Combinatorial Optimization: Integer Linear Programming

sophie.demassey@minesparis.psl.eu


design a system
(dimension, position)
to minimize a cost or
to maximize an usage

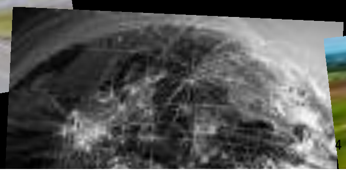
strategical decision is
static or long-term optimization

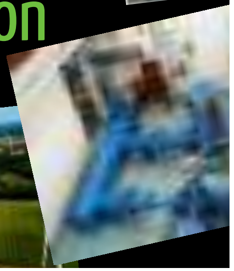





operate a system
(schedule, arrange, assign)
to minimize cost, distance, time, energy
or to maximize profit, performance

operation decision is
short-term optimization



decision-making (specs 1)

- accurate mathematical models of physical systems
- optimality certificates
- flexible algorithms for changing problems
- efficient algorithms for complex/large problems

decision-making (specs 2)

- discrete decisions and logical conditions
  combinatorial optimization
- uncertain data
  stochastic optimization

combinatorial optimization*

*here = Mixed Integer Linear Programming (MILP)

This course is about:

- techniques to model or approximate problems as MILPs
- some applications
- notions of complexity
- generic techniques to solve MILPs: the main ideas
- modern solvers and their usage

why MILP ?

versatility:

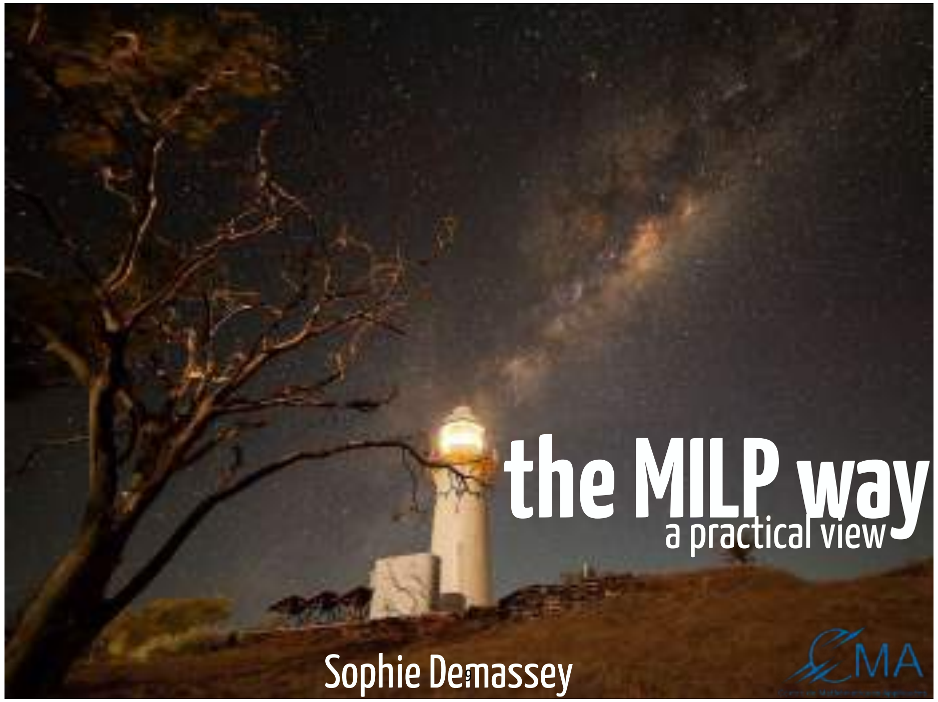
- logical conditions as binary variables and linear inequalities
- physic or economic constraints & objectives as piecewise-linear functions
- convex MINLP solvers incorporate MILP relaxations and solvers

flexibility:

- one generic model = one generic solver
- one specific problem = one generic solver + specific components

efficiency:

- easy LP + enumeration
- sophisticated algorithmic components



1 how to model ?

2 how difficult ?

3 how to solve ?

1 how to model ?

