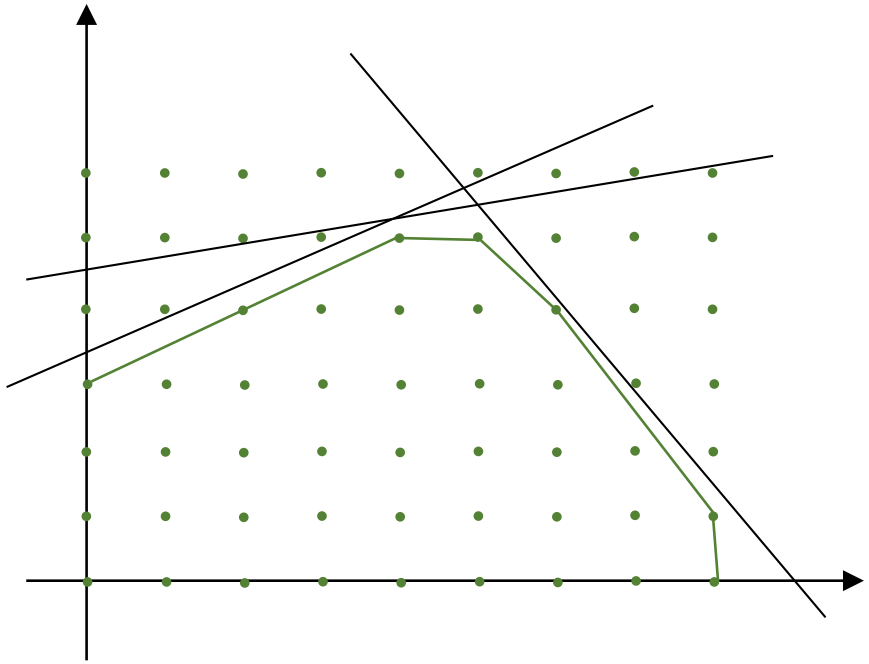
CE 205A
Transportation Logistics

Lecture 4

Branch and Bound

Previously on Transportation Logistics

Most methods for solving integer programs rely on relaxations and LP solutions.

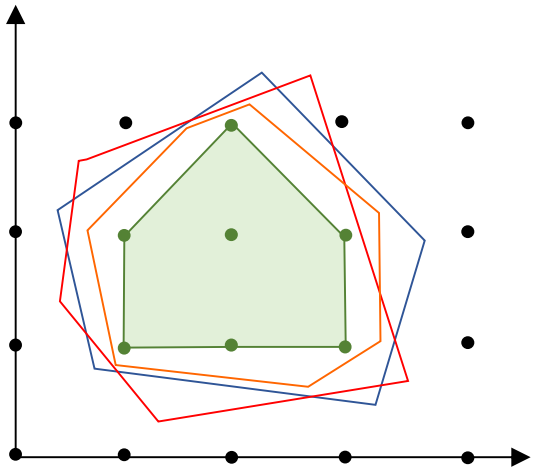


An ideal LP relaxation coincides with the convex hull of feasible points. (Why?)

Previously on Transportation Logistics

Consider the following feasible set of points

$$X = \{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2),(2,3)\}$$



There are infinitely many relaxation formulations that can lead to this feasible region.

Can you tell which of the three formulations P_1, P_2, and P_3 are strong? What about their LP relaxations?

In general, if P_1 and P_2 are two formulations of an IP, P_1 is a *stronger* formulation than P_2 if $P_1 \subset P_2$.

Previously on Transportation Logistics

Given $\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_l\mathbf{x}_{N_l} - \mathbf{B}^{-1}\mathbf{N}_u\mathbf{x}_{N_u}$, the objective can be written as

$$\begin{aligned} z &= \mathbf{c}_B^{\mathsf{T}}(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}_l\mathbf{x}_{N_l} - \mathbf{B}^{-1}\mathbf{N}_u\mathbf{x}_{N_u}) + \mathbf{c}_{N_l}^{\mathsf{T}}\mathbf{x}_{N_l} + \mathbf{c}_{N_u}^{\mathsf{T}}\mathbf{x}_{N_u} \\ &= \mathbf{c}_B^{\mathsf{T}}\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_{N_l}^{\mathsf{T}} - \mathbf{c}_B^{\mathsf{T}}\mathbf{B}^{-1}\mathbf{N}_l)\mathbf{x}_{N_l} + (\mathbf{c}_{N_u}^{\mathsf{T}} - \mathbf{c}_B^{\mathsf{T}}\mathbf{B}^{-1}\mathbf{N}_u)\mathbf{x}_{N_u} \end{aligned}$$

If $\mathbf{c}_{N_l}^{\mathsf{T}} - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}_l \geq \mathbf{0}^{\mathsf{T}}$, increasing the $\mathbf{x}_{N_l}$ values which are at their lower bounds will only increase the objective.

If $\mathbf{c}_{N_u}^{\mathsf{T}} - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}_u \leq 0$, decreasing $\mathbf{x}_{N_u}$ will again increase the objective.

Theorem

Suppose $\mathbf{x}^*$ *is a basic feasible solution and* $\mathbf{c}_{N_l}^{\mathsf{T}} - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}_l \geq \mathbf{0}^{\mathsf{T}}$ *and* $\mathbf{c}_{N_u}^{\mathsf{T}} - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}_u \leq \mathbf{0}^{\mathsf{T}}$, *then* $\mathbf{x}^*$ *is optimal*

What if the optimality conditions are violated for multiple non-basic variables? Which variable do you choose?

Lecture Outline

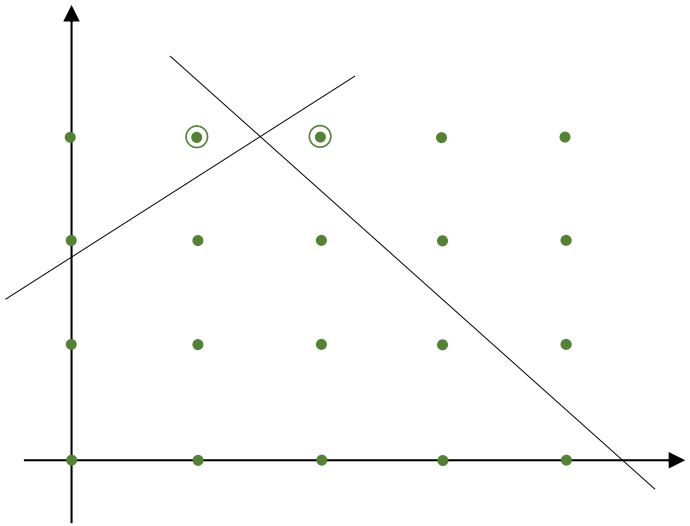
1. IP Solution Methods
2. Branch and Bound
3. Preprocessing

