Theoretical and computational analysis of sizes of branch-and-bound trees

Santanu S. Dey
Georgia Institute of Technology

Oct 2021

Joint work with...

Yatharth Dubey
Georgia Tech

Marco Molinaro
PUC-Rio.

Prachi Shah
Georgia Tech

1
Introduction

Integer program

Integer program

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \qquad \qquad \text{(IP)} \\ & x \in \mathbb{Z}^n \end{aligned}$$

Integer program

Integer program

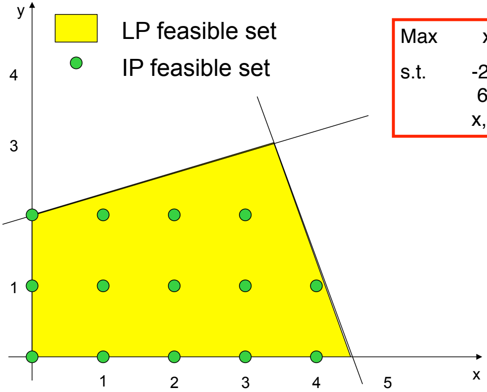
$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n \end{aligned} \qquad \text{(IP)}$$

Many applications:

- Decision making with vast economic and societal impact
- Power systems, Sustainibility, IMRT cancer treatment, Circuit design, Healthcare analytics, Network design, Supply chain Design, Urban mobility, Production planning, National security, etc.

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.



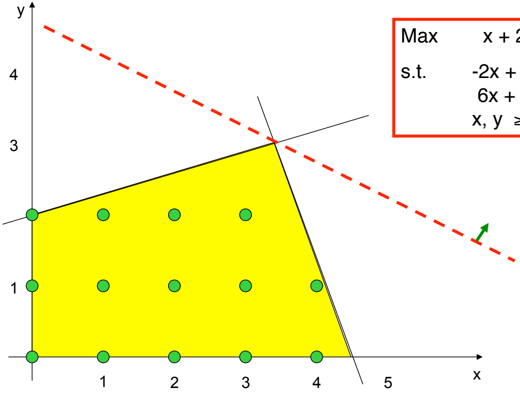
$$\begin{array}{ll} \text{Max} & x + 2y \\ \text{s.t.} & -2x + 7y \leq 14 \\ & 6x + 2y \leq 27 \\ & x, y \geq 0 \text{ and integer} \end{array}$$

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

Solving Root Note



Max $x + 2y$

s.t. $-2x + 7y \leq 14$

$6x + 2y \leq 27$

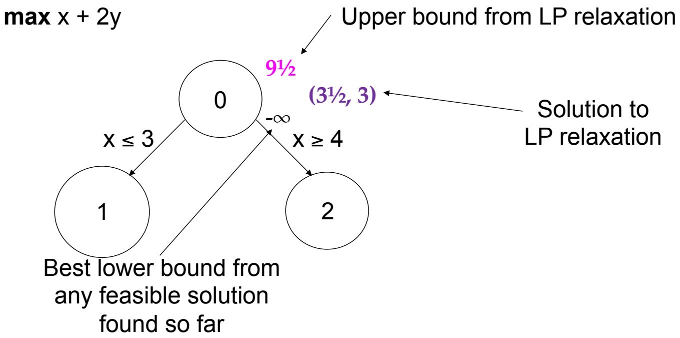
$x, y \geq 0$ and integer

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

Branch-and-Bound Tree

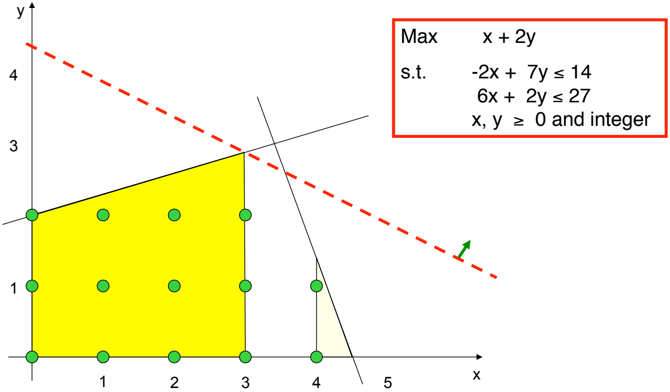


Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

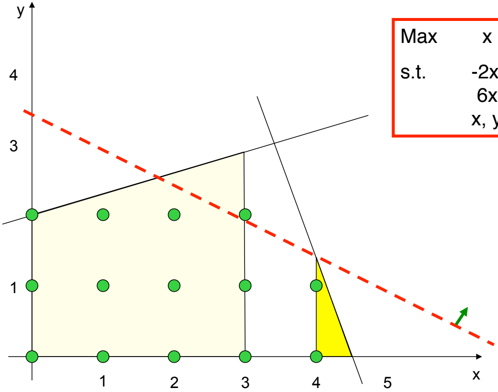
Branching process



Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.



