NOVEL ANALYSIS OF THE BRANCH-AND-BOUND METHOD FOR INTEGER PROGRAMMING

A Dissertation
Presented to
The Academic Faculty

By

Yatharth Dubey

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering
College of Engineering

Georgia Institute of Technology

August 2022

© Yatharth Dubey 2022

NOVEL ANALYSIS OF THE BRANCH-AND-BOUND METHOD FOR INTEGER PROGRAMMING

Thesis committee:

Dr. Santanu S. Dey
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Mohit Singh
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Marco Molinaro
Computer Science Department
Pontifical Catholic University of Rio de Janeiro

Dr. Alejandro Toriello
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Amitabh Basu
Department of Applied Mathematics and Statistics
Johns Hopkins University

Date approved: August 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Mixed-integer linear programming (MILP) has become a pillar of operational decision making and optimization, with large-scale economic and societal impact. MILP solvers drive multi-billion dollar industries and the operation of critical infrastructure, and this ability to use MILPs to effectively make large-scale discrete decisions relies on the ability to solve MILPs efficiently. Despite a half-century of active research on the subject, critical components of these solvers' underlying algorithms remain poorly understood theoretically. This thesis provides novel and fundamental explanations for, and practical insights on, several long-analyzed phenomena in the branch-and-bound method, the workhorse algorithm of all state-of-the-art MILP solvers. In chapter 1, we give some background on branch-and-bound and related works.

These implementations of branch-and-bound typically use variable branching, that is, the child nodes are obtained by fixing some integer constrained variable to one of its possible values. Even though modern MILP solvers are able to solve very large-scale instances efficiently, relatively little attention has been given to understanding why the underlying branch-and-bound algorithm performs so well. In chapter 2, our goal is to theoretically analyze the performance of the standard variable branching based branch-and-bound algorithm. In order to avoid the exponential worst-case lower bounds, we follow the common idea of considering random instances. More precisely, we consider random integer programs where the entries of the coefficient matrix and the objective function are randomly sampled. Our main result is that with good probability branch-and-bound with variable branching explores only a polynomial number of nodes to solve these instances, for a fixed number of constraints. To the best of our knowledge this is the first known such result for a standard version of branch-and-bound. We believe that this result provides an indication as to why branch-and-bound with variable branching works so well in practice.

To understand the difficulties of branch-and-bound, in chapter 3 we study an algorithm that

can be viewed as an abstraction of modern MILP solvers: *general branch-and-bound*. That is, instances that are challenging for general branch-and-bound are likely to also be challenging for MILP solvers. A *general branch-and-bound tree* is a branch-and-bound tree which is allowed to use general disjunctions of the form $\pi^\top x \leq \pi_0 \vee \pi^\top x \geq \pi_0 + 1$, where π is an integer vector and π_0 is an integer scalar, to create child nodes. We construct a packing instance, a set covering instance, and a Traveling Salesman Problem instance, such that any general branch-and-bound tree that solves these instances must be of exponential size. We also verify that an exponential lower bound on the size of general branch-and-bound trees persists even when we add Gaussian noise to the coefficients of the cross-polytope, thus showing that a polynomial-size "smoothed analysis" upper bound is not possible.

Full strong-branching (henceforth referred to as strong-branching) is a well-known variable selection rule that is known experimentally to produce significantly smaller branch-and-bound trees in comparison to all other known variable selection rules. In chapter 4, we attempt an analysis of the performance of the strong-branching rule both from a theoretical and a computational perspective. On the positive side for strong-branching we identify vertex cover as a class of instances where this rule provably works well. In particular, for vertex cover we present an upper bound on the size of the branch-and-bound tree using strong-branching as a function of the additive integrality gap, show how the Nemhauser-Trotter property of persistency which can be used as a pre-solve technique for vertex cover is being recursively and consistently used through-out the strong-branching based branch-and-bound tree, and finally provide an example of a vertex cover instance where not using strong-branching leads to a tree that has at least exponentially more nodes than the branch-and-bound tree based on strong-branching. On the negative side for strong-branching, we identify another class of instances where strong-branching based branch-and-bound tree has exponentially larger tree in comparison to another branch-and-bound tree for solving these instances. On the computational side, we conduct experiments on various types of instances like the lot-sizing problem and its variants, packing integer programs (IP), covering

IPs, chance constrained IPs, vertex cover, etc., to understand how much larger is the size of the strong-branching based branch-and-bound tree in comparison to the optimal branch-and-bound tree. The main take-away from these experiments is that for all these instances, the size of the strong-branching based branch-and-bound tree is within a factor of two of the size of the optimal branch-and-bound tree.

Finally, in chapter 5 we discuss possible extensions of the work covered in this thesis.

CHAPTER 1

INTRODUCTION

Solving combinatorial optimization problems to optimality is a central object of study in Operations Research, Computer Science, and Mathematics. One way to solve a combinatorial optimization problem is to model it as an integer program (IP), namely a problem of the form

$$\begin{aligned} \max\ & \langle c, x \rangle \\ \text{s.t. } & Ax \leq b \\ & x \in \mathbb{Z}^n, \end{aligned} \tag{IP}$$

and then use an IP solver. The branch-and-bound algorithm, invented by Land and Doig in [1], is the underlying algorithm implemented in all modern state-of-the-art MILP solvers.

As is well-known, the branch-and-bound algorithm searches the solution space by recursively partitioning it. The progress of the algorithm is monitored by maintaining a tree. Each node of the tree corresponds to a linear program (LP) solved, and in particular, the root-node corresponds to the LP relaxation of the integer program (i.e., the where the constraint $x \in \mathbb{Z}^n$ in (chapter IP) is removed). After solving the LP corresponding to a node, the feasible region of the LP is partitioned into two subproblems (which correspond to the child nodes of the given node), so that the fractional optimal solution of the LP is not included in either subproblem, but any integer feasible solution contained in the feasible region of the LP is included in one of the two subproblems. This is accomplished by adding an inequality of the form $\pi^\top x \leq \pi_0$ to the first subproblem and the inequality $\pi^\top x \geq \pi_0 + 1$ to the second subproblem (these two inequalities are referred as a disjunction), where π is an integer vector and π_0 is an integer scalar (see Figure 3.1). The process of partitioning at a node stops if (i) the LP at the node is infeasible, or (ii) the LP's optimal solu-

tion is integer feasible, or (iii) the LP's optimal objective function value is worse than an already known integer feasible solution. These three conditions are sometimes referred to as the rules for pruning a node. The algorithm terminates when there are no more "open nodes" to process, that is all nodes have been pruned. A branch-and-bound algorithm is completely described by fixing a rule for partitioning the feasible region at each node and a rule for selecting which open node should be solved and branched on next. If the choice of π is limited to being the canonical basis vectors $e_j = (0, \dots, 0, 1, 0, \dots, 0)$ (with the 1 in the j-th position), then we call such an algorithm a *simple* or *variable* branch-and-bound, and without such a restriction on π we call the algorithm a *general* branch-and-bound. See [2, 3] for more discussion on branch-and-bound and for general background on integer programming.

1.0.1 Bounds on the size of branch-and-bound trees

We will primarily be concerned with the size of branch-and-bound trees, which we define to be the number of nodes in the tree. This is the number of subproblems in the enumeration tree and, in most cases, the algorithm solves at most a linear number of linear programs at each node. For these reasons, the size of a branch-and-bound tree is arguably the standard measure of the algorithm's ability to solve a given problem. This is supported in experimental studies and in practice, where smaller trees correlate with faster total computation times and larger trees with slower times.

Upper bounds on the size of branch-and-bound trees and "positive" results. In 1983, Lenstra [4] showed that integer programs can be solved in polynomial time in fixed dimension. This algorithm can be viewed as a general branch-and-bound algorithm that uses tools from the geometry of numbers, in particular the lattice basis reduction algorithm [5], to decide on π for partitioning the feasible region. Pataki [6] proved that most random packing IPs (i.e., where A and b in (chapter IP) are non-negative) can be solved at the root-node using a partitioning

scheme similar to the one proposed by Lenstra [4]. It has been observed that using such general partitioning rules can result in significantly smaller trees than using a simple branch-and-bound for some instances [7, 8], but most commercial solvers use the latter. Beame et al. [9] recently studied how branch-and-bound can give good upper bounds for certain SAT formulas.

In chapter 2 we show that for certain classes of random integer programs the simple branch-and-bound tree has polynomial size (number of nodes), with good probability; see [10] for the full paper. See also [11] for nice extensions of this direction of results.

Lower bounds on the size of branch-and-bound trees and connections to the size of cutting-plane algorithms. Jeroslow [12] and Chvátal [13] present examples of integer programs where every *simple* branch-and-bound algorithm for solving them has an exponential-size tree. However, these instances can be solved with small (polynomial-size) *general* branch-and-bound trees; see Yang et al. [14] and Basu et al. [15]. Cook et al. [16] present a TSP instance that requires exponential-size branch-and-*cut* trees that uses simple branching (recall that branch-and-cut is branch-and-bound where one is allowed to add cuts to the intermediate LPs). Basu et al. [17] compare the performance of branch-and-bound with the performance of cutting-plane algorithms, providing instances where one outperforms the other and vice-versa. In another paper, Basu et al. [15] compare branch-and-bound with branch-and-cut, providing instances where branch-and-cut solves the instance in exponentially fewer nodes than branch-and-bound. They also present a result showing that the sparsity of the disjunctions can have a large impact on the size of the branch-and-bound tree required to solve a given problem. Beame et al. [9] asked as an open question whether there are superpolynomial lower bounds for general branch-and-bound algorithm. Dadush and Tiwari [18] settled this in the affirmative. In particular, they show that any general branch-and-bound tree that proves the integer infeasibility of the so-called cross-polytope in n-dimensions has at least $\frac{2^n}{n}$ leaf nodes.

In chapter 3 we demonstrate IP formulations of several combinatorial optimization problems

requiring exponential size general branch-and-bound trees. We also present the first "smoothed" exponential lower bound for general branch-and-bound; see [19] for the full paper.

Quantifying the impact of variable selection on the size of simple branch-and-bound trees. When considering a simple branch-and-bound tree, the partitioning scheme boils down to the choice of which variable to branch on. Much of the research in the area of branch-and-bound algorithms has focused on this topic – see for example [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. Most of the above work develops various intricate greedy rules for determining the branching variable. A popular concept is that of *pseudocost branching*: the value of pseudocost (variable with largest pseudocost gets branched on) keeps a history of the success (in terms of improving dual bound) of the variables on which branching has already been done. Many of the papers cited above differ in how pseudocost is initialized and updated during the course of the branch-and-bound tree. Other successful methods like *hybrid branching* and *reliability branching* [32] are combinations of pseudocost branching and *full strong-branching*, that we discuss next.

The focus of chapter 4 is *full strong-branching* [30], henceforth referred to as strong-branching for simplicity. Empirically it is well understood that strong-branching produces very small trees in comparison to other rules. However, to the best of our knowledge there is *no understanding of how good strong-branching is in absolute terms.* In chapter 4, we analyze strong branching, both theoretically and computationally; see [33] for the full paper.

CHAPTER 2

BRANCH-AND-BOUND SOLVES RANDOM BINARY IPS IN POLYTIME

2.1 Introduction

In 1983, Lenstra [4] showed that general integer programs can be solved in polynomial time in fixed dimension. This algorithm, which is essentially a branch-and-bound algorithm, uses tools from geometry of numbers, in particular the lattice basis reduction algorithm [5] to decide on π for partitioning the feasible region. Pataki et al. [6] proved that most random packing integer programs can be solved at the root-node using a partitioning scheme similar to the one proposed by Lenstra [4]. While there are some implementations of such general partitioning rules [7], all state-of-the-art solvers use the much simpler (and potentially significantly weaker and restrictive) variable branch-and-bound for solving binary IPs.

As mentioned above, all state-of-the-art MILP solvers use variable branching, which has proven itself to be very successful in practice [34]. Part of this success could be attributed to the fact that variable branching helps maintain the sparsity structure of the original LP relaxation, which can help in solving LPs in the branch-bound-tree faster (see [35, 36, 37, 38]). Additionally, while in the worst-case there can be exponentially many nodes in the branch-and-bound tree (see [13, 12, 39] for explicit examples for variable branching based branch-and-bound), a major reason for its success is that in practice the size of the tree can be quite small [40]. To the best of our knowledge there is no theoretical study of branch-and-bound algorithm using variable branching that attempts to explain its incredible success in practice.

In order to avoid worst-case lower bounds, a standard idea is to consider random instances. A famous example is the study of smoothed analysis for the simplex method [41]. In this paper, we provide what seems to be the first analysis of the branch-and-bound algorithm with variable

branching for a set of random instances. More precisely, we consider problems of the form

$$\begin{aligned} \max \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \{0,1\}^n. \end{aligned} \tag{IP}$$

By a **random instance** of IP we mean one where we draw the entries of the constraint matrix $A \in \mathbb{R}^{m \times n}$ ($m \ll n$) and the objective vector c uniformly from $[0,1]$ independently. We remark that the right-hand side is still fixed/deterministic. We parametrize it as $b_j = \beta_j \cdot n$, and we will consider the case where $\beta_j \in (0, \frac{1}{2})$ for $j \in \{1, \dots, m\}$. (The case when β_j is high is less interesting since the sum of n i.i.d random variables drawn from Uniform$[0,1]$ concentrates tightly around $\frac{1}{2}$ as $n \longrightarrow \infty$, and so this constraint will be satisfied even if all variables are set to 1.) We show that if the number of constraints m is fixed, then the branch-and-bound tree with variable branching has at most polynomial number of nodes with good probability. More precisely, we show the following result.

Theorem 1. *Consider a branch-and-bound algorithm using the following rules:*

- *Partitioning rule: Variable branching, where any fractional variable can be used to branch on.*
- *Node selection rule: Select a node with largest LP value as the next node to branch on.*

Consider $n \geq m + 1$ and a random instance of the problem IP where $b_j = \beta_j \cdot n$ and $\beta_j \in (0, 1/2)$ for $j \in \{1, \dots, n\}$. Then with probability at least $1 - \frac{1}{n} - 2^{-\alpha \bar{a}_2}$, the branch-and-bound algorithm applied to this random instance produces a tree with at most $n^{f(m;\beta)}$ nodes for all $\alpha \leq \min\{30\,m\,, \frac{\log n}{\bar{a}_2}\}$, where $\bar{a}_2$ is constant depending only on m and β and $f(m;\beta)$ is a function depending only on m and β.

We note that the node selection rule used here is called the *best-bound method* in the literature and often used in practice with minor modifications [31]. It is folklore that this node selection

rule is known to guarantee the smallest tree for any fixed partitioning rule. Also notice that Theorem 1 does not specify a rule for selecting a fractional variable to branch on and therefore even "adversarial" choices lead to a polynomial sized tree with good probability. This indicates that the tree is likely to be even smaller when one uses a "good" variable selection rule, such as *strong branching* [32]. Another reason for the size of the tree to be even smaller in practice is that Theorem 1 relies only on rule (iii) of pruning, i.e., pruning by bounds, to bound the size of the tree. However, pruning may also occur due to rules (i) (LP infeasibility) or (ii) (integer optimality), thus leading to a smaller tree size than predicted by Theorem 1.

Also notice that while IP is written in packing form (e.g., A is non-negative), our bounds work for every deterministic binary IP that is "well-behaved", as discussed in section 2.7. Together with the above observations, we believe Theorem 1 provides compelling indication of why branch-and-bound with variable branching works so well in practice.

Finally, we note that random (packing) instances have been considered in several previous studies, and it has been shown that they can be solved in polynomial time with high probability. As mentioned earlier, Pataki et al. [6] proves that random packing integer programs can be solved in polynomial-time; however it uses the very heavy machinery of lattice basis reduction, which is not often used in practice. Other papers considering random packing problems present algorithms that are custom-made enumeration-based schemes that are **not** equivalent to the general purpose branch-and-bound algorithm. In particular, Lueker [42] showed that the additive integrality gap for random one-row (i.e. $m = 1$) knapsack instances is $O\left(\frac{\log^2 n}{n}\right)$, and using this property Goldberg and Marchetti-Spaccamela [43] presented a polynomial-time enumeration algorithm for these instances. Beier and Vocking [44, 45] showed that the so-called knapsack core algorithm with suitable improvements using enumeration runs in polynomial-time with high probability. Finally, Dyer and Frieze [46] generalize the previous results on integrality gap and enumeration techniques to the random instances we consider here.

There has also been follow-up work to the results presented here. Borst et al. [11] show that,

when the data is drawn from a standard Gaussian (as opposed to the uniform distribution studied here), the integrality gap is sufficiently small that branch-and-bound solves problems of this type in $n^{\text{poly}(m)}$ nodes. Frieze [47] shows that any branch-and-bound tree solving the asymmetric travelling salesperson problem via the assignment problem relaxation has an exponential number of nodes.

The rest of the paper is organized as follows. section 2.2 presents notation, formalizes the set-up and presents some preliminary results needed. section 2.3 establishes a key result that the size of branch-and-bound tree can be bounded if one can bound all possible "near optimal" solutions for IP(b) for varying values of b. section 2.4 and section 2.5 build up machinery to prove that the number of "near optimal" solutions is bounded by a polynomial. section 2.6 completes the proof of Theorem 1. In section 2.7, we present an upper bound on the size of branch-and-bound tree for general IPs.

An earlier version of this paper is published in the conference SODA [10]. The current paper is a significant improvement over [10]. Although the main result (Theorem 1) is the same, we have significantly simplified the proofs and sharpened the key take-away result in section 2.7 that can be applied to obtain upper bounds on the size of branch-and-bound tree for general IP models: see Corollary 3. Comparing with [10], we improve this result by reducing the upper bound on size of branch-and-bound trees by a factor of $O(n^m)$. Also the analysis in section 2.3 and section 2.4 are revamped and simplified as compared the original paper [10].

2.2 Preliminaries

2.2.1 Branch-and-bound

Even though the general branch-and-bound algorithm was already described in the introduction, we describe it again here for maximization-type 0/1 IPs and using variable branching as partitioning rule and best-bound as node selection rule, for a clearer mental image. This is what we

will henceforth refer to as *the branch-and-bound* (BB) algorithm.

The algorithm constructs a tree $\mathcal{T}$ where each node has an associated LP; the LP relaxation of original integer program is the LP of the root node. In each iteration the algorithm:

1. (Node selection) Selects an unpruned leaf N with highest optimal LP value in the current tree $\mathcal{T}$, and obtains an optimal solution $\tilde{x}$ to this LP, if one exists.

2. (Pruning by integrality) If $\tilde{x}$ is integral, and hence feasible to the original IP, and has higher value than the current best such feasible solution, it sets $\tilde{x}$ as the current best feasible solution. The node N is marked as *pruned by integrality*.

3. (Pruning by infeasibility/bound) Else, if the LP is infeasible or its value is no better than the value of the current best feasible solution, the node N is marked as *pruned by infeasibility/bound*.

4. (Branching) Otherwise it selects a coordinate j where $\tilde{x}$ is fractional and adds two children to N in the tree: on one of them it adds the constraint $x_j = 0$ to the LP of N, and on the other it adds the constraint $x_j = 1$ instead.

Note that we do not assume that an optimal IP solution, or even the optimal IP value, is given in the beginning of the procedure: the algorithm starts with no current best feasible solution, which are only found in step *Pruning by integrality*. One simple but important property of this best-bound node selection rule is the following.

Lemma 1 (best-bound node selection)**.** *The execution of branch-and-bound with best-bound node selection rule never branches on a node whose LP value is worse than the optimal value of the original IP.*

Proof. Let IP^* denote the optimal value of the original IP. Notice that throughout the execution, either:

1. The current best feasible solution has value IP^* (i.e. an optimal integer solution has been found)

2. The LP of some unpruned leaf contains an optimal integer solution, and so this LP value is at least as good as IP^*.

This means that in every iteration, the algorithm cannot *select* and *branch* on a leaf with LP value strictly less than IP^*: in Case 2 such leaf is not selected (due to the best-bound rule), and in Case 1 such leaf is pruned by infeasibility/bound (and hence not branched on) if selected. □

2.2.2 (Random) Packing problems

We will use the following standard observation on the number of fractional coordinates in an optimal solution of the LP relaxation of every instance of IP (see for example Section 17.2 of [48]).

Lemma 2. *Consider an instance of IP. Then every LP in the BB tree for this instance has an optimal solution with at most m fractional coordinates.*

Proof. Notice that the LP's in the BB tree for this instance have the form

$$\begin{aligned} \max \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax \leq b \\ & x_j = 0, \qquad \forall j \in J_0 \\ & x_j = 1, \qquad \forall j \in J_1 \\ & x_j \in [0,1], \quad \forall j \notin J_0 \cup J_1, \end{aligned} \tag{2.1}$$

for disjoint subsets $J_0, J_1 \subseteq [n]$ (i.e. the fixings of variables due to the branchings up to this node in the tree).

The feasible region P in (Equation 2.1) is bounded, so there is an optimal solution x^* of the LP that is a vertex of P. This implies that at least n of the constraints of the LP are satisfied by x^* at equality. Since there are m constraints in $Ax \le b$, at least $n - m$ of these equalities are of the form $x_j^* = v_j$ (for some $v_j \in \{0, 1\}$) and so at most m x_j^*'s can be fractional. □

Recall that the *integrality gap* of IP is IPGAP := OPT(LP) – OPT(IP), namely the optimal value of its LP relaxation LP := $\max\{\langle c, x\rangle : Ax \le b,\ x \in [0,1]^n\}$ minus the optimal IP value. Dyer and Frieze proved the following property that will be crucial for our results: for a random instance of IP (defined right after the definition of IP), the integrality gap is $O(\frac{\log^2 n}{n})$ with good probability (setting α to be a sufficiently large constant below).

Theorem 2 (Theorem 1 of [46]). *Consider a vector $\beta \in (0, \frac{1}{2})^m$ and let $b \in \mathbb{R}^m$ be given by $b_j = \beta_j \cdot n$ for $j \in \{1, \dots, m\}$. Then there are scalars $a_1, a_2 \ge 1$ depending only on m and $\min_j \beta_j$ such that the following holds: for a random instance $\mathcal{I}$ of IP*

$$\Pr\left(IPGAP(\mathcal{I}) \ge \alpha a_1 \frac{\log^2 n}{n}\right) \le 2^{-\alpha a_2}$$

for all $\alpha \le \frac{3 \log n}{a_2}$.

2.2.3 Notation

We use the shorthands $\binom{n}{\le k} := \sum_{i=0}^{k} \binom{n}{i}$ and $[n] := \{1, 2, \dots, n\}$. We also use $\binom{[n]}{\le k}$ to denote the family of all subsets of $[n]$ of size at most k. We use A^j to denote the jth column of the matrix A. For any vector $x \in \mathbb{R}^n$, we use x^+ to denote the vector that satisfies $x_i^+ = \max(0, x_i)$ for all $i \in [n]$.

2.3 Branch-and-bound and good integer solutions

In this section we connect the size of the BB tree for any instance of IP and the number of its near-optimal solutions.