Lecture Outline

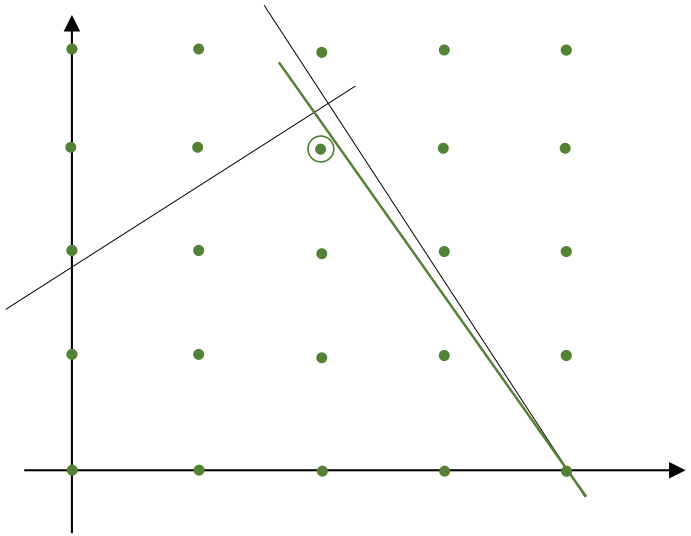
IP Solution Methods

We can always solve linear programming relaxations and round off the values. But this can cause two issues:

Infeasibility



Far from Optimality



IP Solution Methods

Bounds

Most methods for solving IPs rely on calculating bounds. These bounds can often be found quickly and improved with additional effort.

For a minimization problem, *primal bounds* or upper bounds are derived from feasible solutions. They range from being simple in some cases (such as the TSP), to being NP-complete or NP-hard for other kinds of problems such as routing problems with time windows.

Greedy and local-search based heuristics are standard ways to arrive at good upper bounds.

IP Solution Methods

Bounds

The lower bounds or *dual bounds* can be found from relaxations. These can be of two types

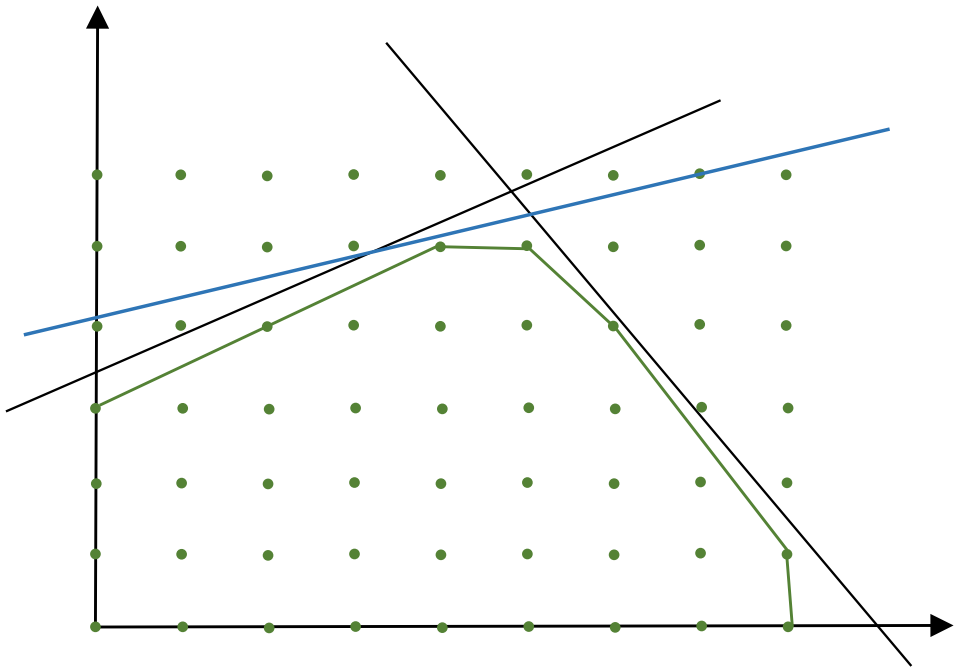
- The feasible region is expanded so that it becomes easier to solve the new problem. E.g., LP relaxations.
- The objective function is replaced by another function which is a lower bound estimate for all feasible solutions.

Other options include Lagrangian relaxation which will be discussed in detail later.

IP Solution Methods

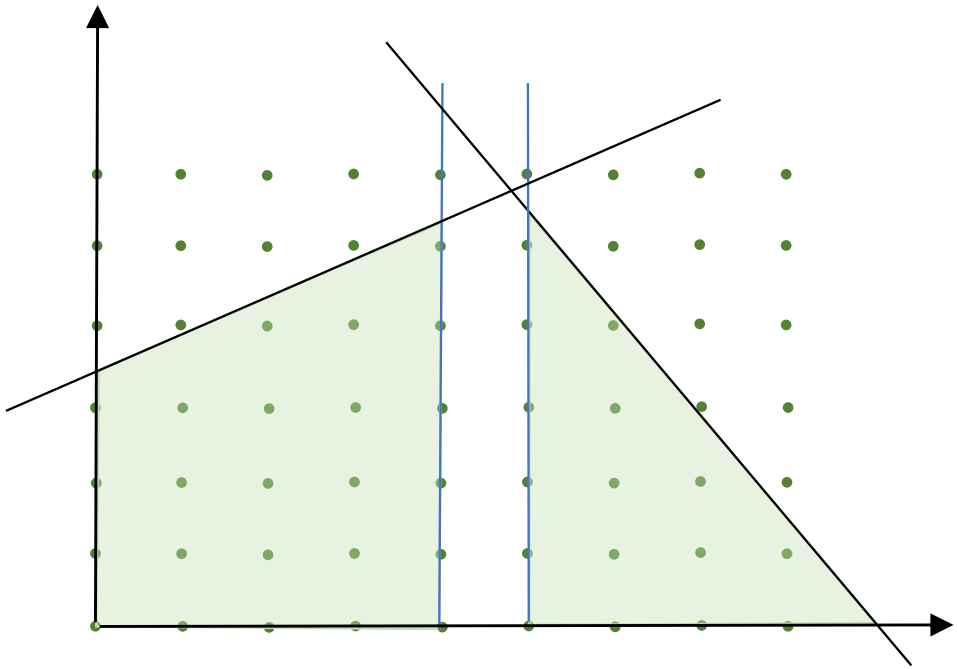
Cutting Planes

The idea behind cutting plane algorithms is to solve the LP relaxations and identify a hyperplane that separates the LP relaxation and integer feasible solutions of the problem.



This is done iteratively by solving a *separation problem* which determines new cuts to be added to the problem.

The branch and bound method is a divide-and-conquer method in which the feasible region is iteratively decomposed into smaller regions.