Max $x + 2y$

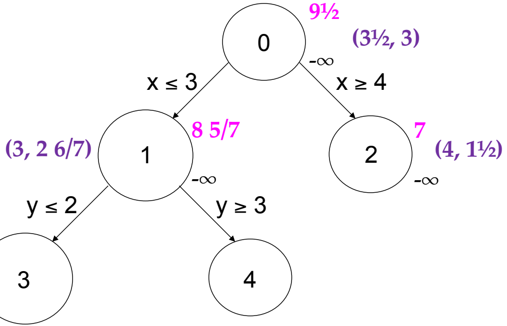
s.t. $-2x + 7y \leq 14$
$6x + 2y \leq 27$
$x, y \geq 0$ and integer

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

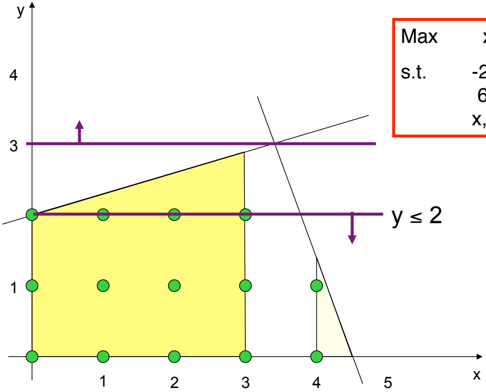
Branch-and-Bound Tree



Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.



Max $x + 2y$

s.t. $-2x + 7y \leq 14$

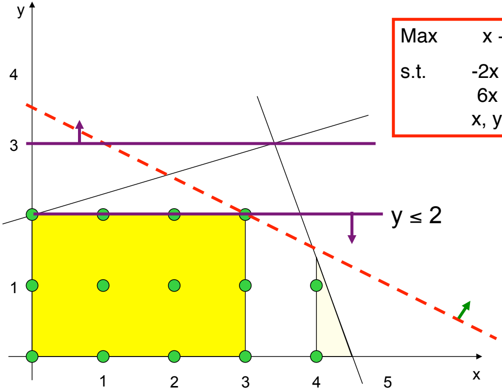
$6x + 2y \leq 27$

$x, y \geq 0$ and integer

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.



Max $\quad x + 2y$

s.t. $\quad -2x + 7y \leq 14$

$\quad 6x + 2y \leq 27$

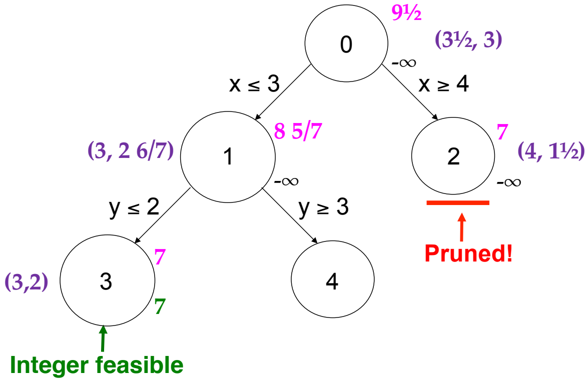
$\quad x, y \geq 0$ and integer

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

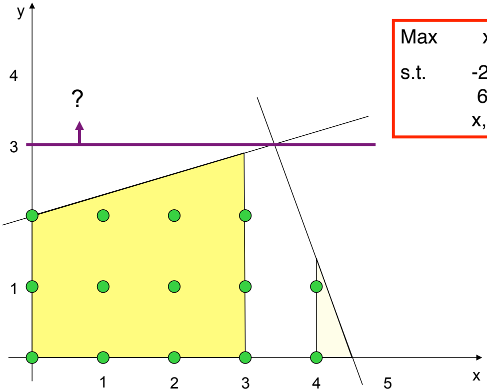
Branch-and-Bound Tree



Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.



