# Lower bound on size of branch-and-bound trees for solving lot-sizing problem

Santanu S. Dey*[1] and Prachi Shah†[1]

[1]School of Industrial and Systems Engineering, Georgia Institute of Technology

**Abstract**

We show that there exists a family of instances of the lot-sizing problem, such that any branch-and-bound tree that solves them requires an exponential number of nodes, even in the case when the branchings are performed on general split disjunctions.

## 1 Introduction

### 1.1 Branch-and-bound procedure

Land and Doig [14] invented the branch-and-bound procedure to solve mixed integer linear programs (MILP). Today, all state-of-the-art MILP solvers are based on the branch-and-bound procedure. An important decision is formalizing a branch-and-bound procedure is to decide the method to partition the feasible region of the linear program corresponding to a node. If the partition is based on variable disjunctions, that is, the feasible region of the linear program at a given node is partitioned by adding the inequality of the form $x_i \leq \eta$ to one child node and the inequality $x_i \geq \eta + 1$ to the other child node where $\eta$ is an integer, then we call the branch-and-bound tree as a *simple* branch-and-bound tree. On the other hand if we allow the use of more general split disjunctions of the form:

$$\left(\pi^\top x \leq \eta\right) \vee \left(\pi^\top x \geq \eta + 1\right),$$

where $\pi$ is an integer vector and $\eta$ is an integer, to create two child nodes, we call the resulting branch-and-bound tree as a *general* branch-and-bound tree. Clearly general branch-and-bound tree are expected to be smaller than simple branch-and-bound tree. However, in practice, MILP solvers use simple branch-and-bound trees (one reason may be to maintain the sparsity of linear programs solved at child nodes; see discussion in [8, 11]).
One way to measure the efficiency of the branch-and-bound algorithm for a given class of instances, is to estimate the size of the branch-and-bound tree to solve the instances, since the size of the branch-and-bound tree corresponds to the number of linear programs solved.

*santanu.dey@isye.gatech.edu
†prachi.shah@gatech.edu

**Upper bounds on size of branch-and-bound trees.** Pataki [15] showed that for certain classes of random integer programs the general branch-and bound-tree has linear number of nodes with high probability, while recently [8, 4] showed that for two different classes of random integer programs even the simple branch-and bound-tree has polynomial size (number of nodes), with good probability. The paper [10] presents upper bounds on size of simple branch-and-bound tree for the vertex cover problem.

**Lower bounds on size of branch-and-bound trees.** The papers [13, 5] present examples of integer programs where every simple branch-and-bound algorithm for solving them has exponential size, although these instances can be solved using polynomial-size general branch-and-bound trees; see [19, 3]. Cook et al. [6] present a travelling salesman problem (TSP) instance that requires exponential-size branch-and-bound tree to solve when using simple branching.
Note that a lower bound on the size of a general branch-and-bound tree is also a lower bound on the size of a simple branch-and-bound tree. The paper [7] was the first to prove an exponential lower bound on the size of general branch-and-bound tree to prove the infeasibility of the cross-polytope. The paper [3] shows that the sparsity of the disjunctions used for branching can have a large impact on the size of the branch-and-bound tree. The paper [9] presents exponential lower bounds on the size of general branch-and-bound trees for solving a particular packing integer program, a particular covering integer program, and a particular TSP instance.
This paper contributes to this literature, by showing an (worst case) exponential lower bound on the size of general branch-and-bound tree for solving lot-sizing problems.

## 1.2 Lot-sizing problem

In this paper, we consider the classical lot-sizing problem of determining production volumes to meet demands in $n$ periods exactly, while minimizing production cost and fixed cost of production. A lot-sizing problem with a time horizon of $n$ periods can be formulated as a mixed-integer linear program (MILP) as follows,

$$\min \quad \sum_{i=1}^{n} p_i\, x_i + \sum_{i=1}^{n} f_i\, y_i \tag{1a}$$

$$\sum_{k=1}^{i} x_k \geq d_{1,i} \text{ for all } i \in \{1,\ldots,n-1\}, \tag{1b}$$