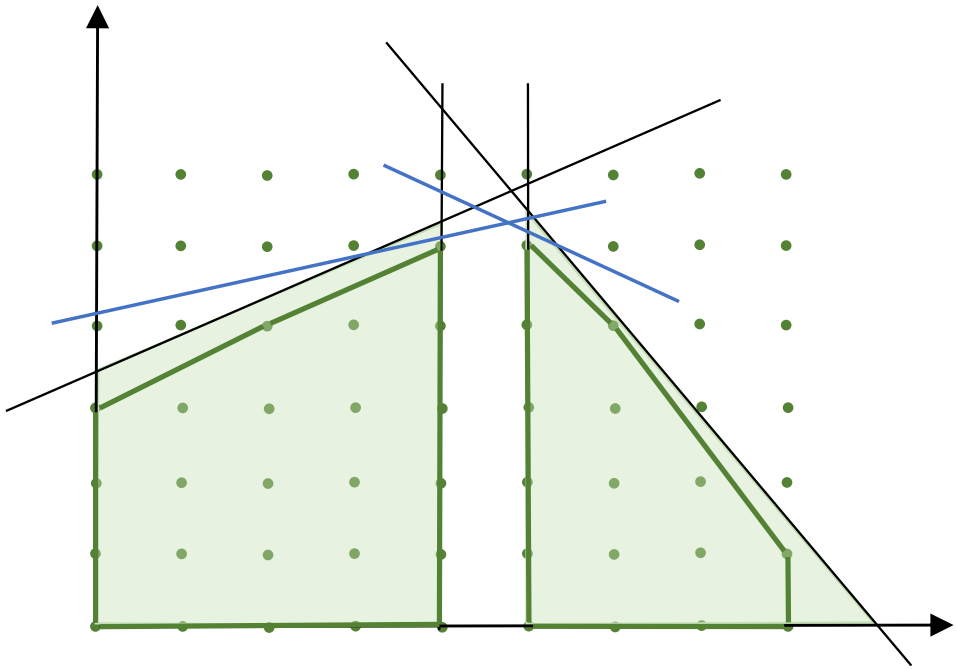


The method determines if a region of the feasible region should be further broken down using the upper and lower bounds of the problem.

IP Solution Methods

Branch and Cut

Cutting plane algorithms sometimes do not improve the LP relaxations by a significant margin. In such cases, they are combined with the branch and bound methods.



In this hybrid approach, one would iteratively decompose the feasible region and add cuts to the sub-regions.

Consider solving the LP relaxations of problems with a large number of variables (sometimes they may be exponential and prohibitive to specify/store) but fewer constraints.

$$\begin{bmatrix} & & & & & \\ \mathbf{A}_{\cdot 1} & \mathbf{A}_{\cdot 2} & \mathbf{A}_{\cdot 3} & \cdots & \mathbf{A}_{\cdot n-1} & \mathbf{A}_{\cdot n} \\ & & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Only a few variables are part of the optimal basis and the rest are zeros. Using column generation, we iteratively pick variables that can enter the basis. This is achieved by solving a *pricing* problem using the dual variables.

IP Solution Methods

Heuristics

Heuristics are a good way to find approximate solutions, especially when there are time constraints. These could be problem/model-based or generic in nature.

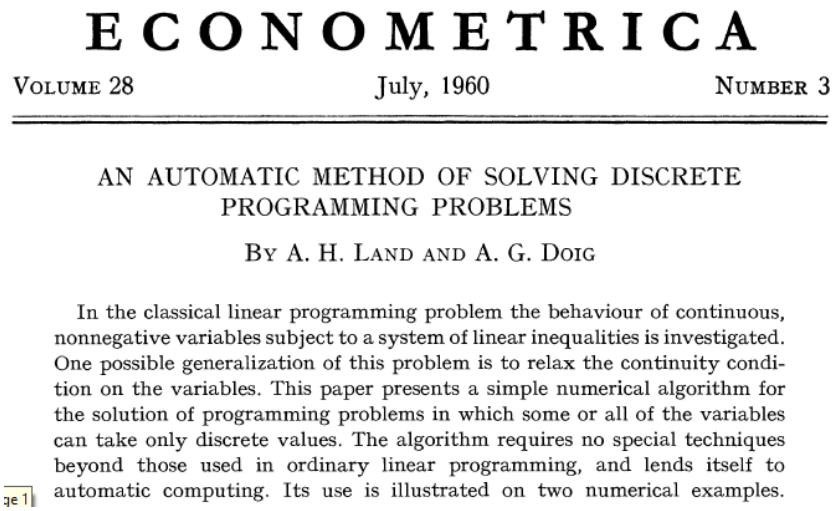
- Adaptive/Large/Variable Neighborhood Search (LNS, VLNS, ALNS)
- Genetic Algorithms
- Simulated Annealing
- Tabu Search

These methods typically start with a single or a pool of feasible solutions and try to find new feasible solutions and evaluate their objective value to see if they can be included or discarded.

Branch and Bound

Branch and Bound

History



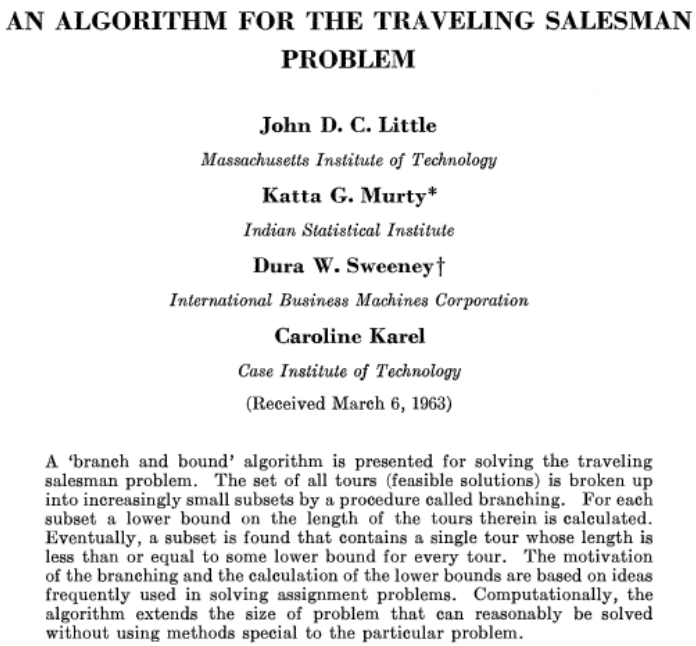
ECONOMETRICA

VOLUME 28 July, 1960 NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

BY A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated. One possible generalization of this problem is to relax the continuity condition on the variables. This paper presents a simple numerical algorithm for the solution of programming problems in which some or all of the variables can take only discrete values. The algorithm requires no special techniques beyond those used in ordinary linear programming, and lends itself to automatic computing. Its use is illustrated on two numerical examples.