To make this precise, first let LP be the LP relaxation of IP

$$\begin{aligned} \max \quad & \langle c, x \rangle \\ \text{s.t.} \quad & Ax \le b \\ & x \in [0,1]^n, \end{aligned} \tag{LP}$$

and its Lagrangian relaxation

$$\min_{\lambda \ge 0} \max_{x \in [0,1]^n} \langle c, x \rangle - \lambda^\top (Ax - b) \quad \equiv \quad \min_{\lambda \ge 0} \max_{x \in [0,1]^n} \sum_{j \in [n]} (c_j - \langle \lambda, A^j \rangle) x_j + \langle b, \lambda \rangle.$$

Let (x^*, λ^*) be a saddle point of this Lagrangian, namely

$$\begin{aligned} x^* &= \operatorname{argmax} \left\{ \sum_{j \in [n]} (c_j - \langle \lambda^*, A^j \rangle) x_j + \langle b, \lambda^* \rangle \ : \ x \in [0,1]^n \right\} \\ \lambda^* &= \operatorname{argmin} \left\{ \sum_{j \in [n]} (c_j - \langle \lambda, A^j \rangle) x_j^* + \langle b, \lambda \rangle \ : \ \lambda \ge 0 \right\}. \end{aligned}$$

Recall that this saddle point solution has the same value as OPT(LP) [49]. Also notice that $c_j - \langle \lambda^*, A^j \rangle$ can be thought as the reduced cost of variable x_j and by optimality of x^*

$$x_j^* = \begin{cases} 1 & \text{if } c_j - \langle \lambda^*, A^j \rangle > 0 \\ 0 & \text{if } c_j - \langle \lambda^*, A^j \rangle < 0. \end{cases} \tag{2.2}$$

For any point $x \in [0,1]^n$, we define the quantity

$$\text{pareto}(x) = \sum_{j \in [n]} (c_j - \langle \lambda^*, A^j \rangle) \cdot (x_j^* - x_j),$$

which is then the difference in value between x and x^* given by the reduced costs. Given this, we

say that a 0/1 point x is *good* if its pareto is at most IPGap, and we use G to denote the set of all good points, namely

$$G := \{x \in \{0,1\}^n : \text{pareto}(x) \leq \text{IPGap}\}. \tag{2.3}$$

The following is the main result of this section, which states that we can bound the size of the branch-and-bound tree based on the number of these good points.

Theorem 3. *Consider the branch-and-bound algorithm with best-bound node selection rule for solving IP. Then its final tree has at most* $2|G|n + 1$ *nodes.*

The remainder of this section is dedicated to proving this result. So let $\mathcal{T}$ denote the final BB tree constructed by the algorithm, and let $\mathcal{N}$ denote the set of its internal nodes. The first observation is that all LP solutions seen throughout the BB tree have small pareto.

Lemma 3. *Let* N *be a node that is branched on in the BB tree* $\mathcal{T}$*, i.e.* $N \in \mathcal{N}$*, and let* x^N *be an optimal solution for the LP of this node with at most* m *fractional coordinates (via Lemma 2). Then,*

$$\textit{pareto}(x^N) \leq \textit{IPGap}.$$

Proof. First notice that for every feasible solution x to LP we have

$$\text{OPT(LP)} \geq \langle c, x\rangle + \text{pareto}(x),$$

since the equality of optimal value of LP and its Lagrangian and then feasibility of x give

$$\text{OPT(LP)} - \langle c, x\rangle = \sum_{j\in[n]} (c_j - \langle \lambda^*, A^j\rangle)x_j^* + \langle b, \lambda^*\rangle - \langle c, x\rangle \geq \sum_{j\in[n]} (c_j - \langle \lambda^*, A^j\rangle)x_j^* + \langle Ax, \lambda^*\rangle - \langle c, x\rangle,$$

and the claim follows by noticing that the right-hand side is precisely pareto(x). Moreover, by

the best-bound node selection rule (Lemma 1) we have that $\langle c, x^N \rangle \geq \text{OPT(LP)} - \text{IPG}_{\text{AP}}$. Putting these observations together proves the result. □

We will now present the proof of Theorem 3. This follows from the construction of an association between BB nodes and good points. In particular, we will show that the association is at least one-to-one and at most n-to-one.

Proof of Theorem 3. We will demonstrate an association $r : \mathcal{N} \longrightarrow G$ that satisfies the following properties:

1. associates any internal node $N \in \mathcal{N}$ to at least one good solution $x \in G$; i.e. for all $N \in \mathcal{N}$ it holds that $|r(N)| \geq 1$
2. associates at most n internal nodes to any one good solution; i.e. for any $x \in G$, it holds that $|r^{-1}(x)| \leq n$

Let $N \in \mathcal{N}$ be any internal node of $\mathcal{T}$ and x^N be its optimal LP solution. Let $J \subset [n]$ denote the set of indices where x^N is 0 or 1. Define $r : \mathcal{N} \longrightarrow G$ to associate N with any

$$x' \in \arg\min\{ \text{pareto}(x) : x \in \{0,1\}^n,\ x_j = x_j^N\ \forall j \in J\}.$$

We will now show that r satisfies property 1. Noting that the objective function pareto$(\cdot)$ is affine (λ^* is fixed) and that x^N is in the convex hull of the 0, 1 points $\{x \in \{0,1\}^n : x_j = x_j^N\ \forall j \in J\}$, it must hold that

$$\text{pareto}(x') \leq \text{pareto}(x^N) \leq \text{IPG}_{\text{AP}},$$

where the last inequality follows from Lemma 3. Then we see that

$$x' \in \{x \in \{0,1\}^n : \text{pareto}(x) \leq \text{IPG}_{\text{AP}}\}.$$

This concludes the proof of property 1.

We will now show that r satisfies property 2. Let $x \in G$. Let N_1, N_2 be two internal nodes of $\mathcal{T}$, with optimal LP solutions $x^{(1)}$ and $x^{(2)}$ respectively, such that $r(x^{(1)}) = r(x^{(2)}) = x'$. In the following paragraph we show that either N_1 is a descendant of N_2, or vice versa. Then, for any $x \in G$ all nodes associated to x (i.e. all $N \in r^{-1}(x)$) must lie in the same path from the root in $\mathcal{T}$ (i.e. must all be descendants of one another). So since a path from the root in $\mathcal{T}$ can have length at most n, we have the desired conclusion.

Suppose, for the sake of contradiction, N_1 and N_2 are not one a descendant of the other in the BB tree $\mathcal{T}$. Let $N \neq N_1, N_2$ be their lowest common ancestor in the tree $\mathcal{T}$, and let $f \in \{1, ..., n\}$ be the index where node N was branched on. Since nodes N_1 and N_2 are on different subtrees under N, without loss of generality assume that N_1 is in the subtree with $x_f = 0$ and N_2 is in the subtree with $x_f = 1$. Then since $x_f^{(1)} = 0$ and $r(x^{(1)}) = x'$, it must be that $x'_f = 0$. However, since $x_f^{(2)} = 1$, we have $x_f^{(2)} \neq x'_f$, so that by definition of $r(\cdot)$ we have $x^{(2)} \notin r^{-1}(x')$, getting the desired contradiction. This concludes the proof of property 2.

The existence of such an association implies that the number of internal nodes of $\mathcal{T}$ (i.e. $|\mathcal{N}|$) is upper bounded by $|G| \cdot n$. Since the total number of nodes in a binary tree is at most twice the number of its internal nodes plus 1, we see that $\mathcal{T}$ has at most $2|G|n + 1$ nodes, giving the desired bound. □

We spend the remainder of this paper obtaining an upper bound of the form $n^{O(m)}$ for the number of good solutions (Equation 2.3), which will then prove Theorem 1.

2.4 Value of solutions and geometry of items

Going back to (Equation 2.2), we can see the saddle point x^* as being obtained by a *linear classification* of columns induced by λ^*, namely x_j^* is set to 0/1 depending on whether

$$c_j - \langle \lambda^*, A^j \rangle \gtrless 0 \quad \equiv \quad \langle (c_j, A^j), (1, -\lambda^*) \rangle \gtrless 0,$$

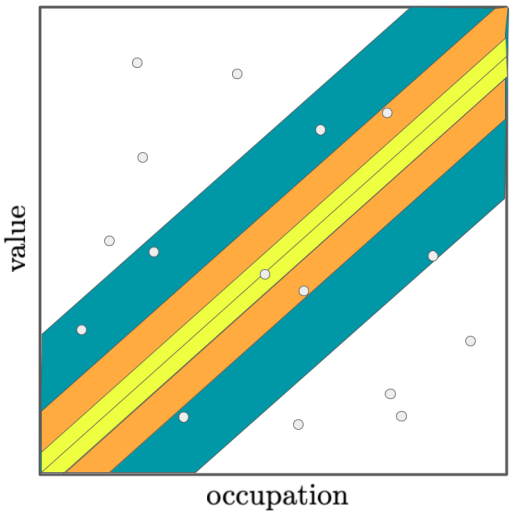

Figure 2.1: Each dot represents an item/column (c_j, A^j) of IP. The grey line coming from the origin is H. The optimal solution has $x^*_j = 1$ for all items j above this line, $x^*_j = 0$ for all items j below this line. Each J_ℓ is the set of items that lie in one of the colored regions; J_{rem} are the items that lie on the inner most region (yellow).

that is, depending where the column (c_j, A^j) lands relative to the hyperplane in $\mathbb{R}^{m+1}$

$$H := \{y \in \mathbb{R}^{m+1} : \langle (1, -\lambda^*), y \rangle = 0\};$$

see Figure Figure 2.1.

The next lemma says that the pareto(x) of a 0/1 point x increases the more it disagrees with x^*, and the penalty for the disagreement on item j depends on the distance of the column (c_j, A^j) to the hyperplane H. We use $d(z, U)$ to denote the Euclidean distance between a point z and a set U, and $\mathbf{1}(P)$ to denote the 0/1 indicator of a predicate P. This generalizes Lemma 3.1 of [43].

Lemma 4. *Consider any $x \in \{0,1\}^n$. Then*

$$\textit{pareto}(x) \geq \sum_{j \in [n]} d\Big((c_j, A^j),\ H\Big) \cdot \mathbf{1}(x_j \neq x^*_j).$$

Proof. Recalling the definition of pareto(·), we can rewrite it as follows:

$$\begin{aligned}\text{pareto}(x) &= \sum_{j\in[n]} (c_j - \langle \lambda^*, A^j \rangle)(x_j^* - x_j) \\ &= \sum_{j\,:\,x_j^*=0} (c_j - \langle \lambda^*, A^j \rangle)(x_j^* - x_j) + \sum_{j\,:\,x_j^*=1} (c_j - \langle \lambda^*, A^j \rangle)(x_j^* - x_j). \end{aligned} \tag{2.4}$$

To analyze the right-hand side of this equation, consider a term in the first sum. Since $x_j^* = 0$, the term is non-zero exactly when the 0/1 point x takes value $x_j = 1$. Moreover, since $x_j^* = 0$ implies that $c_j - \langle \lambda^*, A^j \rangle$ is negative, this means that the terms in the first sum of (Equation 2.4) equal $|c_j - \langle \lambda^*, A^j \rangle| \cdot \mathbf{1}(x_j \neq x_j^*)$. A similar analysis shows that the terms in the second sum of (Equation 2.4) also equal $|c_j - \langle \lambda^*, A^j \rangle| \cdot \mathbf{1}(x_j \neq x_j^*)$. Together these give

$$\text{pareto}(x) = \sum_{j\in[n]} |c_j - \langle \lambda^*, A^j \rangle| \cdot \mathbf{1}(x_j \neq x_j^*). \tag{2.5}$$

Finally, notice that

$$|c_j - \langle \lambda^*, A^j \rangle| \geq d\Big((c_j, A^j),\ H\Big),$$

because, since the point $(\langle \lambda^*, A^j \rangle, A^j)$ belongs to the hyperplane H, we have

$$d\Big((c_j, A^j),\ H\Big) \leq \|(c_j, A^j) - (\langle \lambda^*, A^j \rangle, A^j)\|_2 = |c_j - \langle \lambda^*, A^j \rangle|.$$

Plugging this bound in (Equation 2.5) concludes the proof of the lemma. □

Thus, a good solution, i.e., one with $\text{pareto}(x) \leq \text{IPGap}$, must disagree with x^* on few items j whose columns (c_j, A^j) are "far" from the hyperplane H. To make this quantitative we bucket these distances in powers of 2, so for $\ell \geq 1$ define the set of items

$$J_\ell := \left\{ j \,:\, d\Big((c_j, A^j),\ H\Big) \text{ is in the interval } \left(\tfrac{\log n}{n} 2^\ell,\ \tfrac{\log n}{n} 2^{\ell+1} \right] \right\},$$

and define $J_{\text{rem}} = [n] \setminus \bigcup_{\ell \geq 1} J_\ell$ as the remaining items (see Figure Figure 2.1). Since every item in J_ℓ has distance at least $\frac{\log n}{n} 2^\ell$ from H, we directly have the following.

Corollary 1. *If $x \in \{0,1\}^n$ is a good point (i.e., pareto$(x) \leq$ IPGAP), then for every $\ell \geq 1$ the number of coordinates $j \in J_\ell$ such that $x_j \neq x^*_j$ is at most $\frac{C}{2^\ell}$, where $C := \frac{n}{\log n} \cdot$ IPGAP.*

Then, an easy counting argument gives an upper bound on the total number of good points G.

Lemma 5. *We have the following upper bound:*

$$|G| \leq 2^{|J_{rem}|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell|}{\leq C/2^\ell},$$

where $C := \frac{n}{\log n} \cdot$ IPGAP.[1]

Proof. Notice that every solution in G can be thought of as being created by starting with the vector x^* and then changing some of its coordinates, and because of Corollary 1 we:

- Cannot change the value of x^* in any coordinate j in a J_ℓ with $\ell > \log C$
- Can only flip the value of x^* in at most $\frac{C}{2^\ell}$ of the coordinates $j \in J_\ell$ for each $\ell = 1, \dots, \log C$ (recall that x^* is 0/1 in all such coordinates)
- Set a new arbitrary 0/1 value for (in principle all) coordinates in J_{rem}.

Since there are at most $2^{|J_{\text{rem}}|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell|}{\leq C/2^\ell}$ options in this process, we have the desired upper bound. □

Notice that, ignoring the term $2^{|J_{\text{rem}}|}$, this already gives with good probability a quasi-polynomial bound $|G| \lesssim O(n)^{\text{polylog}(n)}$ for random instances of IP: the upper bound on the integrality gap from

[1] We assume throughout that the $C/2^\ell$'s are integral to simplify the notation, but it can be easily checked that using $\lceil C/2^\ell \rceil$ instead does not change the results.

Theorem 2 (with α set as a large enough constant) gives that with good probability $C \le \log n$ and so we have $\log\log n$ binomial terms, each at most $\binom{n}{\log n} \le n^{\log n}$ (since $|J_\ell| \le n$).

In order to obtain the desired polynomial bound $|G| \le n^{O(m)}$, we need a better control on $|J_\ell|$, namely the number of points at a distance from the hyperplane H.

2.5 Number of items at a distance from the hyperplane

To control the size of the sets J_ℓ we need to consider a random instance of IP and use the fact that the columns (c_j, A^j) are uniformly distributed in $[0,1]^{m+1}$. Recalling the definition of J_ℓ, we see that an item j belongs to this set only if the column (c_j, A^j) lies on the $(m+1)$-dim *slab* of width $\frac{\log n}{n} 2^{\ell+1}$ around H:

$$\left\{ y \in \mathbb{R}^{m+1} \,:\, d(y,H) \le \tfrac{\log n}{n} 2^{\ell+1} \right\}.$$

It can be shown that the volume of this slab intersected with $[0,1]^{m+1}$ is at most proportional to its width, so as long as H is independent of the columns, the probability that a random column (c_j, A^j) lies in this slab is $\approx \frac{\log n}{n} 2^{\ell+1}$. Thus, we would expect that at most $\approx (\log n)\, 2^{\ell+1}$ columns lie in this slab, which gives a much improved upper bound on the (expected) size of J_ℓ (as indicated above, think $\ell \le \log\log n$). Moreover, using independence of the columns (c_j, A^j), standard concentration inequalities show that for *each* such slab with good probability the number of columns that land in it is within a multiplicative factor from this expectation.

Unfortunately, the hyperplane H that we are concerned with, whose normal $\frac{(1,-\lambda^*)}{\|(1,-\lambda^*)\|_2}$ is determined by the data, is *not* independent of the columns. So we will show a much stronger *uniform* bound that shows that with good probability the above phenomenon holds *simultaneously* for all slabs around *all* hyperplanes, which then shows that it holds for H. We abstract this situation and prove such uniform bound. We use $\mathbb{S}^{k-1}$ to denote the unit sphere in $\mathbb{R}^k$.

Theorem 4 (Uniform bound for slabs)**.** *For $u \in \mathbb{S}^{k-1}$ and $w \ge 0$, define the slab of normal u and*

width w as

$$S_{u,w} := \left\{ y \in \mathbb{R}^k \,:\, \langle u, y \rangle \in [-w, w] \right\}.$$

Let $Y^1, \ldots, Y^n$ be independent random vectors uniformly distributed in the cube $[0,1]^k$, for $n \geq k$. Then with probability at least $1 - \frac{1}{n}$, we have that for all $u \in \mathbb{S}^{k-1}$ and $w \geq \frac{\log n}{n}$ at most $60nwk$ of the Y^j's belong to $S_{u,w}$.

While very general bounds of this type are available (for example, appealing to the low VC-dimension of the family of slabs), we could not find in the literature a good enough such *multiplicative* bound (i.e., relative to the expectation $\approx nw$). The proof of Theorem 4 instead relies on an ε-net type argument.

This result directly gives the desired upper bound on the size of the sets J_{rem} and J_ℓ.

Corollary 2. *With probability at least $1 - \frac{1}{n}$ we have simultaneously*

$$|J_{rem}| \leq 120(m+1)\log n$$

$$|J_\ell| \leq 60(m+1)2^{\ell+1}\log n, \quad \forall \ell \in [\log n - 1]$$

2.5.1 Proof of Theorem 4

Let $\mathbb{S}'$ be a minimal ε-net, for $\varepsilon := \frac{\log n}{n\sqrt{k}}$, of the sphere $\mathbb{S}^{k-1}$, namely for each $u \in \mathbb{S}^{k-1}$ there is $u' \in \mathbb{S}'$ such that $\|u' - u\|_2 \leq \varepsilon$. It is well-known that there is such a net of size at most $(\frac{3}{\varepsilon})^k$ (Corollary 4.2.13 [50]). Also define the discretized set of widths $\mathbb{W}' := \{\frac{\log n}{n}, \frac{2\log n}{n}, \ldots, \sqrt{k} + \frac{\log n}{n}\}$ so that $|\mathbb{W}'| = \frac{n\sqrt{k}}{\log n} + 1$.

We start by focusing on a single slab in the net, observing that these slabs are determined independently of the columns. Since the vector Y^j is uniformly distributed in the cube $[0,1]^k$, the probability that it belongs to a set $U \subseteq [0,1]^k$ equals the volume $\text{vol}(U)$. Using this and an upper bound on the volume of a slab (intersected with the cube), we get that the probability that Y^j

lands on a slab is at most proportional to the slab's width.

Lemma 6. *For every slab $S_{u',w'}$, with $u' \in \mathbb{S}'$ and $w' \in \mathbb{W}'$, we have* $\Pr(Y^j \in S_{u',w'}) \le 2\sqrt{2}w'$.

Proof. Since these slabs are determined independently of the columns, it is equivalent to show that the volume $\mathrm{vol}(S_{u',w'} \cap [0,1]^k)$ is at most $2\sqrt{2}w'$. Let $\mathrm{slice}(h) := \{y \in [0,1]^k : \langle y, u\rangle = h\}$ be the slice of the cube with normal u' at height h. It is known that every slice of the cube has $(k-1)$-dim volume vol_{k-1} at most $\sqrt{2}$ [51], and since $S_{u',w'} \cap [0,1]^k = \bigcup_{h\in[-w',w']} \mathrm{slice}(h)$, by integrating we get

$$\mathrm{vol}(S_{u',w'} \cap [0,1]^k) = \int_{-w'}^{w'} \mathrm{vol}_{k-1}(\mathrm{slice}(h))\,\mathrm{d}h \;\le\; 2\sqrt{2}w'$$

as desired. □

Let $N_{u,w}$ be the number of vectors Y^j that land in the slab $S_{u,w}$. From the previous lemma we have $\mathbb{E}N_{u',w'} \le 2\sqrt{2}w'n$. To show that $N_{u',w'}$ is concentrated around its expectation we need Bernstein's Inequality; the following convenient form is a consequence of Appendix A.2 of [52] and the fact $\mathrm{Var}(\sum_j Z_j) = \sum_j \mathrm{Var}(Z_j) \le \sum_j \mathbb{E}Z_j^2 \le \mathbb{E}\sum_j Z_j$ for independent random variables Z_j in $[0,1]$.

Lemma 7. *Let $Z_1, \dots, Z_n$ be independent random variables in $[0,1]$. Then for all $t \ge 0$,*

$$\Pr\left(\sum_j Z_j \ge \mathbb{E}\sum_j Z_j + t\right) \le \exp\left(-\min\left\{\frac{t^2}{4\mathbb{E}\sum_j Z_j},\ \frac{3t}{4}\right\}\right).$$

Lemma 8. *For each $u' \in \mathbb{S}'$ and width $w' \in \mathbb{W}'$, we have*

$$\Pr(N_{u',w'} \ge 20w'nk) \le n^{-4k}.$$

Proof. Let $\mu := \mathbb{E}N_{u',w'}$. Notice that $N_{u',w'}$ is the sum of the independent random variables that indicate for each j whether $Y^j \in S_{u',w'}$. Then applying the previous lemma with $t = 4\mu + \frac{16}{3}k\ln n$

we get

$$\Pr\left(N_{u',w'} \;\geq\; 5\mu + \frac{16}{3}k\ln n\right) \;\leq\; \exp\left(-\min\left\{\frac{16}{3}k\ln n\,,\;\frac{3\cdot(16/3)k\ln n}{4}\right\}\right) \;\leq\; n^{-4k}, \qquad (2.6)$$

where in the first inequality we used $t^2 \geq (4\mu)\cdot(\frac{16}{3}k\ln n)$. From Lemma 6 we get $\mu \leq 2\sqrt{2}nw'$, and further using the assumption $w' \geq \frac{\log n}{n}$ (implied by $w' \in \mathbb{W}'$) we see that

$$20w'nk \;\geq\; \left(10\sqrt{2}+\frac{16}{3}\right)w'nk \;\geq\; 5\mu + \frac{16}{3}k\ln n,$$

and hence inequality (Equation 2.6) upper bounds $\Pr(N_{u',w'} \geq 20w'nk)$. This concludes the proof.

□

To prove the theorem we need to show that with high probability we simultaneously have $N_{u,w} \leq 60nwk$ for all $u \in \mathbb{S}^{k-1}$ and $w \geq \frac{\log n}{n}$. We associate each slab $S_{u,w}$ (for $u \in \mathbb{S}^{k-1}$ and $w \in [0, \sqrt{k}]$) to a "discretized slab" $S_{u',w'}$ by taking u' as a vector in the net $\mathbb{S}'$ so that $\|u' - u\|_2 \leq \varepsilon$ and taking $w' \in \mathbb{W}'$ so that $w' \in [w + \frac{\log n}{n}, w + \frac{2\log n}{n}]$.