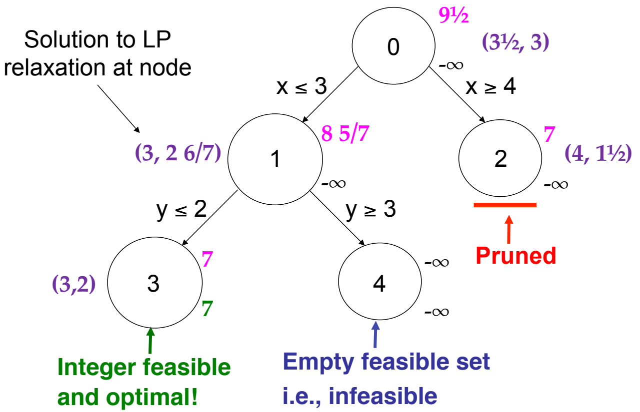
$$\begin{array}{ll} \text{Max} & x + 2y \\ \text{s.t.} & -2x + 7y \leq 14 \\ & 6x + 2y \leq 27 \\ & x, y \geq 0 \text{ and integer} \end{array}$$

Example/Picture credit: Natashia Boland

Branch-and-Bound

Branch-and-bound is the basic algorithm underlying state-of-art IP solvers.

Branch-and-Bound Tree



Example/Picture credit: Natashia Boland

Well-defined branch-and-bound algorithm

- Node selection: Which node should we branch on next?

Well-defined branch-and-bound algorithm

- Node selection: Which node should we branch on next?
  A common rule used: *Worst bound rule* – use the node which has the largest (resp. smallest) LP value for a maximization-type (resp. minimization-type) IP.

Well-defined branch-and-bound algorithm

- Node selection: Which node should we branch on next?
  A common rule used: *Worst bound rule* – use the node which has the largest (resp. smallest) LP value for a maximization-type (resp. minimization-type) IP.
- Partitioning the feasible region of an LP at a node

Well-defined branch-and-bound algorithm

- Node selection: Which node should we branch on next?
  A common rule used: *Worst bound rule* – use the node which has the largest (resp. smallest) LP value for a maximization-type (resp. minimization-type) IP.
- Partitioning the feasible region of an LP at a node
  - Simple branch-and-bound: Used in practise by solvers

    $$\begin{aligned} x_j &\leq \lfloor \hat{x}_j \rfloor & --> \text{Added to left node} \\ x_j &\geq \lceil \hat{x}_j \rceil & --> \text{Added to right node} \end{aligned}$$

    Need rule to decide which variable to branch on: *Full strong branching*, *Reliability branching*, *Pseudocost branching*: will discuss some of these later.
  - General branch-and-bound:

    $$\begin{aligned} \pi^\top x &\leq \pi_0 & --> \text{Added to left node} \\ \pi^\top x &\geq \pi_0 + 1 & --> \text{Added to right node} \end{aligned}$$

    where $\pi \in \mathbb{Z}^n$ is an integer vector and $\pi_0 \in \mathbb{Z}$ is an integer.

Branch-and-bound procedure

- The branch and bound algorithm was invented by Land and Doig in 1960.

Ailsa Land

Picture credit: Wikipedia

Alison Doig

Branch-and-bound procedure

- The branch and bound algorithm was invented by Land and Doig in 1960.
- Almost 60 years now, but there is very little theoretical analysis of the branch-and-bound algorithm!

What kind of questions we want to answer

- What is known: There are simple examples (i.e. knapsack IPs) with n variables that require $\mathcal{O}(2^n)$ nodes when using simple branch-and-bound tree. [Jeroslow (1974)], [Chvátal (1980)]

What kind of questions we want to answer

- What is known: There are simple examples (i.e. knapsack IPs) with n variables that require $\mathcal{O}(2^n)$ nodes when using simple branch-and-bound tree. [Jeroslow (1974)], [Chvátal (1980)]
- However, branch-and bound algorithm (with many bells and whistle) seems to work well in practice.

What kind of questions we want to answer

- What is known: There are simple examples (i.e. knapsack IPs) with n variables that require $\mathcal{O}(2^n)$ nodes when using simple branch-and-bound tree. [Jeroslow (1974)], [Chvátal (1980)]
- However, branch-and bound algorithm (with many bells and whistle) seems to work well in practice.

So the type of questions we want to understand:

- Can we prove for a random model for instances that branch-and-bound works well?

What kind of questions we want to answer

- What is known: There are simple examples (i.e. knapsack IPs) with n variables that require $\mathcal{O}(2^n)$ nodes when using simple branch-and-bound tree. [Jeroslow (1974)], [Chvátal (1980)]
- However, branch-and bound algorithm (with many bells and whistle) seems to work well in practice.

So the type of questions we want to understand:

- Can we prove for a random model for instances that branch-and-bound works well?
- The simple examples above can be solved using a polynomial number of nodes using general branch-and-bounds [Yang, Boland, Savelsbergh (2021)]. Can we understand lower bounds for general branch-and-bound. (Preliminary results: [Dadush, Tiwari (2020)])

What kind of questions we want to answer

- What is known: There are simple examples (i.e. knapsack IPs) with n variables that require $\mathcal{O}(2^n)$ nodes when using simple branch-and-bound tree. [Jeroslow (1974)], [Chvátal (1980)]
- However, branch-and bound algorithm (with many bells and whistle) seems to work well in practice.