AN ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM

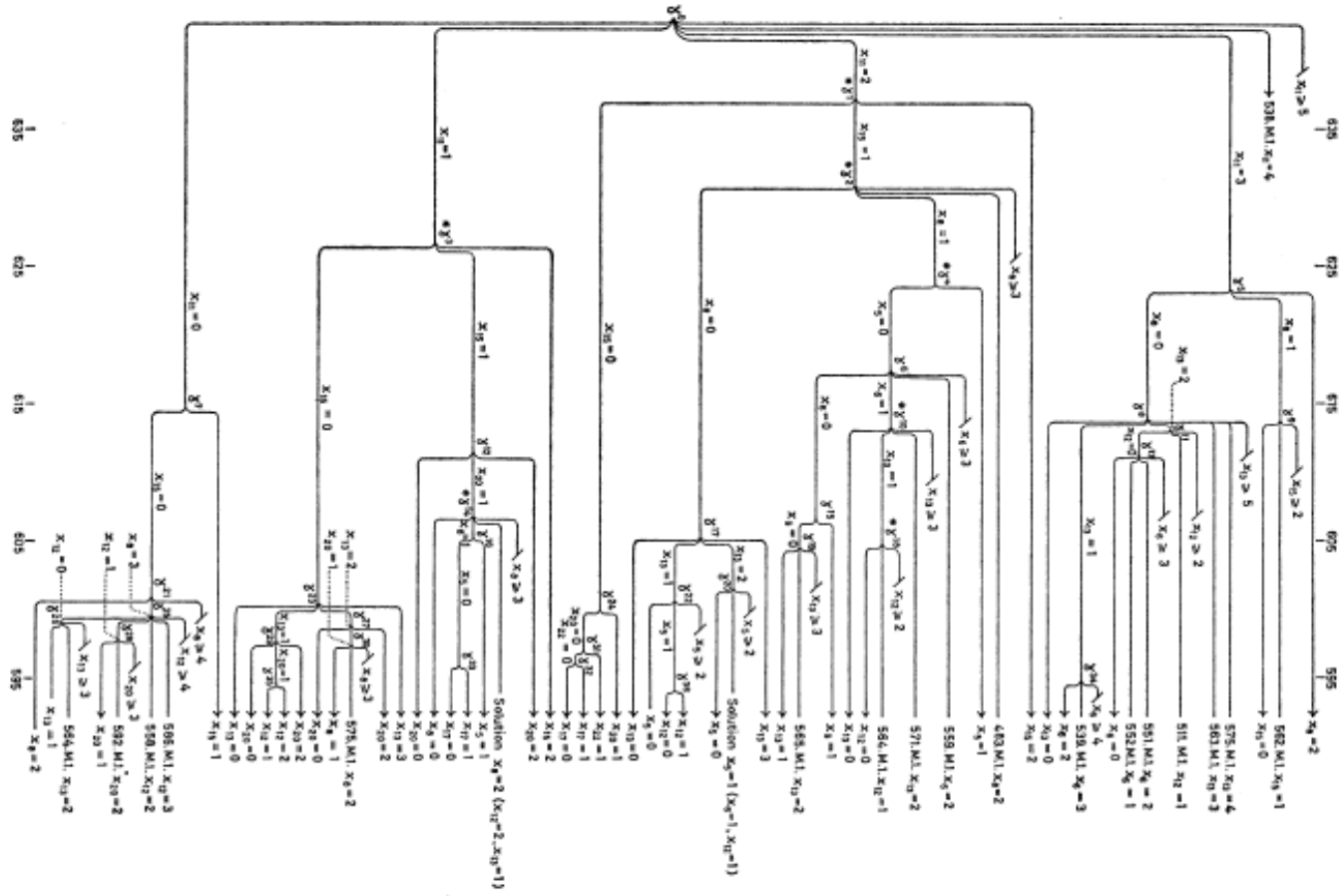
John D. C. Little
Massachusetts Institute of Technology
Katta G. Murty*
Indian Statistical Institute
Dura W. Sweeney†
International Business Machines Corporation
Caroline Karel
Case Institute of Technology

(Received March 6, 1963)

A 'branch and bound' algorithm is presented for solving the traveling salesman problem. The set of all tours (feasible solutions) is broken up into increasingly small subsets by a procedure called branching. For each subset a lower bound on the length of the tours therein is calculated. Eventually, a subset is found that contains a single tour whose length is less than or equal to some lower bound for every tour. The motivation of the branching and the calculation of the lower bounds are based on ideas frequently used in solving assignment problems. Computationally, the algorithm extends the size of problem that can reasonably be solved without using methods special to the particular problem.

See link for some historical notes on the 1963 paper.

Land and Doig's Branch and Bound Tree



Interviews by Land and Doig

Branch and Bound

Introduction

Branch and bound is a enumeration method that tries to split the feasible regions into sub-regions and solve the LP with extra constraints.

Consider an optimization problem $z = \min \mathbf{c}^{\mathsf{T}}\mathbf{x}$ such that $\mathbf{x} \in X$. Suppose X is broken down into smaller sets $X_1, \ldots, X_k$ such that $X = X_1 \cup \ldots \cup X_k$.

Let $z^k = \min \mathbf{c}^{\mathsf{T}}\mathbf{x}$ such that $\mathbf{x} \in X_k$. Then,