Lemma 9. *This association has the following properties: for every $u \in \mathbb{S}^{k-1}$ and $w \in [0, \sqrt{k}]$*

1. *The intersected slab $S_{u,w} \cap [0,1]^k$ is contained in the associated intersected slab $S_{u',w'} \cap [0,1]^k$. In particular, in every scenario $N_{u,w} \leq N_{u',w'}$*
2. *If the width satisfies $w \geq \frac{\log n}{n}$, then $w' \leq 3w$.*

Proof. The second property is immediate, so we only prove the first one. Take a point $y \in S_{u,w} \cap [0,1]^k$. By definition we have $\langle y, u\rangle \in [-w, w]$, and also

$$|\langle y, u'\rangle - \langle y, u\rangle| = |\langle y, u' - u\rangle| \leq \|y\|_2\,\|u' - u\|_2 \leq \sqrt{k}\varepsilon = \frac{\log n}{n},$$

where the first inequality is Cauchy-Schwarz, and the second uses the fact that every vector in $[0,1]^k$ has Euclidean norm at most $\sqrt{k}$. Therefore $\langle y, u'\rangle \in [-(w + \frac{\log n}{n}), w + \frac{\log n}{n}]$, and since the

associated width satisfies $w' \geq w + \frac{\log n}{n}$ we see that y belongs to $S_{u',w'} \cap [0,1]^k$. This concludes the proof. □

Proof of Theorem 4. We need to show

$$\Pr\left[\bigvee_{u \in \mathbb{S}^{k-1}, w \geq \frac{\log n}{n}} (N_{u,w} > 60wnk)\right] \leq \frac{1}{n}. \tag{2.7}$$

Since there are only n Y^j's, we always have $N_{u,w} \leq n$ and hence it suffices to consider $w \in [\frac{\log n}{n}, \frac{1}{60k}]$, in which case we can use the inequalities $N_{u,w} \leq N_{u',w'}$ and $w' \leq 3w$ from the previous lemma; in particular, the event $(N_{u,w} > 60wnk)$ implies the event $(N_{u',w'} > 20w'nk)$. Thus, using Lemma 8

$$\begin{aligned}
\text{LHS of (Equation 2.7)} &\leq \Pr\left[\bigvee_{u' \in \mathbb{S}', w' \in \mathbb{W}'} (N_{u',w'} > 20w'nk)\right] \\
&\leq \sum_{u' \in \mathbb{S}', w' \in \mathbb{W}'} \Pr(N_{u',w'} > 20w'nk) \\
&\leq |\mathbb{S}'|\,|\mathbb{W}'|\, n^{-4k} \\
&\leq \left(\frac{3n\sqrt{k}}{\log n}\right)^{k+1} n^{-4k} \leq \frac{1}{n},
\end{aligned}$$

where the last inequality uses the assumption $n \geq k$. This concludes the proof. □

2.6 Proof of Theorem 1

We can finally conclude the proof of Theorem 1. To simplify the notation we use $O(\text{val})$ to denote $\text{cst} \cdot \text{val}$ for some constant cst. Because of Theorem 3, we show that for a random instance $\boldsymbol{\mathcal{I}}$ of IP, with probability at least $1 - \frac{1}{n} - 2^{-\alpha a_2}$ the number of good points G relative to this instance is at most $n^{f(m;\beta)}$.

For that, let E be the event where all of the following hold:

1. Theorem 2 holds for $\mathcal{I}$, namely $\text{IPGAP}(\mathcal{I})$ is at most $\frac{\alpha a_1 \log^2 n}{n}$,

2. The bound of Corollary 2 on the sizes of J_{rem} and J_ℓ for all $\ell \in [\log n - 1]$, determined by the arrangement of random vectors (c_j, A^j)'s that comprise the columns of $\mathcal{I}$.

By taking a union bound over these results we see that E holds with probability at least $1 - \frac{1}{n} - 2^{-\alpha a_2}$.

From the first item in the definition of E, under E we have that $C := \frac{n}{\log n} \cdot \text{IPGAP}(\mathcal{I})$ is at most $\alpha a_1 \log n$. Therefore, using the standard estimate $\binom{a}{\leq b} \leq (4a/b)^b$ that holds for $a \geq 4b$, we get that, under E:

$$\begin{aligned}
\prod_{\ell=\frac{\log a_1}{2}}^{\log C} \binom{|J_\ell|}{\leq C/2^\ell} &\leq \prod_{\ell=1}^{\log C} \left(\frac{O(m)\, 2^{2\ell} \log n}{\alpha a_1 \log n} \right)^{C/2^\ell} \\
&= \prod_{\ell=1}^{\log C} \left(\frac{O(m)\, 2^{2\ell}}{\alpha a_1} \right)^{C/2^\ell} \\
&\leq \left(\frac{O(m)}{\alpha a_1} \right)^{C \sum_{\ell \geq 1} \frac{1}{2^\ell}} \cdot 2^{2C \sum_{\ell \geq 1} \frac{\ell}{2^\ell}} \\
&\leq \left(\frac{O(m)}{\alpha a_1} \right)^{O(C)},
\end{aligned}$$

where we started the product from $\ell = \frac{\log a_1}{2}$ to ensure we could apply the binomial estimate given only the assumption $\alpha \leq 30m$; for the lower terms we can use the crude upper bound

$$\prod_{\ell < \frac{\log a_1}{2}} \binom{|J_\ell|}{\leq C/2^\ell} \leq \prod_{\ell < \frac{\log a_1}{2}} 2^{|J_\ell|} \leq 2^{O(m a_1 \log a_1 \log n)} = n^{O(m a_1 \log a_1)}.$$

Plugging these bounds on Lemma 5, we get that under E the number of good points G relative to the instance $\mathcal{I}$ is upper bounded as

$$|G| \leq 2^{O(m \log n)} \cdot n^{O(m a_1 \log a_1)} \cdot \left(\frac{O(m)}{\alpha a_1} \right)^{O(\alpha a_1 \log n)} \leq n^{O(m a_1 \log a_1 + \alpha a_1 \log m)}.$$

Finally, plugging this bound on Theorem 3 we get that under E the branch-and-bound tree for

the instance $\mathcal{I}$ has at most

$$2\,n \cdot\ n^{O(ma_1 \log a_1 + \alpha a_1 \log m)} + 1\ \leq\ n^{O(ma_1 \log a_1 + \alpha a_1 \log m)}$$

nodes. Recalling from Theorem 2 that a_1 is a constant depending only on m and β, this concludes the proof of Theorem 1.

2.7 Final remarks

We note that most of the above arguments hold not only for random problems but actually for *every* 0/1 IP. In particular, Theorem 3 and Lemma 5 hold in such generality, which combined give the following.

Corollary 3. *Consider any instance of IP with arbitrary $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then the tree of the best-bound branch-and-bound algorithm applied to this instance has at most*

$$2\left(2^{|J_{rem}|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell|}{\leq\ C/2^\ell}\right) n + 1$$

nodes, where $C := \frac{n}{\log n} \cdot IPG_{AP}$.

This shows that the effectiveness of branch-and-bound actually hold for every "well-behaved" 0/1 IP (or distributions that generate such IPs with good probability), where "well-behaved" means that the integrality gap must be small and there cannot be too many columns (c_j, A^j) concentrated around a hyperplane (in order to control the terms $|J_\ell|$).

CHAPTER 3

LOWER BOUNDS ON THE SIZE OF GENERAL BRANCH-AND-BOUND TREES

3.1 Introduction

Here we take interest in the size of general branch-and-bound trees. This is because instances requiring an exponential number of nodes to solve using general branch-and-bound are likely to also be challenging for MILP solvers. We hope that this study can provide some intuition on when and why solvers struggle with a MILP instance and how to formulate heuristics to combat these bottlenecks.

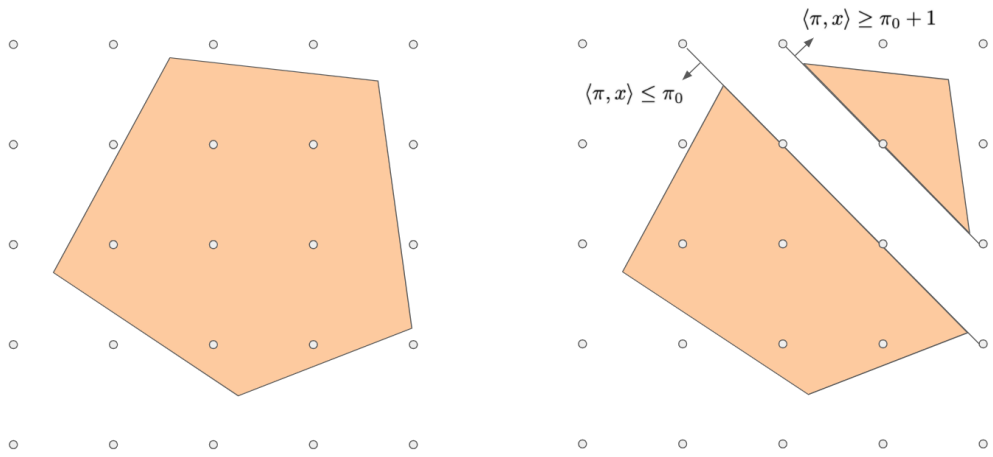

Figure 3.1: The **left** picture is the initial polytope and the **right** picture demonstrates the two subproblems as a result of branching on the disjunction $\langle \pi, x \rangle \leq \pi_0 \vee \langle \pi, x \rangle \geq \pi_0 + 1$.

Concurrent to the development of our work, Fleming et al. [53] showed a fascinating relationship between general branch-and-bound proofs and cutting-plane proofs using Chvátal-Gomory (CG) cutting-planes:

Theorem 5 (Theorem 3.7 from [53]). *Let $P \subseteq [0,1]^n$ be an integer-infeasible polytope and suppose*

there is a general branch-and-bound proof of infeasibility of size s and with maximum coefficient c. Then there is a CG proof of infeasibility of size at most

$$s(cn)^{\log s}.$$

The following simple corollary allows one to infer exponential lower bounds for branch-and-bound trees for polytopes for which we have exponential lower bounds for CG proofs.

Corollary 4. *Let $P \subseteq [0,1]^n$ be an integer-infeasible polytope such that any CG proof of integer-infeasibility of P (see Definition 1) has length at least L. Then any general branch-and-bound proof of integer-infeasibility of P with maximum coefficient c has size at least*

$$L^{\frac{1}{1+\log(cn)}}.$$

The above result makes progress in answering questions raised in Basu et al. [17] related to the comparison between the size of general branch-and-bound trees and the size of CG proofs. Moreover, Pudlak [54] and Dash [55] provide exponential lower bounds for CG proofs for the "clique vs. coloring" problem, which is of note since this problem is defined by only polynomially-many inequalities. Thus, Corollary 4 taken together with results in [54] and [55] also settles the question raised in Dadush and Tiwari [18] as long as the maximum coefficient in the disjunctions used in the tree is bounded by a polynomial in n.

3.1.1 Contributions of this paper and relationship to existing results

Contributions. We construct an instance of packing-type and a set-cover instance such that any general branch-and-bound tree that solves these instances must be of exponential (with respect to the ambient dimension) size. We note that the packing and covering instances are described using an exponential number of constraints, and so unfortunately this does not settle the

question raised by Dadush and Tiwari [18]. We also present a simple proof that any branch-and-bound tree proving the integer infeasibility of the cross-polytope in n dimensions must have 2^n leaves. We then extend this result to give (high-probability) exponential lower bounds for perturbed instances of the cross-polytope where independent Gaussian noise is added to the entries of the constraint matrix. To our knowledge this is the first result that shows that a "smoothed analysis" [56] polynomial upper bound on the size of branch-and-bound trees is not possible. Finally, we show an exponential lower bound on the size of any general branch-and-bound tree for the Traveling Salesman Problem (TSP).

Comparison to previous results. We now discuss our results in the context of the recent landscape, in particular with the results of [18] and [53].

1. *New problems with exponential lower bounds on the size of general branch-and-bound tree:* As mentioned earlier, recently Dadush and Tiwari [18] provided the first exponential lower bound on the size of general branch-and-bound tree for the cross-polytope. The other additional result, Corollary 4 from [53], only implies branch-and-bound lower bounds for polytopes for which we already have CG hardness. These come few and far between in the existing literature, and these instances are often a bit artificial; see [54] and [55]. In contrast, in this paper we provide lower bounds for the size of general branch-and-bound tree for packing and set-cover instances, which are more natural combinatorial problems than those mentioned above.

2. *Improved quality of bounds:* Dadush and Tiwari [18] show that any branch-and-bound proof of infeasibility of the cross-polytope has at least $\frac{2^n}{n}$ leaves. We improve on this result by providing a simple proof that any such proof of infeasibility must have 2^n leaves.

   Chvatal et al. [57] provide a $\frac{1}{3n}2^{n/8}$ lower bound on CG proofs for TSP. Combined with Corollary 4, this can be used to show a lower bound of $2^{O\left(\frac{n}{\log cn}\right)}$ for branch-and-bound trees

for TSP using maximum coefficient c for the disjunctions. We are able to achieve a stronger lower bound of $2^{\Omega(n)}$.

3. *Removing the dependence on the maximum coefficient size used in the branch-and-bound proof:* The bound given in Corollary 4 depends on the maximum coefficient size used in the branch-and-bound proof. In [53], the authors mention that they "view this as a step toward proving [branch-and-bound] lower bounds (with no restrictions on the [coefficient sizes])". Our results satisfy this property, as none of the bounds presented in this work depend on the coefficients of the inequalities of the general branch-and-bound proof.

Finally, the results presented here can be easily combined with Theorem 1.14 of [15] to apply to branch-and-cut proofs. In particular, our results imply exponential lower bounds, for the polytopes shown here, on the size of branch-and-cut proofs that are allowed to branch on split disjunctions and employ any cutting-plane paradigm that is "not sufficiently different" from split disjunctive cuts (see [15] for details).

3.1.2 Roadmap and notation

Since this paper focuses on lower bounds for general branch-and-bound trees (obviously implying lower bounds for simple branch-and-bound tree), we drop the term "general" for the rest of the paper. The paper is organized as follows. In section 3.2 we present the necessary definitions. In section 3.3, we present key reduction results that allow transferring lower bounds on the size of branch-and-bound trees from one optimization problem to another. In section 3.4, we present a lower bound on the size of branch-and-bound trees for packing and set covering instances. In section 3.5, we present a lower bound on the size of branch-and-bound trees for the cross-polytope and some other related technical results. In section 3.6, we show that even after adding Gaussian noise to the coefficients of the cross-polytope, with good probability branch-and-bound still requires an exponentially large tree to prove infeasibility. Finally, in section 3.7, we use results

from section 3.5 and section 3.3 to provide an exponential lower bound on the size of branch-and-bound trees for solving TSP instances.

For a positive integer n, we denote the set $\{1, \dots, n\}$ as $[n]$. When the dimension is clear from context, we use the notation $\mathbf{1}$ to be a vector whose every entry is 1. Let C be a set of linear constraints of the form $(\pi^i)^\top x \leq \pi_0^i,\ \forall i \in [m]$. Then let $\{x : C\}$ denote the set of all $x \in [0,1]^n$ such that all of the constraints C are valid for x (i.e. the polytope defined by the set of constraints C). Note that for a subset of these constraints $B \subseteq C$, it holds that $\{x : B\} \supseteq \{x : C\}$. Also note that for two sets of constraints B, C, it holds that $\{x : B \cup C\} = \{x : B\} \cap \{x : C\}$. Given a set S, we denotes its convex hull by $\text{conv}(S)$. Given a polytope $P \subseteq \mathbb{R}^n$, we denote its integer hull, that is the set $\text{conv}(P \cap \mathbb{Z}^n)$, as P_I. We call P integer-infeasible if $P \cap \mathbb{Z}^n = \emptyset$.

3.2 Abstract branch-and-bound trees and notions of hardness

In order to present lower bounds on the size of branch-and-bound (BB) trees, we simplify our analysis by removing two typical condition assumed in a BB algorithm – (i) the requirement that the partitioning into two subproblems (which correspond to the child nodes of the given node) is done in such a way that the optimal LP solution of the parent node is not included in either subproblem, and that (ii) branching is not done on pruned nodes. By removing these conditions, we can talk about a branch-and-bound tree independent of the underlying polytope – it is just a full binary tree (that is, each node has 0 or 2 child nodes). The root-node has an empty set of *branching constraints*. If a node has two child nodes, these are obtained by applying some disjunction $\pi^\top x \leq \pi_0 \vee \pi^\top x \geq \pi_0 + 1$, where each of the child nodes adds one of these constraints to its set of branching constraints together with all the branching constraints of the parent node. Note that here π is an *integer* vector and π_0 an *integer*; we call such disjunctions *legal*. Note that proving lower bounds on the size of such BB trees that solves a given integer program certainly gives a lower bound on the size of BB trees that in addition require (i) and (ii). Finally, note that since a BB tree is a full binary tree, the total number of nodes of a BB tree with N leaf-nodes is

$2N - 1$.

Definition 1. *Given a branch-and-bound tree $\mathcal{T}$, applied to a polytope $P \subseteq \mathbb{R}^n$, and a node v of the tree:*

- *We denote the number of nodes of the branch-and-bound tree $\mathcal{T}$ by $|\mathcal{T}|$. This is what is termed* the size *of this tree.*
- *We denote by C_v the set of branching constraints of v (as explained above, these are the constraints added by the branch-and-bound tree along the path from the root-node to v).*
- *We call the feasible region defined by the LP relaxation P and the branching constraints at node v the* atom *of this node, i.e., $P \cap \{x : C_v\}$ is the atom corresponding to v.*
- *We let $\mathcal{T}(P)$ denote the union of the atoms corresponding to the leaves of $\mathcal{T}$ when run on polytope P, i.e., $\mathcal{T}(P) = \bigcup_{v \in leaves(\mathcal{T})}(P \cap \{x : C_v\})$.*
- *For any $x^* \in P \setminus P_I$, we say that $\mathcal{T}$* separates *x^* from P if $x^* \notin conv(\mathcal{T}(P))$.*
- *Given a vector $c \in \mathbb{R}^n$, we say $\mathcal{T}$ solves $\max_{x \in P \cap \mathbb{Z}^n} \langle c, x \rangle$ if for all the leaf nodes v of $\mathcal{T}$, one of the following three conditions hold: (i) the atom of v is empty, (ii) there exists at least one optimal solution of the linear program $\max_{x \in \text{ atom of } v} \langle c, x \rangle$ that is integral, or (iii) $\max_{x \in \text{ atom of } v} \langle c, x \rangle$ is at most the objective function value of another atom whose optimal solution is integral.*

  If $P \cap \mathbb{Z}^n = \emptyset$, note that (ii) and (iii) are not possible, and in this case we use the term "proves integer-infeasibility" instead of "solves" the problem.

Given a polytope $P \subseteq \mathbb{R}^n$, we define its *BB hardness* as

$$BBhardness(P) = \max_{c \in \mathbb{R}^n} \left(\min \left\{ |\mathcal{T}| : \mathcal{T} \text{ solves } \max_{x \in P \cap \mathbb{Z}^n} \langle c, x \rangle \right\} \right).$$

Our goal for most of this paper is to provide lower bounds on the BB hardness of certain polytopes.

To get exponential lower bounds on BB hardness for some P, we will often present a particular point $x^* \in P \setminus P_I$ such that any $\mathcal{T}$ that separates x^* from P must have exponential size. We formalize this below.

Definition 2 (BBdepth)**.** *Let $P \subseteq \mathbb{R}^n$ be a polytope and consider any $x^* \in P \setminus P_I$. Let $\mathcal{T}$ be a smallest BB tree that separates x^* from P. Then, define BBdepth(x^*, P) to be $|\mathcal{T}|$.*

Definition 3 (BBrank)**.** *Define BBrank$(P) = \max_{x \in P \setminus P_I}$ BBdepth(x, P).*

Lemma 10 (BB rank lower bounds BB hardness)**.** *Let $P \subseteq \mathbb{R}^n$ be a polytope. Then, there exists $c \in \mathbb{R}^n$ such that any BB tree solving $\max_{x \in P \cap \mathbb{Z}^n} \langle c, x \rangle$ must have size at least BBrank(P), that is BBhardness$(P) \geq$ BBrank(P).*

Proof. Let $x^* \in \operatorname{argmax}_{x \in P \setminus P_I} \text{BBdepth}(x, P)$, so that $\text{BBrank}(P) = \text{BBdepth}(x^*, P)$. Since x^* does not belong to the convex set P_I, by the hyperplane separation theorem [58] there exists c with the separation property $\langle c, x^* \rangle > \max_{x \in P_I} \langle c, x \rangle$. By choice of x^*, for any BB tree $\mathcal{T}$ with $|\mathcal{T}| <$ BBrank(P) it holds that $x^* \in \text{conv}(\mathcal{T}(P))$. Then such tree $\mathcal{T}$ must have a leaf whose optimal LP solution has value at least $\langle c, x^* \rangle > \max_{x \in P_I} \langle c, x \rangle$, and therefore must still not be pruned, showing that $\mathcal{T}$ does not solve $\max_{x \in P \cap \mathbb{Z}^n} \langle c, x \rangle$. □

We now show that, under some conditions, the reverse of this kind of relationship also holds. We will use this reverse relationship to prove the BB hardness of optimizing over an integer feasible polytope given the BB hardness of proving the infeasibility of another "smaller" polytope.

Lemma 11 (Infeasibility-to-optimization)**.** *Let $P \subseteq \mathbb{R}^n$ be a polytope and $\langle c, x \rangle \leq \delta$ be a facet defining inequality of P_I that is not valid for P. Assume that the affine hull of P and P_I are the same. Then, there exists $\varepsilon_0 > 0$ such that for every $\varepsilon \in (0, \varepsilon_0]$*

$$\textit{BBrank}(P) \geq \textit{BBhardness}(\{x \in P : \langle c, x \rangle \geq \delta + \varepsilon\}).$$

Before we can present the proof of Lemma 11 we require a technical lemma from [59]. The full-dimensional case $L = \mathbb{R}^n$ is Lemma 3.1 of [59], and the general case follows directly by applying it to the affine subspace L.

Lemma 12 ([59])**.** *Consider an affine subspace $L \subseteq \mathbb{R}^n$ and a hyperplane $H = \{x \in \mathbb{R}^n : \langle c, x\rangle = \delta\}$ that does not contain L. Consider $\dim(L)$ affinely independent points $s^1, s^2 \dots, s^{\dim(L)}$ in $L \cap H$. Consider $\delta' > \delta$ and let G be a bounded and non-empty subset of $L \cap \{x \in \mathbb{R}^n : \langle c, x\rangle \geq \delta'\}$. Then there exists a point x in $\bigcap_{g\in G} conv(s^1, \dots, s^{\dim(L)}, g)$ satisfying the strict inequality $\langle c, x\rangle > \delta$.*

Proof of Lemma 11. Let L be the affine hull of P_I. Then there exist $d := \dim(P_I) = \dim(L)$ affinely independent vertices of $\{x \in P_I : \langle c, x\rangle = \delta\}$. Let $s^1, \dots, s^d$ be d such affinely independent vertices and note that since they are vertices of P_I, they are all integral. Let $\varepsilon_0 := (\max_{x\in P}\langle c, x\rangle) - \delta$, and notice that since the inequality $\langle c, x\rangle \leq \delta$ is not valid for P we have $\varepsilon_0 > 0$. Then for any $\varepsilon \in (0, \varepsilon_0]$, let $G := \{x \in P : \langle c, x\rangle \geq \delta + \varepsilon\}$, which is then non-empty. Also notice that G is a bounded set, since P is bounded. Let $N := \text{BBhardness}(G)$.

