Bounds on the Size of Branch-and-Bound Proofs for Integer Knapsacks

Bala Krishnamoorthy[*]

Department of Mathematics

P.O. Box 643113, Washington State University

Pullman, WA 99164

Abstract

Using a direct counting argument, we derive lower and upper bounds for the number of nodes enumerated by linear programming-based branch-and-bound (B&B) method to prove the infeasibility of an integer knapsack problem. We prove by example that the size of the B&B tree could be exponential in the worst case.

Keywords: integer programming, knapsack problem, branch-and-bound proofs.

1 Introduction

Linear programming-based branch-and-bound (B&B) methods and cutting plane methods are the standard techniques used to solve integer programming (IP) problems. Most commercial solvers use cutting planes in conjunction with B&B (termed branch-and-cut methods). The feasibility version of the integer programming problem, as is the 0–1 integer programming problem, is known to be NP-complete [11, Chap. 15]. It is highly unlikely that there exists a polynomial-time algorithm that uses branch-and-cut methods to solve integer programs. In fact, some classes of branch-and-bound algorithms have been shown to have worst-case exponential time complexity for solving 0–1 integer programs. Jeroslow [7] studied the problem max $\{x_1 | 2x_1 + \cdots + 2x_n = n,\ x_1, \ldots, x_n \in \{0,1\}\}$. For odd n, he proved that any B&B procedure, independent of the order in which the variables are fixed and the method for choosing the nodes, must expand out at least $2^{\lceil n/2 \rceil}$ nodes before it discovers that the above system is inconsistent. Chvátal [4] considered the performance of a class of recursive algorithms that combined the powers of B&B, dynamic programming, and rudimentary divisibility arguments

[*]email: kbala@wsu.edu (corresponding author; postal address as given above.)

for solving the 0–1 knapsack problem. He identified a class of 0–1 knapsack instances for which the running time of every recursive algorithm studied is bounded below by $2^{n/10}$ for an overwhelming majority of these problems, when the number of variables, n, is sufficiently large.

Even though exponential lower bounds have been derived for the worst case complexity of several branch-and-cut algorithms applied to 0–1 integer programs, similar results are found lacking when it comes to general (or pure) integer programming problems. One reason for not studying them separately might be the fact that an instance of pure integer programming problem can be transformed in polynomial time to an instance of a 0–1 integer programming problem [10, p126]. In this paper, we provide a lower bound for the number of nodes examined by B&B to prove the infeasibility of a general integer knapsack problem that could be exponential in the number of variables for a class of these problems. The method used to derive this bound is fundamentally different from those used by Jeroslow [7] and Chvátal [4]. In the problems studied by these authors, unless a sufficiently large number of variables are fixed to integer values by the B&B algorithm considered, the LP-relaxation at the node would be feasible, and hence the node cannot be pruned. Following this idea, it is shown that there could exist an exponential number of such nodes. In contrast to these approaches, we try to enumerate the total number of B&B nodes that need to be examined for the proof of infeasibility of an integer knapsack problem, and derive lower and upper bounds for the same. For a class of decomposable instances of the knapsack problem where the coefficients have the form $a_i = p_i M + r_i$, for $i = 1, \ldots, n$, with $p_i, M, \in \mathbb{Z}_{>0}$, $r_i \in \mathbb{Z}$, and M sufficiently large, we show that this lower bound varies as M^{n-1}. Hence we provide a unique and direct method for showing the exponential size of a branch-and-bound proof of infeasibility of a general integer program, without considering the equivalent 0–1 integer program.

We begin the detailed discussions by formally describing the branch-and-bound algorithm applied to integer feasibility problems.

2 Linear programming-based branch-and-bound

The branch-and-bound algorithm used in most commercial solvers is usually described for the optimization version of the integer programming problem [16, see Chap. 7]. We describe the branch-and-bound algorithm for the feasibility version, where we try to answer the following question with a YES or a NO.

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}, \quad \text{Is } P \cap \mathbb{Z}^n = \emptyset\,? \tag{1}$$

Here, A is an $m \times n$ matrix and b is an m-vector. The aim is to find an integral x that is in P, or else prove that no such x exists. We formally describe the binary branching version of the B&B algorithm below. Here, $\mathcal{S}$ is the set of sub-problems (nodes) maintained throughout the execution of the algorithm and P is as defined in Equation (1).

B&B for IP-Feasibility

```
Step 1. Initialization       $P^0 = P, \mathcal{S} = \{P^0\},\ ans = \{\};$ goto Step 2.
Step 2. Examine node         if  $\mathcal{S} = \emptyset,\ ans =$ YES, STOP;
                             else choose $P^k \in \mathcal{S},\ \mathcal{S} = \mathcal{S} \setminus \{P^k\}$
                                 if  $P^k = \emptyset,$ goto Step 2.
                                 else find $x^k \in P^k,$ goto Step 3.                  (BBIPFeas)
Step 3. Termination          if  $x^k \in \mathbb{Z}^n,\ ans =$ NO, STOP;
                             else goto Step 4.
Step 4. Binary branching     choose $x_i^k \notin \mathbb{Z},$ let $P^{k+1} = P^k \cap \{x \in \mathbb{R}^n \mid x_i \leq \lfloor x_i^k \rfloor\},$
                                        $P^{k+2} = P^k \cap \{x \in \mathbb{R}^n \mid x_i \geq \lceil x_i^k \rceil\},$
                             $\mathcal{S} = \mathcal{S} \cup \{P^{k+1}, P^{k+2}\},$ goto Step 2.
```