$$\sum_{k=1}^{n} x_k = d_{1,n}, \tag{1c}$$

$$x_i \leq d_{i,n}\, y_i \text{ for all } i \in \{1,\ldots,n\}, \tag{1d}$$

$$x \in \mathbb{R}^n_+,\ y \in [0,1]^n, \tag{1e}$$

$$y \in \mathbb{Z}^n, \tag{1f}$$

where variable $x_i$ is the quantity produced in period $i$ and $y_i$ is a binary variable with value 1 if production occurs in period $i$ and 0 otherwise. Unit cost of production, fixed cost of production, and demand in period $i$ are denoted by $p_i$, $f_i$ and $d_i$ respectively. We represent the cumulative demand from period $i$ to period $j$ by,

$$d_{i,j} = \left( \sum_{k=i}^{j} d_k \right).$$

The lot-sizing problem is a very well-studied problem [16] with many important applications. The classical dynamic programming algorithm to solve lot-sizing problem uses the so-called Wagner-Whitin property and runs in $\mathcal{O}(n^2)$ [18]. This running-time was later improved to $\mathcal{O}(n\log(n))$ [1, 12, 17]. The full polyhedral description of the convex hull of feasible solutions is presented in [2]. In this paper, we show that even though lot-sizing is such a "simple" problem, that is there is a polynomial-time dynamic programming algorithm to solve it, general branch-and-bound tree in the worst case may take exponential-time to solve the problem. Formally we prove the following:

**Theorem 1.** *Consider the lot-sizing instance on $n$ time periods with*

$$f_j = 1, \quad p_j = n - j + 1, \quad d_j = 1 \quad \text{for all } j \in \{1, \dots, n\}. \tag{2}$$

*Then any general branch-and-bound tree that solves this instance has at least $2^{(n/2)-1}$ leaf nodes.*

We provide a proof of Theorem 1 in the next section.

## 2 Proof of Theorem 1

We begin by finding the optimal objective function value for the class of instances (2).

**Claim 1.** *For the lot-sizing instance (2) with $n$ time periods, the optimal objective function value is $\dfrac{n(n+1)}{2} + n$.*

*Proof.* Observe that $y_1 = 1$ for any feasible solution since $d_1 > 0$. Now, consider the demand at period $j \geq 2$ and $y_j$. There are two possible cases:

(a) $y_j = 1$. Then we may assume that the demand at $j$ is met by production in period $j$ [16]. The total cost incurred for satisfying demand at $j$ is,

$$f_j + p_j = n - j + 2.$$

(b) $y_j = 0$. Then we may assume that the entire demand at $j$ is met by production in some period $i < j$ [16]. In this case, the unit cost of production of at period $i$ is

$$p_i = n - i + 1 \geq n - j + 2.$$

In other words, the increase in unit cost incurred by producing in an earlier time period, is at least as much as the fixed cost for period $j$. Therefore, we conclude that setting $y_j = 1$ for all $j \in \{1, \dots, n\}$ leads to an optimal solution. This optimal solution is:

$$\begin{aligned} \hat{x}_j &= d_j = 1, \ \text{ for all } j \in \{1, \dots, n\}, \\ \hat{y}_j &= 1, \ \text{ for all } j \in \{1, \dots, n\}. \end{aligned}$$

Therefore, the optimal cost (OPT) of the MILP is

$$\begin{aligned} OPT &= \sum_{i=1}^{n} p_i \, \hat{x}_i + \sum_{i=1}^{n} f_i \, \hat{y}_i \\ &= \sum_{i=1}^{n} (n - j + 1) + \sum_{i=1}^{n} 1 = \ \frac{n(n+1)}{2} + n. \end{aligned} \tag{3}$$