Let $\mathcal{T}$ be a BB tree such that $|\mathcal{T}| < N$. Then we have that $\mathcal{T}(G) \neq \emptyset$, that is, there exists $x^*(\mathcal{T}) \in \mathcal{T}(G)$. In particular $x^*(\mathcal{T}) \in G$. Moreover, since $G \subseteq P$, we have $\mathcal{T}(G) \subseteq \mathcal{T}(P)$ (see Lemma 13 in the next section for a formal proof of this), and so we have $x^*(\mathcal{T}) \in \mathcal{T}(P)$. Also note that since $s^1, \dots, s^d \in P \cap \mathbb{Z}^n$, we have that these points also belong to $\mathcal{T}(P)$. Thus,

$$\text{conv}\left(s^1, \dots, s^d, x^*(\mathcal{T})\right) \subseteq \text{conv}(\mathcal{T}(P)).$$

Now applying Lemma 12, with $\delta' = \delta + \varepsilon$, we have that there exists x^* such that

$$x^* \in \bigcap_{\mathcal{T} : |\mathcal{T}|<N} \text{conv}\left(s^1, \dots, s^d, x^*(\mathcal{T})\right) \subseteq \bigcap_{\mathcal{T} : |\mathcal{T}|<N} \text{conv}(\mathcal{T}(P)) \tag{3.1}$$

and such that $\langle c, x^*\rangle > \delta$. Clearly, $x^* \notin P_I$, since $\langle c, x\rangle \leq \delta$ is a valid inequality for P_I. Thus, since (Equation 3.1) implies $x^* \in \text{conv}(\mathcal{T}(P))$ for all $\mathcal{T}$ with $|\mathcal{T}| < N$, we have that $\text{BBdepth}(x^*, P) \geq N$

and consequently, BBrank$(P) \geq N$. □

3.3 Framework for BB hardness reductions

In this section we present key reduction results that allow transferring lower bounds on the size of BB trees from one optimization problem to another. We begin by showing monotonicity of the operator $\mathcal{T}(\cdot)$.

Lemma 13 (Monotonicity of leaves)**.** *Let $Q \subseteq P \subseteq \mathbb{R}^n$ be polytopes. Then $\mathcal{T}(Q) \subseteq \mathcal{T}(P)$.*

Proof. For any leaf $v \in \mathcal{T}$, recall that C_v is the set of branching constraints of v. Then $\mathcal{T}(Q) = \bigcup_{v \in \text{leaves}(\mathcal{T})}(Q \cap \{x : C_v\}) = Q \cap \bigcup_{v \in \text{leaves}(\mathcal{T})}\{x : C_v\} \subseteq P \cap \bigcup_{v \in \text{leaves}(\mathcal{T})}\{x : C_v\} = \bigcup_{v \in \text{leaves}(\mathcal{T})}(P \cap \{x : C_v\}) = \mathcal{T}(P)$. □

The following corollary follows easily from Lemma 13. In particular, consider a smallest BB tree $\mathcal{T}$ that separates x^* from P. By Lemma 13, the same tree, when applied to $Q \subseteq P$, will not have x^* in the convex hull of its leaves and therefore separates x^* from Q.

Corollary 5 (Monotonicity of depth)**.** *Let $Q \subseteq P \subseteq \mathbb{R}^n$ be polytopes. Then for every $x^* \in (Q \setminus Q_I) \cap (P \setminus P_I) = Q \setminus P_I$ we have*

$$BBdepth(x^*, Q) \leq BBdepth(x^*, P).$$

Inspired by the lower bounds for cutting-plane rank from [57], we show that integral affine transformations conserve the hardness of separating a point via branch-and-bound, i.e. they conserve BBdepth. Then, we give a condition where BBrank is also conserved. These will be used to obtain lower bounds in the subsequent sections.

We say that $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ is an *integral affine function* if it has the form $f(x) = Cx + d$, where $C \in \mathbb{Z}^{m \times n}$, $d \in \mathbb{Z}^m$.

Lemma 14 (Simulation for integral affine transformations)**.** *Let $P \subseteq \mathbb{R}^n$ be a polytope, $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ an integral affine function, and denote $Q := f(P) \subseteq \mathbb{R}^m$. Let $\hat{\mathcal{T}}$ be any BB tree. Then, there exists*

a BB tree $\mathcal{T}$ *such that* $|\mathcal{T}| = |\hat{\mathcal{T}}|$ *and*

$$f(\mathcal{T}(P)) \subseteq \hat{\mathcal{T}}(Q).$$

Proof. Let $f(x) = Cx + d$ with $C \in \mathbb{Z}^{m\times n}$, $d \in \mathbb{Z}^m$. Given a BB tree $\hat{\mathcal{T}}$, we construct a BB tree $\mathcal{T}$ with the desired properties as follows: $\mathcal{T}$ has the same nodes as $\hat{\mathcal{T}}$ but each branching constraint $\langle a, y\rangle \le b$ of $\hat{\mathcal{T}}$ is replaced by the constraint $\langle C^T a, x\rangle \le b - \langle a, d\rangle$ in $\mathcal{T}$.

First we verify that $\mathcal{T}$ only uses legal disjunctions: First note that $C^T a \in \mathbb{Z}^n$ and $b - \langle a, d\rangle \in \mathbb{Z}$. If a node of $\hat{\mathcal{T}}$ has $\langle a, y\rangle \le b \vee \langle a, y\rangle \ge b+1$ as its disjunction, the corresponding node in $\mathcal{T}$ has the disjunction $\langle C^T a, x\rangle \le b - \langle a, d\rangle \vee \langle -C^T a, x\rangle \le -b - 1 - \langle -a, d\rangle$ (notice $\langle a, y\rangle \ge b+1 \equiv \langle -a, y\rangle \le -b - 1$). Since the second term in the latter disjunction is equivalent to $\langle C^T a, x\rangle \ge b - \langle a, d\rangle + 1$, we see that this disjunction is a legal one.

To conclude the proof, we show that $f(\mathcal{T}(P)) \subseteq \hat{\mathcal{T}}(Q)$. Let S be the atom of a leaf v of $\mathcal{T}$ and $\hat{S}$ be the atom of the corresponding leaf $\hat{v}$ of $\hat{\mathcal{T}}$. We show that for all $x \in S$, it must be that $f(x) \in \hat{S}$. To see this, notice that if x satisfies an inequality $\langle C^T a, x\rangle \le b - \langle a, d\rangle$ then $f(x)$ satisfies $\langle a, f(x)\rangle \le b$:

$$\langle a, f(x)\rangle = \langle a, Cx + d\rangle = \langle a, Cx\rangle + \langle a, d\rangle = \langle C^T a, x\rangle + \langle a, d\rangle \le b.$$

Since any $x \in S$ belongs to P and satisfies all the branching constraints of the leaf v, this implies $f(x)$ belongs to Q and satisfies all the branching constraints of the leaf $\hat{v}$, and hence belongs to the atom $\hat{S}$. Therefore, $f(S) \subseteq \hat{S}$. Taking a union over all leaves/atoms then gives $f(\mathcal{T}(P)) \subseteq \hat{\mathcal{T}}(Q)$ as desired. □

Corollary 6. *Let* P, Q, *and* f *satisfy the assumptions of Lemma 14. Further, suppose* P *and* Q *are both integer-infeasible. Then,*

$$BBhardness(Q) \ge BBhardness(P).$$

Proof. Let $\hat{\mathcal{T}}$ be the smallest BB tree such that $\hat{\mathcal{T}}(Q) = \emptyset$. Then, by Lemma 14, it must hold that $\mathcal{T}(P) = \emptyset$. The desired result follows. □

Corollary 7. *Let P, Q, and f satisfy the assumptions of Lemma 14. Then for every $x^* \in \mathbb{R}^n$ such that $x^* \notin P_I$ and $f(x^*) \notin Q_I$, we have*

$$BBdepth(f(x^*), Q) \geq BBdepth(x^*, P).$$

Proof. Let $\hat{\mathcal{T}}$ be a smallest BB tree that separates $f(x^*)$ from Q, and let $\mathcal{T}$ be a tree given by Lemma 14. Together with the fact that f is affine, this implies that if $x \in \text{conv}(\mathcal{T}(P))$ then $f(x) \in \text{conv}(\hat{\mathcal{T}}(Q))$: there exists $x^1, ..., x^k \in \mathcal{T}(P)$ and $\lambda_1, ..., \lambda_k \in [0, 1]$ such that $\sum_{i\in[k]} \lambda_i = 1$ and $x = \sum_{i\in[k]} \lambda_i x^i$. Thus,

$$\begin{aligned} f(x) &= C(\sum_{i\in[k]} \lambda_i x^i) + d = \sum_{i\in[k]} \lambda_i (Cx^i) + \sum_{i\in[k]} \lambda_i d \\ &= \sum_{i\in[k]} \lambda_i (Cx^i + d) = \sum_{i\in[k]} \lambda_i f(x^i) \in \text{conv}(\hat{\mathcal{T}}(Q)), \end{aligned}$$

where the last containment is by definition of $\mathcal{T}$. Since we know $f(x^*) \notin \text{conv}(\hat{\mathcal{T}}(Q))$, this implies that $x^* \notin \text{conv}(\mathcal{T}(P))$, namely $\mathcal{T}$ separates x^* from P as desired. □

Lemma 15 (Hardness lemma)**.** *Let $P \subseteq \mathbb{R}^n$ and $T \subseteq \mathbb{R}^m$ be polytopes and $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ an integral affine function such that $f(P) \subseteq T$. Suppose f is also one-to-one and $T \cap \mathbb{Z}^m \subseteq f(P \cap \mathbb{Z}^n)$. Then,*

$$BBrank(T) \geq BBrank(P).$$

Proof. First we show that $x \notin P_I$ implies $f(x) \notin T_I$ by proving the contrapositive. Suppose $f(x) \in T_I$; then $\exists y^1, ..., y^k \in T \cap \mathbb{Z}^m$ and $\lambda_1, ..., \lambda_k \in [0, 1]$ such that $\sum_{i\in[k]} \lambda_i = 1$ and $f(x) = \sum_{i\in[k]} \lambda_i y^i$.

Since $T \cap \mathbb{Z}^m \subseteq f(P \cap \mathbb{Z}^n)$, for each i there is $x^i \in P \cap \mathbb{Z}^n$ such that $y^i = f(x^i)$. Then

$$f(x) = \sum_{i\in[k]} \lambda_i f(x^i) = \sum_{i\in[k]} \lambda_i (Cx^i + d) = C \sum_{i\in[k]} \lambda_i x^i + d = f(\sum_{i\in[k]} \lambda_i x^i),$$

and so $f(x)$ belongs to $f(P_I)$. Since f is one-to-one, this implies that x belongs to P_I, as desired.

Now let $x^* = \text{argmax}_{x\in P\setminus P_I} \text{BBdepth}(x, P)$. Since $x^* \notin P_I$, by the above claim $f(x^*) \notin T_I$. By assumption $f(P) \cap \mathbb{Z}^m \subseteq T \cap \mathbb{Z}^m$, and so $(f(P))_I \subseteq T_I$, and therefore $f(x^*) \notin (f(P))_I$. Then by Corollary 7 we have

$$\text{BBdepth}(f(x^*), f(P)) \geq \text{BBdepth}(x^*, P).$$

Since by assumption $f(P) \subseteq T$, $f(x^*) \in f(P)$, and $f(x^*) \notin T_I$, by Corollary 5 we have

$$\text{BBdepth}(f(x^*), T) \geq \text{BBdepth}(f(x^*), f(P)).$$

Putting it all together we get

$$\text{BBrank}(T) = \max_{y\in T\setminus T_I} \text{BBdepth}(y, T) \geq \text{BBdepth}(f(x^*), T) \geq \text{BBdepth}(f(x^*), f(P)) \geq \text{BBdepth}(x^*, P)$$

$$= \max_{x\in P\setminus P_I} \text{BBdepth}(x, P) = \text{BBrank}(P),$$

which concludes the proof of the lemma. □

In the rest of the paper, we will use Corollary 6, Corollary 7 or Lemma 15 together with some appropriate affine transformation to reduce the BB hardness of one problem to another. The three affine one-to-one functions we will use (and their compositions) are Flipping, Embedding, and Duplication as defined below.

Definition 4 (Flipping)**.** *We say* $f : [0,1]^n \to [0,1]^n$ *is a* flipping operation *if it "flips" some*

coordinates, that is, there exists $J \subseteq [n]$ *such that*

$$y = f(x) \implies y_i = \begin{cases} x_i & \text{if } i \notin J \\ 1 - x_i & \text{if } i \in J \end{cases}.$$

In other words, $f(x) = Cx + d$*, where (recall* e_i *is the* i*-th canonical basis vector)*

$$C^i = \begin{cases} e_i & \text{if } i \notin J \\ -e_i & \text{if } i \in J \end{cases}$$

$$d_i = \begin{cases} 0 & \text{if } i \notin J \\ 1 & \text{if } i \in J \end{cases}.$$

Definition 5 (Embedding)**.** *We say* $f : [0,1]^n \longrightarrow [0,1]^{n+k}$ *is an* embedding operation *if*

$$y = f(x) \implies y_i = \begin{cases} x_i & \text{if } 1 \leq i \leq n \\ 0 & \text{if } n < i \leq n + k_1 \\ 1 & \text{if } n + k_1 < i \leq n + k \end{cases},$$

for some $0 \leq k_1 \leq k$*. In other words,* $f(x) = Cx + d$*, where*

$$C^i = \begin{cases} e_i & \text{if } 1 \leq i \leq n \\ 0 & \text{otherwise} \end{cases}$$

$$d_i = \begin{cases} 1 & \text{if } n + k_1 < i \leq n + k \\ 0 & \text{otherwise} \end{cases}.$$

Note that we can always renumber the coordinates so that the additional coordinates with values 0 *or* 1 *are interspersed with the original ones and not grouped at the end.*

Definition 6 (Duplication)**.** *Consider a k-tuple of coordinates $(j_1, ..., j_k)$ that are not necessarily distinct, where $j_i \in \{1, ..., n\}$ for $i = 1, ..., k$. We say that $f : [0,1]^n \longrightarrow [0,1]^{n+k}$ is a* duplication operation *using this tuple if*

$$y = f(x) \implies y_i = \begin{cases} x_i & \text{if } 1 \le i \le n \\ x_{j_{i-n}} & \text{if } n < i \le n+k \end{cases}.$$

Further, let $J_j = \{i \in \{1, ..., k\} : y_{n+i} = x_j\}$ be the indices of y that are duplicates of x_j. Then, in other words, $f(x) = Cx$ where

$$C^i = \begin{cases} e_i & \text{if } 1 \le i \le n \\ e_1 & \text{if } i - n \in J_1 \\ \vdots & \\ e_n & \text{if } i - n \in J_n \end{cases}$$

3.4 BB hardness for packing polytopes and set-cover

In this section, we will begin by presenting a packing polytope with BBrank of $2^{\Omega(n)}$. The proof of this result will be based on a technique developed by Dadush and Tiwari [18]. Then we will employ affine maps that satisfy Lemma 15 to obtain lower bounds on BBrank for a set-cover instance.

We present a slightly generalized version of a key result from [18]. The proof is essentially the same as of the original version, but we present it for completeness.

Lemma 16 (Generalized Dadush-Tiwari Lemma)**.** *Let $P \subseteq \mathbb{R}^n$ be an integer-infeasible non-empty polytope. Further, suppose P is defined by the set of constraints C_P (i.e. $P = \{x : C_P\}$) and let $D \subseteq C_P$*

be a subset of constraints such that if we remove any constraint in D, the polytope becomes integer feasible (i.e. for all subsets $C \subset C_P$ such that $D \setminus C \neq \emptyset$, it holds that $\{x : C\} \cap \mathbb{Z}^n \neq \emptyset$). Then, any branch-and-bound tree $\mathcal{T}$ proving the integer-infeasibility of P has at least $\frac{|D|}{n}$ leaf nodes, that is $|\mathcal{T}| \geq 2\frac{|D|}{n} - 1$.

Proof. Let $\mathcal{T}$ denote any branch-and-bound proof of infeasibility for P and let N denote the number of leaf nodes of $\mathcal{T}$. Suppose for sake of contradiction, that $N < \frac{|D|}{n}$. Consider any leaf node v of $\mathcal{T}$. Let C_v be the set of branching constraints on the path to v. Since v is a leaf and $\mathcal{T}$ is a proof of infeasibility, we note $\{x : C_v \cup C_P\} = \{x : C_v\} \cap P = \emptyset$.

By Helly's Theorem [58], there exists a set of $n+1$ constraints $K_v \subseteq C_v \cup C_P$ such that $\{x : K_v\} = \emptyset$. Also, we see that

$$|K_v \cap C_P| \leq n. \tag{3.2}$$

This is because if we had $|K_v \cap C_P| = n+1$, this would imply $K_v \subseteq C_P$, hence $\{x : C_P\} \subseteq \{x : K_v\}$, and since $\{x : K_v\} = \emptyset$; this would imply $\{x : C_P\} = P = \emptyset$, which is clearly a contradiction because we know P is non-empty.

Next, observe that the set $\tilde{P} := \left\{x : \bigcup_{v \in \text{leaves}(\mathcal{T})}(K_v \cap C_P)\right\}$ is integer-infeasible, because in fact $\mathcal{T}$ certifies this: since $\tilde{P} \cap \{x : C_u\} = \left\{x : C_u \cup \bigcup_{v \in \text{leaves}(\mathcal{T})}(K_v \cap C_P)\right\} \subseteq \{x : K_u\} = \emptyset$ for all $u \in \text{leaves}(\mathcal{T})$. On other hand, observe that by (Equation 3.2) we have that $|\bigcup_{v \in \text{leaves}(\mathcal{T})}(K_v \cap C_P)| \leq nN < |D|$, so one of the inequalities in D is not used in the description of $\tilde{P}$ and hence $\tilde{P}$ contains an integer point, a contradiction. This concludes the proof. □

3.4.1 Packing polytopes

Consider the following packing polytope

$$P_{PA} = \left\{x \in [0,1]^n : \sum_{i \in S} x_i \leq k-1 \text{ for all } S \subseteq [n] \text{ such that } |S| = k\right\},$$

where we assume $2 \leq k \leq \frac{n}{2}$.

Lemma 17. *There exists an $x^* \in P_{PA} \setminus (P_{PA})_I$ such that any branch-and-bound tree that separates x^* from P_{PA} has at least $\frac{2}{n}\left(\binom{n}{k}+1\right)-1$ nodes. Therefore, BBrank$(P_{PA}) \geq \frac{2}{n}\left(\binom{n}{k}+1\right)-1$.*

The starting point for proving this lemma is the following following proposition.

Proposition 1. *The polytope $Q = P_{PA} \cap \{x : \langle \mathbf{1}, x\rangle \geq k\}$ is integer-infeasible, and any branch-and-bound tree proving its integer-infeasibility has at least $\frac{2}{n}(\binom{n}{k}+1)-1$ nodes.*

Proof. We show $Q \cap \{0,1\}^n = \emptyset$. Suppose for sake of contradiction there is some $x^* \in Q \cap \{0,1\}^n$. Since $\sum_{i\in[n]} x_i^* \geq k$, there is a set $S \subseteq [n]$ of size k such that $\sum_{i\in S} x_i^* = k$. This violates the cardinality constraint corresponding to S, so $x^* \notin Q$, a contradiction.

For the lower bound on BB trees that prove the integer-infeasibility of Q, we show that Q satisfies all of the requirements of Lemma 16. First we show that $Q \neq \emptyset$. Consider the point $\hat{x} \in \mathbb{R}^n$ where $\hat{x}_i = \frac{k}{n}$ for $i \in [n]$. Then, for any $S \subseteq [n]$ with $|S| = k$, we have $\sum_{i\in S} \hat{x}_i = k \cdot \frac{k}{n} \leq k \cdot \frac{1}{2} \leq k-1$, where the last two inequalities are implied by the assumption $2 \leq k \leq \frac{n}{2}$. Also, $\sum_{i\in[n]} \hat{x}_i = k$. Thus, $\hat{x}$ satisfies all the constraints of Q.

Next, we show that there is a set of $\binom{n}{k}+1$ constraints D such that removing any of these constraints makes Q integer feasible. Suppose we remove the constraint $\sum_{i\in S} x_i \leq k-1$, denote this new polytope Q'. Then let $x_i^* = 1$ for all $i \in S$ and $x_i^* = 0$ for all $i \notin S$. Clearly $\sum_{i\in[n]} x_i^* \geq k$ and since for all $S' \subseteq [n], |S'| = k$ it holds that $|S' \cap S| \leq k-1$, it is also the case that $\sum_{i\in S'} x_i^* \leq k-1$. So $x^* \in Q' \cap \{0,1\}^n$. Now suppose we remove instead the constraint $\sum_{i\in[n]} x_i \geq k$, resulting in polytope P_{PA}. Clearly P_{PA} is down monotone, and therefore $0 \in P_{PA}$.

Therefore, by Lemma 16, any branch-and-bound proof of infeasibility for Q has at least $\frac{2}{n}(\binom{n}{k}+1)-1$ nodes. □

Now, combining Proposition 1 with Lemma 11, we are ready to prove Lemma 17.

Proof of Lemma 17. We will show that P_{PA} and $\langle \mathbf{1}, x\rangle \leq k-1$ satisfy the conditions on P and $\langle c, x\rangle \leq \delta$ set by Lemma 11. First, $\langle \mathbf{1}, x\rangle \leq k-1$ is a valid inequality for $(P_{PA})_I$: this follows from the integer-infeasibility of $Q = P_{PA} \cap \{x : \langle \mathbf{1}, x\rangle \geq k\}$, as proven in Proposition 1. Also, clearly $\langle \mathbf{1}, x\rangle \leq k-1$ is not valid for P_{PA}, since it cuts off the point $\hat{x} = \frac{k}{n}\mathbf{1} \in P_{PA}$. In the following paragraph we will show that $\{x \in (P_{PA})_I : \langle \mathbf{1}, x\rangle = k-1\}$ has dimension $n-1$, that is, $\langle \mathbf{1}, x\rangle \leq k-1$ is facet-defining for $(P_{PA})_I$. With this at hand we can apply Lemma 11 to obtain

$$\text{BBrank}(P_{PA}) \geq \text{BBhardness}(P_{PA} \cap \{x : \langle \mathbf{1}, x\rangle \geq k\}) = \frac{2}{n}\left(\binom{n}{k} + 1\right) - 1$$

where the last inequality follows from Proposition 1, which will then prove the lemma.

