khintchine. "Über dyadische Brüche". *Mathematische Zeitschrift* 18.1 (1923), pp. 109–116. DOI: 10.1007/BF01192399.

[KLP12] Greg Kuperberg, Shachar Lovett, and Ron Peled. "Probabilistic Existence of Rigid Combinatorial Structures". *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing (STOC).* New York, NY: Association for Computing Machinery, 2012, pp. 1091–1106. DOI: 10.1145/2213977.2214075.

[KMT11] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. "Online bipartite matching with unknown distributions". *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC).* 2011, pp. 587–596.

[KP00] Bala Kalyanasundaram and Kirk R. Pruhs. "An Optimal Deterministic Algorithm for Online B-Matching". *Theoretical Computer Science* 233.1-2 (2000), pp. 319–325. DOI: 10.1016/S0304-3975(99)00140-1.

[KP09] Nitish Korula and Martin Pál. "Algorithms for Secretary Problems on Graphs and Hypergraphs". *Automata, Languages and Programming (ICALP).* Vol. 2. Berlin: Springer, 2009, pp. 508–520. DOI: 10.1007/978-3-642-02930-1_42.

[KP94] Bala Kalyanasundaram and Kirk R. Pruhs. "Constructing Competitive Tours from Local Information". *Theoretical Computer Science* 130.1 (1994), pp. 125–138. DOI: 10.1016/0304-3975(94)90155-4.

[KRTV13] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. "An Optimal Online Algorithm for Weighted Bipartite Matching and Extensions to Combinatorial Auctions". *Algorithms – ESA 2013.* Red. by David Hutchison et al. Vol. 8125. Berlin: Springer, 2013, pp. 589–600. DOI: 10.1007/978-3-642-40450-4_50.

[KS18] Franklin Kenter and Daphne Skipper. "Integer-Programming Bounds on Pebbling Numbers of Cartesian-Product Graphs". *Combinatorial Optimization and Applications.* 2018, pp. 681–695. DOI: 10.1007/978-3-030-04651-4_46.

[KSW86] Richard M. Karp, Michael Saks, and Avi Wigderson. "On a Search Problem Related to Branch-and-Bound Procedures". *27th Annual Symposium on Foundations of Computer Science (FOCS).* Toronto, ON, Canada: IEEE, 1986, pp. 19–28. DOI: 10.1109/SFCS.1986.34.

[KTRV14] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. "Primal Beats Dual on Online Packing LPs in the Random-Order Model". *Proceedings of the forty-sixth annual ACM symposium on Theory of computing* (2014), pp. 303–312. DOI: 10.1145/2591796.2591810.

[Kuz96] N. N. Kuzyurin. "An Integer Linear Programming Algorithm Polynomial in the Average Case". *Discrete Analysis and Operations Research.* Mathematics and Its Applications. Dordrecht: Springer Netherlands, 1996, pp. 143–152. DOI: 10.1007/978-94-009-1606-7_11.

[KVV90] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. "An Optimal Algorithm for On-Line Bipartite Matching". *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing - STOC '90.* Baltimore, Maryland, United States: ACM Press, 1990, pp. 352–358. DOI: 10.1145/100216.100262.

[LD60] Alisa H. Land and Alison G. Doig. "An Automatic Method of Solving Discrete Programming Problems". *Econometrica* 28.3 (1960), pp. 497–520. DOI: 10.2307/1910129.

[Len83] Hendrik W. Lenstra. "Integer Programming with a Fixed Number of Variables". *Mathematics of Operations Research* 8.4 (1983), pp. 538–548. DOI: 10.1287/moor.8.4.538.

[Lib12] Leo Liberti. "Symmetry in Mathematical Programming". *Mixed Integer Nonlinear Programming.* The IMA Volumes in Mathematics and Its Applications. New York, NY: Springer, 2012, pp. 263–283. DOI: 10.1007/978-1-4614-1927-3_9.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. "Factoring Polynomials with Rational Coefficients". *Mathematische Annalen* 261.4 (1982), pp. 515–534. DOI: 10.1007/BF01457454.

[LM00] Beatrice Laurent and Pascal Massart. "Adaptive Estimation of a Quadratic Functional by Model Selection". *The Annals of Statistics* 28.5 (2000). DOI: 10.1214/aos/1015957395.

[LM15] Shachar Lovett and Raghu Meka. "Constructive Discrepancy Minimization by Walking on the Edges". *SIAM Journal on Computing* 44.5 (2015), pp. 1573–1582. DOI: 10.1137/130929400.

[Lok+14] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. "Faster Parameterized Algorithms Using Linear Programming". *ACM Transactions on Algorithms* 11.2 (2014), 15:1–15:31. DOI: 10.1145/2566616.

