Branch-and-Bound Solves Random Binary IPs in Polytime

Santanu S. Dey*[1], Yatharth Dubey†[1], and Marco Molinaro‡[2]

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology
[2]Computer Science Department, Pontifical Catholic University of Rio de Janeiro

Abstract

Branch-and-bound is the workhorse of all state-of-the-art mixed integer linear programming (MILP) solvers. These implementations of branch-and-bound typically use variable branching, that is, the child nodes are obtained by fixing some variable to an integer value v in one node and to $v + 1$ in the other node. 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 this paper 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 a compelling indication of why branch-and-bound with variable branching works so well in practice.

*santanu.dey@isye.gatech.edu
†yatharthdubey7@gatech.edu
‡molinaro@inf.puc-rio.br

1 Introduction

The branch-and-bound algorithm, first proposed by Land and Doig in [20], is the workhorse of all modern state-of-the-art mixed integer linear programming (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. 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 $\pi x \leq \pi_0$ to first subproblem and the inequality $\pi^\top x \geq \pi_0 + 1$ to the second subproblem, where π is an integer vector and π_0 is an integer scalar. The process of partitioning at a node stops if (i) the LP at the node is infeasible, (ii) the LP's optimal solution 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, i.e. 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. See [12, 32] for more discussion on the branch-and-bound algorithm.

In 1983, Lenstra [22] 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 [21] to decide on π for partitioning the feasible region. Pataki [27] 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 [22]. While there are some implementations of such general partitioning rules [1], all state-of-the-art solvers use a much simpler (and potentially significantly weaker and restrictive) partitioning rule for solving binary IPs, namely: If x_j is fractional in the optimal solution of the LP of a given node, then one child node is obtained by the addition of the constraint $x_j \leq 0$ and the other with the constraint $x_j \geq 1$ (i.e., $\pi = e_j$, the unit vector in direction j for some j in $\{1, \ldots, n\}$). The rule for deciding how to partition the feasible region at a node then reduces to choosing which fractional variable should be branched on. This kind of partitioning rule is henceforth referred to as **variable branching**.

As mentioned above, all state-of-the-art MILP solvers use variable branching, which has proven itself to be very successful in practice [8]. 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 [11, 14, 15, 31]). Additionally, while in the worst-case there can be exponentially many nodes in the branch-and-bound tree (see [10, 13] 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 [9]. 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 [28]. 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

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

By a **random instance** of IP(b) 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. For the right-hand-side, we will use $b_j = \beta_j \cdot n$, 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 all items fit with probability 1 as $n \longrightarrow \infty$ and m is fixed.) 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(b) 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

$$n^{\bar{a}_1 \cdot (m + \alpha \log m)}$$

nodes for all $\alpha \leq \min\{30m, \frac{\log n}{\bar{a}_2}\}$, where $\bar{a}_1$ and $\bar{a}_2$ are constant 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 [23]. This node selection rule is known to guarantee the smallest tree for any fixed partitioning rule [32]. 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* [2]. 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(b) 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 8. 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 [27] 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 [24] 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 [18] presented a polynomial-time enumeration algorithm for these instances. Beier and Vocking [5, 6] showed that the so-called knapsack core algorithm with suitable improvements using enumeration runs in polynomial-time with high probability. Finally, Dyer and Frieze [16] generalize the previous results on integrality gap and enumeration techniques to the random instances we consider here.

In terms of techniques for proving Theorem 1, the first main idea is to relate the number of nodes of the branch-and-bound tree to the number of near-optimal solutions for all problems IP(b') with different right-hand sides b'. Then we show that for random instances there are polynomially many such solutions *across all right-hand sides b'*. This is a much stronger statement compared to, for example, the analysis

in Goldberg and Marchetti-Spaccamela [18], which only needs to show that there are polynomially many near-optimal solutions for the original right-hand side b.

The rest of the paper is organized as follows. Section 2 presents notation, formalizes the set-up and presents some preliminary results needed. Section 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 4, Section 5, and Section 6 build up machinery to prove that the number of "near optimal" solutions is bounded by a polynomial. Section 7 completes the proof of Theorem 1.

2 Preliminaries

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.

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 worse 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.

One simple but important property of this best-bound node selection rule is the following. (Note that we do not assume that an optimal IP solution 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*.)

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 an 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 (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(b) (see for example Section 17.2 of [29]).

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

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

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

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 (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(b) is IPGap(b) := OPT(LP(b)) − OPT(IP(b)), namely the optimal value its LP relaxation LP(b) := $\max\{\langle c, x\rangle : Ax \le b,\ x \in [0,1]^n\}$ 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(b) (defined right after the definition of IP(b)), the integrality gap is $O(\frac{\log^2 n}{n})$.

Theorem 2 (Theorem 1 of [16])**.** *Consider a vector $\beta \in (0, \frac{1}{2})^n$ and let $b \in \mathbb{R}^n$ be given by $b_j = \beta_j \cdot n$ for $j \in \{1, \dots, n\}$. 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(b)*

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

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

2.3 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.