Mixed Integer Linear Program

$\min f(x) \mid g(x) \geq 0,\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$
with linear functions f and g:

$$\begin{aligned} &\min cx \\ &Ax \geq b \\ &x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} a_{ij} x_j \geq b_i \ \forall i = 1..m \\ &x_j \in \mathbb{Z} \ \forall j = 1..p \\ &x_j \in \mathbb{R} \ \forall j = p+1..n \end{aligned}$$

Mixed Integer Linear Program

$\min f(x) \mid g(x) \geq 0,\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$
with linear functions f and g:

$$\begin{aligned} &\min cx \\ &Ax \geq b \\ &x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \end{aligned}$$

$c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$

- objective cx
- linear constraints $Ax \geq b$
- integrity constraints $x_1, \ldots, x_p \in \mathbb{Z}$
- constraint rhs (right hand side) b
- cost vector c
- solution space $\mathbb{R}^n$
- feasible set $\{x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \mid Ax \geq b\}$

modeling with $\mathbb{B}$

~~true~~ 1 or ~~false~~ 0

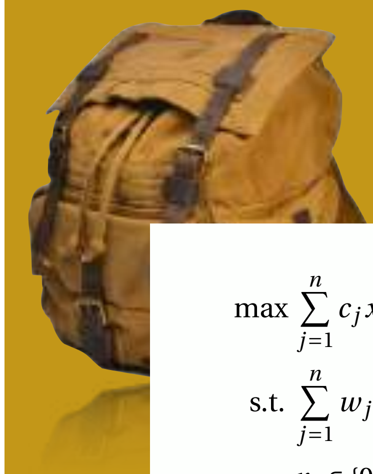
in an optimal solution...

- is item j selected ? $x_j \in \{0,1\}$
- is item j associated to item i ? $x_{ij} \in \{0,1\}$
- is non-negative y greater than a ? $y \geq ax, x \in \{0,1\}$
- at most n items $x_1, \ldots, x_n \in \{0,1\}$

Integer Knapsack Problem

Input n items, value c_j and weight w_j for each item j, capacity K.
Output a maximum value subset of items whose total weight does not exceed K.

x_j is item j packed ?



Integer Knapsack Problem

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \leq K$$

$$x_j \in \{0,1\} \qquad j = 1..n$$

x_j is item j packed ?

logic with binaries

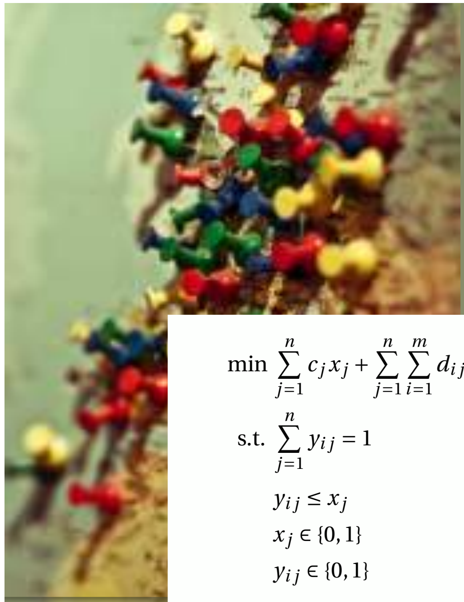
x,y binary variables; f continuous variable; a, k, n constants

- either x or y $\quad x + y = 1$
- if x then y $\quad y \geq x$
- if x then f ≤ a $\quad f \leq ax + M(1 - x)$
- at most 1 out of n $\quad x_1 + \cdots + x_n \leq 1$
- at least k out of n $\quad x_1 + \cdots + x_n \geq k$

Uncapacitated Facility Location Problem

Input n facility locations, m customers, cost c_j to open facility j, cost d_{ij} to serve customer i from facility j
Output a mimimum (opening and service) cost assignment of customers to facilities.

x_j is location j open ? y_{ij} is customer i served from j ?



Uncapacitated Facility Location Problem

$$\min \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij} y_{ij}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{j=1}^{n} y_{ij} = 1 && i = 1..m \\ & y_{ij} \leq x_j && j = 1..n,\ i = 1..m \\ & x_j \in \{0,1\} && j = 1..n \\ & y_{ij} \in \{0,1\} && j = 1..n,\ i = 1..m \end{aligned}$$

tions, m
open
to serve
ity j
ning and
ent of
es.

x_j is location j open ? y_{ij} is customer i served from j ?

K-median clustering

Input n data points, distance d_{ij} between each two points i,j, number k of clusters.
Output k centers minimizing the sum of distances between each point and its nearest center.

x_j is j a center ? y_{ij} is j the nearest center of i ?

K-median clustering

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} y_{ij}$$

$$\text{s.t.} \sum_{j=1}^{n} y_{ij} = 1 \qquad i = 1..n$$

$$y_{ij} \le x_j \qquad i,j = 1..n$$

$$\sum_{j=1}^{n} x_j = k$$

$$y_{ij} \in \{0,1\}, x_j \in \{0,1\} \qquad i,j = 1..n$$

distance
points
sters.
imizing
between
earest

x_j is j a center ? y_{ij} is j the nearest center of i ?