□

Given any node $\mathcal{N}$ in a branch-and-bound tree, we have two types of constraints defining the feasible region of the linear program corresponding to $\mathcal{N}$: (i) the constraints describing the original lot-sizing formulation (1b)-(1e), and (ii) the additional inequalities that were added to this node and its ancestors (starting at the child of the root node). We call the second set of constraints as branching constraints at node $\mathcal{N}$.
We will now show the size of any general branch-and-bound tree for lot-sizing instances (2) is exponential in the number of time periods, that is present a proof of Theorem 1.
For simplicity of exposition, we first consider the case where $n$ is odd. Let $\mathcal{T}$ be any general branch-and-bound tree that solves (2). Let $\mathcal{S} := \{y \in \{0,1\}^n \mid y_j = 1 \text{ if j is odd}\}$ and note that $|\mathcal{S}| = 2^{(n-1)/2} \geq 2^{(n/2)-1}$. Since the elements of $\mathcal{S}$ are integer vectors, they must satisfy the branching constraints of some leaf node.
Now, consider any $u, v \in \mathcal{S}$ such that $u \neq v$ and let $\mathcal{N}$ be a node of $\mathcal{T}$ such that both $u$ and $v$ are feasible for the branching constraints at $\mathcal{N}$. We will show that $\mathcal{N}$ is not a leaf node of $\mathcal{T}$ by constructing a solution $(\hat{x}, \hat{y})$ that satisfies the constraints describing $\mathcal{N}$ and whose objective value is strictly better than the MILP optimal objective function value. Equivalently, any leaf node of $\mathcal{T}$ contains at most one element of $\mathcal{S}$. Thus $\mathcal{T}$ must have at least $|S|$ leaf nodes, completing the proof.
It remains to show that $\mathcal{N}$ is not a leaf node when $u, v$ satisfy the branching constraint of $\mathcal{N}$, where $u, v \in \mathcal{S}$ and $u \neq v$. Let $\hat{y} := \frac{(u+v)}{2}$. By convexity, $\hat{y}$ also satisfies the branching constraints at $\mathcal{N}$.
Construct $\hat{x}$ as follows,

$$\hat{x}_j = \begin{cases} 1 & \text{if } j = n \\ 1 & \text{if } j < n \text{ is odd and } \hat{y}_{j+1} \in \{1, 0.5\} \\ 2 & \text{if } j < n \text{ is odd and } \hat{y}_{j+1} = 0 \\ 1 & \text{if } j \text{ is even and } \hat{y}_j \in \{1, 0.5\} \\ 0 & \text{if } j \text{ is even and } \hat{y}_j = 0. \end{cases}$$

It is straightforward to verify that $(\hat{x}, \hat{y})$ satisfies (1b)-(1e). Since $\hat{y}$ satisfies the branching constraints at $\mathcal{N}$, this shows that $(\hat{x}, \hat{y})$ is a feasible solution for linear program corresponding to $\mathcal{N}$.
We will now compute the objective function value of $(\hat{x}, \hat{y})$ by considering consecutive pairs of time periods, $j$ and $j+1$ for odd values of $j < n$. There are three possible cases:

(i) $\hat{y}_{j+1} = 0 \implies \hat{x}_j = 2$, $\hat{x}_{j+1} = 0$:

$$OBJ_j + OBJ_{j+1} = (1 + (n-j+1) \cdot 2) + (0) = (n-j+1) + (n-j) + 2.$$

(ii) $\hat{y}_{j+1} = 1 \implies \hat{x}_j = 1$, $\hat{x}_{j+1} = 1$:

$$OBJ_j + OBJ_{j+1} = (1 + (n-j+1) \cdot 1) + (1 + (n-j) \cdot 1) = (n-j+1) + (n-j) + 2.$$

(iii) $\hat{y}_{j+1} = 0.5 \implies \hat{x}_j = 1$, $\hat{x}_{j+1} = 1$:

$$OBJ_j + OBJ_{j+1} = (1 + (n-j+1) \cdot 1) + (0.5 + (n-j) \cdot 1) = (n-j+1) + (n-j) + 2 - 0.5.$$