To show that $\langle \mathbf{1}, x\rangle \leq k-1$ is facet-defining, let $T \subseteq [n]$ be such that $|T| = k-1$. Let $\chi(T)$ denote the characteristic vector of T, so that $\chi(T)_i = 1$ if and only if $i \in T$. We know that all these points belong to the hyperplane $\{x : \langle \mathbf{1}, x\rangle = k-1\}$. Thus, there can be at most n affinely independent points among $\{\chi(T)\}_{T\subseteq[n],|T|=k-1}$. We first verify that there are exactly n affinely independent points among $\{\chi(T)\}_{T\subseteq[n],|T|=k-1}$ by showing that the affine hull of the points in $\{\chi(T)\}_{T\subseteq[n],|T|=k-1}$ is the hyperplane $\{x : \langle \mathbf{1}, x\rangle = k-1\}$. Consider the system in variables a, b:

$$\langle a, \chi(T)\rangle = b, \quad \forall T \subseteq [n] \text{ such that } |T| = k-1.$$

We have to show that all non-zero solutions of the above system are a scaling of $(\mathbf{1}, k-1)$. For that, let $T^1 = \{1, \ldots, k-1\}$ and $T^2 := \{2, \ldots, k\}$. Subtracting the equation corresponding to T^1 from that of T^2, we obtain $a_1 = a_k$. Using the same argument by suitably selecting T^1 and T^2, we obtain: $a_1 = a_2 = \cdots = a_n$. Therefore, without loss of generality (excluding the solution where a is identically 0, since that would lead to $b = 0$), we may rescale all the a_i's to 1. Then we see the only possible value for b is $k-1$. This shows that the only affine subspace containing the points $\{\chi(T)\}_{T\subseteq[n],|T|=k-1}$ is $\{x : \langle \mathbf{1}, x\rangle = k-1\}$, in other words, there are n affinely independent points among them. This concludes the proof. □

Finally, since BB hardness is always at least the BB rank (Lemma 10), Lemma 17 gives the desired hardness bound.

Corollary 8. *Consider the polytope* $P_{PA} = \{x \in [0,1]^n : \sum_{i \in S} x_i \leq \frac{n}{2},$ *for all* $S \subseteq [n]$ *such that* $|S| = \frac{n}{2} + 1\}$. *Then,* $BBhardness(P_{PA}) \geq 2^{\Omega(n)}$, *i.e. there exists a vector* $c \in \mathbb{R}^n$ *such that the smallest branch-and-bound tree that solves*

$$\max_{x \in P_{PA} \cap \{0,1\}^n} \langle c, x \rangle$$

has size at least $2^{\Omega(n)}$.

3.4.2 Set-cover

In order to obtain a set-cover instance that requires an exponential-size branch-and-bound tree, we will use Lemma 15 together with the flipping affine mapping (Defintion 4) applied to the packing instance from subsection 3.4.1.

More precisely, consider the following set-cover polytope:

$$T_{\text{SC}} = \left\{ y \in [0,1]^n : \sum_{i \in S} y_i \geq 1 \text{ for all } S \subseteq [n] \text{ such that } |S| = k \right\}.$$

Proposition 2. *Let* P_{PA} *still be the packing polytope from subsection 3.4.1. Let* $f : [0,1]^n \longrightarrow [0,1]^n$ *be the flipping function with* $J = [n]$. *Then:*

- $T_{SC} = f(P_{PA})$
- $T_{SC} \cap \{0,1\}^n \subseteq f(P_{PA} \cap \{0,1\}^n)$.

Proof. By substituting $y_i = 1 - x_i$ for $i \in [n]$ in the polytope P_{PA}, we obtain that $T_{\text{SC}} = f(P_{PA})$.

For the second item, consider any $\hat{y} \in T_{\text{SC}} \cap \{0,1\}^n$. Notice $\hat{x} := 1 - \hat{y}$ belongs to $P_{PA} \cap \{0,1\}^n$ and $\hat{y} = f(\hat{x})$, and hence $\hat{y} \in f(P_{PA} \cap \{0,1\}^n)$. This gives $T_{\text{SC}} \cap \{0,1\}^n \subseteq f(P_{PA} \cap \{0,1\}^n)$. □

Then by Lemmas 15 and 17 we have that $\text{BBrank}(T_{\text{SC}}) \geq \text{BBrank}(P_{PA}) \geq 2^{\Omega(n)}$. Further employing Lemma 10 we obtain the desired hardness bound.

Corollary 9. *BBhardness*$(T_{SC}) \geq 2^{\Omega(n)}$*, i.e. there exists a vector* $c \in \mathbb{R}^n$ *such that the smallest branch-and-bound tree that solves*

$$\max_{x \in T_{SC} \cap \{0,1\}^n} \langle c, x \rangle$$

has size at least $2^{\Omega(n)}$*.*

3.5 BB hardness for cross-polytope

In this section, we present in Proposition 3 a simple proof of BB hardness for the cross-polytope. As mentioned before, this result slightly improves on the result that can be directly obtained by applying Lemma 16 to the cross-polytope.

Next in this section we develop Proposition 4 that shows that there is a point in the cross polytope that is hard to separate using BB trees of small size. This allows us to use the machinery of Lemma 15 and a composition of the affine functions described in section 3.3 to connect the BB hardness of TSP to that of the cross-polytope, which we do in section 3.7.

The cross-polytope is defined as

$$P_n = \left\{ x \in [0,1]^n \,:\, \sum_{i \in J} x_i + \sum_{i \notin J} (1 - x_i) \geq \frac{1}{2} \quad \forall J \subseteq [n] \right\}.$$

Recall that the cross-polytope is integer-infeasible: every 0/1 point $\hat{x} \in \{0,1\}^n$ is cut off by the inequality given by the set $J = \{i \in [n] \,:\, \hat{x}_i = 0\}$.

Proposition 3. *Let* $\mathcal{T}$ *be a BB tree for* P_n *that certifies the integer-infeasibility of* P_n*. Then* $|\mathcal{T}| \geq 2^{n+1} - 1$ *(i.e. BBhardness*$(P_n) \geq 2^{n+1} - 1$*).*

Proof. In order to certify the integer-infeasibility of P_n, the atom of every leaf-node must be an empty set. We will verify that in order for the atom of a leaf v to be empty, no more than one integer point is allowed to satisfy the branching constraints C_v of v. This will complete the proof, since we then must have at least 2^n leaves.

Consider any leaf v of $\mathcal{T}$ such that two distinct integer points are feasible for its branching constraints. Then the average of these two points is a point in $\{0, 1, \frac{1}{2}\}^n$ with at least one component equal to $\frac{1}{2}$, which also satisfies the branching constraints. However, a point in $\{0, 1, \frac{1}{2}\}^n$ with at least one component equal to $\frac{1}{2}$ satisfies the constraints defining P_n. Thus the atom of the leaf v is non-empty. □

Corollary 10. *Let $F \subseteq \mathbb{R}^n$ be a face of P_n with dimension d. Then BBhardness$(F) \geq 2^{d+1} - 1$.*

Proof. Notice that F is a copy of P_d with $n - d$ components fixed to 0 or 1. Thus, there exists an appropriate embedding affine transformation f (Definition 5) such that $f(P_d) = F$. Also since $F \cap \mathbb{Z}^n = \emptyset$, we obtain that f, P_d and F satisfy all the conditions of Corollary 6. Thus, BBhardness$(F) \geq$ BBhardness$(P_d) \geq 2^{d+1} - 1$, where the last inequality follows from Proposition 3. □

Next we show that the point $\frac{1}{2}\mathbf{1}$ is hard to separate from P_n. For that we need a technical result that any halfspace that contains $\frac{1}{2}\mathbf{1}$ must also contain a face of $[0, 1]^n$ of dimension at least $\lfloor n/2 \rfloor$.

Lemma 18. *Consider any $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}$ such that $\langle \pi, \frac{1}{2}\mathbf{1} \rangle > \pi_0$. Let $G = \{x \in [0, 1]^n : \langle \pi, x \rangle > \pi_0\}$. There exists a face F of $[0, 1]^n$ of dimension at least $\lfloor n/2 \rfloor$ contained in G.*

Proof. By assumption we have $\pi_0 < \frac{1}{2} \sum_{i=1}^n \pi_i$. First consider the case where the vector π is non-negative. By renaming the coordinates we can further assume that $\pi_1 \geq \pi_2 \geq \ldots \geq \pi_n \geq 0$. Then the face $F = \{x \in [0, 1]^n : x_i = 1, \forall i \leq \lceil n/2 \rceil\}$ has the desired properties: it has dimension $n - \lceil n/2 \rceil = \lfloor n/2 \rfloor$, and any $\hat{x} \in F$ has

$$\langle \pi, \hat{x} \rangle \geq \sum_{i=1}^{\lceil n/2 \rceil} \pi_i \geq \frac{1}{2} \sum_{i=1}^{n} \pi_i > \pi_0,$$

where the second inequality follows from the ordering of the coordinates of π, and hence F is contained in G.

The case when π is not non-negative can be reduced to the above case by flipping coordinates. More precisely, let J be the set of coordinates i where $\pi_i < 0$, and consider the coordinate flipping operation (Definition 4) $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ that flips all coordinates in J. Notice that

$$f(G) = \left\{ x \in [0,1]^n \;:\; \sum_{i \in J} -\pi_i x_i + \sum_{i \notin J} \pi_i x_i \leq \pi_0 - \sum_{i \in J} \pi_i \right\},$$

and that defining the vector π' as $\pi'_i = -\pi_i$ for $i \in J$ and $\pi'_i = \pi_i$ for $i \notin J$ and $\pi'_0 := \pi_0 - \sum_{i \in J} \pi_i$ we still have $\langle \pi', \frac{1}{2}\mathbf{1} \rangle > \pi'_0$. Since π' is non-negative, the previous argument shows that $f(G)$ has a face F of $[0,1]^n$ of desired dimension, and hence $f^{-1}(F) = f(F)$ is a desired face of $[0,1]^n$ contained in G. □

Proposition 4. *For every n such that $\lfloor n/2 \rfloor > 1$, BBdepth $\left(\frac{1}{2}\mathbf{1}, P_n\right) \geq 2^{\lfloor n/2 \rfloor + 1} - 1$.*

Proof. For sake of contradiction suppose there exists a tree $\mathcal{T}$ of size less than $2^{\lfloor n/2 \rfloor + 1} - 1$ such that $\frac{1}{2}\mathbf{1} \notin \text{conv}(\mathcal{T}(P_n))$. By the hyperplane separation theorem, there exists $(\pi, \pi_0) \in \mathbb{R}^n \times \mathbb{R}$ such that $\langle \pi, \frac{1}{2}\mathbf{1} \rangle > \pi_0$ and $\langle \pi, x \rangle \leq \pi_0$ for all $x \in \text{conv}(\mathcal{T}(P_n))$. By Lemma 18, let F be a face of $[0,1]^n$ of dimension $\lfloor n/2 \rfloor$ contained in $\{x \in \mathbb{R}^n \mid \langle \pi, x \rangle > \pi_0\}$; notice that $P_n \cap F$ is a face of P_n of the same dimension. Since $\mathcal{T}(P_n) \subseteq \text{conv}(\mathcal{T}(P_n)) \subseteq \mathbb{R}^n \setminus \{x \in \mathbb{R}^n \mid \langle \pi, x \rangle > \pi_0\} \subseteq \mathbb{R}^n \setminus F$ and $\mathcal{T}(F) \subseteq F$, we have that $\mathcal{T}(P_n)$ and $\mathcal{T}(F)$ are disjoint and hence from Lemma 13 we get $\mathcal{T}(P_n \cap F) \subseteq \mathcal{T}(P_n) \cap \mathcal{T}(F) = \emptyset$, i.e., the atoms of the leaves of $\mathcal{T}$ applied to $P_n \cap F$ are all empty. Thus, $\mathcal{T}$ is a branch-and-bound tree to certify the integer-infeasibility of $P_n \cap F$ of size less than $2^{\lfloor n/2 \rfloor + 1} - 1$. However, this contradicts Corollary 10.

□

3.6 BB hardness for perturbed cross-polytope

We now show that exponential BB hardness for the cross-polytope persists even after adding Gaussian noise to the entries of the contraint matrix. This implies an exponential lower bound even for a "smoothed analysis" of general branch-and-bound.

We consider the cross-polytope P_n but where we add an independent gaussian noise $N(0, 1/20^2)$ with mean 0 and variance $1/20^2$ to each coefficient in the left-hand side of its defining inequalities, and replace the right-hand sides by approximately $\frac{n}{20}$ instead of the traditional $\frac{1}{2}$. More precisely, we consider the following random polytope Q:

$$Q := \left\{ x \in [0,1]^n \;:\; \sum_{i \in I} \left(1 + N\left(0, \tfrac{1}{20^2}\right)\right) x_i + \sum_{i \notin I} \left(1 - \left(1 + N\left(0, \tfrac{1}{20^2}\right)\right) x_i\right) \geq \frac{1.6n}{20}, \quad \forall I \subseteq [n] \right\}$$

where each occurrence of $N(0, \frac{1}{20^2})$ is independent.

Theorem 6. *With probability at least $1 - \frac{2}{e^{n/2}}$ the polytope Q is integer-infeasible and every BB tree proving its infeasibility has at least $2^{\Omega(n)}$ nodes.*

Recall that for independent gaussians $Y \sim N(\mu, \sigma^2)$ and $Y' \sim N(\mu', (\sigma')^2)$, their sum $Y + Y'$ is distributed as $N(\mu + \mu', \sigma^2 + (\sigma')^2)$, and for a centered gaussian $Y \sim N(0, \sigma^2)$ the scaled random variable αY is distributed as $N(0, \alpha^2\sigma^2)$ for all $\alpha \in \mathbb{R}$.

We need the following standard tail bound for the Normal distribution (see equation (2.10) of [60]).

Fact 1. *Let $X \sim N(0, \sigma^2)$ be a mean zero gaussian with variance σ^2. Then for every $p \in (0,1)$, with probability at least $1 - p$ we have $X \leq \sigma\sqrt{2\ln(1/p)}$, and with probability at least $1 - p$ we have $X \geq -\sigma\sqrt{2\ln(1/p)}$.*

Let $LHS_I(x)$ be the left-hand-side of the constraint of Q indexed by the set I evaluated at the point x.

Lemma 19. *With probability at least $1 - \frac{1}{e^{n/2}}$ the polytope Q is integer-infeasible.*

Proof. Fix a 0/1 point $x \in \{0,1\}^n$, and let $I \subseteq [n]$ be the set of coordinates i where $x_i = 0$. Let $I^c = [n] \setminus I$. Notice $LHS_I(x)$ is a gaussian random variable with mean 0 and variance $\frac{|I^c|}{20^2} \leq \frac{n}{20^2}$, and

so from Fact 1, with probability at least $1 - \frac{1}{e^{n/2}2^n}$ we have

$$LHS_I(x) \le \frac{\sqrt{n}}{20}\sqrt{2\ln(e^{n/2}2^n)} = \frac{\sqrt{n}}{20}\sqrt{(1+2\ln 2)n} < \frac{1.6n}{20},$$

i.e., the point x does not satisfy the inequality of Q indexed by I, and so does not belong to Q. Taking a union bound over all 2^n points $x \in \{0,1\}^n$, with probability at least $1 - \frac{1}{e^{n/2}}$ none of them belong to Q. This concludes the proof. □

Lemma 20. *With probability at least* $1 - \frac{1}{e^{n/2}}$ *the polytope* Q *contains all points* $\{0, \frac{1}{2}, 1\}^n$ *that have at least* $s = \frac{4n}{10}$ *coordinates with value* $\frac{1}{2}$*. (We call this set of points* Half_s*.)*

Proof. Consider $x \in \mathrm{Half}_s$. Fix $I \subseteq [n]$. Let $n_{\mathrm{half}} \ge s = \frac{4n}{10}$ be the number of coordinates of x with value $\frac{1}{2}$, n_{ones} be the number of coordinates with value 1, and let n_{diff} be the number of coordinates i where either $i \in I$ and $x_i = 1$, or $i \notin I$ and $x_i = 0$. We see that $LHS_I(x)$ distributed as

$$\begin{aligned} LHS_I(x) &=_d \frac{n_{\mathrm{half}}}{2} + n_{\mathrm{diff}} + \frac{1}{2}N(0, \tfrac{n_{\mathrm{half}}}{20^2}) + N(0, \tfrac{n_{\mathrm{ones}}}{20^2}) \\ &=_d \frac{n_{\mathrm{half}}}{2} + n_{\mathrm{diff}} + N(0, (\tfrac{n_{\mathrm{half}}}{4} + n_{\mathrm{ones}}) \cdot \tfrac{1}{20^2}), \end{aligned}$$

where again the occurrences of $N(0,\cdot)$ are independent. Since the last term is a gaussian with variance at most $\frac{n}{20^2}$, we get that with probability at least $1 - \frac{1}{e^{n/2}\cdot 2^n \cdot 3^n}$

$$LHS_I(x) \ge \frac{n_{\mathrm{half}}}{2} + n_{\mathrm{diff}} - \frac{1}{20}\sqrt{n}\sqrt{2\log(e^{n/2}\cdot 2^n \cdot 3^n)} \ge \frac{4n}{20} - \frac{2.4n}{20} = \frac{1.6n}{20},$$

that is, x satisfies the constraint of Q indexed by I.

Taking a union bound over all $x \in \mathrm{Half}_s$ and all subsets $I \subseteq [n]$, we see that all points in Half_s satisfy all constraints of Q with probability at least $1 - \frac{1}{e^{n/2}}$. This concludes the proof. □

Lemma 21. *Let* $F \subseteq \{0,1\}^n$ *be a set of 0/1 points. For any* k*, if* $|F| > \sum_{i \le k-1} \binom{n}{i}$*, then* $conv(F)$ *contains a point with at least* k *coordinates of value 1/2.*

Proof. By the Sauer-Shelah Lemma (Lemma 11.1 of [61]), there is a set of coordinates $J \subseteq [n]$ of size $|J| = k$ such that the points in F take all possible values in coordinates J, i.e., the projection F_J onto the coordinates J equals $\{0,1\}^k$. So the point $\frac{1}{2}\mathbf{1}$ belongs to $\text{conv}(F_J)$, which implies that $\text{conv}(F)$ has the desired point. □

Proof of Theorem 6. Let E be the event that both the bounds from Lemmas 19 and 20 hold. By a union bound, this event happens with probability at least $1 - \frac{2}{e^{n/2}}$. So it suffices to show that there is a constant $c > 0$ such that for every scenario in E, every BB tree proving the infeasibility of Q has at least 2^{cn} leaves.

In hindsight, again let $s = \frac{4n}{10}$ and set $c := 1 - h(\frac{s}{n})$, where h is the binary entropy function $h(p) := p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$. Notice that $c > 0$, since h is strictly increasing in the interval $[0, \frac{1}{2}]$ and hence $h(\frac{s}{n}) < h(\frac{1}{2}) = 1$.

Fix a scenario in the event E, so we know Q is integer-infeasible and $\text{Half}_s \subseteq Q$. Consider any tree $\mathcal{T}$ that proves integer-infeasibility of Q, and we claim that it has more than 2^{cn} leaves. By contradiction, suppose not. Then $\mathcal{T}$ has a leaf v whose branching constraints C_v are satisfied by at least $\frac{2^n}{2^{cn}} = 2^{n \cdot h(s/n)}$ 0/1 points (recall that each integer point satisfies all of the branching constraints of at least some leaf). But since $2^{n \cdot h(s/n)} > \sum_{i \le s-1} \binom{n}{i}$ (see e.g. Lemma 5 of [62]), by Lemma 21 we know that the convex set $\{x : C_v\}$ contains a point $\hat{x} \in [0,1]^n$ with at least s coordinates of value 1/2. Moreover, notice that $\hat{x}$ also belongs to $\text{conv}(\text{Half}_s)$, which is contained in Q. Hence $\hat{x} \in \{x : C_v\} \cap Q$, namely the atom of the leaf v. But this contradicts that this atom is empty (which is required since $\mathcal{T}$ proves integer infeasibility of Q). This concludes the proof. □

3.7 BB hardness for TSP

Again we use P_k to denote the cross-polytope in k dimensions.

Proposition 5. *Let f be any composition of the flipping (Defintion 4), embedding (Definition 5), and duplication (Definition 6) operations. Let $H \subseteq [0,1]^n$ be a polytope such that $f(P_k) \subseteq H$ and*

$f(\frac{1}{2}\mathbf{1}) \notin H_I$, *where* $k \leq n$. *Then,* $BBrank(H) \geq 2^{\lfloor k/2 \rfloor}$.

Proof. Notice that if f is a composition of the flipping, embedding, and duplication operations, then f is an integral affine transformation. Moreover, if P is integer-infeasible then $f(P)$ is also integer-infeasible. In particular, since P_k is integer-infeasible we have $(f(P_k))_I = \emptyset$, and hence $f(\frac{1}{2}\mathbf{1}) \notin (f(P_k))_I$. Now, Corollary 7 and Proposition 4 give us BBdepth $\left(f\left(\frac{1}{2}\mathbf{1}\right), f(P_k)\right) \geq$ BBdepth $\left(\frac{1}{2}\mathbf{1}, P_k\right) \geq 2^{\lfloor k/2 \rfloor + 1} - 1$. Finally, since $f(P_k) \subseteq H$, Corollary 5 implies that BBdepth$(f(\frac{1}{2}\mathbf{1}), H) \geq 2^{\lfloor k/2 \rfloor + 1} - 1$. This implies the desired result: BBrank$(H) \geq 2^{\lfloor k/2 \rfloor + 1} - 1 \geq 2^{\lfloor k/2 \rfloor}$. □

We next state a key result from the proof of Theorem 4.1 of [63] (see also [57]), that shows how we can apply Proposition 5 to obtain BB hardenss of the TSP polytope. Let T_{TSP_n} be the LP relaxation of the TSP polytope using subtour elimination constraints for n cities:

$$\begin{aligned} x(\delta(v)) &= 2 \quad \forall v \in V \\ x(\delta(W)) &\geq 2 \quad \forall W \subset V, W \neq \emptyset \\ 0 \leq x(e) &\leq 1 \quad \forall e \in E \end{aligned}$$

Proposition 6 (proof of Theorem 4.1 in [63])**.** *There exists a function f which is a composition of flipping, embedding, and duplication such that $f(P_{\lfloor n/8 \rfloor})$ is contained in T_{TSP_n} and $f(\frac{1}{2}\mathbf{1})$ does not belong to the integer hull of T_{TSP_n}.*

Then employing Proposition 5 we obtain BBrank$(T_{\text{TSP}_n}) \geq 2^{\frac{n}{16}-2}$, and again since BB hardness is at least the BB rank (Lemma 10) we obtain the desired hardness.

Corollary 11 (BB hardness for TSP)**.** *$BBhardness(T_{TSP_n}) \geq 2^{\frac{n}{16}-2}$, i.e, there is a vector $c \in \mathbb{R}^{n(n-1)/2}$ such that the smallest branch-and-bound tree that solves*

$$\max_{x \in T_{TSP_n} \cap \{0,1\}^{n(n-1)/2}} \langle c, x \rangle$$

has size at least $2^{\frac{n}{16}-2}$.

CHAPTER 4

THEORETICAL AND COMPUTATIONAL ANALYSIS OF STRONG BRANCHING

4.1 Introduction

Given an underlying LP solver, formally speaking, the *branch-and-bound algorithm* is well-defined by fixing two rules:

- Rule for selecting an open node to be branched on next and,
- Rule for deciding the variable to branch on.

It is natural to measure the efficiency of a branch-and-bound algorithm by the number of nodes (corresponding to number of LPs solved) in the tree, i.e., lesser the number of nodes, faster the algorithm.

Selecting an open node to branch on next (node selection rule). It is well established [2] that the *best-bound rule* (i.e., select node with the maximum LP optimal objective function value for a maximization-type MILP or select node with minimum LP optimal objective function value for a minimization-type MILP) for selecting the next node to branch on, leads to small branch-and-bound trees. The intuition behind this is the following: one cannot ignore the node with best-bound if one wants to solve the MILP. Thus, it is best to select to branch on it first. In the rest of the paper, we always assume to use the best-bound rule.