So the type of questions we want to understand:

- Can we prove for a random model for instances that branch-and-bound works well?
- The simple examples above can be solved using a polynomial number of nodes using general branch-and-bounds [Yang, Boland, Savelsbergh (2021)]. Can we understand lower bounds for general branch-and-bound. (Preliminary results: [Dadush, Tiwari (2020)])
- Can we understand and analyse properties of some well-known rules for partitioning mentioned above? Hopefully this will lead to better rules.

2

Main results

2.1
Branch-and-bound solves (a class of) random IPs in polynomial time

Random model of IPs

We consider the following model of random IPs:

$$\begin{array}{ccl} \max & c^\top x & c \sim \text{Uniform}([0,\ 1]^n) \\ \text{s.t.} & Ax \leq b & A \sim \text{Uniform}([0,\ 1]^{m\times n}) \\ & x \in \{0,1\}^n & , \end{array}$$

where $b = \beta \cdot n, \beta \in \left(0, \frac{1}{2}\right)^m$.

Random model of IPs

We consider the following model of random IPs:

$$\begin{array}{ccc} \max & c^\top x & c \sim \text{Uniform}([0,\ 1]^n) \\ \text{s.t.} & Ax \leq b & A \sim \text{Uniform}([0,\ 1]^{m\times n}) \\ & x \in \{0,1\}^n & , \end{array}$$

where $b = \beta \cdot n, \beta \in \left(0, \frac{1}{2}\right)^m$.
Incomplete literature review:

- Analysis of gap and enumerative algorithms: [Lueker (1982)], [Goldberg, Marchetti-Spaccamela (1984)], [Beier, Vocking (2003)], [Dyer, Frieze (1989)]

Random model of IPs

We consider the following model of random IPs:

$$\begin{array}{ccl} \max & c^\top x & c \sim \text{Uniform}([0,\ 1]^n) \\ \text{s.t.} & Ax \leq b & A \sim \text{Uniform}([0,\ 1]^{m\times n}) \\ & x \in \{0,1\}^n & , \end{array}$$

where $b = \beta \cdot n, \beta \in \left(0, \frac{1}{2}\right)^m$.

Incomplete literature review:

- Analysis of gap and enumerative algorithms: [Lueker (1982)], [Goldberg, Marchetti-Spaccamela (1984)], [Beier, Vocking (2003)], [Dyer, Frieze (1989)]
- General branching: [Pataki, Tural, Wong (2010)]

Result for random IPs

Theorem (D., Dubey, Molinaro)

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: Worst bound rule (Select a node with largest LP value as the next node to branch on.)*

Result for random IPs

Theorem (D., Dubey, Molinaro)

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: Worst bound 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 IP described previously. 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.

Result for random IPs

Simplified...

Theorem (D., Dubey, Molinaro)

Any branch-and-bound tree using the worst bound rule for node selection, solving the above problem has no more than $(n^{\mathcal{O}(m)})$ nodes with good probability.

Result for random IPs

Simplified...

Theorem (D., Dubey, Molinaro)

Any branch-and-bound tree using the worst bound rule for node selection, solving the above problem has no more than $(n^{\mathcal{O}(m)})$ nodes with good probability.

Nice follow up work [Borst, Dadush, Huiberts, Tiwari (2021)]

2.2
Lower bounds on size of general branch-and-bound tree

Lower bounds

- Remember when using general branch-and-bound, we are allowed to use general disjunctions:

$$\begin{array}{lcll} \pi^\top x & \leq & \pi_0 & - - > \text{Added to left node} \\ \pi^\top x & \geq & \pi_0 + 1 & - - > \text{Added to right node} \end{array}$$

  where π is an integer vector and π_0 is an integer.
- As explaned before, most lower bounds are for simple branch-and-bound trees.

Lower bounds

- Remember when using general branch-and-bound, we are allowed to use general disjunctions:

$$\begin{array}{cccl} \pi^\top x & \leq & \pi_0 & --> \text{Added to left node} \\ \pi^\top x & \geq & \pi_0 + 1 & --> \text{Added to right node} \end{array}$$

  where π is an integer vector and π_0 is an integer.

- As explaned before, most lower bounds are for simple branch-and-bound trees. We want results independent of computation complexity assumptions.