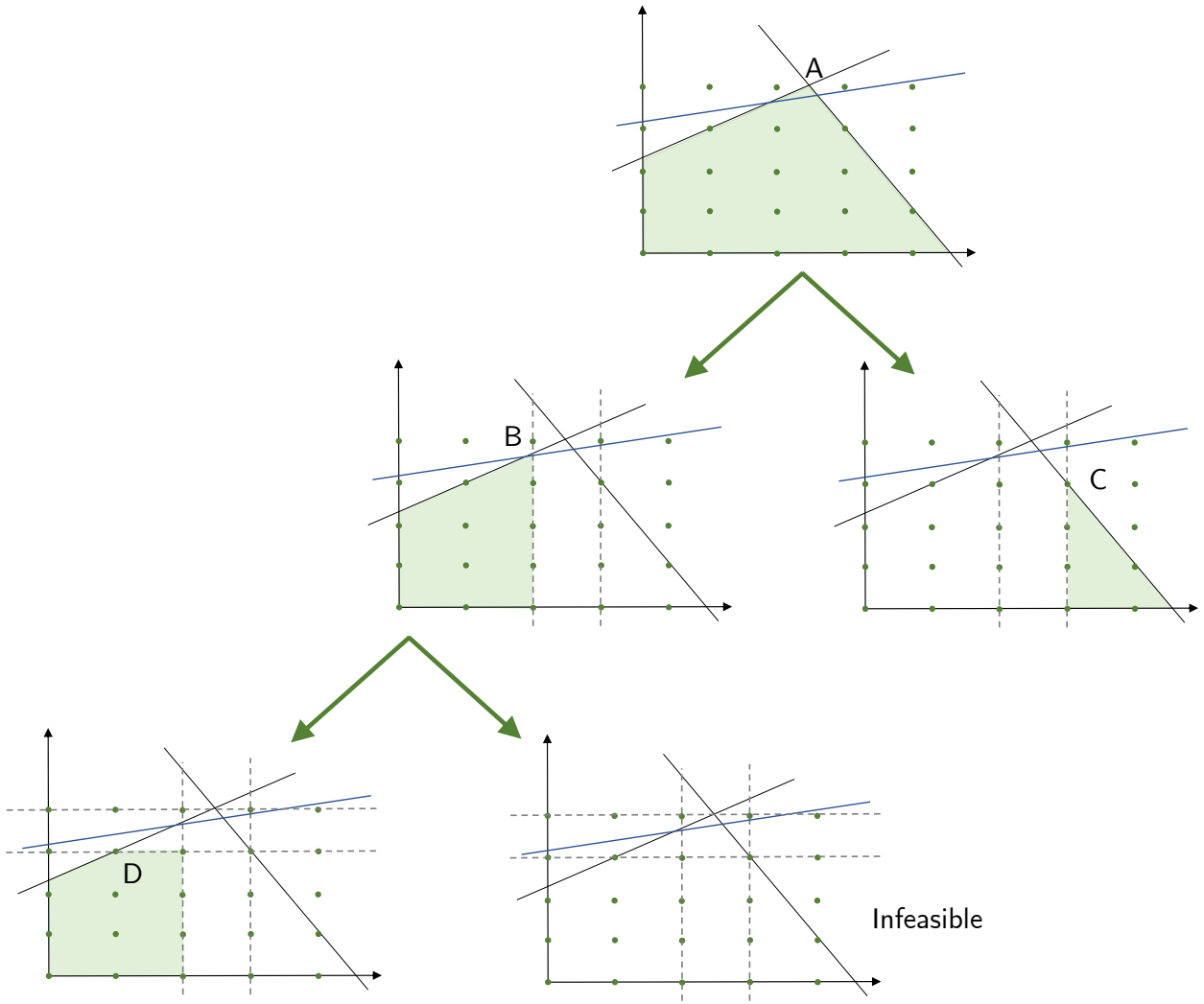
$$z = \min_k z_k$$

The immediate question is how do we create subsets of the feasible region.

Branch and Bound

Geometry

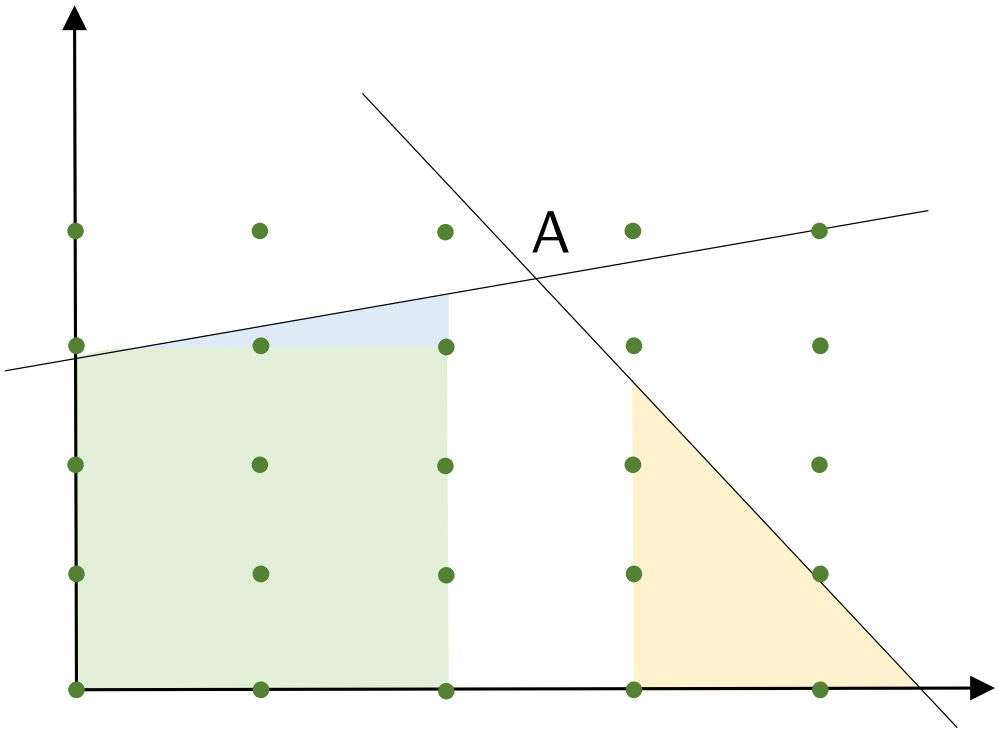
Suppose the blue line represent a level curve of the objective.



Branch and Bound

Geometry

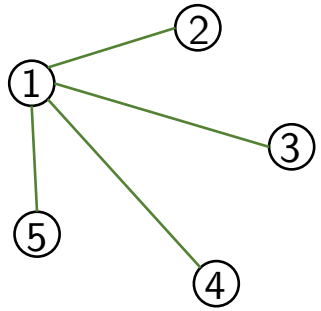
It is not necessary to split the problem into two branches at each node. One can perform multi-way branching as well.



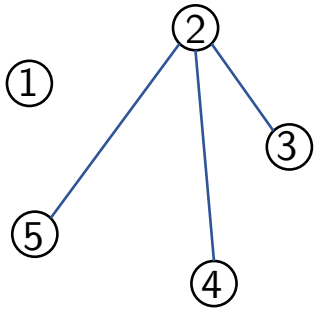
Branch and Bound

Geometry

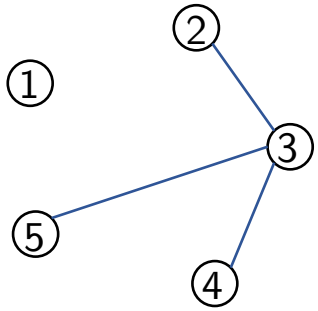
Decomposition strategies can be problem specific. Consider the TSP problem for instance.