3 Branch-and-bound and good integer solutions

In this section we connect the size of the BB tree for any instance of IP(b) and the number of near-optimal solutions for IP(b') for some right-hand side b' (see Figure 1).

To make this precise, first let LP$_=$(b) be the LP relaxation of IP(b) but with *equality* constraints:

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

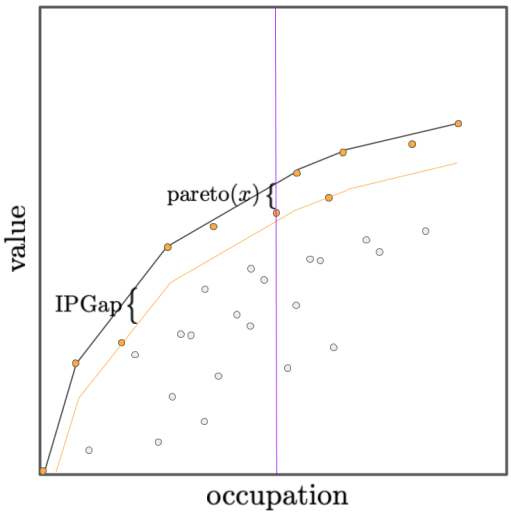

Figure 1: The image depicts the space of (fractional) points x plotted according to their value $\langle c, x\rangle$ and occupation Ax. The dots correspond to the 0/1 points $x \in \{0,1\}^n$. The top (black) curve represents the value of the optimal fractional solutions with each possible occupation, i.e., the pareto curve. Each right-hand side/occupation b' gives a "slice" (vertical line) containing all points with precisely that occupation. If x is in a slice b', pareto(x) is the difference in value between x and the best fractional solution in that slice, namely $\mathrm{LP}_=(b')$. The yellow dots represent the good solution, namely those in G.

With slight overload of notation, we also use $\mathrm{LP}_=(b)$ to denote the optimal value of this program (with $\mathrm{LP}_=(b) = -\infty$ if the program is infeasible). Given a 0/1 solution $x \in \{0,1\}^n$ in a slice b' , i.e., with $Ax = b'$, define its *pareto gap* as the difference between its value and the value of the optimal fractional solution in this slice:

$$\text{pareto}(x) \;:=\; \mathrm{LP}_=(Ax) - \langle c, x\rangle.$$

A 0/1 point x is *good* if its pareto gap is at most $\textsc{IPGap}(b)$, and we use G to denote the set of all good points, namely

$$G \;:=\; \{x \in \{0,1\}^n \;:\; \text{pareto}(x) \le \textsc{IPGap}(b)\}. \tag{2}$$

We then have the following.

Theorem 3. *Consider the branch-and-bound algorithm with best-bound node selection rule for solving IP(b). Then its final tree has at most $2|G|\binom{n}{\le m} + 1$ 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. For any 0/1 point $x \in \{0,1\}^n$ and subset $J \in \binom{[n]}{\le m}$ of at most m coordinates, let $C_J(x) \;:=\; \{x' \in [0,1]^n \;:\; x'_j = x_j \;\; \forall j \notin J,\; x'_j \in (0,1) \;\; \forall j \in J\}$ be the set of vectors that are fractional on coordinates in J and agree with x outside of J. Intuitively $C_J(x)$ is a neighborhood of x. The next lemma shows that when the algorithm branches on a node, it is because its optimal LP solution is a neighbor of one of the good points G.

Lemma 3. *Let N be a node that is branched on in the BB tree $\mathcal{T}$ and let x^N be an optimal solution for the LP of this node with at most m fractional coordinates (via Lemma 2). Then there is a good point $x \in G$ and a set of coordinates $J \in \binom{[n]}{\le m}$ such that $x^N \in C_J(x)$.*

Proof. Let $J = \{j : x^N_j \in (0,1)\}$ be the indices where x^N is fractional, which by assumption has size $|J| \le m$. The vector x^N can be written as a convex combination of 0/1 points, namely there are points $x^1, \dots, x^k \in \{0,1\}^n$ and multipliers $\lambda, \dots, \lambda_k \in [0,1]$ with $\sum_i \lambda_i = 1$ such that $x^N = \sum_i \lambda_i x^i$. Notice that all points x^i agree with x^N the integer coordinates of x^N, and hence $x^N \in C_J(x^i)$ for every x^i. Thus, it suffices to show that one of these x^i's belongs to G.