[LPR20] Giuseppe Lancia, Eleonora Pippia, and Franca Rinaldi. "Using Integer Programming to Search for Counterexamples: A Case Study". *Mathematical Optimization Theory and Operations Research.* 2020, pp. 69–84. DOI: 10.1007/978-3-030-49988-4.

[LRS11] Lap Chi Lau, R. Ravi, and Mohit Singh. *Iterative Methods in Combinatorial Optimization.* 1st ed. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511977152.

[LS99] Jeff T. Linderoth and Martin W. P. Savelsbergh. "A Computational Study of Search Strategies for Mixed Integer Programming". *INFORMS Journal on Computing* 11.2 (1999), pp. 173–187. DOI: 10.1287/ijoc.11.2.173.

[LSV86] László Lovász, Joel Spencer, and Katalin Vesztergombi. "Discrepancy of Set-systems and Matrices". *European Journal of Combinatorics* 7.2 (1986), pp. 151–160. DOI: 10.1016/S0195-6698(86)80041-5.

[Lue82] George S. Lueker. "On the Average Difference between the Solutions to Linear and Integer Knapsack Problems". *Applied Probability-Computer Science: The Interface.* Progress in Computer Science 2. Boston, MA: Birkhäuser, 1982, pp. 489–504. DOI: 10.1007/978-1-4612-5791-2_22.

[LV07] László Lovász and Santosh Vempala. “The Geometry of Logconcave Functions and Sampling Algorithms”. *Random Structures and Algorithms* 30.3 (2007), pp. 307–358. DOI: 10.1002/rsa.20135.

[LZ17] Andrea Lodi and Giulia Zarpellon. “On Learning and Branching: A Survey”. *TOP* 25.2 (2017), pp. 207–236. DOI: 10.1007/s11750-017-0451-6.

[Mat99] Jiří Matoušek. *Geometric Discrepancy*. Algorithms and Combinatorics 18. Berlin: Springer, 1999. DOI: 10.1007/978-3-642-03942-3.

[Meh13] Aranyak Mehta. “Online Matching and Ad Allocation”. *Found. Trends Theor. Comput. Sci.* 8.4 (2013), pp. 265–368.

[Mil+22] Matthias Miltenberger et al. *Scipopt/Soplex: V6.0.0.* Version release-600. Zenodo, 2022. DOI: 10.5281/zenodo.6414886.

[MJSS16] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. “Branch-and-Bound Algorithms: A Survey of Recent Advances in Searching, Branching, and Pruning”. *Discrete Optimization* 19 (2016), pp. 79–102. DOI: 10.1016/j.disopt.2016.01.005.

[MMNS11] R. M. Mcconnell, K. Mehlhorn, S. Näher, and P. Schweitzer. “Survey: Certifying Algorithms”. *Computer Science Review* 5.2 (2011), pp. 119–161. DOI: 10.1016/j.cosrev.2010.09.009.

[MMPP23] Calum MacRury, Tomáš Masařík, Leilani Pai, and Xavier Pérez-Giménez. “The Phase Transition of Discrepancy in Random Hypergraphs”. *SIAM Journal on Discrete Mathematics* 37.3 (2023), pp. 1818–1841. DOI: 10.1137/21M1451427.

[MMS12] Nicole Megow, Kurt Mehlhorn, and Pascal Schweitzer. “Online Graph Exploration: New Results on Old and New Algorithms”. *Theoretical Computer Science* 463 (2012), pp. 62–72. DOI: 10.1016/j.tcs.2012.06.034.

[Mos+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. “Chaff: Engineering an Efficient SAT Solver”. *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*. 2001, pp. 530–535. DOI: 10.1145/378239.379017.

[MP80] J. Ian Munro and Mike S. Paterson. “Selection and Sorting with Limited Storage”. *Theoretical Computer Science* 12.3 (1980), pp. 315–323. DOI: 10.1016/0304-3975(80)90061-4.

[MRST20] Yuhang Ma, Paat Rusmevichientong, Mika Sumida, and Huseyin Topaloglu. “An Approximation Algorithm for Network Revenue Management Under Nonstationary Arrivals”. *Operations Research* 68.3 (2020), pp. 834–855. DOI: 10.1287/opre.2019.1931.

[MS99] João P. Marques-Silva and Karem A. Sakallah. “GRASP: A Search Algorithm for Propositional Satisfiability”. *IEEE Transactions on Computers* 48.5 (1999), pp. 506–521. DOI: 10.1109/12.769433.