1||Cmax Scheduling Problem

Input n tasks, duration pi for each task i, one machine
Output a minimal makespan schedule of the tasks on the machine without overlap

x_{ij} does i precede j ? s_j starting time of j

1||Cmax Scheduling Problem

$$\min s_{n+1}$$

$$\text{s.t. } s_{n+1} \ge s_j + p_j \qquad j = 1..n$$

$$s_j - s_i \ge M x_{ij} + (p_i - M) \qquad i,j = 1..n$$

$$x_{ij} + x_{ji} = 1 \qquad i,j = 1..n; i < j$$

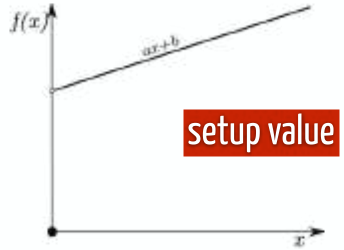
$$s_j \in \mathbb{Z}_+ \qquad j = 1..n+1$$

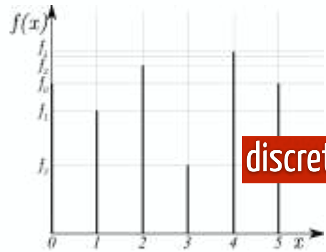
$$x_{ij} \in \{0,1\} \qquad i,j = 1..n$$

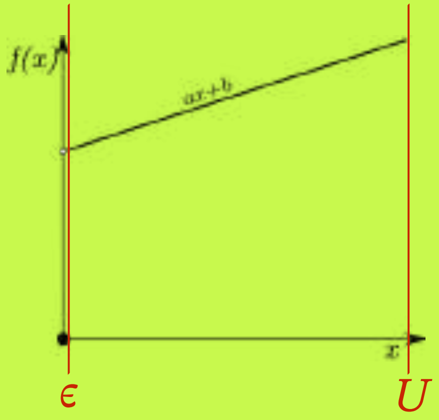
ation pi
he machine
akespan
sks on the
erlap

x_{ij} does i precede j ? s_j starting time of j

non-linear functions



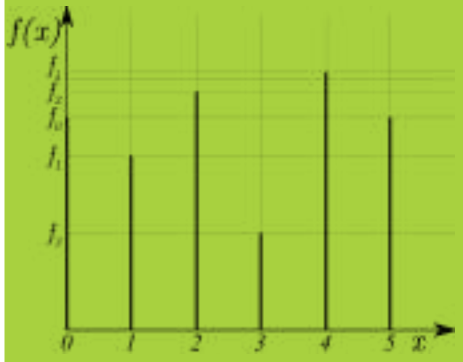




setup value

$$f(x) = ax + b\delta$$
$$\epsilon\delta \le x \le U\delta$$
$$\delta \in \{0,1\}$$

δ is x positive ?

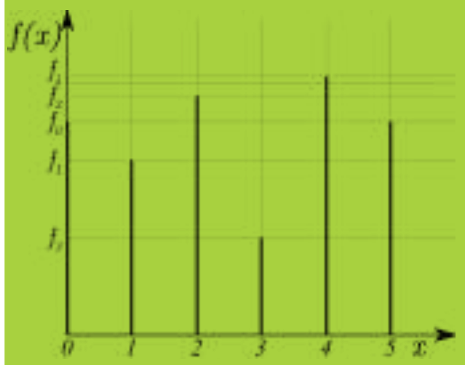


discrete values

$$f(x) = \sum_i \delta_i f_i$$
$$\sum_i i\delta_i = x$$
$$\sum_i \delta_i = 1$$
$$\delta_i \in \{0,1\} \; i = 0..n$$

δ_i is $x=i$ (and $f(x)=f_i$) ?

Special Ordered Set of type 1:
ordered set of variables, all zero except at most one



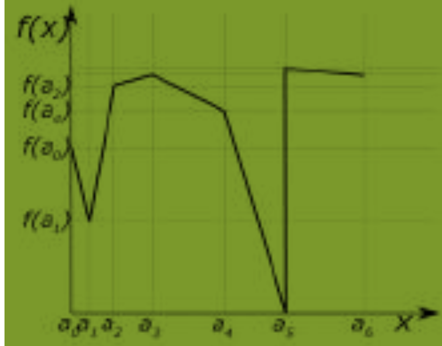
discrete values

$$f(x) = \sum_i \delta_i f_i$$
$$\sum_i i\delta_i = x$$
$$\sum_i \delta_i \geq 1$$
$$\delta_i \in \{0,1\} \; i = 0..n$$

SOS1(δ)

δ_i is $x=i$ (and $f(x)=f_i$) ?