So we need to lower bound the value of the solutions x^i. First, since the node N was not pruned, from Lemma 1 we can lower bound the value of the LP solution x^N as

$$\langle c, x^N \rangle > \mathrm{LP}(b) - \mathrm{IPGap}(b) \ge \mathrm{LP}_=(Ax^N) - \mathrm{IPGap}(b), \tag{3}$$

where the second inequality uses the fact that $Ax^N \le b$ and so any feasible solution for $\mathrm{LP}_=(Ax^N)$ is also feasible for $\mathrm{LP}_=(b)$. Moreover, it is well-known that the function $\mathrm{OPT}_{\mathrm{LP}}(\cdot)$ is concave (Theorem 5.1 of [7]), and hence the function

$$x \mapsto \langle c, x \rangle - \mathrm{LP}_=(Ax)$$

is convex. This implies that there is one of the x^i's such that

$$\langle c, x^i \rangle - \mathrm{LP}_=(Ax^i) \ge \langle c, x^N \rangle - \mathrm{LP}_=(Ax^N),$$

which together with inequality (3) gives $\langle c, x^i \rangle \ge \mathrm{LP}_=(Ax^i) - \mathrm{IPGap}(b)$ and hence $x^i \in G$, as desired. □

We now show that the association between BB nodes and pairs (x, J) made in the previous lemma is unique.

Lemma 4. *Let N_1, N_2 be two different nodes that are branched on in $\mathcal{T}$ and let $x^{(1)}, x^{(2)}$ be their optimal LP solutions respectively. Let $x \in G$ and $J \in \binom{[n]}{\le m}$ be such that $x^{(1)} \in C_J(x)$. Then the other solution $x^{(2)}$ does not belong to $C_J(x)$.*

Proof. First consider the case where N_1 and N_2 are not one a descendant of the other in the BB tree $\mathcal{T}$. Let $N \ne N_1, N_2$ 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 $x^{(1)}_f = 0$ (so integral) and $x^{(1)} \in C_J(x)$, we have $f \notin J$ and $x_f = x^{(1)}_f = 0$. Since $x^{(2)}_f = 1$, by definition of $C_J(x)$ we have $x^{(2)} \notin C_J(x)$ as desired.

Similarly, if N_2 is a descendant of N_1, then letting f be the index where N_1 is branched on we see that $x^{(1)}$ is fractional at coordinate j and hence $j \in J$; but due to this branching, $x^{(2)}$ is integral in this coordinate and hence $x^{(2)} \notin C_J(x)$.

Finally, if N_1 is a descendant of N_2, then letting f be the index where N_2 is branched on we see that due to the branching now $x^{(1)}$ is integral at coordinate j and hence $j \notin J$; but since the branching occurred at a fractional coordinate of $x^{(2)}$ this again gives $x^{(2)} \notin C_J(x)$. This concludes the proof. □

Proof of Theorem 3. From Lemmas 3 and 4 we see that the number of nodes in $\mathcal{T}$ that are branched on (i.e., internal nodes of $\mathcal{T}$) is upper bounded by the number of pairs $(x, J) \in G \times \binom{[n]}{\le m}$. 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 $2|G|\binom{n}{\le m} + 1$ 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 (2), which will then prove Theorem 1.

4 Dual-based solutions

To bound the number of good points the starting idea is to show that the good points in slice b' are all similar to (in a "neighborhood" of) the optimal LP solution for $\mathrm{LP}_=(b')$; but this still has the issue that we need to consider all $b' \in \mathbb{R}^m_+$, so good points can be in infinitely many "neighborhoods", which is not very useful. The main idea is then to consider (partial) solutions to $\mathrm{LP}_=(b')$ induced by a *dual* vector $\lambda \in \mathbb{R}^m$. This is helpful because there are *finitely* many such (partial) LP solutions *across all* b''s.

Thinking of a vector $\lambda \in \mathbb{R}^m$ as setting prices for the constraints $Ax = b'$, it induces a (partial) solution to $\mathrm{LP}_=(b')$ via the reduced costs $c_j - \langle \lambda, A^j \rangle$ in the natural way: pick item j (setting $x_j = 1$) if the reduced cost is positive, do not pick it (setting $x_j = 0$) if the reduced cost is negative, and leave undecided if the reduced cost is 0 (setting $x_j = \star$).

Definition 1 (Dual-based sol, compatibility)**.** *Given a vector $\lambda \in \mathbb{R}^m$, we define the partial solution $x(\lambda) \in \{0, 1, \star\}^n$ induced by it as*

$$x(\lambda)_j = \begin{cases} 1, & \text{if } c_j - \langle \lambda, A^j \rangle > 0 \\ 0, & \text{if } c_j - \langle \lambda, A^j \rangle < 0 \\ \star, & \text{if } c_j - \langle \lambda, A^j \rangle = 0 \end{cases}.$$

We say that a point $x \in \{0,1\}^n$ is compatible *with $x(\lambda)$ if they agree in all coordinates j where $x(\lambda)_j \neq \star$.*

It is well-known that we can obtain an optimal solution for any feasible $\mathrm{LP}_=(b')$ by extending a dual-based solution $x(\lambda)$ (see for example [3, 26]); we sketch the proof for completeness and to give a more concrete view of λ.