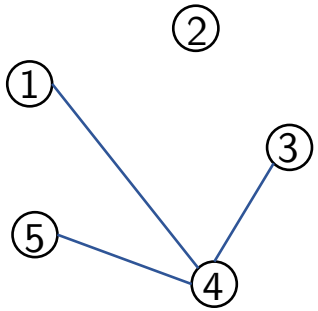
Level 1: 4 Problems



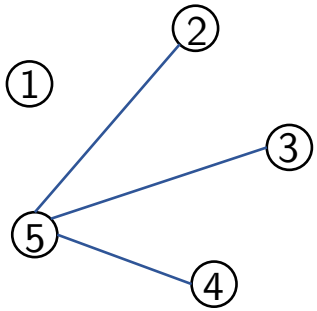
Level 2: 3 Problems



Level 2: 3 Problems



Level 2: 3 Problems



Level 2: 3 Problems

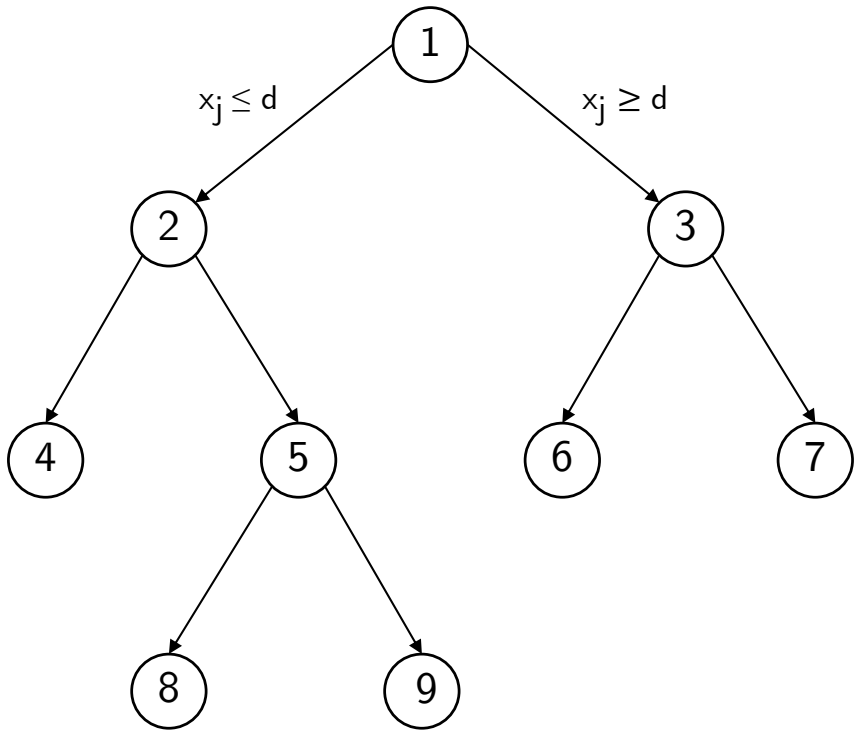
Branch and Bound

Relaxations

To understand the branch and bound method for a minimization problem, it is important to remember the following facts:

- The LP relaxation is a lower bound.
- Any IP solution is feasible and hence an upper bound.
- As we add more constraints to the original LP relaxation, the new LP does strictly worse (Why?).

Assume that an arc from one problem to the other indicates an extra constraint. Can you write mark the LP solutions at each node on the real line?



Branch and Bound

Pruning

However, in practice we do not know where z^* is located. Hence, we start with the LP relaxation lower bound and search for an IP upper bound. The difference between them is called the optimality gap.

The IP upper bound is also called the *incumbent*. Based on the earlier discussion, we can prune a B&B tree using the following three rules:

- Integrality
- Infeasibility
- Incumbent

Branch and Bound

Example

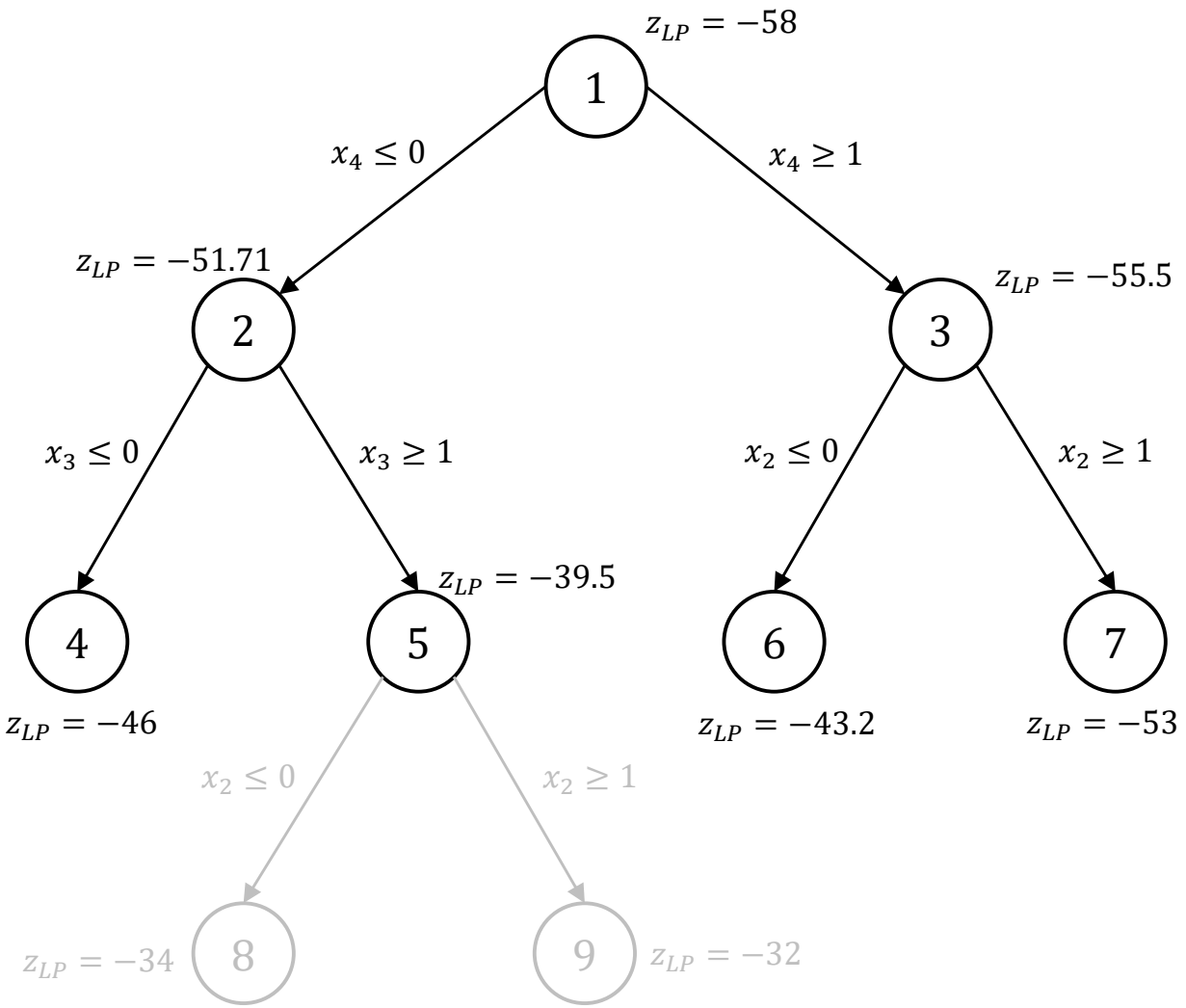
Solve the following optimization problem using branch and bound:

$$
\begin{aligned}
\min \quad & -8x_1 - 22x_2 - 10x_3 - 15x_4 \\
\text{s.t.} \quad & x_1 + 4x_2 + 7x_3 + 5x_4 \leq 11 \\
& 0 \leq x_1 \leq 3 \\
& 0 \leq x_2 \leq 1 \\
& 0 \leq x_3 \leq 1 \\
& 0 \leq x_4 \leq 1
\end{aligned}
$$

Branch and Bound

Example

The gap is measured based on $(z_{UB} - \min_{\text{active nodes}} z_{LB})/z_{UB} * 100$, where



Branch and Bound

Pseudocode

The following pseudocode summarizes the main steps involved in Branch and Bound. Several modifications are possible to speed-up the algorithm.