Deciding which variable to branch on (variable selection rule). Given that the rule for selecting the node to be branched on is well-understood, much of the research in the area of branch-and-bound algorithms has focused on the topic of deciding which variable to branch on – see for example [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. Most of the above work develops

various intricate greedy rules for determining the branching variable. A popular concept is that of *pseudocost branching*: the value of pseudocost (variable with largest pseudocost gets branched on) keeps a history of the success (in terms of improving dual bound) of the variables on which branching has already been done. Many of the papers cited above differ in how pseudocost is initialized and updated during the course of the branch-and-bound tree. Other successful methods like *hybrid branching* and *reliability branching* [32] are combinations of pseudocost branching and *full strong-branching*, that we discuss next.

The focus of this work is *full strong-branching* [30], henceforth referred to as strong-branching for simplicity. This rule works as follows: branching on all the current fractional variables is computed (i.e., the child nodes are solved for every choice of fractional variable) and improvement measured in the left and right child node. Branching is now done on the variable with the most 'combined improvement', where the combined improvement is computed as a 'score' function of the left and right improvement. Formally, let z be the optimal objective function value of the LP at a given node, and let z_j^0 and z_j^1 be the optimal objective function values of the LPs corresponding to the child nodes where the variable x_j is set to 0 and 1 respectively; we define $\Delta_j^+ := z - z_j^1$ and $\Delta_j^- := z - z_j^0$ (assuming the MILP's objective function is of maximizing-type). Note that $\Delta_j^+ = +\infty$ is the child node with x_j set to 0 is infeasible. Similarly for Δ_j^-. Two common score functions used are:

$$\text{score}_L(j) = (1-\mu) \cdot \min\{\Delta_j^-, \Delta_j^+\} + \mu \cdot \max\{\Delta_j^-, \Delta_j^+\}$$

for a constant $\mu \in [0,1]$, and

$$\text{score}_P(j) = \max(\Delta_j^+, \epsilon) \cdot \max(\Delta_j^-, \epsilon),$$

for a constant $\epsilon > 0$, where the first score is recommended in [31, 32] (the paper [32] recommends using $\mu = 1/6$) and the second score function is recommended in [64], where $\epsilon > 0$ is chosen close to 0 (for example, $\epsilon = 10^{-6}$) to break ties. We will refer to the first score function as the *linear score*

function and the second score function as the *product score function*. Finally, the variable selected to branch on belongs to the set:

$$\arg\max_j \{\text{score}(j)\}.$$

Empirically, strong-branching is well-known to produce significantly smaller branch-and-bound trees [32] compared to all other known techniques, but is extremely expensive to implement as one has to solve $2K$ LPs where K is the number of fractional variables for making just one branching decision. This experimentally observed fact is so well established in the literature that almost all recent methods to improve upon branching decisions are based on using *machine learning techniques to mimic strong-branching* that avoid solving the $2K$ LPs, see for example [65, 66, 67, 68, 69, 70, 71]. Finally, see [72] that describes a more sophisticated way to decide the branching variable based on left and right improvement rather than a static 'score' function.

4.1.1 Our contributions

As explained in the previous section, empirically it is well understood that strong-branching produces very small trees in comparison to other rules. However, to the best of our knowledge there is *no understanding of how good strong-branching is in absolute terms.* In particular, we would like to answer questions such as:

- How large is the tree produced by strong-branching in comparison to the smallest possible branch-and-bound tree for a given instance? Answering this question may lead us to finding better rules.
- A more refined line of questioning: Intuitively, we do not expect strong-branching to work well for all types of MILP models and instances. On the other hand, it may be possible that for some classes of MILPs strong-branching based branch-and-bound tree may be quite close to the smallest possible branch-and-bound tree. It would be, therefore, very useful to understand the performance of strong-branching vis-á-vis different classes of instances.

In this paper, we attempt an analysis of the performance of the strong-branching rule – both from a theoretical and a computational perspective, keeping in mind the above questions.

- Strong branching is provably good: We show that for the vertex cover problem, the strong-branching rule has several benefits.

  First, we show that strong branching can take advantage of problem structure. Nemhauser and Trotter [73] proved that one may fix variables that are integral in the optimal solution of the LP relaxation of vertex cover and still find an optimal solution to the IP. Note that in the branch-and-bound tree, every node corresponds to a sub-graph of the original vertex-cover instance, and thus, ideally we would like to continue to use the Nemhauser-Trotter property at each node. Instead of designing a specialized implementation of branch-and-bound algorithm (where we fix variables that have an integer value in the LP optimal solution at each node), our second result is to show that strong-branching naturally incorporates "fixing" the integral variables of the LP solution recursively and consistently thoughout the branch-and-bound tree.

  Next, we present a fixed parameter-type (FPT) result that uses the additive gap between the IP's and the LP's optimal objective function value to bound the size of the branch-and-bound tree using strong-branching.

  Finally, we construct an instance where strong-branching yields a branch-and-bound tree that is exponential-times smaller than a branch-and-bound tree generated using a very reasonable alternative variable selection rule.

- Strong branching is provably bad: We present a class of instances where the size of the strong-branching based branch-and-bound tree is exponentially larger than a special branch-and-bound tree that solves these instances. In fact, the result we prove is stronger – we show that if we only branch on variables that are fractional, then the size of the branch-and-bound tree is exponentially larger than the given special branch-and-bound tree to

solve these instances. This special branch-and-bound tree branches on variables that are integral in the optimal LP solutions at certain nodes.

- Computational evaluation of the size of strong-branching based branch-and-bound tree against the "optimal" branch-and-bound tree[1]: We first present a dynamic programming algorithm for generating the optimal branch-and-bound tree whose running time is $\text{poly}(\text{data}(\mathcal{I}))\cdot 3^{O(n)}$ where n is the number of binary variables. Then we conduct experiments on various types of instances like the lot-sizing problem and its variants, packing IPs, covering IPs, chance constrainted IPs, vertex cover, etc., to understand how much larger is the size of the strong-branching based branch-and-bound tree in comparison to the optimal branch-and-bound tree. The main take-away from these experiments is that for all these instances, the size of the strong-branching based branch-and-bound tree is within a factor of two of the size of the optimal branch-and-bound tree.

To the best of our knowledge, this is the first such study of this kind on strong-branching, that provides a better understanding of why strong-branching often performs so well in practice, and gives insight into when an instance may be challenging for strong-branching.

The rest of the paper is organized in the following fashion. In section 4.2 and section 4.3, we present the theoretical results. In section 4.4, we present the main computational results. In section 4.5 we present the details of the dynamic programming algorithm mentioned above.

4.2 Analysis of Strong Branching for Vertex Cover

There are simple MILPs that require exponential size branch-and-bound trees [12, 13, 19, 15, 18]. A common way to meaningfully analyze an algorithm with exponential worst-case performance

[1]We write "optimal" branch-and-bound tree with quotes, since the size of the optimal branch-and-bound tree depends not only on the branching decisions taken at each node, but also on the properties of the LP solver. We discuss this issue in detail in section 4.4

is to show its performance with respect to some parameter [74, 75]. Arguably the most well-studied problem in parameterized complexity is *vertex cover*.

Definition 7 (Vertex cover)**.** *The vertex cover problem over a graph $G = (V, E)$ can be expressed as the following integer program (IP)*

$$\begin{aligned} \textit{min} \quad & \sum_{v \in V} x_v \\ \textit{s.t.} \quad & x_u + x_v \geq 1, \quad uv \in E \\ & x_v \in \{0, 1\}, \quad v \in V \end{aligned}$$

Given an instance $\mathcal{I}$ of this IP, we let $L(\mathcal{I})$ denote its LP relaxation (i.e. when the variable constraints instead are $x_v \in [0, 1]$). We denote the optimal objective function value of an instance by $OPT(\mathcal{I})$ and the optimal objective function value of its LP relaxation by $OPT(L(\mathcal{I}))$. We denote its additive integrality gap $\Gamma(\mathcal{I}) := OPT(\mathcal{I}) - OPT(L(\mathcal{I}))$. For results pertaining to vertex cover, we use n to denote the number of vertices (i.e. $n := |V|$).

This section has two significant parts. In Section subsection 4.2.1, we present results that give insight into how strong branching takes advantage of the polyhedral structure of vertex cover instances defined above. In Section subsection 4.2.2, we present an upper bound, parameterized by Γ, on the size of branch-and-bound trees using strong branching variable selection when solving vertex cover instances; in Section subsubsection 4.2.2, we that strong branching can result in trees that are an exponential factor smaller than other reasonable variable selection rules.

Note that in this section (Section section 4.2), whenever we refer to strong-branching, we assume that it has been used in conjunction with the product score function [64] score_P, where $\epsilon = 0$ and we use the convention that $0 \cdot \infty = 0$. Finally, we refer to the *size* of a branch-and-bound tree to denote its number of nodes; we note that in any binary tree, the number of nodes is at most $2\ell + 1$, where ℓ is the number of leaves.

Let $\mathcal{T}_S(\mathcal{I})$ represent a branch-and-bound tree for solving instance $\mathcal{I}$ using strong-branching.

Unfortunately, the number of nodes in this tree depends not only on the node selection rule and variable selection rule, but also on the underlying LP solver as well. For example, a solver may report an integral solution at a given node and allow us to prune the node. Another solver might report a different optimal solution to the LP, which is not integral. Therefore, we will be careful to not refer to *the branch-and-bound tree generated by strong-branching*. Instead, we will use $\mathcal{T}_S(\mathcal{I})$ to represent *some branch-and-bound tree generated by strong-branching*.

4.2.1 Strong branching and persistency

Nemhauser and Trotter [73] prove the following property regarding the LP relaxation of vertex cover.

Fact 2 (Persistency; Theorem 2 of [73])**.** *Let $\mathcal{I}$ be an instance of vertex cover, $\hat{x}$ be an optimal solution of $L(\mathcal{I})$ and I be the set of variables on which $\hat{x}$ is integer (i.e. $I = \{j : \hat{x}_j \in \{0, 1\}\}$). Then, there exists an optimal solution $\hat{y}$ to $\mathcal{I}$ such that $\hat{y}$ agrees with $\hat{x}$ on all of its integer components (i.e. $\hat{y}_j = \hat{x}_j$ for all $j \in I$).*

One way to use this property is to use it as a pre-solve routine, i.e., fix variables that are integral in the optimal solution of the LP relaxation, and then work with the sub-graph induced by the vertices with value $\frac{1}{2}$ in the optimal solution of the LP. (The extreme points of the LP relaxation of the vertex cover problem are half integral [76].) However, note that in the branch-and-bound tree, every node corresponds to a sub-graph of the original vertex-cover instance, and thus, ideally we would like to continue to use the Nemhauser-Trotter property at each node. Instead of designing a specialized implementation of branch-and-bound algorithm (where we fix variables that have an integer value in the LP optimal solution at each node), we show that strong-branching naturally incorporates "fixing" the integral variables of the optimal solution of LP at each node recursively throughout the branch-and-bound tree. In fact, it does even better in the following sense: due to dual degeneracy, there may be alternative linear programming optimal

solutions with different corresponding sets of variables being integral. Strong branching "avoids branching" on all the variables that are integral in any of the alternative optimal LP solutions, i.e., strong-branching is not fooled by the LP solver.

In order to present our results, we need to define the notion of *maximal set of integer variables* and present some properties regarding this set of variables.

Fact 3 (Lemma 1 in [77])**.** *Consider an instance of vertex cover and its LP relaxation. Let* x^1, x^2 *be two optimal solutions to this LP relaxation and let* $I^1, I^2 \subseteq [n]$ *be the indices of the integer valued variables in* x^1, x^2 *respectively. Then, there exists an optimal solution of the LP relaxation,* $\hat{x}$*, such that the set of integer valued variables in* $\hat{x}$ *is* $I = I^1 \cup I^2$*.*

Based on the above fact we obtain the following observation: Given a vertex cover instance, all optimal solutions of the LP relaxation that have a maximal number of integral coordinates actually have the same set $I \subseteq [n]$ of integral coordinates. Given an instance of vertex cover $\mathcal{I}$, we call this subset the *maximal set of integer variables* and denote it $I(\mathcal{I})$. Fact 2 then implies that there exists an optimal solution to the vertex-cover instance where the *maximal set of integer variables* are fixed to integer values from the corresponding values of an maximal optimal solution of the LP relaxation.

As discussed before, in a branch-and-bound tree, each node corresponds to a vertex cover instance on a sub-graph. Therefore, we can define the notion of maximal set of integer variables at a given node N, which we denote as $I(\mathcal{I}, N)$. Given an instance of vertex cover $\mathcal{I}$, we refer to $\mathcal{TP}(\mathcal{I})$ as a partial branch-and-bound tree for $\mathcal{I}$ if all the nodes of $\mathcal{TP}(\mathcal{I})$ cannot be pruned. For such a partial branch-and-bound tree, we use $\mathcal{TP}^B(\mathcal{I})$ to refer to the dual bound that one can infer from the partial tree.

The strength of strong-branching with regards to the maximal set of integer variables at a node N is explained by the next result: Let j belong to the maximal set of integer variables at node N. If the LP solver returns an optimal solution with x_j integral, then clearly we do not branch on this variable. However, if the LP solver returns an optimal solution where x_j is fractional and we

decide on branching on this variable based on strong-branching, then it must be that we have "nearly solved" the instance, i.e., the dual bound that can be inferred from the partial tree must be equal to the optimal objective function value of the instance. Formally we have the following:

Theorem 7. *Let $\mathcal{I}$ be any instance of vertex cover. Assume we break ties within the best-bound rule for node selection by selecting a node with the largest depth. Consider a partial tree $\mathcal{TP}_S$ generated by strong-branching with the above version of best-bound node selection rule. Let N be a node of this tree that is not pruned. If $j \in I(\mathcal{I}, N)$ and we decide to branch on variable x_j at node N (using strong-branching), then*

1. *$\mathcal{TP}_S^B(\mathcal{I}) = OPT(\mathcal{I})$.*
2. *After branching on x_j,in at most $\mathcal{O}(n)$ further branchings the algorithm returns an integral optimal solution.*

We will require two preliminary results for the proof of Theorem 7. Throughout this section, we use $N_{j,0}$ to denote the child node of N that results from the branch $x_j = 0$; we use $N_{j,1}$ similarly.

Lemma 22. *Let N be a node of $\mathcal{T}_S(\mathcal{I})$ with optimal objective value less than $OPT(\mathcal{I})$. Then strong-branching will branch on $v \notin I(\mathcal{I}, N)$ at node N and $N_{v,0}, N_{v,1}$ have optimal value at least $\frac{1}{2}$ more than that of N.*

Proof. Since N has optimal objective value less than $\mathrm{OPT}(\mathcal{I})$, the LP relaxation at N must not have an optimal solution that is integer. Note that if at N we branch on x_u where $u \in I(\mathcal{I}, N)$, it must hold that, at least one of $N_{u,0}$ or $N_{u,1}$ have optimal value the same as N. Therefore, $\mathrm{score}_P(u) = 0$. We now show that there exists $v \in [n]$ such that $\mathrm{score}_P(v) > 0$. If there is no optimal solution to N that is integer, then $I(\mathcal{I}, N) \neq [n]$ by Fact 3. Therefore, there exists a $v \notin I(\mathcal{I}, N)$ such that $x_v = \frac{1}{2}$ in every optimal solution to N. It follows that fixing $x_v \in \{0, 1\}$ must result in a feasible solution to N that has strictly greater value, and so branching on x_v at N leads to child nodes $N_{v,0}$ and $N_{v,1}$ where both have optimal value more than that of N. Further, since $N_{v,0}$ and $N_{v,1}$ have

objective value strictly more than N and all basic feasible solutions are half-integral, it holds that $N_{v,0}$ and $N_{v,1}$ have objective value at least $\frac{1}{2}$ more than N. □

Lemma 23. *Let N be a node of $\mathcal{T}_S(\mathcal{I})$ with an optimal LP solution that is integer. Then the sub-tree rooted at N will have size $\mathcal{O}(n)$, where it finds an integer optimal solution.*

Proof. Let $y \in \{0,1\}^n$ be an optimal solution to the LP of node N. If the LP solver returns an integer solution, we are done. Suppose not, and suppose we branch on some variable v, and without loss of generality let $y_v = 0$. Then, of course $N_{v,0}$ will have y as an optimal solution and therefore have value $\text{OPT}(\mathcal{I})$. If $N_{v,1}$ has value greater than $\text{OPT}(\mathcal{I})$, this node gets pruned by bound and we continue by branching on $N_{v,0}$ since it is now the open node with largest depth. Suppose instead $N_{v,1}$ has value $\text{OPT}(\mathcal{I})$. Then, we argue in the next paragraph that there is an optimal solution to the LP corresponding to N that is integer feasible, denote z, with $z_v = 1$; in this case we can break the tie between $N_{v,0}, N_{v,1}$ arbitrarily. Then, for every node N' in the sub-tree rooted at N, either one child of N' has value greater than $\text{OPT}(\mathcal{I})$ and is pruned by bound, or both children of N' have optimal solutions that are integer. In the second case, only one of these two children will ever be branched on, since we break ties by branching on the node with largest depth, and therefore will find an integer solution before revisiting a shallower node. The result of the lemma follows.

Here we argue that if $N_{v,1}$ has a LP optimal objective function value $\text{OPT}(\mathcal{I})$, then there is an optimal solution to the LP corresponding to N that is integral, denote z, with $z_v = 1$. Let x' denote any optimal solution of $N_{v,1}$ and observe that x', y are both optimal solutions to N. It follows from the proof of Lemma 1 in [77] that: given two half-integral optimal solutions x', y,

there exists an optimal solution z constructed as follows:

$$z_j = \begin{cases} 1 & \text{if } x'_j = 1 \text{ or } x'_j = \frac{1}{2}, y_j = 1 \\ 0 & \text{if } x'_j = 0 \text{ or } x'_j = \frac{1}{2}, y_j = 0 \\ \frac{1}{2} & \text{if } x'_j = y_j = \frac{1}{2} \end{cases}$$

In particular, [77] shows that that z constructed as above is feasible and satisfies, $\sum_j z_j = \sum_j x'_j = \sum_j y_j$. Clearly $z_v = x'_v = 1$ and since $y \in \{0,1\}^n$, it follows that $z \in \{0,1\}^n$ and is an optimal solution of the LP corresponding to N. □

Proof of Theorem 7. We begin by proving Property 1. Suppose for sake of contradiction, there exists an open node N' in $\mathcal{TP}^B_S(\mathcal{I})$ with optimal objective value less than OPT($\mathcal{I}$). It follows from the best-bound rule and Lemma 22 that strong-branching will choose to branch on j' at N' with $j' \notin I(\mathcal{I}, N')$. Thus j' and N' are not the same as j and N, giving the desired contradiction.

Property 2 follows directly from Lemma 23 and the fact that $I(\mathcal{I}, N) = [n]$, since otherwise strong-branching would branch on some $j \notin I(\mathcal{I}, N)$. □

The above result shows that we "almost never" branch on maximal set of integer variables at a node if we use strong-branching. However, after branching on a variable that is not in the maximal set of integer variables, what happens to the set of maximal set of integer variables at the child nodes? A very favorable property would be if the maximal set of integer variables of the parent node is inherited by the child nodes. Otherwise, while we may not branch on x_j where $j \in I(\mathcal{I}, N)$ at node N, but we may end up branching on j for a child N' of N — in other words, we are then not really "fixing" variable x_j. As it turns out the above bad scenario does not occur. Formally we have the following:

Theorem 8. *Let $\mathcal{I}$ be an instance of vertex cover. Consider any internal node N of $\mathcal{T}_S(\mathcal{I})$ and let N' be a child node of N that results from branching on x_v where $v \notin I(\mathcal{I}, N)$. Then, $I(\mathcal{I}, N) \subset I(\mathcal{I}, N')$.*

Proof. Throughout this proof, we use $N_{j,0}$ to denote the child node of N that results from the branch $x_j = 0$; we use $N_{j,1}$ similarly. We use $\delta(S)$ to denote the neighbors of any subset of vertices $S \subseteq V$. Also, throughout the proof, let x^* be an optimal solution to N with maximal set of integer components (i.e. $x^*_j \in \{0, 1\}$ for all $j \in I(\mathcal{I}, N)$).

We will first consider the child $N_{v,1}$. Consider the solution y to $N_{v,1}$ constructed from x^* as follows: y takes the same values as x^*, but $y_v = 1$ instead of $\frac{1}{2}$. Note that this is feasible for $N_{v,1}$. Since $v \notin I(\mathcal{I}, N)$ we have that $x_v = \frac{1}{2}$ in every optimal solution to N. In other words, optimal objective function value of $N_{v,1}$ is at least $\frac{1}{2}$ more than that of N. Also, $\langle 1, y\rangle = \langle 1, x^*\rangle + \frac{1}{2}$, and so, y is an optimal solution to $N_{v,1}$. So $I(\mathcal{I}, N) \subset I(\mathcal{I}, N')$ follows in the case of $N' = N_{v,1}$.

We will now consider the child $N_{v,0}$. Let $V^*_1 = \{u \in V : x^*_u = 1\}$ and V^*_0 be defined similarly. We will need the following technical result later in the proof.

Lemma 24. *$|\delta(S) \cap V^*_0| \geq |S|$ for all $S \subseteq V^*_1$.*

Proof. Assume, for the sake of contradiction, there is a subset $S \subseteq V^*_1$ such that $|\delta(S) \cap V^*_0| < |S|$, then it must hold that x^* is not an optimal solution. This is because, we can set all of S and $\delta(S) \cap V^*_0$ to be $\frac{1}{2}$. This remains feasible, since we are only decreasing the value of the vertices in S, but all of the vertices with value 0 adjacent to these vertices are also being raised to $\frac{1}{2}$. It also decreases the cost, since $\frac{1}{2}(|\delta(S) \cap V^*_0| + |S|) < |S|$. □

Let x' be any optimal solution to LP corresponding to $N_{v,0}$. Let $\Phi = \{u \in V^*_1 : x'_u \neq 1\}$. We will construct a feasible solution z such that $z_u = 1$ for all $u \in V^*_1$ with cost no more than that of x'. Note that since $\delta(V^*_0) \subseteq V^*_1$, this implies the existence of an optimal solution to $N_{v,0}$ with the same integer variables as x^* (since any optimal solution with all variables in V^*_1 set to 1 will have all variables in V^*_0 set to 0).