Lemma 5. *For every optimal solution x^* of $LP_=(b')$ there exists a compatible $x(\lambda)$ for some $\lambda \in \mathbb{R}^m$.*

Proof. Consider the dual problem to $\mathrm{LP}_=(b')$ (where λ are the dual variables relative to the equality knapsacks, and μ is relative to the constraints $x \leq 1$)

$$\begin{aligned} \min \quad & \langle b', \lambda \rangle + \langle \mathbf{1}, \mu \rangle \\ \text{s.t.} \quad & \langle \lambda, A^j \rangle + \mu_j \geq c_j \quad \forall j \in [n] \\ & \mu \geq 0. \end{aligned}$$

Consider any optimal solutions x^* to $\mathrm{LP}_=(b')$ and (λ^*, μ^*) to this dual. We claim that x^* and $x(\lambda^*)$ are compatible. To see this, notice that for j's where $c_j - \langle \lambda^*, A^j \rangle > 0$ (or equivalently $x(\lambda^*)_j = 1$), dual feasibility implies that $\mu^*_j > 0$, and complementary slackness gives that we also have $x^*_j = 1$. To show that for all j with $c_j - \langle \lambda, A^j \rangle < 0$ (or equivalently $x(\lambda^*)_j = 0$) we have $x^*_j = 0$, we prove the contrapositive: when $x^*_j > 0$, complementary slackness gives $c_j = \langle \lambda^*, A^j \rangle + \mu^*_j \geq \langle \lambda^*, A^j \rangle$, giving the contrapositive. □

Crucially, as mentioned above, even though there are infinitely many duals λ, it is known that there are only finitely many dual-based solutions $x(\lambda)$ [3, 26] (notice this is independent of any notion of right-hand side b'): this is because we can think of the *hyperplane arrangement* in $\mathbb{R}^m$ (the space of duals λ) induced by the n hyperplanes $c_j - \langle \cdot, A^j \rangle = 0$, and $x(\lambda)$ is defined based on which face of the arrangement λ falls into, and there are at most $O(n)^m$ faces in any arrangement with n hyperplanes (Section 6.1 of [25] or [17]).

Lemma 6. *The collection of all dual-based solutions $\{x(\lambda)\}_{\lambda \in \mathbb{R}^m}$ has at most $(cst \cdot n)^m$ distinct members, for some constant cst.*

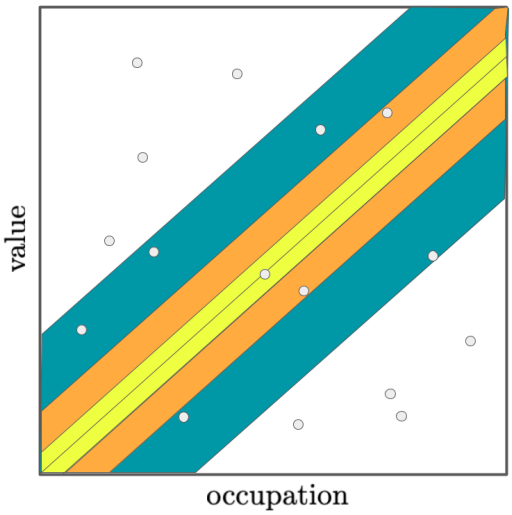

Figure 2: Each dot represents an item/column (c_j, A^j) of IP(b) (unlike Figure 1). The grey line coming from the origin is $H(\lambda)$. The partial solution $x(\lambda)$ is $x(\lambda)_j = 1$ for all items j above this line, $x(\lambda)_j = 0$ for all items j below this line, and $x(\lambda)_j = \star$ for all of those on the line. Each $J_\ell(\lambda)$ is the set of items that lie in one of the colored regions; $J_{\text{rem}}(\lambda)$ are the items that lie on the inner most region (yellow).

5 Value of solutions and geometry of items

We now group the right-hand sides b' based on the compatibility of the optimal solution of $\text{LP}_=(b')$ with the partial solutions $x(\lambda)$'s: given a partial solution $\tilde{x} \in \{0, 1, \star\}^n$, let

$$B_{\tilde{x}} := \{b' \in \mathbb{R}^m_+ : \text{LP}_=(b') \text{ has an optimal solution compatible with } \tilde{x}\}.$$

We should think that $\tilde{x}$ is essentially the "right" solution for all slices $b' \in B_{\tilde{x}}$, and we will show that every good 0/1 point in such a slice must be similar to $\tilde{x}$. Since there is a bounded number of such partial solutions (Lemma 6), there cannot be too many good points over all slices.

To make this notion of similarity precise, consider a partial solution $x(\lambda)$. We can see it as being given by a *linear classification* induced by λ, where $x(\lambda)$ picks an items or not depending on whether

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

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

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

see Figure 2.

The next lemma says that the pareto gap pareto(x) of a 0/1 point x in a slice $b' \in B_{x(\lambda)}$ increases the more it disagrees with $x(\lambda)$, and the penalty for the disagreement on item j depends on the distance of the column (c_j, A^j) to the hyperplane $H(\lambda)$. 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 [18].