An alternative, and arguably smarter, method for creating the sub-problems was suggested by Aardal et al. [1]. They modified the branching step as follows.

```
Step 4. Integer branching    choose $x_i^k \notin \mathbb{Z},$  find $\delta_i = \min\{x_i \mid x \in P^k\}$ and
                                                $\gamma_i = \max\{x_i \mid x \in P^k\}.$
                             for $j = 1$ to $\lfloor \gamma_i \rfloor - \lceil \delta_i \rceil + 1$
                                 $P^{k+j} = P^k \cap \{x \in \mathbb{R}^n \mid x_i = \lceil \delta_i \rceil + j - 1\};$        (2)
                                 $\mathcal{S} = \mathcal{S} \cup \{P^{k+j}\};$
                             end_for
                             goto Step 2.
```

The size of the B&B tree created by this algorithm is given by the number of times *Step* 2 is executed. We prune nodes only based on *infeasibility* [16, see p.94], as given in *Step* 2 (when $if\ P^k = \emptyset$ is TRUE). Since the algorithm will terminate when the first integer solution is found (if there exists an integer solution), pruning by *optimality* does not apply here. Again, since we are not working with a particular objective function, pruning by *bound* does not apply either. In order to analyze the worst-case performance of this B&B algorithm, we concentrate on IP feasibility problems that are known to be infeasible to start with. To prove infeasibility, the B&B for IP-Feasibility algorithm described above will have to examine *all* possible nodes, and have to branch on all the fractional variables at each node that is LP-feasible. Hence, the size of the B&B tree is independent of the order in which the nodes are examined and the order in which the variables are branched on.

3 An infeasible, equality-constrained knapsack problem

In order to analyze the performance of the B&B algorithm (BBIPFeas) using both binary and integer branching, we study a special case of the problem 1, where A has only one row ($m = 1$). Let $a_1, a_2, \ldots, a_n$ be positive integers with $\gcd(a_1, \ldots, a_n) = 1$. Without loss of generality, we assume $a_1 < a_2 < \cdots < a_n$. Thus $A = (a_1, \ldots, a_n)$, and for ease of notation, we set $a = (a_1, \ldots, a_n)$. Also, let a_0 be a positive integer such that $a_i \le a_0$ and $\gcd(a_i, a_0) = 1$ for $i = 1, \ldots, n$. The feasibility IP in Equation (1) becomes

$$P = \{x \in \mathbb{R}^n \mid a\,x = a_0,\ x \ge 0\}, \quad \text{Is } P \cap \mathbb{Z}^n = \emptyset\,? \tag{3}$$

We concentrate on instances for which the answer to the above question is YES. Hence the task in hand is to *prove* that P does not contain any integer vectors. We say that P is *integer-infeasible* if P does not contain any integer vectors. When P is integer-infeasible, the branch-and-bound algorithm (BBIPFeas) could not possibly get "lucky" by identifying any integer vectors early on (for example, by using an appropriate objective function). The largest a_0 such that P is integer-infeasible for a given instance of a is called the *Frobenius number* of $a_1, \ldots, a_n$ and is denoted by $F(a)$ or $F(a_1, \ldots, a_n)$. Problem (3) is directly related to the problem of Frobenius, which aims to find the largest natural number that cannot be expressed as a positive integer linear combination of a given set of natural numbers [13, pg.376]. Finding the Frobenius number is NP-hard. For $n = 2$, it is known that $F(a_1, a_2) = a_1 a_2 - a_1 - a_2$. For $n = 3$, $F(a)$ can be computed in polynomial time - for example, see [15, 6, 12]. Kannan [8] developed a polynomial time algorithm to compute the Frobenius number for every fixed n. Further references on the Frobenius problem (for general n) are available in the paper of Hujter and Vizvári [6], and also in the paper by Selmer [14].

Certain upper bounds on $F(a)$ are known for the general case. Brauer [3] showed that $F(a) \le a_1 a_n - a_1 - a_n$. Other upper bounds are provided by Erdös and Graham [5], and also by Selmer [14]. We are interested in the Frobenius problem because an instance of problem (3) could be hard to solve by branch-and-bound if it is integer-infeasible, and the value of the right-hand-side constant a_0 is large.

The B&B algorithm (BBIPFeas) will have to examine most of the possible combinations of $x_1, \dots, x_n$ with $0 \le x_i \le a_0/a_i$. In this sense, the *hardest* instance of problem (3) for a given a is obtained when we set $a_0 = F(a)$. Hence we could use the bounds on $F(a)$ mentioned above when trying to analyze the performance of the B&B algorithm (BBIPFeas) on infeasible instances of the problem (3). Aardal and Lenstra [2] derived the first known lower bound on $F(a)$ for the class of problems where the numbers a_i decompose as $p_iM + r_i,\ for\ i = 1, \dots, n$, with $p_i, M \in \mathbb{Z}_{>0}$ and $r_i \in \mathbb{Z}$. This lower bound will be used in our proof to show the exponential size of the B&B tree generated to prove the infeasibility of an instance of problem (3).