We begin by constructing an intermediate solution y. Notice Φ can be partitioned into $\Phi^0 = \{u \in \Phi : x'_u = 0\}$ and $\Phi^{\frac{1}{2}} = \{u \in \Phi : x'_u = \frac{1}{2}\}$. Let y be defined by setting the variables in Φ^0 to $\frac{1}{2}$ and setting the variables in $\delta(\Phi^0) \cap V^*_0$ to $\frac{1}{2}$ (note that $x'_u = 1$ for all $u \in \delta(\Phi^0) \cap V^*_0$). This maintains

feasibility, since the variables with decreasing value are a subset of V_0^*, which are only adjacent to vertices in V_1^* which now all have value at least $\frac{1}{2}$ in y. This also maintains optimality, since the cost increases by $\frac{1}{2}|\Phi^0|$ and decreases by $\frac{1}{2}|\delta(\Phi^0) \cap V_0^*|$, and by Lemma 24, $|\delta(\Phi^0) \cap V_0^*| \geq |\Phi^0|$. We now construct z similarly, from this intermediate solution y. Notice now, that all variables in Φ have value $\frac{1}{2}$ in y. We construct z by setting all of these variables to 1 and setting all variables in $\delta(\Phi) \cap V_0^*$ to 0 (note that $y_u \geq \frac{1}{2}$ for all $u \in \delta(\Phi) \cap V_0^*$). This maintains feasibility, since the variables with decreasing value are a subset of V_0^*, which are adjacent only to vertices in V_1^*, which all have value at least 1 in z. This also maintains optimality, since the cost increases by $\frac{1}{2}|\Phi|$ and decreases by at least $\frac{1}{2}|\delta(\Phi) \cap V_0^*|$, and by Lemma 24, $|\delta(\Phi) \cap V_0^*| \geq |\Phi|$. Finally, we note that z maintains feasibility for $N_{v,0}$, since $x'_v = 0$ and therefore $x'_u = 1$ for all $u \in \delta(v)$; since $\delta(v) \cap \Phi = \emptyset$, all vertices v and $\delta(v)$ keep their value in z. This concludes the proof. □

Therefore, using Theorem 7 and Theorem 8 together, we can conclude that when using strong-branching, we are essentially repeatedly using Nemhauser-Trotter property recursively and consistenly through-out the branch-and-bound tree: If j is in the maximal set of integer variables at node N, it continues to remain in the maximal set of integer variables for all the child nodes of N; and we do not branch on such a variable x_j at node N or any of its children unless the instance is essentially solved (i.e., dual bound inferred from the tree equals the values of the IP).

Here we present an example to illustrate that there are variable selection rules for which the property described in Theorem 8 does not hold. See the instance $\mathcal{I}^*$ shown in Figure Figure 4.1. Consider the following partial branch-and-bound tree, letting N denote the root node. Observe that there is an optimal LP solution that sets $x_a = x_c = 1$ and $x_b = x_d = 0$; therefore, $I(\mathcal{I}^*, N) = \{a, b, c, d\}$. However, suppose that an adversarial LP solver returns the optimal basic feasible solution $x_a = x_b = x_c = x_d = \frac{1}{2}$. Suppose we branch on x_b and consider the sub-problem resulting from $x_b = 1$, which we denote N'. The unique optimal solution to N' sets $x_b = 1$ and $x_a = x_c = x_d = \frac{1}{2}$; therefore, $I(\mathcal{I}^*, N') = \{b\} \subset I(\mathcal{I}^*, N)$.

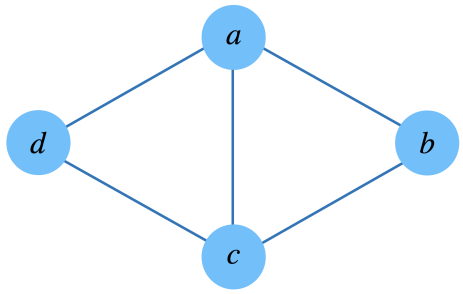

Figure 4.1: Instance $\mathcal{I}^*$: example illustrating that the property of Theorem 8 is not true for every variable selection rule.

4.2.2 Upper bound on the size of branch-and-bound tree using strong-branching

We show an upper bound on strong-branching for vertex cover parameterized by its additive integrality gap $\Gamma(\mathcal{I})$. We note that the result of Theorem 9 matches the guarantee of the classic parameterized-complexity algorithm that uses bounded search trees tailored to this problem; see Theorem 3.8 of [75]. One can view the result of Theorem 9 as indicating that full strong branching is "automating" the standard FPT result (Theorem 3.8 of [75]) via generic integer programming techniques (i.e. no particularly instance-specific knowledge is used).

Theorem 9. *Let $\mathcal{I}$ be any instance of vertex cover. Assume we break ties within the best-bound rule for node selection rule by selecting a node with the largest depth. Let $\mathcal{T}_S(\mathcal{I})$ be some branch-and-bound tree generated by strong-branching (with product scoring rule) with the above version of best-bound node selection rule that solves $\mathcal{I}$. Then independent of the underlying LP solver used,*

$$|\mathcal{T}_S(\mathcal{I})| \leq 2^{2\Gamma(\mathcal{I})+2} + \mathcal{O}(n).$$

Proof. Fix any branch-and-bound tree (using strong-branching) for $\mathcal{I}$. Let N be a node at depth $2\Gamma(\mathcal{I}) + 1$. Suppose that no ancestor of N had an optimal LP solution that was integer; it follows from Lemma 22 that N has optimal objective value at least $\text{OPT}(\mathcal{I}) + \frac{1}{2}$. Denote the set of such nodes $\mathcal{N}_{\text{no ancestor}}$ and note that these nodes are pruned by bound. Observe all other nodes (if any)

at depth $2\Gamma(\mathcal{I}) + 1$ do have such an ancestor. Let $\mathcal{N}_{\text{integer}}$ denote the set of nodes at depth $\leq 2\Gamma(\mathcal{I})$ that have an optimal solution that is integer, but none of its ancestors have an optimal solution that is integer. Finally, observe that $|\mathcal{N}_{\text{no ancestor}}| + |\mathcal{N}_{\text{integer}}| \leq 2^{2\Gamma(\mathcal{I})+1}$. We conclude the proof by observing that Lemma 23 gives: if N is the node in $\mathcal{N}_{\text{integer}}$ with largest depth, then the sub-tree rooted at N has size $\mathcal{O}(n)$ where it finds an integer solution with value $\text{OPT}(\mathcal{I})$. Therefore, no other nodes in $\mathcal{N}_{\text{integer}}$ will be branched on. Since the number of leaves in this tree is at most $2^{2\Gamma(\mathcal{I})+1} + \mathcal{O}(n)$, the result of the theorem follows. □

Intuitively, Theorem 9 follows from the observation that strong branching will never branch on a variable in $I(\mathcal{I}, N)$ since these variables have score 0, while variables outside of this set have score at least $\frac{1}{2}$, which is formalized by Lemma 22 above.

Moreover, it is impossible to find another branch-and-bound rule that has a much better upper bound, so in this sense strong-branching is in the worst-case almost optimal for vertex cover parametrized by integrality gap. This is because of the following bad example.

Remark 1. *There is an instance $\mathcal{I}$ of vertex cover such that any branch-and-bound tree that solves $\mathcal{I}$ has size $2^{2\Gamma(\mathcal{I})+1} - 1$.*

This is the instance of m disjoint triangles presented in [17]. Note that the smallest vertex cover in this instance has value $2m$ while the optimal solution to the LP relaxation has value $\frac{3}{2}m$, therefore $\Gamma(\mathcal{I}) = \frac{1}{2}m$. It follows from the discussion in [17], that all branch-and-bound trees for this instance have 2^m leaves, i.e., at least $2^{m+1} - 1$ nodes.

Superiority of strong-branching

There are very few papers that give upper bounds on sizes of branch-and-bound tree (when we use 0-1 branching) [10, 11]. These papers show that certain class of IPs with random data can be solved using polynomial-size branch-and-tree with high probability. However, these results do not depend on the variable selection rule used. Theorem 9 above is the first result of its kind

that we are aware of, which uses a specific variable selection rule to prove upper bounds on size of branch-and-bound tree. To further highlight the importance of strong-branching in obtaining this upper bound result, we next show that if we do not use the strong-branching rule and we have an "adversarial" LP solver, then we may need exponentially larger trees to solve the instance.

In particular, we next demonstrate the superiority of strong-branching by comparing it with a lexicographic variable selection rule for the vertex cover problem when the LP solver is adversarial, i.e. the LP solver always gives the most fractional extreme point solution. The instance in consideration vertex cover on the *lollipop graph* $L_{3,p}$, where p is an even integer (defined and indexed as in Figure Figure 4.2).

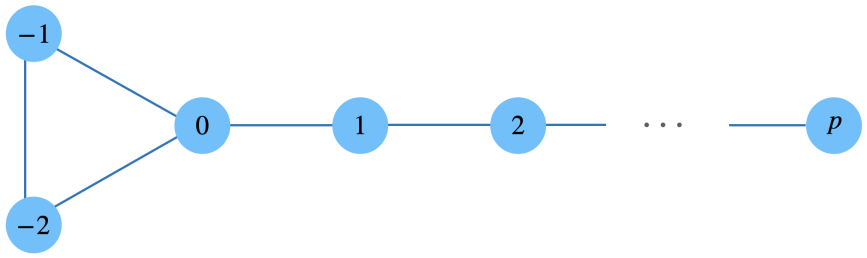

Figure 4.2: Lollipop graph $L_{3,p}$.

Remark 2. *Consider an LP solver with the following property: Among all optimal extreme point solutions, it reports an optimal extreme point with maximal number of fractional components. Let $\mathcal{I}$ be the vertex cover problem on the lollipop graph $L_{3,p}$, where p is an even integer. Let $\mathcal{T}_S(\mathcal{I})$ be some branch-and-bound tree that solves instance $\mathcal{I}$ obtained using the strong-branching rule and $\mathcal{T}_L(\mathcal{I})$ be some branch-and-bound tree that solves instance $\mathcal{I}$ obtained using the following lexicographic rule: branch on the fractional variable with largest index.*

Then, $\Gamma(\mathcal{I}) = \frac{1}{2}$, and $|\mathcal{T}_L(\mathcal{I})| \geq 2^{\Omega(n)}$, while $|\mathcal{T}_S(\mathcal{I})| \leq O(n)$.

Proof. First observe $n = p + 3$. The fact that $\mathrm{OPT}(L(\mathcal{I})) = \frac{p+3}{2}$ follows directly from the proof of Theorem 1 in [78]. Furthermore, it is easy to see that setting $x_{-2} = x_{-1} = 1$, $x_0 = 0$, $x_j = 1$ for all odd $j \in [p]$ and $x_j = 0$ for all even $j \in [p]$ results in a feasible solution to $\mathcal{I}$; therefore,

$\text{OPT}(\mathcal{I}) = \frac{p}{2} + 2 = \text{OPT}(L(\mathcal{I})) + \frac{1}{2}$, and so $\Gamma = \frac{1}{2}$. The lower bound on the size of the tree using the lexicographic rule, $|\mathcal{T}_L(\mathcal{I})| \geq 2^{\Omega(n)}$, follows directly from Theorem 1 of [78]. Finally, the upper bound on the strong branching tree $|\mathcal{T}_S(\mathcal{I})| \leq O(n)$ follows directly from Theorem 9. □

4.3 Strong branching does not work well for some instances

Next we present a negative result regarding strong-branching, showing that strong-branching based branch-and-bound tree can have an exponential times as many nodes as compared to number of nodes in an alternative tree. In fact, the example shows something even stronger: any tree that branches only on variables fractional in the current nodes optimal solution will have exponential size, while an alternative tree has linear size.

We begin by showing a seemingly surprising result about the existence of an extended formulation for any binary IP, that leads to a linear size branch-and-bound tree.

Proposition 7. *For any integer program $\mathcal{I}$ with n binary variables, there is an equivalent integer program that uses an extended formulation of the feasible region of $\mathcal{I}$ with $2n$ binary variables, which we refer to as $BDG(\mathcal{I})$, that has the following property: there exists a branch and bound tree $\mathcal{T}^*(BDG(\mathcal{I}))$ that solves the instance $BDG(\mathcal{I})$ and $|\mathcal{T}^*(BDG(\mathcal{I}))| \leq 4n + 1$.*

Proof. Consider the instance $\mathcal{I}$:

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & x \in P,\ x \in \{0,1\}^n. \end{aligned} \tag{$\mathcal{I}$}$$

where $P \subseteq [0,1]^n$ is a polytope. In [79], Bodur, Dash and Gunluk construct the extended formulation $Q \subseteq [0,1]^{2n}$ of P as follows. For every vertex x of P, construct a vertex (x, y) of Q

where

$$y_i = \begin{cases} 1 & \text{if } x_i \in \{0,1\} \\ 0 & \text{if } x_i \in (0,1) \end{cases}.$$

Define Q to be the convex hull of these vertices, we call this the *BDG extended formulation for P*. We construct the equivalent IP, $BDG(\mathcal{I})$, as follows:

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & (x,y) \in Q,\ x \in \{0,1\}^n,\ y \in \{0,1\}^n. \end{aligned} \tag{BDG($\mathcal{I}$)}$$

For any IP $\mathcal{I}$, there exists a branch-and-bound tree with at most $4n + 1$ nodes that solves $BDG(\mathcal{I})$. However, this tree does not remove the current LP optimal fractional point when branching. See Figure Figure 4.3.

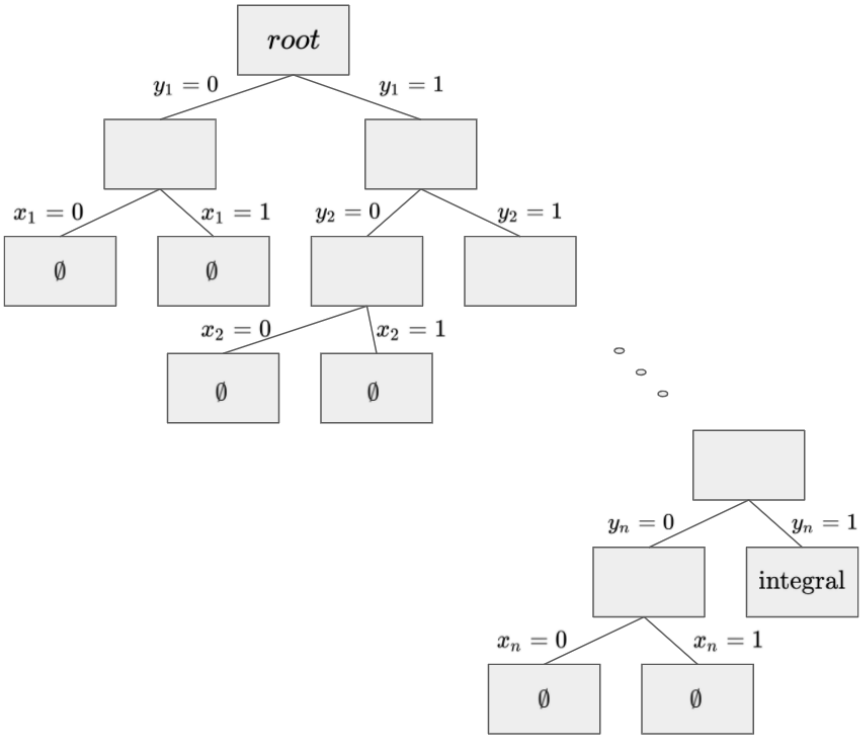

Figure 4.3: Branch-and-bound on BDG extended formulation

This follows since, by the definition of Q, all of its vertices that have $y_j = 0$ must have $x_j \in (0, 1)$. Therefore, the branch that has $y_j = 0$ and $x_j = 0$ (and similarly $x_j = 1$) must be empty. Also, note that the branch that has $y_1 = 1, ..., y_n = 1$ must be integral. □

The extended formulation corresponding to $BDG(\mathcal{I})$ used in Proposition 7 was first introduced in [79] to show that every binary integer program has an extended formulation with split rank of 1. We also remark here that Proposition 7 does not imply that the decision version of binary IPs is in *co-NP*. This is because the $BDG(\mathcal{I})$ formulation may be of exponential-size in comparison to the original formulation.

In Corollary 12 below, we take the cross-polytope [19] and apply the extended formulation of Proposition 7 to obtain an example where strong-branching based branch-and-bound tree can have an exponential times as many nodes as compared to number of nodes in an alternative tree.

Corollary 12. *There exists an instance $\mathcal{I}^*$ with $2n$ binary variables, such that the following holds: Let $\mathcal{T}(\mathcal{I}^*)$ be any tree that solves $\mathcal{I}^*$ satisfying the following property: if x is the optimal solution to an internal node N of $\mathcal{T}(\mathcal{I}^*)$, then the variable j branched on at N must be such that $x_j \in (0, 1)$. Then, $|\mathcal{T}(\mathcal{I}^*)| \geq 2^{n+1} - 1$. In particular, if $\mathcal{T}_S(\mathcal{I}^*)$ is a branch-and-bound tree generated using strong-branching that solve $\mathcal{I}^*$, then $|\mathcal{T}_S(\mathcal{I}^*)| \geq 2^{n+1} - 1$. On the other hand, there exists a tree $\mathcal{T}^*$ that solves $\mathcal{I}^*$ such that $|\mathcal{T}^*(\mathcal{I}^*)| \leq 4n + 1$.*

Proof. Note that the y variables are not fractional in any vertex of Q. So when the tree of Figure Figure 4.3 branches on a y variable, it does not remove the current LP optimal fractional point (because it does not remove any vertex of Q).

Now suppose we restrict ourselves to branching on a variable that does remove the current optimal point. Such a tree would only branch on x variables (i.e. the original variables). Then, if P is the n-dimensional cross polytope [19], and Q is its BDG extended formulation, we know that branching on only the x variables will require a tree of size at least $2^{n+1} - 1$, as shown in [19].

The final statement follows from Proposition 7. □

Typically, when implementing a branch-and-bound algorithm, one might be inclined to restrict the algorithm to branch on variables that are fractional in the current optimal solution. Therefore the above example is counter-intuitive in that it shows there can be a significant separation between branch-and-bound trees that are restricted to branch on variables that are fractional in the current optimal solution and branch-and-bound trees that are allowed to branch on integer valued variables. To our knowledge, this is the first such explicit example in the literature.

4.4 Computational results

In the previous section, we have shown that strong-branching works well for vertex cover and on the other hand, strong-branching can sometimes produce exponentially larger trees than alternative trees to solve an instance. However, in general it seems very difficult to analyze strong-branching for general MILPs on a case-to-case basis. Moreover, we would really like to answer the question: how good is the strong-branching based branch-and-bound tree in comparison to the optimal tree? In this section we try to shed light on this question using computational experiments.

Optimal branch-and-bound tree. We begin with a discussion of an "optimal branch-and-bound tree" for a given instance.

First note that it is clear that the optimal branch-and bound tree uses the worst bound rule for node selection [2]. Moreover, we will consider branching on all variables at a given node, whether it is integral or not in the optimal solution of the LP relaxation. Since we are using the worst bound rule for node selection, we only branch on nodes whose objective function value is at least as good as that of the MILP optimal objective function value.

Now consider a node whose LP optimal objective function value is equal to that of the MILP optimal objective function value. There are two possible scenarios:

- The optimal face of this LP is integral, i.e., all the vertices of the optimal face are integer

feasible and therefore the LP solver (like simplex) is guaranteed to find an integral solution and prune the node.

- At least one vertex of the optimal face is fractional. In this case, depending on whether the LP solver returns an integral vertex or not, we are able to prune the node or not. Moreover, if we arrive at a fractional vertex, then depending on properties of other nodes and how we break ties for node selection among nodes with same objective function value, we may or may not end up branching on this node.

In other words, in the second case, the size of tree may depend on the type of optimal vertex reported by the LP solver. In order to simplify our analysis and to remove all ambiguity regarding the definition of the optimal branch-and-bound tree, we make the following assumption for results presented in this section.

Assumption 1. *We assume that if there exists an optimal solution to the LP relaxation at a given node that is integral, then the LP solver finds it.*

Observation 1. *When using an LP solver which satisfies Assumption 1, a branch-and-bound tree using the worst bound rule never branches on a node whose optimal objective function value is equal to that of the IP solver.*

Proof of Observation 1 is presented in subsection 4.5.1. The above observation clearly makes the notion of "optimal branch-and-bound tree" for a given instance well-defined.

Dynamic programming algorithm for finding the optimal branch-and-bound tree. In subsection 4.5.2 we present a dynamic programming (DP) algorithm that computes an optimal branch-and-bound tree for a given instance under Assumption 1 for the LP solver.

Theorem 10. *Under Assumption 1, there exists an algorithm with running time* $\text{poly}(data) \cdot 3^{O(n)}$ *time to compute an optimal branch-and-bound tree for any binary MILP instance* $\mathcal{I}$ *defined on* n *binary variables.*

subsection 4.5.2 also presents some computational enhancements (like exploiting parallel computing) to improve the wall clock run time of the DP algorithm.

Computationally comparison of various variable selection rules against the optimal branch-and-bound tree. We conduct the first study comparing branch-and-bound trees employing different variable selection rules to the optimal branch-and-bound tree. Detailed results are presented in [33].

We consider the following variable selection rules: strong-branching with linear score function, strong-branching with product score function, most infeasible, and random. We evaluate the performance of these rules on a wide range of problems: general packing and covering IPs (packing-type, covering-type, and mixed packing and cover instances), lot-sizing and variants, vertex cover, chance constraint programming (CCP) models for multi-period power planning and portfolio optimization, and stable set on a bipartite graph with knapsack sides constraint (100 instances for each model). Note that all of these instances have up to 20 binary variables since we are unable to run the DP algorithm of Theorem 10 for larger instances. See [33] for a detailed description of all these instances.

We present a discussion of all our results (together with tables and explanatory figures) in [33]. Here are some of our notable findings on sizes of branch-and-bound tree trees:

- Random consistently performs the worst.
- Strong branching *always* performs the best.
- While the performance of two variants of strong-branching is comparable on all problems considered in this study, strong branching with product score function (SB-P) dominates over strong branching with linear score function (SB-L) on 8 out of 10 problems, although by a small margin.
- The geometric mean of branch-and-bound tree size using strong-branching remains less

than twice the size of optimal branch-and-bound tree for all problems considered in this study. This is not the case for all branching rules: the Random rule (and sometimes even the most-infeasible rule) is typically many more times larger than the optimal branch-and-bound tree.

Finally, see Figure Figure 4.4 for a summary of the computational results. It is clear that while strong-branching does quite well (within a factor of 2), there is scope for coming up with better rules for deciding branching variables, for example for problems such as CCP portfolio optimization. It would be interesting to see if one can use machine learning techniques to learn from the optimal branch-and-bound trees.

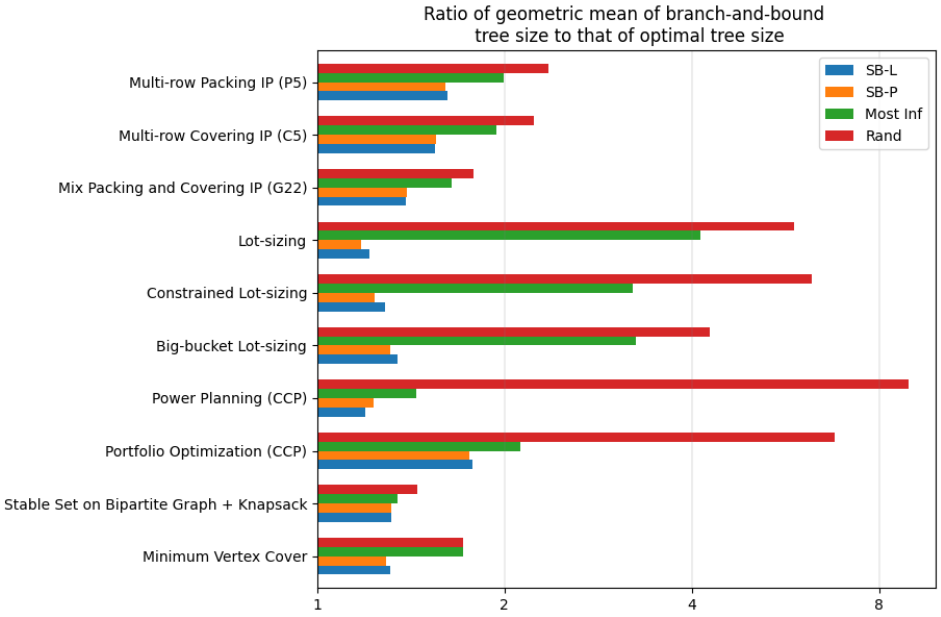

Figure 4.4: Ratio of geometric mean of branch-and-bound tree sizes to geometric mean of optimal tree sizes over all instances of a problem for various branching strategies. "Rand" stands for random, "Most Inf" stands for most infeasible, "SB-P" stands for strong-branching with product score function, and "SB-L" stands for strong-branching with linear score function.