Lower bounds

- Remember when using general branch-and-bound, we are allowed to use general disjunctions:

$$\begin{array}{lcll} \pi^\top x & \leq & \pi_0 & --> \text{Added to left node} \\ \pi^\top x & \geq & \pi_0 + 1 & --> \text{Added to right node} \end{array}$$

  where π is an integer vector and π_0 is an integer.
- As explaned before, most lower bounds are for simple branch-and-bound trees. We want results independent of computation complexity assumptions.
- Very recently, [Dadush,Tiwari (2020)] showed the following:

Theorem (Dadush, Tiwari)

Any (general) branch-and-bound tree that certifies the following instance is integer feasible requires at least $\frac{2^n}{n}$ leaf nodes:

$$C := \left\{ x \in \{0,1\}^n \,|\, \sum_{j\in S} x_j + \sum_{j \in [n]\setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0-1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \ \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0-1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).
- Then observe, by convexity, the vector $\frac{v+w}{2}$ satisfies the branching constraints at this node.

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \ \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0-1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).
- Then observe, by convexity, the vector $\frac{v+w}{2}$ satisfies the branching constraints at this node.
- Observe that any vector $u \in \left\{0, 1, \frac{1}{2}\right\}^n$ with at least one component $\frac{1}{2}$ satisfies the LP relaxation of C:

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0-1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).
- Then observe, by convexity, the vector $\frac{v+w}{2}$ satisfies the branching constraints at this node.
- Observe that any vector $u \in \left\{0, 1, \frac{1}{2}\right\}^n$ with at least one component $\frac{1}{2}$ satisfies the LP relaxation of C:

$$\frac{v+w}{2} \in \left\{ x \in [0,1]^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0 - 1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).
- Then observe, by convexity, the vector $\frac{v+w}{2}$ satisfies the branching constraints at this node.
- Observe that any vector $u \in \left\{0, 1, \frac{1}{2}\right\}^n$ with at least one component $\frac{1}{2}$ satisfies the LP relaxation of C:

$$\frac{v+w}{2} \in \left\{ x \in [0,1]^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \; \forall S \subseteq [n] \right\}$$

- So $\frac{v+w}{2}$ satisfies all the constraints at the node. Equivalently, the node is non-empty.

Lets consider the cross-polytope again..

$$C := \left\{ x \in \{0,1\}^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \ \forall S \subseteq [n] \right\}$$

We can present a sligthly better result:

- Consider any node of a general branch-and-bound tree where two distinct $0-1$ vectors v and w are feasible for the branching-constraints added to that node (v and w ofcourse cannot belong to C or its LP relaxation).
- Then observe, by convexity, the vector $\frac{v+w}{2}$ satisfies the branching constraints at this node.
- Observe that any vector $u \in \left\{0, 1, \frac{1}{2}\right\}^n$ with at least one component $\frac{1}{2}$ satisfies the LP relaxation of C:

$$\frac{v+w}{2} \in \left\{ x \in [0,1]^n \mid \sum_{j \in S} x_j + \sum_{j \in [n] \setminus S} (1 - x_j) \geq \frac{1}{2} \ \forall S \subseteq [n] \right\}$$

- So $\frac{v+w}{2}$ satisfies all the constraints at the node. Equivalently, the node is non-empty.
- In order to prove integer-infeasiblilty of C, every leaf node should be infeasible. So from above, there must at at least 2^n leaf nodes!

More results –I: Hardness of some combinatorial problems

Theorem (D., Dubey, Molinaro)

Let n be a even positive integer. Any branch-and-bound tree, solving the following instance

$$\begin{aligned} \max \quad & \sum_{j \in [n]} x_j \\ \text{s.t.} \quad & \sum_{k \in S} x_j \leq \frac{n}{2} - 1, \ \forall S \subseteq [n], |S| = \frac{n}{2} \\ & x \in \{0,1\}^n \end{aligned}$$

requires at least $2^{\Omega(n)}$ nodes.

More results –II: Hardness for travelling saleman problem

We develope techniques to reduce branch-and-bound hardness, and together with

More results –II: Hardness for travelling saleman problem

We develope techniques to reduce branch-and-bound hardness, and together with the cross polytope result, we can obtain the following result:

Theorem (D., Dubey, Molinaro)

Let P be LP relaxation of the usual travelling saleman problem formulation (with sub tour elimination) with n cities.

More results –II: Hardness for travelling saleman problem

We develope techniques to reduce branch-and-bound hardness, and together with the cross polytope result, we can obtain the following result:

Theorem (D., Dubey, Molinaro)

Let P be LP relaxation of the usual travelling saleman problem formulation (with sub tour elimination) with n cities. There exist an objective function c, such that any branch-and-bound tree, solving the following instance

$$\begin{array}{ll} \max & c^\top x \\ \text{s.t.} & x \in P \\ & x \in \{0,1\}^{\frac{(n)(n-1)}{2}} \end{array}$$

requires at least $2^{\Omega(n)}$ nodes.