4 Branch-and-bound on the knapsack problem

We derive lower and upper bounds on the total number of branch-and-bound nodes (sub-problems) that the B&B algorithm (BBIPFeas) will examine when applied to an integer-infeasible instance of the equality-constrained knapsack feasibility problem (3). We provide an inductive argument. The instance of this problem for $n = 2$ is

$$P = \{x_1, x_2 \in \mathbb{R} \mid a_1x_1 + a_2x_2 = a_0,\ x_1 \ge 0, x_2 \ge 0\}, \quad \text{Is } P \cap \mathbb{Z}^2 = \emptyset\,? \tag{4}$$

Figure 1 illustrates the performance of B&B using binary branching on an infeasible instance with $n = 2$. Starting from the point on the vertical axis, the B&B algorithm examines all the points marked. Also, at each of these marked points, one node would be pruned due to infeasibility. Hence the total number of B&B nodes that are examined for proving infeasibility would be $2\lceil a_0/a_1\rceil + 2\lceil a_0/a_2\rceil$. Note that the total number of sub-problems remains the same irrespective of which marked point on the line (representing the LP-feasible set of points) is examined to start with. *All* the marked points (or the corresponding nodes) are examined.

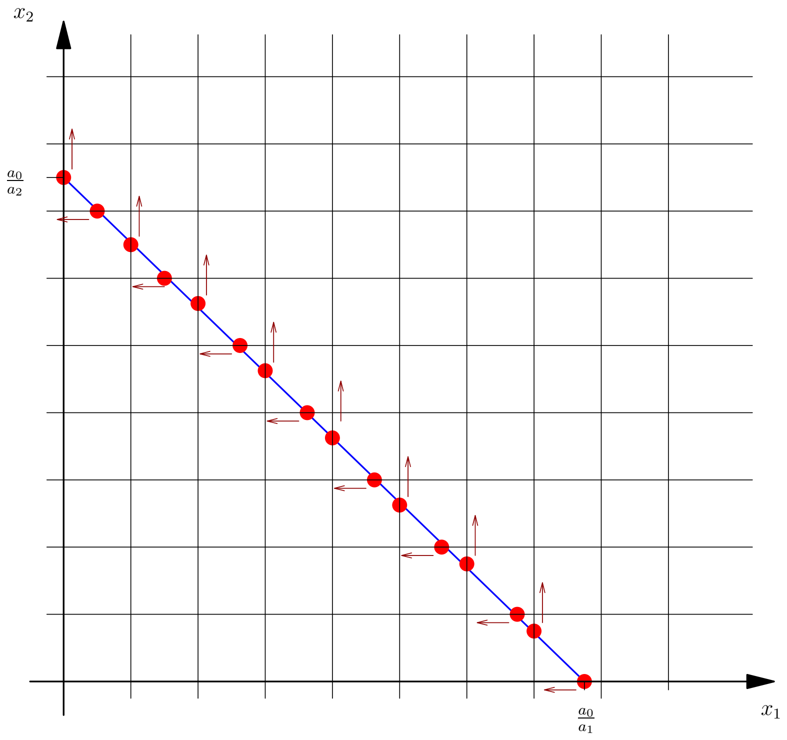

Figure 1: B&B on the infeasible 2D instance. Marked points represent all the nodes that are examined. The arrows at the points indicate the direction of rounding that gives an infeasible node in each case.

We generalize this idea of counting the number of nodes examined to higher dimensions. Let $\mathrm{N}(a_r, k)$ denote the number of B&B nodes that need to be examined when k of the integer variables are fractional (i.e. available for branching on) and the remaining unused capacity of the knapsack equality is a_r. In other words, a_r is obtained by subtracting from the original capacity a_0 the contributions corresponding to all the $n - k$ x_i's that have been fixed to integral values. We derive a relation between $\mathrm{N}(a_r, k+1)$ and $\mathrm{N}(a_r, k)$ by counting the branches created when x_{k+1} is branched on. Given the unused capacity of a_r, we need to consider the possible integer values that x_{k+1} can take in the range $0 \leq x_{k+1} \leq a_r/a_{k+1}$. Similar to the case of (say) x_1 in the 2D instance, branching on x_{k+1} would create $2\lceil a_0/a_{k+1} \rceil$ B&B nodes, half of which are pruned due to infeasibility and the other half of the nodes have to be branched on further. We obtain the following recursion.

$$\mathrm{N}(a_r, k+1) = 2\left\lceil \frac{a_r}{a_{k+1}} \right\rceil + \sum_{i=0}^{\left\lfloor \frac{a_r}{a_{k+1}} \right\rfloor} \mathrm{N}(a_r - i a_{k+1}, k) \tag{5}$$