Lemma 7. *Consider a partial solution $x(\lambda)$ and let $x \in \{0,1\}^n$ be a 0/1 point in a slice $b' \in B_{x(\lambda)}$, i.e., $Ax \in B_{x(\lambda)}$. Then*

$$\mathit{pareto}(x) \geq \sum_{j\,:\,x(\lambda)_j \neq \star} d\Big((c_j, A^j),\ H(\lambda)\Big) \cdot \mathbf{1}(x_j \neq x(\lambda)_j).$$

Proof. Let $b' := Ax$, and since $b' \in B_{x(\lambda)}$ let x^* be an optimal solution for $\mathrm{LP}_=(b')$ compatible with $x(\lambda)$. We want to lower bound the quantity $\mathrm{pareto}(x) = \langle c, x^* \rangle - \langle c, x \rangle$, which equals

$$\begin{aligned}
\mathrm{pareto}(x) &= \langle c, x^* - x \rangle \\
&= \langle c - A^T\lambda, x^* - x \rangle + \langle \lambda, A(x^* - x) \rangle \\
&= \langle c - A^T\lambda, x^* - x \rangle \\
&= \sum_{j\,:\,x(\lambda)_j = 0} (c_j - \langle \lambda, A^j \rangle)(x^*_j - x_j) \\
&\quad + \sum_{j\,:\,x(\lambda)_j = 1} (c_j - \langle \lambda, A^j \rangle)(x^*_j - x_j),
\end{aligned} \tag{4}$$

where the third equation follows from $A\bar{x} = b' = Ax^*$, and the fourth equation follows because $c_j - \langle \lambda, A^j \rangle = 0$ when $x(\lambda)_j = \star$. To analyze the right-hand side of this equation, consider a term in the first sum. Since $x(\lambda)_j = 0$, by compatibility we have $x^*_j = 0$, and hence the term is non-zero exactly when the 0/1 point x takes value $x_j = 1$. Moreover, since $x(\lambda)_j = 0$ implies that $c_j - \lambda A^j$ is negative, this means that the terms in the first sum of (4) equal $|c_j - \langle \lambda, A^j \rangle| \cdot \mathbf{1}(x_j \neq x(\lambda)_j)$. A similar analysis shows that the terms in the second sum of (4) also equal $|c_j - \langle \lambda, A^j \rangle| \cdot \mathbf{1}(x_j \neq x(\lambda)_j)$. Together these give

$$\mathrm{pareto}(x) = \sum_{j\,:\,x(\lambda)_j \neq \star} |c_j - \langle \lambda, A^j \rangle| \cdot \mathbf{1}(x_j \neq x(\lambda)_j). \tag{5}$$

Finally, notice that

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

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

$$\begin{aligned}
d\Big((c_j, A^j),\ H(\lambda)\Big) &\leq \|(c_j, A^j) - (\langle \lambda, A^j \rangle, A^j)\|_2 \\
&= |c_j - \langle \lambda, A^j \rangle|.
\end{aligned}$$

Plugging this bound on (5) concludes the proof of the lemma. □

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

$$J_\ell(\lambda) := \Big\{ j \,:\, x(\lambda)_j \neq \star, \text{ and } d\Big((c_j, A^j),\ H(\lambda)\Big) \text{ is in the interval } \Big(\tfrac{\log n}{n} 2^\ell,\ \tfrac{\log n}{n} 2^{\ell+1}\Big] \Big\},$$

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

Corollary 1. *Consider a partial solution $x(\lambda)$. If $x \in \{0,1\}^n$ is a good point (i.e., pareto$(x) \le$ IPGap(b)) in a slice $b' \in B_{x(\lambda)}$ (i.e., $Ax \in B_{x(\lambda)}$), then for every $\ell \ge 1$ the number of coordinates $j \in J_\ell(\lambda)$ such that $x_j \ne x(\lambda)_j$ is at most $\frac{C}{2^\ell}$, where $C := \frac{n}{\log n} \cdot$ IPGap(b).*

By considering all the possible partial solutions $x(\lambda)$ (there are $O(n)^m$ many, by Lemma 6), an easy counting argument gives an upper bound on the total number of good points G (see Appendix A for the proof).

Lemma 8. *We have the following upper bound:*

$$|G| \le (cst \cdot n)^m \cdot \max_{\lambda \in \mathbb{R}^m} 2^{|J_{rem}(\lambda)|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell(\lambda)|}{\le C/2^\ell},$$

where cst is the constant from Lemma 6 and $C := \frac{n}{\log n} \cdot$ IPGap(b).[1]

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

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

6 Number of items at a distance from the hyperplane

To control the size of the sets $J_\ell(\lambda)$ we need to consider a random instance of IP(b) and use the fact that the columns (c_j, A^j) are uniformly distributed in $[0,1]^{m+1}$. Recalling the definition of $J_\ell(\lambda)$, 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 $H(\lambda)$:

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

It can be showed that the volume of this slab intersected with $[0,1]^{m+1}$ is proportional to its width, and so 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 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 $J_\ell(\lambda)$ (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.