More results –III: Smoothened analysis not possible

Random family of instances:

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

$$x \in [0, 1]^n,$$

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

Theorem (D., Dubey, Molinaro)

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

2.3

Analysis of full strong branching rule for partitioning

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0,\ 1)$ for $j \in F \subseteq [n]$.

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0, \ 1)$ for $j \in F \subseteq [n]$.
- Let OPT $:=$ the optimal value of the LP at this node.

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0,\ 1)$ for $j \in F \subseteq [n]$.
- Let $\text{OPT} :=$ the optimal value of the LP at this node.
- Let $\text{OPT}_{j,0} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \leq 0$.
  Let $\text{OPT}_{j,1} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \geq 1$.

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0,\ 1)$ for $j \in F \subseteq [n]$.
- Let OPT $:=$ the optimal value of the LP at this node.
- Let $\text{OPT}_{j,0} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \leq 0$.
  Let $\text{OPT}_{j,1} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \geq 1$.
- Let

$$\text{score}_j = \max\{|\text{OPT} - \text{OPT}_{j,1}|, \epsilon\} \cdot \max\{|\text{OPT} - \text{OPT}_{j,0}|, \epsilon\},$$

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0, 1)$ for $j \in F \subseteq [n]$.
- Let $\mathrm{OPT} :=$ the optimal value of the LP at this node.
- Let $\mathrm{OPT}_{j,0} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \leq 0$.
  Let $\mathrm{OPT}_{j,1} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \geq 1$.
- Let

  $$\mathrm{score}_j = \max\{|\mathrm{OPT} - \mathrm{OPT}_{j,1}|, \epsilon\} \cdot \max\{|\mathrm{OPT} - \mathrm{OPT}_{j,0}|, \epsilon\},$$

  where $\epsilon > 0$ is a small number related to machine precision.
- Let $j^* := \mathrm{argmax}_{j \in F}(\mathrm{score}_j)$.

Full strong branching

- Full strong branching is a partitioning rule for simple branch-and-bound trees.
- Supose $\hat{x}$ is a LP optimal solution. Let $\hat{x}_j \in (0, \ 1)$ for $j \in F \subseteq [n]$.
- Let $\text{OPT} :=$ the optimal value of the LP at this node.
- Let $\text{OPT}_{j,0} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \leq 0$.
  Let $\text{OPT}_{j,1} :=$ the optimal value of the LP of the child node where we have the inequality $x_j \geq 1$.
- Let

  $$\text{score}_j = \max\left\{|\text{OPT} - \text{OPT}_{j,1}|, \epsilon\right\} \cdot \max\left\{|\text{OPT} - \text{OPT}_{j,0}|, \epsilon\right\},$$

  where $\epsilon > 0$ is a small number related to machine precision.
- Let $j^* := \text{argmax}_{j \in F}\,(\text{score}_j)$.
- Branch on j^*.

More on full strong branching

- Experimentally, full strong branching, works better than any other rule for simple branch-and-bound trees [Achterberg, Koch, and Martin (2005)].

More on full strong branching

- Experimentally, full strong branching, works better than any other rule for simple branch-and-bound trees [Achterberg, Koch, and Martin (2005)].
- However, this rule is not used in practise, because we need to solve $2|F|$ LPs just to decide one branching decision.

More on full strong branching

- Experimentally, full strong branching, works better than any other rule for simple branch-and-bound trees [Achterberg, Koch, and Martin (2005)].
- However, this rule is not used in practise, because we need to solve $2|F|$ LPs just to decide one branching decision.
- Recently there has been many attempts made to mimic full strong branching using machine learning. [Alvarez, Louveaux, and Wehenkel (2017)], [Gasse, Chetelat, Ferroni, Charlin, Lodi (2019)], [Khalil, Le Bodic, Song, Nemhauser, Dilkina (2016)], [Nair et al. (2020)]

Some questions..

- How large is the tree produced by strong-branching in comparison to the smallest possible branch-and-bound tree for a given instance? Answering this question may lead us to finding better rules.

Introduction

Main results

Random IPs

Lower bounds on size of general branch-and-bound tree

Analysis of full strong branching rule for partitioning

Strong branching applied to specific problems

A computational evaluation of strong branching

Proof Outlines

Some questions..

- How large is the tree produced by strong-branching in comparison to the smallest possible branch-and-bound tree for a given instance? Answering this question may lead us to finding better rules.
- A more refined questions is the following: it would be useful to understand the performance of strong-branching vis-á-vis different classes of instances.