In the case of integer branching on x_{k+1}, we obtain $\lceil a_0/a_{k+1} \rceil$ sub-problems by fixing x_{k+1} to each of the integers $0, 1, \ldots, \lfloor a_0/a_{k+1} \rfloor$, and each of these sub-problems need to be examined further. Hence we get

$$\mathrm{N}(a_r, k+1) = \left\lceil \frac{a_r}{a_{k+1}} \right\rceil + \sum_{i=0}^{\left\lfloor \frac{a_r}{a_{k+1}} \right\rfloor} \mathrm{N}(a_r - i a_{k+1}, k) \tag{6}$$

For the two-dimensional case, we create $\lceil a_0/a_2 \rceil$ branches by branching on x_2 (as we assume $a_1 \leq a_2$, it is beneficial to branch on x_2 first), and each of these branches will be pruned by infeasibility. Hence, the total number of B&B nodes examined is only $\lceil a_0/a_2 \rceil$. Working with the recursions (5) and (6), we can use inductive arguments to derive lower and upper bounds on $\mathrm{N}(a_r, k)$. Of course, our goal is to derive these bounds for $\mathrm{N}(a_0, n)$. We state and prove the following theorem that provides the bounds in question.

Theorem 4.1. *The following statements are true for $k \geq 2$.*

1. *Under binary branching* $\mathrm{N}(a_0, k)$ *satisfies*

$$\frac{4}{(k-1)!} \frac{a_0^{(k-1)}}{(a_2 a_3 \cdots a_k)} \;\leq\; \mathrm{N}(a_0, k) \;\leq\; 4\frac{a_0}{a_k} + \frac{4\,(2)^{k(k+1)/2}}{k!} \frac{a_0^k}{(a_1 a_2 \cdots a_k)} \tag{7}$$

2. *Under integer branching* $\mathrm{N}(a_0, k)$ *satisfies*

$$\frac{1}{(k-1)!} \frac{a_0^{(k-1)}}{(a_2 a_3 \cdots a_k)} \;\leq\; \mathrm{N}(a_0, k) \;\leq\; \frac{a_0}{a_k} + \frac{(2)^{k(k+1)/2}}{k!} \frac{a_0^k}{(a_1 a_2 \cdots a_k)} \tag{8}$$

Proof of Statement 1: We use induction on k. First, for $k = 2$, we gave the exact value for $\mathrm{N}(a_0, 2)$ in Section 4. Using that result, we obtain

$$\mathrm{N}(a_0, 2) = 2\left\lceil \frac{a_0}{a_1} \right\rceil + 2\left\lceil \frac{a_0}{a_2} \right\rceil \geq 4\left\lceil \frac{a_0}{a_2} \right\rceil \geq 4\left(\frac{a_0}{a_2}\right)$$

since $a_1 \leq a_2 \leq \cdots \leq a_n$. It is also straightforward to check the upper bound for $k = 2$.

$$\mathrm{N}(a_0, 2) = 2\left\lceil \frac{a_0}{a_1} \right\rceil + 2\left\lceil \frac{a_0}{a_2} \right\rceil \leq 4\left\lceil \frac{a_0}{a_1} \right\rceil \leq 4\left(\frac{a_0}{a_1} + 1\right) \leq 4\left(\frac{a_0}{a_1}\right)\left(4\frac{a_0}{a_2}\right) + 4\left(\frac{a_0}{a_2}\right)$$

To prove the induction step for general $k \geq 2$, we will use the following lemma.

Lemma 4.2.

$$\textit{For } 0 < \rho \leq 1, \qquad \frac{1}{k\rho} \;\leq\; \sum_{i=0}^{\left\lfloor \frac{1}{\rho} \right\rfloor} (1 - i\rho)^{k-1} \;\leq\; \frac{(1+\rho)^k}{k\rho} \tag{9}$$

Proof: Let $m = \left\lfloor \frac{1}{\rho} \right\rfloor$. We can write $m = \frac{1}{\rho} - \epsilon$ for some $0 \leq \epsilon < 1$. Hence, the above sum becomes

$$\begin{aligned} \sum_{i=0}^{\left\lfloor \frac{1}{\rho} \right\rfloor} (1 - i\rho)^{k-1} &= \sum_{i=0}^{m} \left(1 - i\frac{1}{m+\epsilon}\right)^{k-1} \\ &= \frac{1}{(m+\epsilon)^{k-1}} \sum_{i=0}^{m} (m - i + \epsilon)^{k-1} = \frac{1}{(m+\epsilon)^{k-1}} \sum_{i=0}^{m} (i + \epsilon)^{k-1} \end{aligned}$$

We note that $\int_0^m (x+\epsilon)^{k-1} \mathrm{d}x$ is a lower bound for $\sum_{i=0}^{m} (i+\epsilon)^{k-1}$. Similarly, $\int_0^{m+1} (x+\epsilon)^{k-1} \mathrm{d}x$ is an upper bound for the same sum. The value of the first integral is the area under the curve $(x+\epsilon)^{k-1}$ from $x = 0$ to $x = m$, and that of the second integral is the area under the same curve from $x = 0$ to $x = m + 1$. The sum, on the other hand, can be considered as the addition of the areas of the unit-width rectangles that rise above (or below) the curve at each unit interval. These relationships are illustrated in Figure 2.