[MSV23] Javier Marinkovic, José A. Soto, and Victor Verdugo. *Online Combinatorial Assignment in Independence Systems*. Version 1. 2023. DOI: 10.48550/arXiv.2311.00890. Pre-published.

[MSVV07] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. “AdWords and Generalized Online Matching”. *Journal of the ACM* 54.5 (2007), p. 22. DOI: 10.1145/1284320.1284321.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: An Introduction to Randomized Algorithms and Probabilistic Analysis*. New York: Cambridge University Press, 2005. DOI: 10.1017/CBO9780511813603.

[MY] Mohammad Mahdian and Qiqi Yan. "Online Bipartite Matching with Random Arrivals: An Approach Based on Strongly Factor-Revealing LPs". DOI: 10.1145/1993636.1993716.

[NW88] George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Ltd, 1988. DOI: 10.1002/9781118627372.

[Ost09] James Ostrowski. "Symmetry in Integer Programming". PhD thesis. Lehigh University, 2009. URL: https://coral.ise.lehigh.edu/~pubs/files/ostrowski_phd_thesis.pdf.

[Pap81] Christos H. Papadimitriou. "On the Complexity of Integer Programming". *Journal of the ACM* 28.4 (1981), pp. 765–768. DOI: 10.1145/322276.322287.

[Pot18] Aditya Potukuchi. *Discrepancy in Random Hypergraph Models*. 2018. DOI: 10.48550/arXiv.1811.01491. Pre-published.

[PPSV15] Andrea Pietracaprina, Geppino Pucci, Fransesco Silvestri, and Fabio Vandin. "Space-Efficient Parallel Algorithms for Combinatorial Search Problems". *Journal of Parallel and Distributed Computing* 76 (2015), pp. 58–65. DOI: 10.1016/j.jpdc.2014.09.007.

[Pré71] András Prékopa. "Logarithmic Concave Measures with Application to Stochastic Programming". *Acta Universitatis Szegediensis. Acta Scientiarum Mathematicarum* 32 (1971), pp. 301–316.

[PSST22] Marco Pavone, Amin Saberi, Maximilian Schiffer, and Matt Wu Tsao. "Technical Note—Online Hypergraph Matching with Delays". *Operations Research* 70.4 (2022), pp. 2194–2212. DOI: 10.1287/opre.2022.2277.

[PTW10] Gábor Pataki, Mustafa Tural, and Erick B. Wong. "Basis Reduction and the Complexity of Branch-and-Bound". *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, 2010, pp. 1254–1261. DOI: 10.1137/1.9781611973075.100.

[Pul20] Jonad Pulaj. "Cutting Planes for Families Implying Frankl's Conjecture". *Mathematics of Computation* 89.322 (2020), pp. 829–857. DOI: 10.1090/mcom/3461.

[Rot17] Thomas Rothvoss. "Constructive Discrepancy Minimization for Convex Sets". *SIAM Journal on Computing* 46.1 (2017), pp. 224–234. DOI: 10.1137/141000282.

[RR23] Victor Reis and Thomas Rothvoss. "The Subspace Flatness Conjecture and Faster Integer Programming". *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. Santa Cruz, CA: IEEE, 2023, pp. 974–988. DOI: 10.1109/FOCS57990.2023.00060.

[RT87] Prabhakar Raghavan and Clark D. Tompson. "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs". *Combinatorica* 7.4 (1987), pp. 365–374. DOI: 10.1007/BF02579324.

[RV07] Heiko Röglin and Berthold Vöcking. "Smoothed Analysis of Integer Programming". *Mathematical Programming* 110.1 (2007), pp. 21–56. DOI: 10.1007/s10107-006-0055-7.

[Sav94] M. W. P. Savelsbergh. "Preprocessing and Probing Techniques for Mixed Integer Programming Problems". *ORSA Journal on Computing* 6.4 (1994), pp. 445–454. DOI: 10.1287/ijoc.6.4.445.

[SBD19] Youcef Sahraoui, Pascale Bendotti, and Claudia D'Ambrosio. "Real-World Hydro-Power Unit-Commitment: Dealing with Numerical Errors and Feasibility Issues". *Energy* 184 (2019), pp. 91–104. DOI: 10.1016/j.energy.2017.11.064.

[Sch03] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Algorithms and Combinatorics 24. Berlin: Springer, 2003.

[Sch11] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester Weinheim: Wiley, 2011.

[SS93] Leena M. Suhl and Uwe H. Suhl. "A Fast LU Update for Linear Programming". *Annals of Operations Research* 43.1 (1993), pp. 33–47. DOI: 10.1007/BF02025534.