Special Ordered Set of type 2:
ordered set of variables, all zero except at most two consecutive



piecewise linear

$$f(x) = \sum_i \lambda_i f(a_i)$$
$$\sum_i a_i\lambda_i = x$$
$$\sum_i \lambda_i = 1$$
$$\lambda_i \in [0,1] \; i = 0..n$$

SOS2(λ)

λ_i is $x=a_i$? (then $\lambda_i a_i + \lambda_{i+1} a_{i+1}$ in $[a_i, a_{i+1}]$ if $\lambda_i + \lambda_{i+1} = 1$)

$$x_i = 5$$

to order i is the 5th item

to count 5 items are selected

to measure time task i starts at time 5

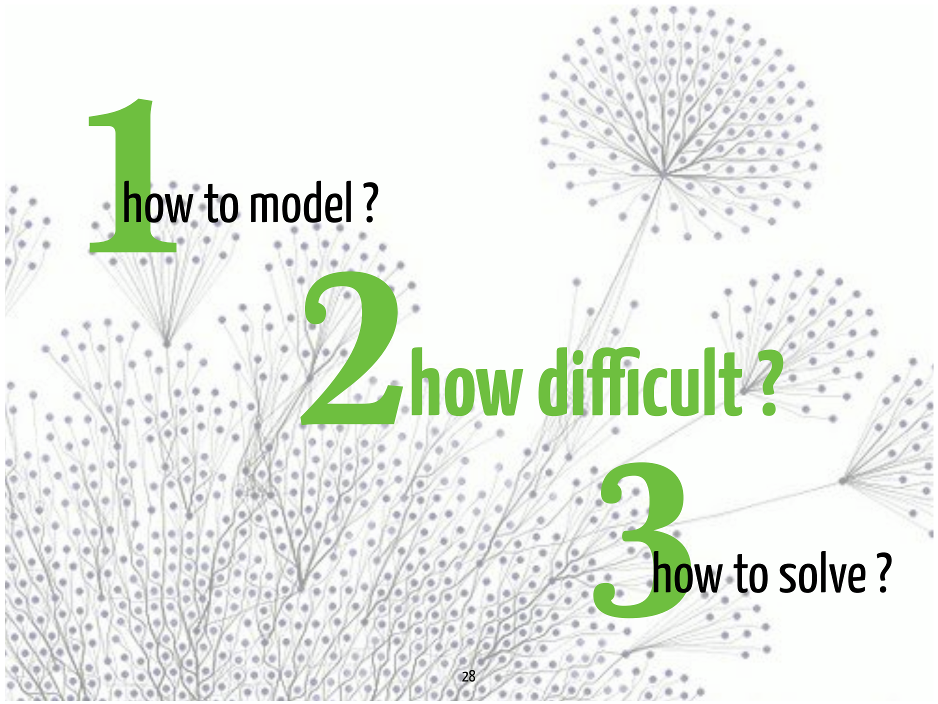
to measure space item i is located on floor 5

$$\simeq \delta_{i5} = 1$$

Binary Integer Linear Program (BIP) $\{0,1\}^n$

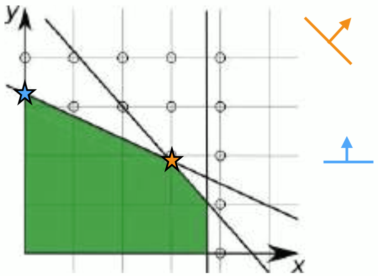
Integer Linear Program (IP) $\mathbb{Z}^n$