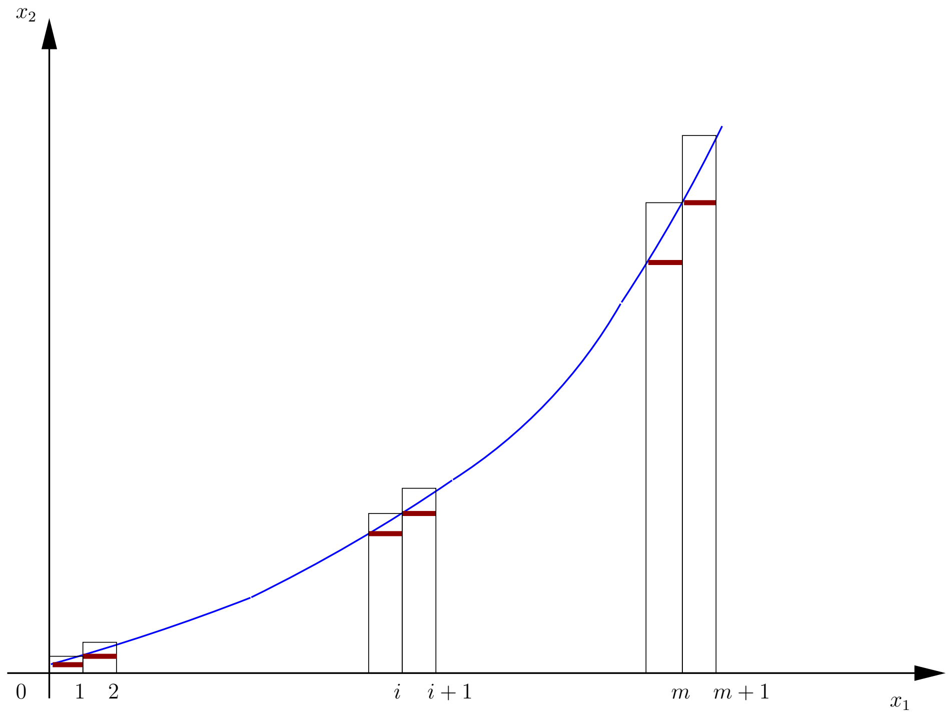

Figure 2: The area under the curve is a lower bound for the sum of the areas of the m unit-width rectangles rising above the curve (the right edges of these rectangles are located at $x = 1, 2 \ldots, m$). The area under the curve is also an upper bound if we consider the areas of the $m + 1$ rectangles below the curve (the left edges of these rectangles are located at $x = 0, 1, \ldots, m + 1$). The top edges of the second set of $m + 1$ rectangles are drawn as thicker lines.

Thus, we obtain

$$\sum_{i=0}^{m}(i+\epsilon)^{k-1} \;\geq\; \epsilon^{k-1} + \int_0^m (x+\epsilon)^{k-1}\mathrm{d}x \;=\; \epsilon^{k-1} + \frac{(m+\epsilon)^k}{k} \;-\; \frac{\epsilon^k}{k} \;\geq\; \frac{(m+\epsilon)^k}{k},$$

and $$\sum_{i=0}^{m}(i+\epsilon)^{k-1} \;\leq\; \int_0^{m+1} (x+\epsilon)^{k-1}\mathrm{d}x \;=\; \frac{(m+1+\epsilon)^k}{k} \;-\; \frac{\epsilon^k}{k} \;\leq\; \frac{(m+1+\epsilon)^k}{k}.$$

Hence, we have

$$\frac{m+\epsilon}{k} \;\leq\; \frac{1}{(m+\epsilon)^{k-1}}\sum_{i=0}^{m}(i+\epsilon)^{k-1} \;\leq\; \frac{m+\epsilon}{k}\left(1+\frac{1}{m+\epsilon}\right)^k.$$

The lemma follows directly from the above set of inequalities.

Returning to the proof of Theorem 4.1, let us assume that the result holds for $\mathrm{N}(a_0, k)$. To prove the lower bound for $\mathrm{N}(a_0, k+1)$, we ignore the first term on the right hand side of the recursion in Equation (5).

$$\mathrm{N}(a_0, k+1) \geq \sum_{i=0}^{\left\lfloor \frac{a_0}{a_{k+1}} \right\rfloor} \frac{4}{(k-1)!}\frac{(a_0 - i a_{k+1})^{k-1}}{(a_2 a_3 \cdots a_k)} = \frac{4a_0^{k-1}}{(k-1)! a_2 \cdots a_k} \sum_{i=0}^{\left\lfloor \frac{a_0}{a_{k+1}} \right\rfloor} \left(1 - i\frac{a_{k+1}}{a_0}\right)^{k-1}$$

Applying the lower bound given in lemma 4.2 to the above sum, we obtain

$$\mathrm{N}(a_0, k+1) \geq \frac{4a_0^{k-1}}{(k-1)! a_2 \cdots a_k}\left(\frac{1}{k}\frac{a_0}{a_{k+1}}\right) = \frac{4a_0^k}{k! a_2 \cdots a_{k+1}},$$

which is the desired result for $k+1$.