[SS95] Anand Srivastav and Peter Stangier. "Weighted Fractional and Integral K-Matching in Hypergraphs". *Discrete Applied Mathematics* 57.2-3 (1995), pp. 255–269. DOI: 10.1016/0166-218X(94)00107-O.

[ST04] Daniel A. Spielman and Shang-Hua Teng. "Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time". *Journal of the ACM* 51.3 (2004), pp. 385–463. DOI: 10.1145/990308.990310.

[Sti45] George J. Stigler. "The Cost of Subsistence". *Journal of Farm Economics* 27.2 (1945), p. 303. DOI: 10.2307/1231810.

[SW72] Elias M. Stein and Guido Weiss. *Introduction to Fourier Analysis on Euclidean Spaces.* Princeton Mathematical Series 32. Princeton University Press, 1972. DOI: 10.1515/9781400883899.

[Tro15] Joel A. Tropp. "An Introduction to Matrix Concentration Inequalities". *Foundations and Trends® in Machine Learning* 8.1-2 (2015), pp. 1–230. DOI: 10.1561/2200000048.

[TU24] Thorben Tröbst and Rajan Udwani. *Almost Tight Bounds for Online Hypergraph Matching*. 2024. DOI: 10.48550/arXiv.2402.08775. Pre-published.

[Ver18] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science.* Cambridge Series in Statistical and Probabilistic Mathematics 47. Cambridge, NY: Cambridge University Press, 2018. DOI: 10.1017/9781108231596.

[Vig+23] Stefan Vigerske et al. *Scipopt/Scip: V8.0.3.* Version v803. Zenodo, 2023. DOI: 10.5281/zenodo.8105539.

[Wah17] Magnus Wahlström. "LP-branching Algorithms Based on Biased Graphs". *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms.* Society for Industrial and Applied Mathematics, 2017, pp. 1559–1570. DOI: 10.1137/1.9781611974782.102.

[WBH17] Jakob Witzig, Timo Berthold, and Stefan Heinz. "Experiments with Conflict Analysis in Mixed Integer Programming". *Integration of AI and OR Techniques in Constraint Programming.* Lecture Notes in Computer Science 10335. Cham: Springer International Publishing, 2017, pp. 211–220. DOI: 10.1007/978-3-319-59776-8_17.

[WBH21] Jakob Witzig, Timo Berthold, and Stefan Heinz. "Computational Aspects of Infeasibility Analysis in Mixed Integer Programming". *Mathematical Programming Computation* (2021). DOI: 10.1007/s12532-021-00202-0.

[WHJ14] Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. “DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs”. *Theory and Applications of Satisfiability Testing - SAT 2014*. Lecture Notes in Computer Science 8561. Springer, 2014, pp. 422–429. DOI: 10.1007/978-3-319-09284-3_31.

[WLH00] Kent Wilken, Jack Liu, and Mark Heffernan. “Optimal Instruction Scheduling Using Integer Programming”. *SIGPLAN Notices* 35.5 (2000), pp. 121–133. DOI: 10.1145/358438.349318.

[Wol21] Laurence A. Wolsey. *Integer Programming*. 2nd ed. Hoboken, NJ: Wiley, 2021. DOI: 10.1002/9781119606475.

[WW15] Yajun Wang and Sam Chiu-wai Wong. “Two-Sided Online Bipartite Matching and Vertex Cover: Beating the Greedy Algorithm”. *Automata, Languages, and Programming*. Lecture Notes in Computer Science 9134. Berlin: Springer, 2015, pp. 1070–1081. DOI: 10.1007/978-3-662-47672-7_87.

[Yao77] Andrew Chi-Chin Yao. “Probabilistic Computations: Toward a Unified Measure of Complexity”. *18th Annual Symposium on Foundations of Computer Science (FOCS)*. Washington DC: IEEE Computer Society, 1977, pp. 222–227. DOI: 10.1109/SFCS.1977.24.

Samenvatting

Optimalisatieproblemen zijn overal: van het maken van lesroosters tot het ontwerpen van computerchips. Veel van deze problemen kunnen worden geformuleerd als een *integer linear program* (ILP), een universeel optimalisatieprobleem. Hierdoor kunnen ze worden opgelost met een generiek computerprogramma genaamd MIP solver. Er bestaan vele MIP solvers, die vrijwel allemaal gebaseerd zijn op hetzelfde algoritme: het *branch-and-bound algoritme*.