Mixed Integer Linear Program (MIP) $\mathbb{Z}^n \cup \mathbb{Q}^n$

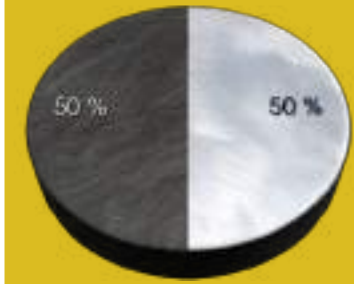


Linear Programming cheat sheet

LP is easy

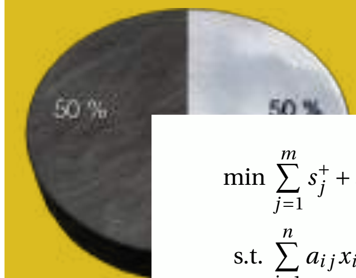


- MILP without integrality = LP-relaxation
- linear inequality = halfspace
- LP feasible set = polyhedron
- convex optimization
- if LP is feasible and bounded, at least one vertex is optimal
- primal simplex algorithm: visit adjacent vertices as cost decreases
- strong duality: $\min_x\{cx \mid Ax \geq b, x \geq 0\} = \max_u\{ub \mid uA \leq c, u \geq 0\}$
- interior point method runs in polynomial time (simplex can be better in practice)



Market Split Problem

Input 1 company, 2 divisions, m products with availabilities d_j, n retailers with demands a_{ij} in each product j.
Output an assignment of the retailers to the divisions approaching a 50/50 production split.

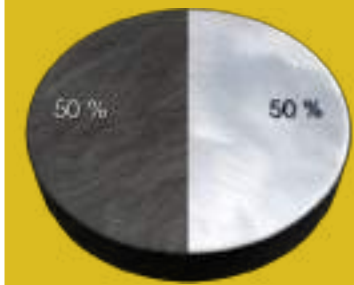


Market Split Problem

$$
\begin{aligned}
\min \ & \sum_{j=1}^{m} s_j^+ + s_j^- & \\
\text{s.t.} \ & \sum_{i=1}^{n} a_{ij} x_i + s_j^+ - s_j^- = \frac{d_j}{2} & j = 1..m \\
& x_i \in \{0,1\} & i = 1..n \\
& s_j^+ \geq 0, s_j^- \geq 0 & j = 1..m
\end{aligned}
$$

sions, m
ities d_j,
s a_{ij} in
the
ons
duction

x_i is retailer i assigned to division 1 ?
s_j gap to the 50% split goal for product j

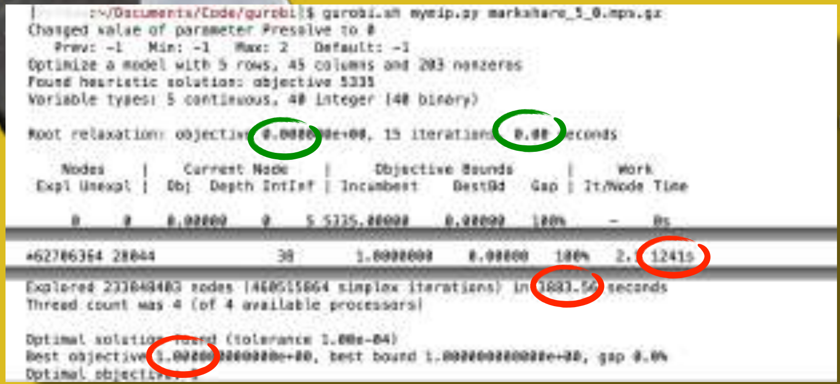


MIPLIB markshare_5_0

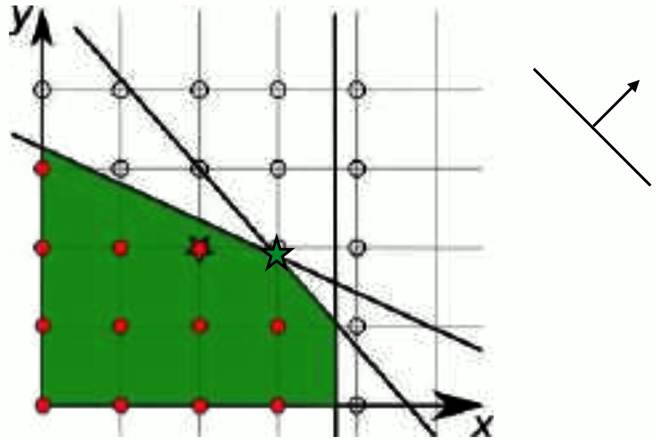
Input 5 products, 40 retailers
Output . . . . (hold the line please) . . . .