To prove the upper bound, we again start with Equation (5).

$$\mathrm{N}(a_0, k+1) \;\leq\; 4\frac{a_0}{a_{k+1}} \;+\; \sum_{i=0}^{\left\lfloor \frac{a_0}{a_{k+1}} \right\rfloor} 4\left(\frac{a_0 - i\, a_{k+1}}{a_k}\right) \;+\; \sum_{i=0}^{\left\lfloor \frac{a_0}{a_{k+1}} \right\rfloor} \frac{4\,(2)^{k(k+1)/2}}{k!}\frac{(a_0 - i\, a_{k+1})^k}{(a_1 a_2 \ldots a_k)}$$

Applying the upper bound given in lemma 4.2 to the sums in the above expression gives

$$\mathrm{N}(a_0, k+1) \;\leq\; 4\frac{a_0}{a_{k+1}} \;+\; 4\frac{(a_0 + a_{k+1})^2}{2\, a_k a_{k+1}} \;+\; \frac{4\,(2)^{k(k+1)/2}}{k!(a_1 a_2 \ldots a_k)}\frac{(a_0 + a_{k+1})^{k+1}}{(k+1) a_{k+1}}$$

$$\Longrightarrow \mathrm{N}(a_0, k+1) \;\leq\; 4\frac{a_0}{a_{k+1}} \;+\; \frac{4\,(2)^{\frac{k(k+1)}{2}}\,(2)^{k+1}\, a_0^{k+1}}{(k+1)!(a_1 a_2 \ldots a_{k+1})},$$

which is the upper bound for $k+1$. □

The last step in the proof might demand some explanation. We replace the a_{k+1} in the term $(a_0 +$

$a_{k+1})^{k+1}$ by a_0 (and $a_0 \geq a_{k+1}$ by assumption). By rewriting $(a_0 + a_0)^{k+1}$ as $(a_0 + a_{k+1} \ + \ (a_0 - a_{k+1})\ )^{k+1}$ and considering the binomial expansion of the same, one could see that the quadratic term in the right hand side of the inequality $\left(4\frac{(a_0 + a_{k+1})^2}{2\ a_k a_{k+1}}\right)$ would be dominated by the terms added on. The proof for statement 2 (integer branching case) is similar to the one given above.

5 A decomposable knapsack problem

Aardal and Lenstra [2] studied a class of the equality-constrained knapsack problem (3) in which the coefficients decompose as $a_i = p_i M + r_i$ for $i = 1, \ldots, n,$ where p_i's and M are natural numbers and r_i's are integers. For this class of problems, they derived a lower bound on $F(a)$ in which the dominating term was quadratic in M. The authors also calculated the upper bound on $F(a)$ using the formula provided by Brauer (see Section 3). We reproduce the expression for the lower bound (denoted by $f(p, M, r)$) from their paper.

$$f(p, M, r) = \frac{(M^2 p_j p_k + M(p_j r_k + p_k r_j) + r_j r_k)\left(1 - \frac{2}{M + (r_j/p_j)}\right)}{p_k r_j - p_j r_k} - \left(M + \frac{r_j}{p_j}\right) \tag{10}$$

In this expression, $r_j/p_j = \max_{i=1,\ldots,n}\{r_i/p_i\}$ and $r_k/p_k = \min_{i=1,\ldots,n}\{r_i/p_i\}$. We already noted that the hardest instances of the problem (3) for the B&B algorithm to solve are the ones in which $a_0 = F(a)$. The authors demonstrated by a computational study that the lower bound on $F(a)$ given in Equation (10) is very close to the actual value when the vectors p and r are chosen to be short compared to M. At the same time, these integer-infeasible instances (with huge M values) prove to be very hard for branch-and-bound to solve (prove integer infeasibility).

In order to study the worst case complexity of the B&B algorithm (BBIPFeas), we consider an integer-infeasible instance of the knapsack problem (3) with $a_0 = F(a)$. We assume that the coefficients decompose as $a = pM + r$, with M chosen sufficiently large so that the lower bound on $F(a)$ is valid (specifically, we require $M \geq \max\{(r_j/p_j) - 2(r_k/p_k),\ 2 - (r_j/p_j)\}$ - see [2]). In fact, we assume $M \gg p_i, r_i$ for all $i = 1, \ldots, n$. We analyze the lower bound on the number of B&B nodes for this problem given by Theorem 4.1, $\mathrm{N}(a_0, n)$, by using the lower bound $f(p, M, r)$ on $F(a)$. Noting the fact that $f(p, M, r)$ is quadratic in M, one can guess that the dominating term in the expression for the lower bound for $\mathrm{N}(a_0, n)$ in Equation (7) is M^{n-1}. We derive an expression for a lower bound on $\mathrm{N}(a_0, n)$ that contains precisely this dominating term. For ease of calculation, we work with a lower bound on $F(a)$ that is slightly different from the one given in Equation (10), as expressed below.

$$f_1(p, M, r) = \frac{(M + r_j/p_j)(M + 2r_k/p_k - r_j/p_j - 2)}{(r_j/p_j - r_k/p_k)} \tag{11}$$