In previous section (see Corollary 12), we have seen an example where strong-branching performs badly while alternative branch-and-bound tree that has exponentially lesser nodes, branches on integer variables. So a natural question we would like to understand is the percent-

age of times the optimal branch-and-bound tree branches on integer variables for the various instances. These results are presented in Table Table 4.1. We note that there are multiple optimal trees. So it may be possible that these exists other optimal trees with a slightly different number of branchings on integer variables.

Here are some of our notable findings:

- Strong branching does relatively poorly on general packing and covering IPs which has a high fraction of integer branchings in the optimal tree and it does very well on lot-sizing where the optimal tree rarely branches on integers. So it would appear consistent with our hypothesis that more branchings on integer variables in the optimal tree implies strong-branching performs poorly.

- On the other hand, for stable set on bipartite graph with knapsack side constraint, strong-branching does very well in spite of a lot of integer branchings in the optimal tree.

Table 4.1: Summary of average tree sizes (geometric) and percentage of branching on integral variable in optimal tree for all problems across 100 instances

Problem	Opt Tree	SB-L	SB-P	Most inf	Rand	% Int Branch
Multi-row Packing IP (P5)	25.4	41.2	40.9	50.8	59.9	48.2%
Multi-row Covering IP (C5)	25.2	39.0	39.1	48.9	56.4	49.0%
Mix Packing and Covering IP (G22)	10.2	14.2	14.2	16.8	18.3	48.0%
Lot-sizing	111.5	135.0	131.1	461.0	651.6	0.2%
Constrained Lot-sizing	101.9	131.1	125.9	328.0	635.0	0.2%
Big-bucket Lot-sizing	81.8	110.3	107.1	266.4	349.6	1.1%
Power Planning (CCP)	37.9	45.3	46.8	54.8	337.9	1.1%
Portfolio Optimization (CCP)	97.4	172.8	171.1	206.5	659.1	4.4%
Stable Set on Bipartite Graph + Knapsack	137.6	180.8	180.8	185.5	199.6	47.7%
Minimum Vertex Cover	7.1	9.3	9.1	12.1	12.2	0.0%

In other words, there does not seem to be a direct relationship between the performance of strong-branching and the number of branching on integer variable of the optimal branch-and-bound tree.

Finally we end this section with a word of caution regarding over-interpreting the computational results above: As mentioned above, due of the exponential nature of the DP algorithm to compute the optimal branch-and-bound tree, computational experiments could be performed only on relatively smaller problem sizes with up to 20 binary variables. Some of the observations derived here may not extrapolate to larger instances.

4.5 Computing an optimal branch-and-bound tree

4.5.1 Proof of Observation 1.

Observation 2. *When using an LP solver which satisfies Assumption 1, a branch-and-bound tree using the worst bound rule never branches on a node whose optimal objective function value is equal to that of the IP solver.*

Proof. Consider a linear program at a node whose optimal function value is equal to that of the MILP optimal objective function value. There are two cases to consider. Case 1: if there exists an integral optimal solution to the LP relaxation at a given node, then the LP solver finds it and the node is pruned. Case 2: If not, then this node does not contain an integral solution with objective function value equal to that of the MILP optimal objective function value. This implies that there must exist another node whose feasible region contains an integer feasible solution with the same objective function value as that of the MILP optimal objective function value. In particular, this implies that there must exist a node whose optimal linear programming solution is such an integer feasible solution. Since the LP solver will discover this solution before branching on the current node, we will never branch on the current node. □

4.5.2 The dynamic programming algorithm to compute optimal branch-and-bound tree.

We will now present the algorithm to compute the size of an optimal branch-and-bound tree for a fixed IP $\max_{x \in P \cap \{0,1\}^n} \langle c, x \rangle$ under Assumption 1. Let $\mathcal{F}$ denote the set of faces of $[0,1]^n$ and

note that each face can be defined as a string in $\{\star, 0, 1\}^n$. For example, $(0, \star, 1)$ denotes the face $\{x \in [0,1]^3 \,:\, x_1 = 0, x_3 = 1\}$. Thus, $|\mathcal{F}| = 3^n$. Also, $\mathcal{F}$, is in one-to-one correspondence with all the possible nodes in the branch-and-bound tree. Let $\overline{\text{OPT}}(F)$ denote the size of the optimal branch-and-bound tree for the sub-problem restricted to F, i.e., $\max_{x \in F \cap P \cap \{0,1\}^n} \langle c, x \rangle$.

Based on Assumption 1 and Observation 1, with the WDB rule for node selection, a node in the branch-and-bound tree is pruned if and only if it is either infeasible, or the optimal objective function value of its LP relaxation is less than or equal to the optimal MILP optimal objective value. In Phase-1 of our algorithm (Algorithm Algorithm 1), this fact is used to identify nodes that are pruned by infeasibility or by bound, thus, $\overline{\text{OPT}}(F) = 0$ for corresponding faces.

Now, given a face F that is not pruned in the branch-and-bound tree, and variable x_j that is free in F, define $F_{j,0}, F_{j,1}$ to be the faces of F that result from fixing x_j to 0 and 1 respectively, i.e. $F_{j,0} = \{x \in F \,:\, x_j = 0\}$. The fact that the optimal sub-tree at the node corresponding to F branches on the variable that produces two child nodes having the smallest optimal sub-trees, leads to the following recurrence relation,

$$\overline{\text{OPT}}(F) = 1 + \min_{j \in J_F} \left\{ \overline{\text{OPT}}(F_{j,0}) + \overline{\text{OPT}}(F_{j,1}) \right\},$$

where J_F denotes the set of variables that are free in F. We use this recurrence relation in the bottom-up computation of $\overline{\text{OPT}}(F)$ for the remaining faces (i.e. faces where $\overline{\text{OPT}}(F) \neq 0$) in $\mathcal{F}$ as Phase-2 of the algorithm. Thus, it can be inductively seen that the algorithm is correct. Additionally, the actual branch-and-bound tree can be found by storing $\arg\min_j \left\{ \overline{\text{OPT}}(F_{j,0}) + \overline{\text{OPT}}(F_{j,1}) \right\}$ at every iteration.

Notice it takes $2^{O(n)}$ time to execute line 1 and poly($data$) time to execute line 3 of Algorithm Algorithm 1 for a particular face, where poly($data$) is the running time for solving an LP. Therefore, Phase I takes at most $2^{O(n)} + \text{poly}(data) \cdot 3^n$ time. Also notice Phase-2 takes $n \cdot 3^n$ time; this is because line 11 of Algorithm Algorithm 1 takes at most n comparisons. So, in total Phase-1 and

Algorithm 1 Computing Optimal Branch-and-bound Tree

Phase-1: Pruning by Infeasibility or Bound

Solve $\max_{x \in P \cap \{0,1\}^n} \langle c, x \rangle$; let x^* be the solution
Initialise: $\mathcal{S} \leftarrow \mathcal{F}$
for F in $\mathcal{S}$ **do**
    Solve $\max_{x \in F \cap P} \langle c, x \rangle$; let x_F^* be the optimal solution ($x_F^* = \emptyset$ if LP is infeasible)
    if $x_F^* = \emptyset$ **or** $\langle c, x_F^* \rangle \leq \langle c, x^* \rangle$ **then**
        $\overline{\mathrm{OPT}}(F) \leftarrow 0$
        $\mathcal{S} \leftarrow \mathcal{S} \setminus \{F\}$
    end if
end for

Phase-2: Recursive bottom-up computation

Sort $\mathcal{S}$ in order of increasing dimension
for F in $\mathcal{S}$ **do**
    $\overline{\mathrm{OPT}}(F) \leftarrow 1 + \min_j(\overline{\mathrm{OPT}}(F_{j,0}) + \overline{\mathrm{OPT}}(F_{j,1}))$
end for
return $\overline{\mathrm{OPT}}([0,1]^n)$

Phase-2 take $2^{O(n)} + (\mathrm{poly}(data) + n) \cdot 3^n = \mathrm{poly}(data) \cdot 3^{O(n)}$ time.

CHAPTER 5

FUTURE WORK

5.1 Probabilistic Analysis of Branch-and-Bound

In chapter 2, we conduct a probabilistic analysis of branch-and-bound with variable branching. We are able to show that branch-and-bound does well for random integer programs with few (at most constant) constraints. A natural extension would be to show that branch-and-bound does well for some instances with more (e.g. $\Omega(n)$) constraints, even if that means we have to introduce more structure. For example, we could study how branch-and-bound performs in solving the maximum weight matching problem (defined only by the degree constraints) on random graphs. A similar study was attempted by [47], where Alan Frieze studied how branch-and-bound performs on random instances of the asymmetric traveling salesperson problem. In this study, he found that, in expectation, branch-and-bound results in exponential size trees.

5.2 Lower Bounds on the Size of General Branch-and-Bound Trees

In chapter 3, we demonstrated several examples of combinatorial optimization problems whose integer programming formulations require exponential size general branch-and-bound trees. An important caveat to these results is that each of these polytopes are described by an exponential number of constraints (i.e. have an exponential number of facets). Moreover, the proof techniques used crucially rely on this property. This leaves the question of whether there exist polytopes with a small description requiring large general branch-and-bound trees. In an attempt to gain more insight into this question, [80] show the affirmative is true for the analogous question for split cuts, which is a cutting plane paradigm based on the same family of disjunctions as general branch-and-bound (see [3] for more information on split cuts). Unfortunately, [9] showed that

the same family of polytopes is solved relatively easily by general branch-and-bound, resulting in trees of depth at most $O(\log^2 n)$. However, it is possible that a *similar* family of polytopes is a good candidate for showing the desired result. That being said, it would be especially nice if one could show an exponential lower bound for stable set, since the results of chapter 3 would imply exponential lower bounds for several other problems like vertex cover, set partition, and knapsack.

5.3 Quantifying the Impact of Variable Selection

In chapter 4, we quantify in several ways how strong branching performs well for vertex cover. It would be especially interesting to study the success of strong branching in more generality. The positive results of chapter 4 are largely due to properties of the vertex cover polytope, including half-integrality and persistency. [81] gives a general framework that can get half-integral and persistent linear programming relaxations for several combinatorial optimization problems (e.g. they find new such relaxations for the group feedback vertex set and unique label cover problems). Perhaps we can show that strong branching performs well integer programming formulations constructed via the techniques of [81].

Finally, we think a challenging, but potentially rewarding direction might be to study other successful branching schemes—maybe in relationship to strong branching or most fractional branching, for example. Resolution search from [82] is one that comes to mind.

REFERENCES

[1] A. H. Land and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[2] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*. John Wiley & Sons, 1999, vol. 55.

[3] M. Conforti, G. Cornuéjols, G. Zambelli, *et al.*, *Integer programming*. Springer, 2014, vol. 271.

[4] H. W. Lenstra Jr, "Integer programming with a fixed number of variables," *Mathematics of operations research*, vol. 8, no. 4, pp. 538–548, 1983.

[5] H. W. Lenstra, A. K. Lenstra, L. Lovász, *et al.*, "Factoring polynomials with rational coeficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[6] G. Pataki, M. Tural, and E. B. Wong, "Basis reduction and the complexity of branch-and-bound," in *Proceedings of the twenty-first annual ACM-SIAM symposium on discrete algorithms*, SIAM, 2010, pp. 1254–1261.

[7] K. Aardal, R. E. Bixby, C. A. Hurkens, A. K. Lenstra, and J. W. Smeltink, "Market split and basis reduction: Towards a solution of the cornuéjols-dawande instances," *INFORMS Journal on Computing*, vol. 12, no. 3, pp. 192–202, 2000.

[8] G. Cornuéjols, L. Liberti, and G. Nannicini, "Improved strategies for branching on general disjunctions," *Mathematical Programming*, vol. 130, no. 2, pp. 225–247, 2011.

[9] P. Beame *et al.*, "Stabbing planes," in *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[10] S. S. Dey, Y. Dubey, and M. Molinaro, "Branch-and-bound solves random binary ips in polytime," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, SIAM, 2021, pp. 579–591.

[11] S. Borst, D. Dadush, S. Huiberts, and S. Tiwari, "On the integrality gap of binary integer programs with gaussian data.," in *IPCO*, 2021, pp. 427–442.

[12] R. G. Jeroslow, "Trivial integer programs unsolvable by branch-and-bound," *Mathematical Programming*, vol. 6, no. 1, pp. 105–109, 1974.

[13] V. Chvátal, "Hard knapsack problems," *Operations Research*, vol. 28, no. 6, pp. 1402–1411, 1980.

[14] Y. Yang, N. Boland, and M. Savelsbergh, "Multivariable branching: A 0-1 knapsack problem case study," *INFORMS Journal on Computing*, 2021.

[15] A. Basu, M. Conforti, M. Di Summa, and H. Jiang, "Complexity of branch-and-bound and cutting planes in mixed-integer optimization-ii," *Integer Programming and Combinatorial Optimization (IPCO) 2021*, 2021.

[16] W. J. Cook and M. Hartmann, "On the complexity of branch and cut methods for the traveling salesman problem.," *Polyhedral Combinatorics*, vol. 1, pp. 75–82, 1990.

[17] A. Basu, M. Conforti, M. Di Summa, and H. Jiang, "Complexity of branch-and-bound and cutting planes in mixed-integer optimization," *Mathematical Programming*, pp. 1–24, 2022.

[18] D. Dadush and S. Tiwari, "On the complexity of branching proofs," in *Proceedings of the 35th Computational Complexity Conference*, 2020, pp. 1–35.

[19] S. S. Dey, Y. Dubey, and M. Molinaro, "Lower bounds on the size of general branch-and-bound trees," *Mathematical Programming*, pp. 1–21, 2022.

[20] W. Healy Jr, "Multiple choice programming (a procedure for linear programming with zero-one variables)," *Operations Research*, vol. 12, no. 1, pp. 122–138, 1964.

[21] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *The computer journal*, vol. 8, no. 3, pp. 250–255, 1965.

[22] N. J. Driebeek, "An algorithm for the solution of mixed integer programming problems," *Management Science*, vol. 12, no. 7, pp. 576–587, 1966.

[23] M. Bénichou, J.-M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent, "Experiments in mixed-integer linear programming," *Mathematical Programming*, vol. 1, no. 1, pp. 76–94, 1971.

[24] G. Mitra, "Investigation of some branch and bound strategies for the solution of mixed integer linear programs," *Mathematical Programming*, vol. 4, no. 1, pp. 155–170, 1973.

[25] J. Forrest, J. Hirst, and J. A. Tomlin, "Practical solution of large mixed integer programming problems with umpire," *Management Science*, vol. 20, no. 5, pp. 736–773, 1974.

[26] R. Breu and C.-A. Burdet, "Branch and bound experiments in zero-one programming," in *Approaches to Integer Programming*, Springer, 1974, pp. 1–50.

[27] J.-M. Gauthier and G. Ribiere, "Experiments in mixed-integer linear programming using pseudo-costs," *Mathematical Programming*, vol. 12, no. 1, pp. 26–47, 1977.

[28] A. Land and S. Powell, "Computer codes for problems of integer programming," in *Annals of Discrete Mathematics*, vol. 5, Elsevier, 1979, pp. 221–269.

[29] J. Eckstein, "Parallel branch-and-bound algorithms for general mixed integer programming on the cm-5," *SIAM journal on optimization*, vol. 4, no. 4, pp. 794–814, 1994.

[30] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "Finding cuts in the tsp (a preliminary report)," Citeseer, Tech. Rep., 1995.

[31] J. T. Linderoth and M. W. Savelsbergh, "A computational study of search strategies for mixed integer programming," *INFORMS Journal on Computing*, vol. 11, no. 2, pp. 173–187, 1999.

[32] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Operations Research Letters*, vol. 33, no. 1, pp. 42–54, 2005.

[33] S. S. Dey, Y. Dubey, M. Molinaro, and P. Shah, "A theoretical and computational analysis of full strong-branching," *arXiv preprint arXiv:2110.10754*, 2021.

[34] R. Bixby and E. Rothberg, "Progress in computational mixed integer programming–a look back from the other side of the tipping point," *Annals of Operations Research*, vol. 149, no. 1, p. 37, 2007.

[35] "Chapter 3 large sparse linear programming," in *Large Sparse Numerical Optimization*, T. F. Coleman, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 35–46, ISBN: 978-3-540-38796-1.

[36] M. Walter, "Sparsity of lift-and-project cutting planes," in *Operations Research Proceedings 2012*, Springer, 2014, pp. 9–14.

[37] S. S. Dey, M. Molinaro, and Q. Wang, "Approximating polyhedra with sparse inequalities," *Mathematical Programming*, vol. 154, no. 1-2, pp. 329–352, 2015.

[38] ——, "Analysis of sparse cutting planes for sparse milps with applications to stochastic milps," *Mathematics of Operations Research*, vol. 43, no. 1, pp. 304–332, 2018.

[39] S. Dash, "An exponential lower bound on the length of some classes of branch-and-cut proofs," in *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2002, pp. 145–160.

[40] K. K. Cheung, A. Gleixner, and D. E. Steffy, "Verifying integer programming results," in *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2017, pp. 148–160.

[41] D. A. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 385–463, 2004.

[42] G. S. Lueker, "On the average difference between the solutions to linear and integer knapsack problems," in *Applied Probability – Computer Science, The Interface*, vol. 1, Birkhäuser, 1982.

[43] A. V. Goldberg and A. Marchetti-Spaccamela, "On finding the exact solution of a zero-one knapsack problem," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984, pp. 359–368.

[44] R. Beier and B. Vöcking, "Random knapsack in expected polynomial time," *Journal of Computer and System Sciences*, vol. 69, no. 3, pp. 306–329, 2004.

[45] R. Beier and B. Vöcking, "Probabilistic analysis of knapsack core algorithms," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, 2004, pp. 468–477.

[46] M. E. Dyer and A. M. Frieze, "Probabilistic analysis of the multidimensional knapsack problem," *Mathematics of Operations Research*, vol. 14, no. 1, pp. 162–176, 1989.

[47] A. M. Frieze, "On the expected efficiency of branch and bound for the asymmetric tsp," 2020.

[48] V. Vazirani, *Approximation algorithms*. Springer, 2001, ISBN: 9783540653677.

[49] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[50] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

[51] K. Ball, "Cube slicing in rn," *Proceedings of the American Mathematical Society*, vol. 97, no. 3, pp. 465–473, 1986.

[52] V. Koltchinskii, *Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems*. Springer-Verlag, 2011.

[53] N. Fleming *et al.*, "On the power and limitations of branch and cut," in *Proceedings of the 36th Computational Complexity Conference*, 2021, p. 1.

[54] P. Pudlák, "Lower bounds for resolution and cutting plane proofs and monotone computations," *Journal of Symbolic Logic*, pp. 981–998, 1997.

[55] S. Dash, "Exponential lower bounds on the lengths of some classes of branch-and-cut proofs," *Mathematics of Operations Research*, vol. 30, no. 3, pp. 678–700, 2005.

[56] T. Roughgarden, Ed., *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020, ISBN: 9781108637435.

[57] V. Chvátal, W. Cook, and M. Hartmann, "On cutting-plane proofs in combinatorial optimization," *Linear algebra and its applications*, vol. 114, pp. 455–499, 1989.

[58] A. I. Barvinok, *A course in convexity*, ser. Graduate studies in mathematics. American Mathematical Society, 2002, vol. 54, ISBN: 978-0-8218-2968-4.

[59] S. Dash, O. Günlük, and M. Molinaro, "On the relative strength of different generalizations of split cuts," *Discrete Optimization*, vol. 16, pp. 36–50, 2015.

[60] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*. Cambridge university press, 2018, vol. 47.

[61] S. Jukna, *Extremal Combinatorics: With Applications in Computer Science*, 1st. Springer Publishing Company, Incorporated, 2010, ISBN: 3642085598.

[62] L. Gottlieb, A. Kontorovich, and E. Mossel, "VC bounds on the cardinality of nearly orthogonal function classes," *Discret. Math.*, vol. 312, no. 10, pp. 1766–1775, 2012.

[63] W. Cook and S. Dash, "On the matrix-cut rank of polyhedra," *Mathematics of Operations Research*, vol. 26, no. 1, pp. 19–30, 2001.

[64] T. Achterberg, "Constraint integer programming," *PhD Thesis*, 2007.

[65] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[66] A. Lodi and G. Zarpellon, "On learning and branching: A survey," *Top*, vol. 25, no. 2, pp. 207–236, 2017.

[67] A. M. Alvarez, Q. Louveaux, and L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 185–195, 2017.

[68] M.-F. Balcan, T. Dick, T. Sandholm, and E. Vitercik, "Learning to branch," in *International conference on machine learning*, PMLR, 2018, pp. 344–353.

[69] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," *arXiv preprint arXiv:1906.01629*, 2019.

[70] P. Gupta, M. Gasse, E. B. Khalil, M. P. Kumar, A. Lodi, and Y. Bengio, "Hybrid models for learning to branch," *arXiv preprint arXiv:2006.15212*, 2020.

[71] V. Nair *et al.*, "Solving mixed integer programs using neural networks," *arXiv preprint arXiv:2012.13349*, 2020.

[72] P. Le Bodic and G. Nemhauser, "An abstract model for branching and its application to mixed integer programming," *Mathematical Programming*, vol. 166, no. 1, pp. 369–405, 2017.

[73] G. L. Nemhauser and L. E. Trotter, "Vertex packings: Structural properties and algorithms," *Mathematical Programming*, vol. 8, no. 1, pp. 232–248, 1975.

[74] T. Roughgarden, "Beyond worst-case analysis," *Communications of the ACM*, vol. 62, no. 3, pp. 88–96, 2019.

[75] M. Cygan *et al.*, *Parameterized algorithms*, 4. Springer, 2015, vol. 5.

[76] G. L. Nemhauser and L. E. Trotter, "Properties of vertex packing and independence system polyhedra," *Mathematical programming*, vol. 6, no. 1, pp. 48–61, 1974.

[77] J.-C. Picard and M. Queyranne, "On the integer-valued variables in the linear vertex packing problem," *Mathematical Programming*, vol. 12, no. 1, pp. 97–101, 1977.

[78] M. J. Naderi, A. Buchanan, and J. L. Walteros, "Worst-case analysis of clique mips," *Mathematical Programming*, pp. 1–35, 2021.

[79] M. Bodur, S. Dash, and O. Günlük, "Cutting planes from extended lp formulations," *Mathematical Programming*, vol. 161, no. 1-2, pp. 159–192, 2017.

[80] S. Dash and Y. Dubey, "On polytopes with linear rank with respect to generalizations of the split closure," *arXiv preprint arXiv:2110.04344*, 2021.

[81] Y. Iwata, M. Wahlstrom, and Y. Yoshida, "Half-integrality, lp-branching, and fpt algorithms," *SIAM Journal on Computing*, vol. 45, no. 4, pp. 1377–1411, 2016.

[82] V. Chvatal, "Resolution search," *Discrete Applied Mathematics*, vol. 73, no. 1, pp. 81–99, 1997.