2.3.1
Strong branching applied to specific problems

Vertex cover

Definition (Vertex cover)

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

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

Given an instance I of this IP, we let OPT(I) denote optimal objective function value and $\mathrm{OPT}_L(I)$ be the optimal objective function of the LP relaxation (i.e. when the variable constraints are $x_v \in [0, 1]$). Then let

$$\Gamma(I) := \mathrm{OPT}(I) - \mathrm{OPT}_L(I).$$

Strong branching works well for vertex cover

Theorem (D., Dubey. Molinaro, Shah)

Let I be any instance of vertex cover on n nodes. Assume we break ties within the worst-bound rule for node selection rule by selecting a node with the largest depth.

Strong branching works well for vertex cover

Theorem (D., Dubey. Molinaro, Shah)

Let I be any instance of vertex cover on n nodes. Assume we break ties within the worst-bound rule for node selection rule by selecting a node with the largest depth. Let $\mathcal{T}_S(I)$ be some branch-and-bound tree generated by strong-branching with the above version of worst-bound node selection rule that solves I.

Strong branching works well for vertex cover

Theorem (D., Dubey. Molinaro, Shah)

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

$$|\mathcal{T}_S(I)| \leq 2^{2\Gamma(I)+1} + \mathcal{O}(n).$$

Strong branching does not work well for other IP models

$$\left\{(x, y) \in \{0,1\}^n \times \{0,1\}^n \,|\, y_i \leq 2x_i, y_i \leq 2 - 2x_i, \forall i \in [n], \sum_{i=1}^{n} y_i = 1\right\}$$

Theorem

The smallest branch-and-bound tree that shows that the above set is integer infeasible requires no more than $4n + 1$ nodes. On the other hand, strong branching requires at least 2^n nodes.

2.3.2

A computational evaluation of strong branching

Optimal branch-and-bound tree

Optimal branch-and-bound tree

- It is not possible to analyse different problems analytically.

Optimal branch-and-bound tree

- It is not possible to analyse different problems analytically.
- So we came up with a Dynamic programming algorithm to compute the optimal branch-and-bound tree.

Theorem (D., Dubey, Molinaro, Shah)

There is a DP algorithm with running time $\mathrm{poly}(data) \cdot 3^{O(n)}$ *time to compute an optimal branch-and-bound tree for any binary MILP instance defined on n binary variables.*

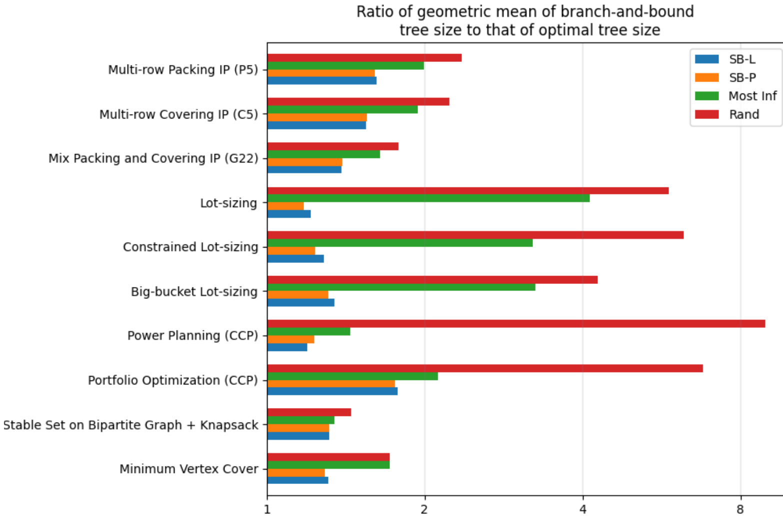


Figure: Ratio of geometric mean of branch-and-bound tree sizes to geometric mean of optimal tree sizes over all instances of a problem for various branching strategies. “Rand" stands for random, “Most Inf" stands for most infeasible, “SB-P" stands for strong-branching with product score function, and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

3

Proof Outlines

3.1
Random IPs Theorem

Reminder...

$$
\begin{array}{lll}
\max & c^\top x & c \sim \text{Uniform}([0,\ 1]^n) \\
\text{s.t.} & Ax \le b & A \sim \text{Uniform}([0,\ 1]^{m\times n}) \\
 & x \in \{0,1\}^n & ,
\end{array}
$$

where $b = \beta \cdot n, \beta \in \left(0, \frac{1}{2}\right)^m$.

Theorem (D., Dubey, Molinaro)

Any branch-and-bound tree using the worst bound rule for node selection, solving the above problem has no more than $(n^{\mathcal{O}(m)})$ nodes (with good probability).