However, in order for this to be useful in Lemma 8 we need a much stronger *uniform* bound that shows that with good probability this holds *simultaneously* for all slabs around all hyperplanes $\{H(\lambda)\}_{\lambda \in \mathbb{R}^m}$. We abstract this situation and prove such uniform bound. We use $\mathbb{S}^{k-1}$ to denote the unit sphere in $\mathbb{R}^k$.

Lemma 9 (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 $Y^1, \dots, Y^n$ be independent random vectors uniformly distributed in the cube $[0,1]^k$, for $n \ge k$. Then with probability at least $1 - \frac{1}{n}$, we have that for all $u \in \mathbb{S}^{k-1}$ and $w \ge \frac{\log n}{n}$ 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 Lemma 9 instead relies on an ε-net type argument.

[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.

6.1 Proof of Lemma 9

We start by focusing on a single slab. 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 $\mathrm{vol}(U)$. 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 10. *For every slab $S_{u,w}$ of width w we have $\Pr(Y^j \in S_{u,w}) \le 2\sqrt{2}w$.*

Proof. 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 $\sqrt{2}$ [4], 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 $\mathbb{E}N_{u,w} \le 2\sqrt{2}wn$. 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 [19] 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 11. *Let $Z_1, \ldots, Z_n$ 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 12. *For each $u \in \mathbb{S}^{k-1}$ and width $w \ge \frac{\log n}{n}$, we have*

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

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

$$\begin{aligned}\Pr\left(N_{u,w} \;\ge\; 5\mu + \frac{16}{3}k\ln n\right) &\\ \le\; \exp\left(-\min\left\{\frac{16}{3}k\ln n\,,\ \frac{3\cdot(16/3)k\ln n}{4}\right\}\right) &\\ \le\; n^{-4k}, &\end{aligned} \tag{6}$$

where in the first inequality we used $t^2 \ge (4\mu)\cdot(\frac{16}{3}k\ln n)$. From Lemma 10 we get $\mu \le 2\sqrt{2}nw$, and further using the assumption $w \ge \frac{\log n}{n}$ we see that

$$20wnk \;\ge\; \left(10\sqrt{2} + \frac{16}{3}\right)wnk \;\ge\; 5\mu + \frac{16}{3}k\ln n,$$

and hence inequality (6) upper bounds $\Pr(N_{u,w} \ge 20wnk)$. This concludes the proof. □

To prove the theorem we need to show that with high probability we simultaneously have $N_{u,w} \le 60nwk$ for all $u \in \mathbb{S}^{k-1}$ and $w \ge \frac{\log n}{n}$. By discretizing the u's and w's, we will construct a finite family of $N_{u,w}$'s in a way that it will be enough to control this family.

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 $\|u' - u\|_2 \le \varepsilon$. It is well-known that there is such a net of size at most $(\frac{3}{\varepsilon})^k$ (Corollary 4.2.13 [30]). Also define the discretized set of widths $\mathbb{W}' := \{\frac{\log n}{n}, \frac{2\log n}{n}, ..., \sqrt{k} + \frac{\log n}{n}\}$ so that $|\mathbb{W}'| = \frac{n\sqrt{k}}{\log n} + 1$. 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 $\mathcal{S}'$ so that $\|u' - u\|_2 \le \varepsilon$ and taking $w' \in \mathbb{W}'$ so that $w' \in [w + \frac{\log n}{n}, w + \frac{2\log n}{n}]$.

Lemma 13. *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 $N_{u,w} \le N_{u',w'}$*
2. *If the width satisfies $w \ge \frac{\log n}{n}$, then $w' \le 3w$.*

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

$$\begin{aligned} |\langle y, u'\rangle - \langle y, u\rangle| = |\langle y, u' - u\rangle| &\le \|y\|_2 \|u' - u\|_2 \\ &\le \sqrt{k}\varepsilon = \frac{\log n}{n}, \end{aligned}$$

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

Proof of Theorem 9. We need to show

$$\Pr\left[\bigvee_{u \in \mathbb{S}^{k-1}, w \ge \frac{\log n}{n}} (N_{u,w} > 60wnk)\right] \le \frac{1}{n}. \tag{7}$$

Since there are only n Y^j's, we always have $N_{u,w} \le n$ and hence it suffices to consider $w \in [\frac{\log n}{n}, \frac{1}{60k}]$, in which case we can use the inequalities $N_{u,w} \le N_{u',w'}$ and $w' \le 3w$ from the previous lemma; in particular, the event $(N_{u,w} > 60wnk)$ implies the event $(N_{u',w'} > 20w'nk)$. Thus, using Lemma 12

$$\begin{aligned} \text{LHS of (7)} &\le \Pr\left[\bigvee_{u' \in \mathbb{S}', w' \in \mathbb{W}'} (N_{u',w'} > 20w'nk)\right] \\ &\le \sum_{u' \in \mathbb{S}', w' \in \mathbb{W}'} \Pr(N_{u',w'} > 20w'nk) \\ &\le |\mathbb{S}'|\,|\mathbb{W}'|\, n^{-4k} \\ &\le \left(\frac{3n\sqrt{k}}{\log n}\right)^{k+1} n^{-4k} \le \frac{1}{n}, \end{aligned}$$

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

7 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 $\mathcal{I}$ of $\mathrm{IP}(b)$, with probability at least $1 - \frac{1}{n} - 2^{-\alpha a_2}$ the number of good points G relative to this instance at most $n^{O(m+\alpha a_1 \log m)}$.