1. Set $z_{UB} = \infty$ and add the root to a list of active nodes.
2. Pick an element from the list of active nodes, solve its LP relaxation, and delete it. Terminate if the list is empty.
3. If the problem is infeasible or if $z^{LP} > z_{UB}$, go to Step 2. Else, two cases are possible.
   - If the solution is integral, update $z_{UB} = z^{LP}$ and go to Step 2.
   - If the solution is fractional, create two child nodes with $x_j \leq \lfloor x_j^{LP} \rfloor$ and $x_j \geq \lceil x_j^{LP} \rceil$, respectively and add them to the list of active nodes and go to Step 2.

Branch and Bound

Implementation Choices

Minor implementation choices can have a significant impact on the run time of a Branch and Bound algorithm.

- **Can we prioritize variable selection from the list of active nodes?** Typically, we pick the node whose parent LP solution is the smallest. This is the most 'promising' child node. Such a strategy is called the *best node first* approach.

  However, in the early stages, we do not have an upper bound (unless it is supplied from another heuristic). Hence, we first prefer to go down the tree to get an integer feasible solution. This is called the *depth first search* strategy. Once, an integer solution is found, one can switch to the best node first strategy.

Branch and Bound

Implementation Choices

- **How to choose a variable to branch on when multiple variables are fractional?** A popular option is to choose the most fractional variable, i.e., the variable closest to 0.5. Suppose F is the set of fractional variables, then choose

$$j \in \arg\max_{F} \min \left\{ x_j^{LP} - \lfloor x_j^{LP} \rfloor, 1 - x_j^{LP} + \lfloor x_j^{LP} \rfloor \right\}$$

- **Is there a way to solve the LPs faster?** Use Dual Simplex when descending from a parent to a child since the new constraint will not violate reduced cost optimality conditions. Otherwise, store the basis of the parent and use it as a starting point. Exploit simplex with bounds instead of adding the variables as constraints.

Branch and Bound

Implementation Choices

- **How can we reduced the number of nodes explored?**The number of nodes explored can grow very quickly and increase memory usage. Hence, one can delete all active nodes that are not promising as soon as the upper bounds change using the LP relaxation solution of their parent. (How?)

  Other options include:

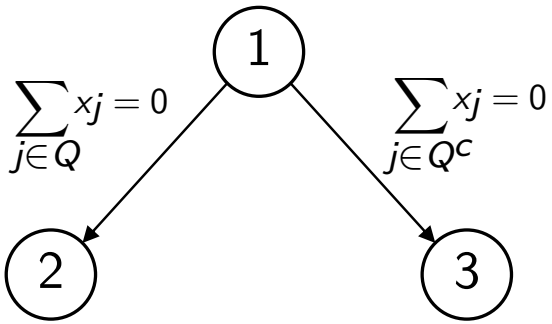
  1. Generalized upper bound (GUB) or special order sets (SOS)
  2. Strong branching
  3. Adding cuts
  4. Branching strategies that make the tree balanced (e.g., imagine two-way and multi-way branching of TSP)

Branch and Bound

GUB or SOS Branching

Consider the constraints of the following form which we saw in the last lecture $\sum_{j=1}^{k} x_j = 1$.

When we branch on a single fractional variable, one side of the tree where $x_j = 1$ has only one node, but the other child has $k - 1$ nodes and hence the tree becomes unbalanced.



To avoid this issue, we split the feasible region using constraints of the form $\sum_{j \in Q} x_j = 0$ as shown in the figure.

We find an index r such that $r = \min\left\{t : \sum_{i=1}^{t} x_i^{LP} \geq 1/2\right\}$. The two child nodes are created by setting $Q = \{1, \ldots, r\}$.

Branch and Bound

GUB or SOS Branching

Apply GUB branching on the following problem after branching on x_{13} at the root.

$$\begin{aligned}
\max\ & 50x_1 + 47x_2 + 44x_3 + 41x_4 + 38x_5 + 36x_6 + 31x_7 + 29x_8 \\
& + 27x_9 + 25x_{10} + 23x_{11} + 21x_{12} + 20x_{13} \\
\text{s.t.}\ & \sum_{j=1}^{13} (21 - j)x_j \leq 22 \\
& \sum_{j=1}^{12} x_j = 1 \\
& x_j \in \{0, 1\}\ \forall\ j = 1, \ldots, 13
\end{aligned}$$

Can you visualize how the feasible region is being split to generate the child nodes?

In the strong branching strategy, CPLEX explores the up and down branches to either optimality or for a certain number of levels and updates the bounds.

Restricting the variables to a lower and higher values are also called down and up branches, respectively.

The direction which is most promising is chosen as the branching variable at the parent node.

Branch and Bound

CPLEX Output and Settings

Check out the details that CPLEX prints for different nodes of the branch and bound tree. Note that integer solution is labeled 'cutoff' in CPLEX node log.

- Presolve
- Branching directions
- Multiple solutions
- Number of nodes explored
- Node selection
- Variable selection

Preprocessing

Preprocessing

Introduction

Before running a branch and bound algorithm, most solvers have a preprocessing stage that may perform one or more of the following operations:

- Tighten bounds
- Remove redundant constraints
- Fix variables

These steps can reduce the number of variables and constraints as well as help solve the LP relaxations faster.

Preprocessing

Tighten Bounds

Consider the inequality $\sum_{j=1}^{n} a_j x_j \leq b$, where $l_j \leq x_j \leq u_j$ for all $j = 1, \ldots, n$.