One can verify that $f_1(p, M, r) = f(p, M, r) - 2$. Hence, $F(a) \geq f_1(p, M, r)$. Using this lower bound in Equation (7) gives

$$\begin{aligned}
\mathrm{N}(a_0, n) &\geq \frac{4}{(n-1)!} \frac{[f_1(p, M, r)]^{n-1}}{(a_2 a_3 \cdots a_n)} = \frac{4}{(n-1)!} \frac{\left(M + \dfrac{r_j}{p_j}\right)^{n-1} \left(M + \dfrac{2r_k}{p_k} - \dfrac{r_j}{p_j} - 2\right)^{n-1}}{\left(\dfrac{r_j}{p_j} - \dfrac{r_k}{p_k}\right)^{n-1} (p_2 M + r_2) \cdots (p_n M + r_n)} \\
&= \frac{4}{(n-1)!} \frac{1}{\left(\dfrac{r_j}{p_j} - \dfrac{r_k}{p_k}\right)^{n-1} (p_2 \cdots p_n)} \frac{\left(M + \dfrac{r_j}{p_j}\right)^{n-1} \left(M + \dfrac{2r_k}{p_k} - \dfrac{r_j}{p_j} - 2\right)^{n-1}}{\left(M + \frac{r_2}{p_2}\right) \cdots \left(M + \frac{r_n}{p_n}\right)} \\
&\geq \mathcal{C}(n) \left(M + \frac{2r_k}{p_k} - \frac{r_j}{p_j} - 2\right)^{n-1}, \qquad \text{where } \mathcal{C}(n) \text{ is independent of } M.
\end{aligned}$$

Under the assumption that $M \gg p_i, r_i$ for $i = 1, \ldots, n$, the above expression for the lower bound on $\mathrm{N}(a_0, n)$ directly leads to the main result of this paper.

Theorem 5.1. *The number of B&B nodes examined by the LP-based branch-and-bound algorithm BBIPFeas applied to an integer feasibility problem is exponential in the number of variables in the worst case.*

6 Analysis of the bounds

For the two-dimensional case, the number of nodes examined by the B&B algorithm (BBIPFeas), denoted by $\mathrm{N}(a_0, 2)$, is known exactly (see Section 4). Examining the form of the lower bound on $\mathrm{N}(a_0, n)$ as given in Theorem 4.1 for $n \geq 3$, one can see that its value will be large if a_0 is large compared to the individual coefficients a_i. For example, consider the $n = 3$ instance of the decomposable knapsack problem given by Aardal and Lenstra (Example 1 in [2]).

$$P = \{x \in \mathbb{R}^3 \mid 12223\, x_1 + 12224\, x_2 + 36672\, x_3 = 149389505,\ x \geq \mathbf{0}\}, \quad \text{Is } P \cap \mathbb{Z}^3 = \emptyset\,? \tag{12}$$

The coefficients decompose as $pM + r$, with $M = 12223$, $p = (1, 1, 3)$, and $r = (0, 1, 3)$. The Frobenius number for this instance is 149389505, which is also the capacity of the knapsack constraint. Using this value of a_0 in the bounds in inequalities (7) and (8) gives the lower bound on $\mathrm{N}(a_0, 3)$ as $99,568,562$ and the upper bound as 2.5961×10^{13} B&B nodes using binary branching, and $24,892,140$ and 6.4902×10^{12} B&B nodes using integer branching. If this problem is solved using CPLEX 8.0 under the default settings, the number of B&B nodes examined to prove infeasibility is $23,481,314$. With all cutting planes and presolve options in CPLEX 8.0 turned off, the number of B&B nodes goes up to $24,900,061$. The reader should note that the branch-and-bound algorithm employed by CPLEX is somewhat different from the basic B&B algorithm (BBIPFeas). The difference is quite striking even for the case of $n = 2$. Consider the knapsack instance with $a_1 = 26$, $a_2 = 49$ and $a_0 = 391$ (which is the Frobenius number for this instance). We obtain $\mathrm{N}(391, 2) = 48$ using binary branching and $\mathrm{N}(391, 2) = 12$ using integer branching. On the other hand, CPLEX 8.0 proves the infeasibility of this instance in a single B&B node, even with all the cutting planes and presolve turned off. Still, the number of B&B nodes examined by CPLEX without the use of cutting plane generation and presolve routines gives a fair idea of the actual number of sub-problems enumerated by the branch-and-bound algorithm (BBIPFeas) for the decomposable knapsack problems studied. Consider another example of this problem for $n = 4$ where the coefficients are given by $a = (322, 323, 645, 968)$. The lower bound on the Frobenius number is 103361. Using this value for a_0 actually creates an integer-infeasible instance. The lower bound on $\mathrm{N}(a_0, 4)$ is $3,600,171$ B&B nodes using binary branching ($900,043$ under integer branching). When solved using CPLEX 8.0 (with no cutting planes or presolve), $507,111$ nodes are examined.