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

1. Theorem 2 holds for $\mathcal{I}$, namely $\mathrm{IPGap}(\mathcal{I})$ is at most $\frac{\alpha a_1 \log^2 n}{n}$
2. The bound of Lemma 9 for the 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 second item (Lemma 9), under E we have that for every $\lambda \in \mathbb{R}^m$

$$|J_{\mathrm{rem}}(\lambda)| \le 60(m+1)2\log n,$$

since $x(\lambda)_j = \star$ iff $d((c_j, A^j), H(\lambda)) \le \frac{2\log n}{n}$ and hence $J_{\mathrm{rem}}(\lambda) = \{j : d((c_j, A_j), H(\lambda)) \le \frac{2\log n}{n}\}$ is relative to a slab of width $\frac{2\log n}{n}$, and similarly

$$|J_\ell(\lambda)| \le 60(m+1)2^{\ell+1}\log n$$

for all $\ell \ge 1$. In addition, from the first item in the definition of E, under E we have that $C := \frac{n}{\log n}\cdot\mathrm{IPGap}(\mathcal{I})$ is at most $\alpha a_1 \log n$. Therefore, using the standard estimate $\binom{a}{\le b} \le (4a/b)^b$ that holds for $a \ge 4b$, we get that, under E, for every $\lambda \in \mathbb{R}^m$

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

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

$$\begin{aligned}
\prod_{\ell<\frac{\log a_1}{2}} \binom{|J_\ell(\lambda)|}{\le C/2^\ell} &\le \prod_{\ell<\frac{\log a_1}{2}} 2^{|J_\ell(\lambda)|} \\
&\le 2^{O(ma_1 \log a_1 \log n)} \\
&= n^{O(ma_1 \log a_1)}.
\end{aligned}$$

Plugging these bounds on Lemma 8, we get that under E the number of good points G relative to the instance $\mathcal{I}$ is upper bounded as

$$\begin{aligned}
|G| &\le O(n)^m \cdot 2^{m\log n} \cdot n^{O(ma_1\log a_1)} \cdot \left(\frac{O(m)}{\alpha a_1}\right)^{O(\alpha a_1 \log n)} \\
&\le n^{O(ma_1 \log a_1 + \alpha a_1 \log m)}.
\end{aligned}$$

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

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

nodes. This concludes the proof of Theorem 1.

8 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 8 hold in such generality, which combined give the following.

Corollary 2. *Consider any instance of IP(b) 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*

$$2\left((cst \cdot n)^m \cdot \max_{\lambda\in\mathbb{R}^m} 2^{|J_{rem}(\lambda)|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell(\lambda)|}{\leq C/2^\ell}\right)\binom{n}{\leq m} + 1$$

nodes, where cst is the constant from Lemma 6 and $C := \frac{n}{\log n} \cdot IPGAP(b)$.

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 $|J_\ell(\lambda)|$).

Acknowledgments

We thank the anonymous reviewers for their very helpful comments, in particular about the generality of the results.

References

[1] Karen Aardal, Robert E Bixby, Cor AJ Hurkens, Arjen K Lenstra, and Job W Smeltink. Market split and basis reduction: Towards a solution of the cornuéjols-dawande instances. *INFORMS Journal on Computing*, 12(3):192–202, 2000.

[2] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[3] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Oper. Res.*, 62(4):876–890, 2014. ISSN 0030-364X. doi: 10.1287/opre.2014.1289. URL http://dx.doi.org/10.1287/opre.2014.1289.

[4] Keith Ball. Cube slicing in rn. *Proceedings of the American Mathematical Society*, 97(3):465–473, 1986. ISSN 00029939, 10886826. URL http://www.jstor.org/stable/2046239.

[5] René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. In Lawrence L. Larmore and Michel X. Goemans, editors, *STOC*, pages 232–241. ACM, 2003. ISBN 1-58113-674-9. URL http://dblp.uni-trier.de/db/conf/stoc/stoc2003.html#BeierV03.

[6] René Beier and Berthold Vöcking. Probabilistic analysis of knapsack core algorithms. In J. Ian Munro, editor, *SODA*, pages 468–477. SIAM, 2004. ISBN 0-89871-558-X. URL http://dblp.uni-trier.de/db/conf/soda/soda2004.html#BeierV04.

[7] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

[8] Robert Bixby and Edward Rothberg. Progress in computational mixed integer programming–a look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37, 2007.