Can you improve the upper bound on x_1 if $a_1 > 0$?

$$x_1 \leq \min \left\{ u_1, \frac{1}{a_1} \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) \right\}$$

Can you improve the lower bound on x_1 if $a_1 < 0$?

$$x_1 \geq \max \left\{ l_1, \frac{1}{a_1} \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) \right\}$$

Can you do better? You can update them using the new bounds and round up the new lower bounds and round down the upper bounds if they are fractional, i.e., $\lceil l_i \rceil \leq x_i \leq \lfloor u_i \rfloor$

Preprocessing

Remove Redundant Constraints

When is a constraint $\sum_{j=1}^{n} a_j x_j \leq b$, where $l_j \leq x_j \leq u_j$ for all $j = 1, \ldots, n$ redundant? Check if

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b$$

Likewise the problem is infeasible if

$$\sum_{j:a_j>0} a_j l_j + \sum_{j:a_j<0} a_j u_j > b$$

Preprocessing

Fix Variables

Consider an optimization problem of the form $\min \mathbf{c}^{\mathsf{T}}\mathbf{x}$, s.t., $\mathbf{Ax} \geq \mathbf{b}$ and let $x_j \in [l_j, u_j] \forall\, j = 1, \ldots, n$.

- Can you fix the value of a variable x_j if $a_{ij} \geq 0 \,\forall\, i = 1, \ldots, m$ and $c_j < 0$? Set $x_j = u_j$.
- Can you fix the value of a variable x_j if $a_{ij} \leq 0 \,\forall\, i = 1, \ldots, m$ and $c_j > 0$? Set $x_j = l_j$.

Prove the above results using a duality argument.

Preprocessing

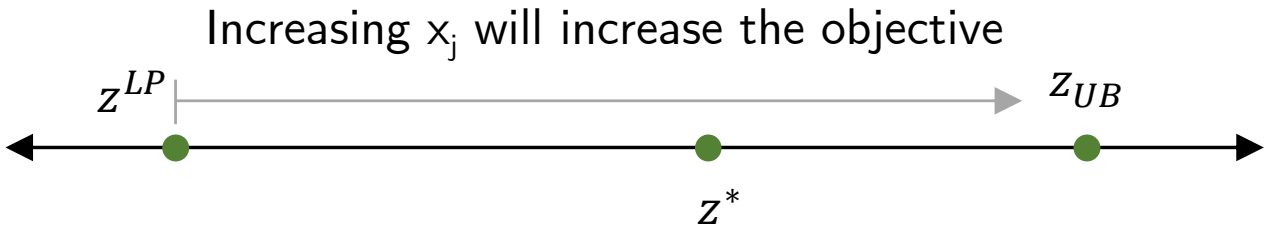
Fix Variables

Recall that the LP relaxation after simplex terminates can be written as

$$z^{LP} = \mathbf{c}_B^{\mathsf{T}}\mathbf{B}^{-1}\mathbf{b} + \bar{\mathbf{c}}_{N_l}^{\mathsf{T}}\mathbf{x}_{N_l} + \bar{\mathbf{c}}_{N_u}^{\mathsf{T}}\mathbf{x}_{N_u}$$

Suppose z_{UB} is an primal/upper bound of the integer program. We can update the upper bound of a non-basic variable j that is currently at its lower bound (if $\bar{c}_j \neq 0$) as follows

$$x_j \leq \min\left\{u_j, \left\lfloor \frac{z_{UB} - z^{LP}}{\bar{c}_j} \right\rfloor\right\}$$



Likewise, for a non-basic variable at its upper bound, the lower bound can be adjusted to $\max\{l_j, \lceil \frac{z^{LP} - z_{UB}}{\bar{c}_j} \rceil\}$.

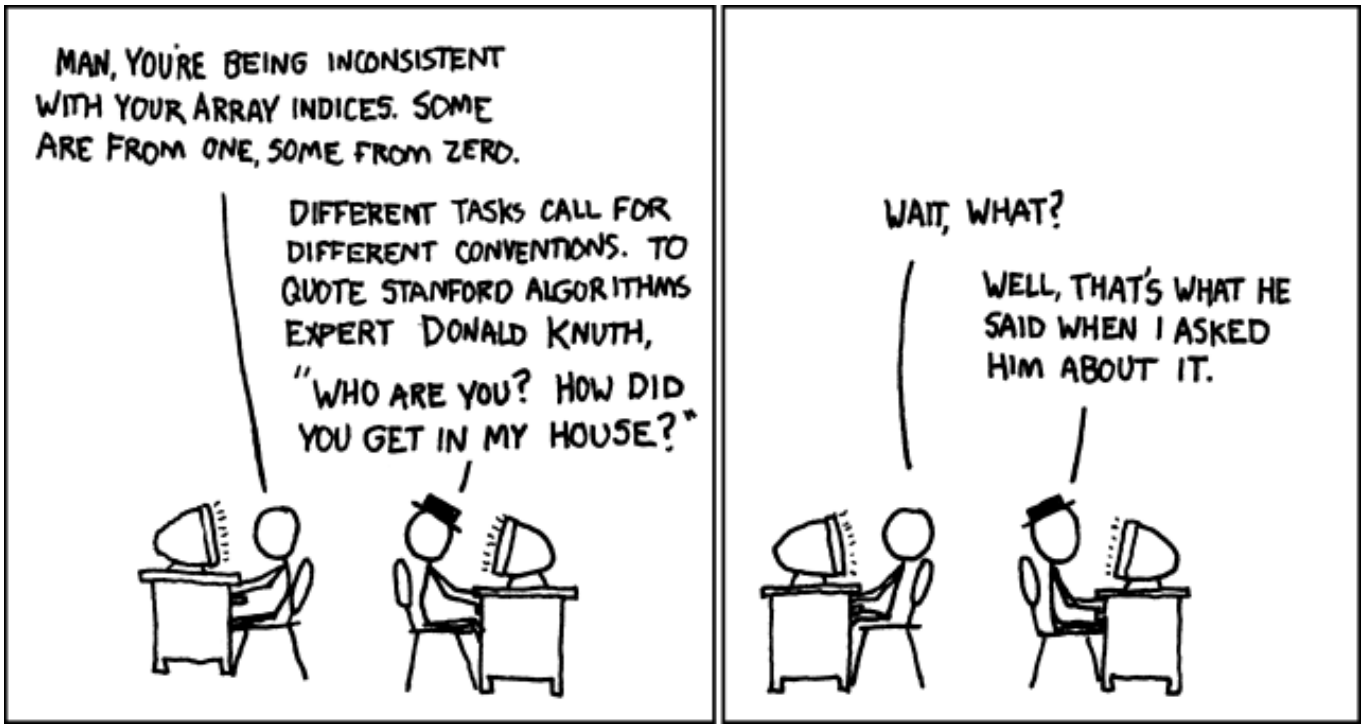
Preprocessing

Example

Apply the above preprocessing rules on the following LP.

$$
\begin{aligned}
\min \quad & -2x_1 - x_2 + x_3 \\
\text{s.t.} \quad & 5x_1 - 2x_2 + 8x_3 \leq 15 \\
& 8x_1 + 3x_2 - x_3 \geq 9 \\
& x_1 + x_2 + x_3 \leq 6 \\
& 0 \leq x_1 \leq 3 \\
& 0 \leq x_2 \leq 1 \\
& 1 \leq x_3
\end{aligned}
$$

Your Moment of Zen



Source: xkcd