In dit proefschrift onderzoeken we verschillende aspecten van dit algoritme. Allereerst onderzoeken we in Hoofdstuk 3 de oplossingstijd die het algoritme nodig heeft om willekeurig gegenereerde ILP's op te lossen. We laten zien dat voor diverse klassen van willekeurige ILP's, de verwachte oplossingstijd enkel polynomiaal afhangt van het aantal variabelen in het ILP, terwijl de oplossingstijd in het slechtste geval exponentieel is. Dit resultaat is gebaseerd op nieuwe bovengrenzen voor de *integrality gap* van deze ILPs. Om deze bovengrenzen te bewijzen, introduceren we nieuwe theorie over de *discrepancy* van willekeurige matrices.

Een belangrijk aspect van het branch-and-bound algoritme is de zogenaamde *zoekstrategie*, die bepaalt in welke volgorde deelproblemen worden opgelost. De moeilijkheid van het ontwerpen van een goede zoekstrategie is dat een zoekrichting moet worden gekozen zonder kennis van de volledige branch-and-bound boom. In Hoofdstuk 5 analyseren we zoekstrategiën op een theoretische manier, in het *explorable heap model*. We introduceren een nieuwe zoekstrategie, en bewijzen dat deze strategie in dit model beter presteert dan bestaande zoekstrategiën.

In Hoofdstuk 6 bestuderen we een ander probleem dat draait om het maken van keuzes op basis van onvolledige informatie: het *online matching* probleem voor k-uniforme hypergrafen. Één voor één arriveren de *online* knopen samen met hun incidente hyperkanten, en het algoritme moet direct beslissen welke hyperkant toe te voegen aan de *matching*. We richten ons op het geval $k = 3$ en laten zien dat het optimale competitieve ratio van een online algoritme voor de fractionele versie van dit probleem gelijk is aan $\frac{e-1}{e+1} \approx 0.4621$. Verder presenteren we een algoritme voor de geheeltallige versie van het probleem dat op hypergrafen met begrensde graad

een competitieve ratio strikt beter dan $\frac{1}{k}$ bereikt.

ILP's worden in het algemeen opgelost met behulp van *floating-point* getallen, aangezien dit in de praktijk veel sneller is dan het werken met exacte breuken. Dit kan echter leiden tot numerieke problemen, zoals afrondingsfouten. In Hoofdstuk 7 onderzoeken we het oplossen van ILP's met behulp van exacte breuken. We maken hierbij gebruik van de exacte MIP solver SCIP. De exacte versie van deze solver heeft geen ondersteuning voor *constraint propagation* en *conflictanalyse*, twee technieken die het oplossen van ILP's erg kunnen versnellen. We laten zien dat het mogelijk is om deze technieken te implementeren in de exacte solver, en dat dit leidt tot een significante verbetering van de oplossingstijd.

Curriculum vitae

Sander Borst was born on January 29, 1996 in Leidschendam, the Netherlands. In 2014 he started his studies at the Delft University of Technology, where he obtained BSc degrees in both mathematics (cum laude) and computer science in 2018. He then continued his master studies in mathematics at the same university, specializing in algorithms and optimization. As part of the program, he did an internship at ORTEC, working on a project related to vehicle routing. In 2020, he obtained his MSc degree in mathematics cum laude. He then started his PhD at Centrum Wiskunde & Informatica (CWI) in Amsterdam, advised by Daniel Dadush.

Acknowledgements

I am grateful to all the people who have helped me in the process of writing this thesis. First and foremost, I would like to thank Daniel. You have been a great advisor. Thank you for your guidance in all the research projects that we have worked on together and also for all the freedom that you gave me to explore other topics. You have introduced me to many people in the field and gave me a lot of opportunities to collaborate with them. I would also like to thank Gunther for being my other promotor and for his great feedback on my thesis.

I am grateful to the thesis committee members, Karen, Hans, Santanu, Alan and Anupam, for their time and effort in reviewing my thesis.

This thesis would not have been possible without all great coauthors I have worked with during my PhD: Aleksei, Matthijs, Daniel, Leon, Ambros, Sophie, Leo, Danish, Steven, Cedric, Dan, Neil, Makrand, Fedor and Samarth. Thank you for all the nice collaborations and for all the fun times we had together. Also, thank you, Sophie, for letting me use your thesis template.

Doing my PhD project at CWI has been a great experience thanks to all the great colleagues from the Networks and Optimization group. Thank you all!

I would very much like to thank Ambros for hosting me at the Zuse-Institute in Berlin, Anupam for hosting me at CMU in Pittsburgh and Moritz and Marek for hosting me at Bocconi University in Milan. Thanks also to all the other friendly people I have met at ZIB, CMU, EPFL, Bocconi University, and at workshops and conferences.

Finally, I would like to thank my family and friends for their support during my PhD project. Last but not least, I want to thank someone very special to me: Annelotte. Thank you for all your love and support.