Proof Sketch

$$\Delta(x) = \sum_{j \in [n]} \underbrace{(c_j - \langle \lambda^*, A^j \rangle)}_{\text{reduced cost}} \cdot (\underbrace{x_j^*}_{\text{LP Opt.}} - x_j),$$

Proof Sketch

$$\Delta(x) = \sum_{j\in[n]} \underbrace{(c_j - \langle \lambda^*, A^j \rangle)}_{\text{reduced cost}} \cdot (\underbrace{x_j^*}_{\text{LP Opt.}} - x_j),$$

$$G := \left\{ x \in \{0,1\}^n \,|\, \Delta(x) \leq \mathrm{OPT} - \mathrm{OPT}_{\mathrm{LP}} \right\}$$

Proof Sketch

$$\Delta(x) = \sum_{j \in [n]} \underbrace{(c_j - \langle \lambda^*, A^j \rangle)}_{\text{reduced cost}} \cdot (\underbrace{x_j^*}_{\text{LP Opt.}} - x_j),$$

$$G := \left\{ x \in \{0,1\}^n \,|\, \Delta(x) \leq \text{OPT} - \text{OPT}_{\text{LP}} \right\}$$

1. The number of internal nodes in a branch-and-bound tree is at most n times the number of good integer solutions G.

Proof Sketch

$$\Delta(x) = \sum_{j\in[n]} \underbrace{(c_j - \langle \lambda^*, A^j \rangle)}_{\text{reduced cost}} \cdot (\underbrace{x_j^*}_{\text{LP Opt.}} - x_j),$$

$$G := \left\{x \in \{0,1\}^n \,|\, \Delta(x) \leq \text{OPT} - \text{OPT}_{\text{LP}}\right\}$$

1. The number of internal nodes in a branch-and-bound tree is at most n times the number of good integer solutions G.
2. The number of good integer solutions is at most $n^{\mathcal{O}(m)}$ (with good probability).

We construct a mapping...

r : internal nodes $\rightarrow G$ as follows:

We construct a mapping...

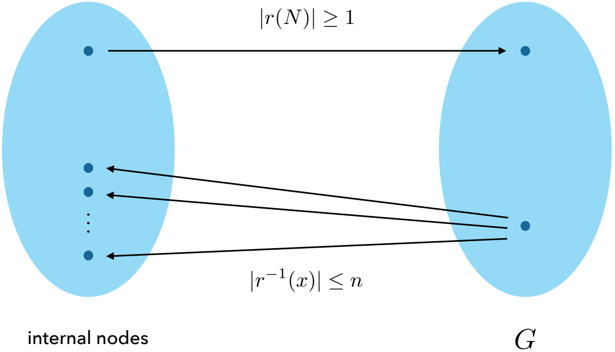
r : internal nodes $\rightarrow G$ as follows:

$$r(N) = x' \in \operatorname{argmin}\{\Delta(x) \,|\, x'_j = \underbrace{x_j^N}_{\text{Opt. Sol of N}} \text{ if } x_j^N \in \{0,1\}\}.$$

We construct a mapping...

r : internal nodes $\rightarrow G$ as follows:

$$r(N) = x' \in \operatorname{argmin}\{\Delta(x) \mid x'_j = \underbrace{x^N_j}_{\text{Opt. Sol of N}} \quad \text{if } x^N_j \in \{0, 1\}\}.$$



Picture credit: Yatharth Dubey

3.2

Optimal branch-and-bound tree

Dynamic programming algorithm

$$\max_{x \in P \cap \{0,1\}^n} c^\top x$$

Dynamic programming algorithm

$$\max_{x \in P \cap \{0,1\}^n} c^\top x$$

Let $\mathcal{F}$ denote the set of faces of $[0,\ 1]^n$, i.e. $|\mathcal{F}| = 3^n$.

Dynamic programming algorithm

$$\max_{x \in P \cap \{0,1\}^n} c^\top x$$

Let $\mathcal{F}$ denote the set of faces of $[0,\ 1]^n$, i.e. $|\mathcal{F}| = 3^n$.

Algorithm 1 Computing Optimal Branch-and-bound Tree

Phase-1: Pruning by Infeasibility or Bound

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

Phase-2: Recursive bottom-up computation

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

Thank You.

- Dey, Santanu S., Yatharth Dubey, and Marco Molinaro. "Branch-and-bound solves random binary ips in polytime." Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA). Society for Industrial and Applied Mathematics, 2021.
- Dey, Santanu S., Yatharth Dubey, and Marco Molinaro. "Lower Bounds on the Size of General Branch-and-Bound Trees." arXiv preprint arXiv:2103.09807 (2021).
- Dey, Santanu S., Yatharth Dubey, and Marco Molinaro. "A Theoretical and Computational Analysis of Full Strong-Branching." arXiv preprint arXiv:2110.10754 (2021).