The lower bound on $\mathrm{N}(a_0, n)$ is small when a_0 is of the same order of magnitude as the a_i's. For the class of instances where the knapsack coefficients do not necessarily decompose as described in Section 5, the Frobenius number is usually not much larger than the coefficients themselves (see the problem instances of this kind reported in [2] – these are the instances named prob11 to prob20 given in Table 1 of this paper). Consider the instance prob11 as an example. $n = 10$ in this case, and the coefficients range from 11615 to 130204, but do not decompose into two component vectors that are short, with a comparatively large multiplier for one of them. The Frobenius number for this instance is reported as 577134. Using this value for a_0, the lower and upper bounds on $\mathrm{N}(a_0, 10)$ become 7623 and 1.3645×10^{21} (under binary branching). At the same time, CPLEX 8.0 enumerates 41633 nodes, even if we allow the use of cutting planes and presolve. With the cutting planes and presolve options turned off, CPLEX 8.0 examines 88860 B&B nodes. Hence, one could surmise that the lower bound on $\mathrm{N}(a_0, n)$ could underestimate the actual number of B&B nodes enumerated when the coefficients are not decomposable.

The upper bound on $\mathrm{N}(a_0, n)$ provided by Theorem 4.1 appears to be a highly conservative estimate of the actual number of B&B nodes enumerated. Irrespective of the relative magnitudes of a_0 and the a_i's, the dominating factor turns out to be the $2^{n(n+1)/2}$ term in the expression for this upper bound. This factor was introduced mainly for the ease of proving the induction step in proof of Theorem 4.1.

7 Conclusions and Remarks

We study the performance of linear programming-based branch-and-bound algorithms applied to integer programming feasibility problems.We derive lower and upper bounds on the number of nodes that need to be examined by linear programming-based B&B to prove the infeasibility of an equality-constrained knapsack problem. A class of integer-infeasible knapsack problems where the coefficients decompose as $a = pM + r$, with $p_i, M \in \mathbb{Z}_{>0}$ for $i = 1, \ldots, n$ and $r_i \in \mathbb{Z}$ with $M \gg p_i, r_i$ yields an expression for the lower bound on the number of nodes examined by B&B that varies as M^{n-1}. This result provides a direct and unique proof for the worst-case exponential size of branch-and-bound proofs (using binary or integer branching) for integer feasibility problems.

The results in this paper could be extended to the case of an infeasible system of knapsack inequalities of the form $\delta \leq ax \leq \gamma$. The line in Figure (1) will be replaced by a thin band created by the two parallel lines $ax = \delta$ and $ax = \gamma$, still avoiding all the integer points. A couple of other key observations are in order as well. First, cutting planes are not considered in our analysis at all. At this point, it is not very clear how we could incorporate one or more classes of cutting planes into the framework used to explicitly count the number of nodes examined by the B&B algorithm. A direct proof for the exponential size of a cutting plane proof for a general integer program (as compared to that for a 0–1 program) would be interesting in its own right. Secondly, we show in another recent paper [9] that the class of decomposable knapsack problems that give rise to the exponential size of the B&B tree are trivial to solve if we branch on hyperplanes (or on integer linear combinations of the variables) rather than on the individual variables themselves. A column basis reduction-based reformulation easily identifies the favorable hyperplanes that should be branched on. We also show that the infeasibility of the system in question could be proved by branching on a *single* hyperplane. Similar results were in fact suggested earlier by Aardal and Lenstra [2].

8 Acknowledgments

The author would like to thank Gábor Pataki from the University of North Carolina at Chapel Hill for having constructive discussions and providing helpful suggestions on the material presented in this paper.

References

[1] Karen Aardal, Robert E. Bixby, Cor A. J. 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] Karen Aardal and Arjen K. Lenstra. Hard equality constrained integer knapsacks. *Mathematics of Operations Research*, 29(3):724–738, 2004. Revised version available at http://homepages.cwi.nl/∼aardal/mor_corr.pdf.

[3] Alfred Brauer and James E. Shockley. On a problem of Frobenius. *Journal für reine und angewandte Mathematik*, 211:399–408, 1962.

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

[5] Peter L. Erdös and Ronald L. Graham. On a linear Diophantine problem of Frobenius. *Acta Arithmetica*, 21:399–408, 1972.

[6] Mihály Hujter and Béla Vizvári. The exact solution to the Frobenius problem with three variables. *Journal of the Ramanujan Mathematical Society*, 2:117–143, 1987.

[7] Robert G. Jeroslow. Trivial integer programs unsolvable by branch-and-bound. *Mathematical Programming*, 6:105–109, 1974.

[8] Ravi Kannan. Lattice translates of a polytope and the Frobenius problem. *Combinatorica*, 12(2):161–177, 1992.

[9] Bala Krishnamoorthy and Gábor Pataki. Column basis reduction and decomposable knapsack problems. *Discrete Optimization*, 2006. Under review - see http://www.math.wsu.edu/faculty/bkrishna/PapersDiss/cbr1-final.pdf.

[10] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[11] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.

[12] Herbert E. Scarf and David F. Shallcross. The Frobenius problem and maximum lattice free bodies. *Mathematics of Operations Research*, 18:511–515, 1993.

[13] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley, Chichester, United Kingdom, 1986.

[14] Ernst S. Selmer. On the linear Diophantine problem of Frobenius. *Journal für reine und angewandte Mathematik*, 293/294:1–17, 1977.

[15] Ernst S. Selmer and Övind Beyer. On the linear Diophantine problem of Frobenius in three variables. *Journal für reine und angewandte Mathematik*, 301:161–178, 1978.

[16] Laurence A. Wolsey. *Integer Programming.* Wiley-Interscience, 1998.