Lastly, note that the contribution to the objective function from the last period is 2, i.e. $OBJ_n = 2$.
Let $N_{0.5}$ be the number of coordinates of $\hat{y}$ that are 0.5 and $OPT$ be the optimal MILP objective

value from Claim 1. Then, the objective function value for $(\hat{x}, \hat{y})$ is,

$$\begin{aligned}
\sum_{j=1}^{n} OBJ_j &= \sum_{j=1}^{n-1} OBJ_j + OBJ_n \\
&= \sum_{\substack{j=1, \\ j \text{ is odd}}}^{n-2} ((n-j+1)+(n-j)+2) - 0.5\, N_{0.5} + 2 \\
&= \sum_{j=1}^{n} ((n-j+1)+1) - 0.5\, N_{0.5} \\
&= \frac{n(n+1)}{2} + n - 0.5\, N_{0.5} \\
&= OPT - 0.5\, N_{0.5} \\
&< OPT,
\end{aligned}$$

where the last inequality follows since $\hat{y}$ must have at least one component equal to 0.5 since $u \neq v$. Thus, $\mathcal{N}$ cannot be a leaf node. This completes the proof.
In the case where $n$ is even, the proof follows similarly, by defining

$$\mathcal{S} := \{y \in \{0,1\}^n \,|\, y_1 = 1,\, y_j = 1 \text{ if j is even}\}$$

and noting that $|\mathcal{S}| = 2^{(n/2)-1}$.

## References

[1] Alok Aggarwal and James K Park. Improved algorithms for economic lot size problems. *Operations research*, 41(3):549–571, 1993.

[2] Imre Barany, Tony Van Roy, and Laurence A Wolsey. Uncapacitated lot-sizing: The convex hull of solutions. In *Mathematical programming at Oberwolfach II*, pages 32–43. Springer, 1984.

[3] Amitabh Basu, Michele Conforti, Marco Di Summa, and Hongyi Jiang. Complexity of branch-and-bound and cutting planes in mixed-integer optimization–ii. *arXiv preprint arXiv:2011.05474*, 2020.

[4] Sander Borst, Daniel Dadush, Sophie Huiberts, and Samarth Tiwari. On the integrality gap of binary integer programs with gaussian data. *arXiv preprint arXiv:2012.08346*, 2020.

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

[6] William J Cook and Mark Hartmann. On the complexity of branch and cut methods for the traveling salesman problem. *Polyhedral Combinatorics*, 1:75–82, 1990.

[7] Daniel Dadush and Samarth Tiwari. On the complexity of branching proofs. In *Proceedings of the 35th Computational Complexity Conference*, pages 1–35, 2020.

[8] Santanu S Dey, Yatharth Dubey, and Marco Molinaro. Branch-and-bound solves random binary ips in polytime. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 579–591. SIAM, 2021.

[9] Santanu S Dey, Yatharth Dubey, and Marco Molinaro. Lower bounds on the size of general branch-and-bound trees. *arXiv preprint arXiv:2103.09807*, 2021.

[10] Santanu S Dey, Yatharth Dubey, Marco Molinaro, and Prachi Shah. A theoretical and computational analysis of full strong-branching. *arXiv preprint arXiv:2110.10754*, 2021.

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

[12] Awi Federgruen and Michal Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in 0 (n log n) or 0 (n) time. *Management Science*, 37(8):909–925, 1991.

[13] Robert G Jeroslow. Trivial integer programs unsolvable by branch-and-bound. *Mathematical Programming*, 6(1):105–109, 1974.

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

[15] Gábor Pataki, Mustafa Tural, and Erick B Wong. Basis reduction and the complexity of branch-and-bound. In *Proceedings of the twenty-first annual ACM-SIAM symposium on discrete algorithms*, pages 1254–1261. SIAM, 2010.

[16] Yves Pochet and Laurence A Wolsey. *Production planning by mixed integer programming.* Springer Science & Business Media, 2006.

[17] Albert Wagelmans, Stan Van Hoesel, and Antoon Kolen. Economic lot sizing: An o (n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research*, 40(1-supplement-1):S145–S156, 1992.

[18] Harvey M Wagner and Thomson M Whitin. Dynamic version of the economic lot size model. *Management science*, 5(1):89–96, 1958.

[19] Yu Yang, Natashia Boland, and Martin Savelsbergh. Multivariable branching: A 0-1 knapsack problem case study. *INFORMS Journal on Computing*, 2021.