[9] Kevin K. H. Cheung, Ambros M. Gleixner, and Daniel E. Steffy. Verifying integer programming results. In Friedrich Eisenbrand and Jochen Könemann, editors, *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, volume 10328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, 2017. doi: 10.1007/978-3-319-59250-3"_13. URL https://doi.org/10.1007/978-3-319-59250-313.

[10] Vasek Chvátal. Hard knapsack problems. *Operations Research*, 28(6):1402–1411, 1980.

[11] Thomas F. Coleman, editor. *Chapter 3 Large sparse linear programming*, pages 35–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 1984. ISBN 978-3-540-38796-1. doi: 10.1007/3-540-12914-6_3. URL https://doi.org/10.1007/3-540-12914-63.

[12] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

[13] Sanjeeb 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*, pages 145–160. Springer, 2002.

[14] Santanu S Dey, Marco Molinaro, and Qianyi Wang. Approximating polyhedra with sparse inequalities. *Mathematical Programming*, 154(1-2):329–352, 2015.

[15] Santanu S Dey, Marco Molinaro, and Qianyi Wang. Analysis of sparse cutting planes for sparse milps with applications to stochastic milps. *Mathematics of Operations Research*, 43(1):304–332, 2018.

[16] M. E. Dyer and A. M. Frieze. Probabilistic analysis of the multidimensional knapsack problem. *Mathematics of Operations Research*, 14(1):162–176, 1989. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3689842.

[17] Komei Fukuda, Shigemasa Saito, Akihisa Tamura, and Takeshi Tokuyama. Bounding the number of k-faces in arrangements of hyperplanes. *Discrete Applied Mathematics*, 31(2):151 – 165, 1991. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(91)90067-7. URL http://www.sciencedirect.com/science/article/pii/0166218X91900677.

[18] Andrew V. Goldberg and Alberto Marchetti-Spaccamela. On finding the exact solution of a zero-one knapsack problem. In Richard A. DeMillo, editor, *STOC*, pages 359–368. ACM, 1984. URL http://dblp.uni-trier.de/db/conf/stoc/stoc84.html#GoldbergM84.

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

[20] Alisa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[21] Hendrik Willem Lenstra, Arjen K Lenstra, Lászl'o Lovász, et al. Factoring polynomials with rational coeficients. *Mathematische Annalen*, 261:515–534, 1982.

[22] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[23] Jeff T Linderoth and Martin WP Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

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

[25] Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002. ISBN 0387953744.

[26] Marco Molinaro and R. Ravi. The geometry of online packing linear programs. *Math. Oper. Res.*, 39 (1):46–59, February 2014. ISSN 0364-765X. doi: 10.1287/moor.2013.0612. URL http://dx.doi.org/10.1287/moor.2013.0612.

[27] Gábor Pataki, Mustafa Tural, and Erick B. Wong. *Basis Reduction and the Complexity of Branch-and-Bound*, pages 1254–1261. doi: 10.1137/1.9781611973075.100. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611973075.100.

[28] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[29] V. Vazirani. *Approximation algorithms*. Springer, 2001. ISBN 9783540653677. URL http://books.google.com/books?id=EILqAmzKgYIC.

[30] Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. doi: 10.1017/9781108231596.

[31] Matthias Walter. Sparsity of lift-and-project cutting planes. In *Operations Research Proceedings 2012*, pages 9–14. Springer, 2014.

[32] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

Appendix

A Proof of Lemma 8

From Lemma 6, there are vectors $\lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R}^m$, for $k \leq (\text{cst} \cdot n)^m$, such that every dual-based solution $x(\lambda)$ equals one of the $x(\lambda_i)$'s. Recall that $B_{x(\lambda_i)}$ is the set of all slices b' where $\text{LP}_=(b')$ has an optimal solution compatible with $x(\lambda_i)$, and from Lemma 5 every slice b' belongs to one such set $B_{x(\lambda_i)}$. Let $G_i = \{x \in G : Ax \in B_{x(\lambda_i)}\}$ be the good solutions in a slice in $B_{x(\lambda_i)}$. Then $G = \bigcup_i G_i$ and it suffices to upper bound the size of each G_i.

We claim that

$$|G_i| \leq 2^{|J_{\text{rem}}(\lambda_i)|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell(\lambda_i)|}{\leq C/2^\ell}. \tag{8}$$

To see this, notice that every solution in G_i can be though as being created by starting with the vector $x(\lambda_i)$ and then changing some of its coordinates, and because of Corollary 1 we:

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

Since there are at most $2^{|J_{\text{rem}}(\lambda_i)|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell(\lambda_i)|}{\leq C/2^\ell}$ options in this process, we have the upper bound of (8).

Adding this bound over all i's, we get

$$|G| \leq \sum_{i=1}^{k} \left[2^{|J_{\text{rem}}(\lambda_i)|} \cdot \prod_{\ell=1}^{\log C} \binom{|J_\ell(\lambda_i)|}{\leq C/2^\ell} \right],$$

and the lemma directly follows.