Int Opt = 1
Solution time = 20 minutes
Proof time = > 1 hour

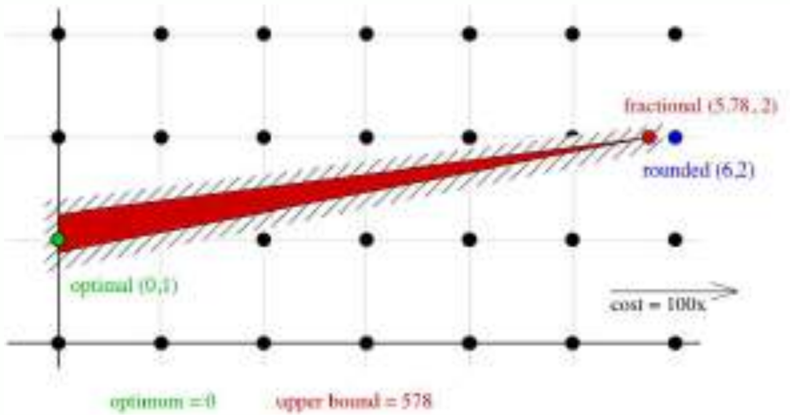
MIPLIB markshare_5_0



ILP ≠ LP-relaxation



ILP ≠ round LP-relaxation



general ILP is NP-hard

small problems are easy
some specific problems are easy

1||Cmax Scheduling Problem

$$
\begin{array}{lll}
\min & s_{n+1} \quad \boxed{= p_1+\ldots+p_n} & \\
\text{s.t.} & s_{n+1} \geq s_j + p_j & j = 1..n \\
& s_j - s_i \geq M x_{ij} + (p_i - M) & i,j = 1..n \\
& x_{ij} + x_{ji} = 1 & i,j = 1..n; i<j \\
& s_j \in \cancel{\mathbb{Z}_+} \geq 0 & j = 1..n+1 \\
& x_{ij} \in \{0,1\} & i,j = 1..n
\end{array}
$$

ation pi
ne machine
akespan
sks on the
erlap

Capacitated Transhipment Problem

Input digraph (V,A), demand or supply b_i at each node i, capacity h_{ij} and unit flow cost c_{ij} for each arc (i,j)
Output a mimimum cost integer flow to satisfy the demand

x_{ij} flow on arc (i,j)

Capacitated Transhipment Problem

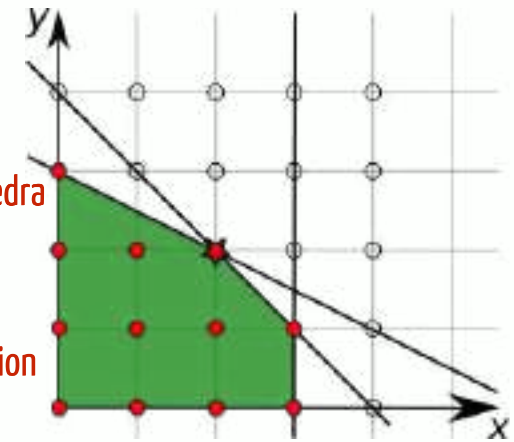
$$
\begin{array}{lll}
\min & \sum_{(i,j)\in A} c_{ij} x_{ij} & \\
\text{s.t.} & \sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ij} = b_i & i \in V \\
& x_{ij} \leq h_{ij} & (i,j) \in A \\
& x_{ij} \in \cancel{\mathbb{Z}_+} \geq 0 & (i,j) \in A
\end{array}
$$

, demand
h node i,
t flow
c (i,j)
st integer
e demand

x_{ij} flow on arc (i,j)

LP = ILP sometimes

integral polyhedra
=
convex hull
=
ideal formulation



totally unimodular matrix
(theory)

$$(P) = \max\{\ cx \mid Ax \leq b, x \in \mathbb{Z}_+^n\ \}$$

- basic feasible solutions of the LP relaxation $(\bar{P})$ take the form: $\bar{x} = (\bar{x}_B, \bar{x}_N) = (B^{-1}b, 0)$ where B is a square submatrix of (A, I_m)
- Cramer's rule: $B^{-1} = B^*/det(B)$ where B^* is the adjoint matrix (made of products of terms of B)
- Proposition: if (P) has integral data (A, b) and if $det(B) = \pm 1$ then $\bar{x}$ is integral

Definition

A matrix A is totally unimodular (TU) if every square submatrix has determinant $+1$, -1 or 0.

Proposition

If A is TU and b is integral then any optimal solution of $(\bar{P})$ is integral.

totally unimodular matrix
(practice)

How to recognize TU ?

Sufficient condition

A matrix A is TU if

- all the coefficients are $+1$, -1 or 0
- each column contains at most 2 non-zero coefficient
- there exists a partition (M_1, M_2) of the set M of rows such that each column j containing two non zero coefficients satisfies $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

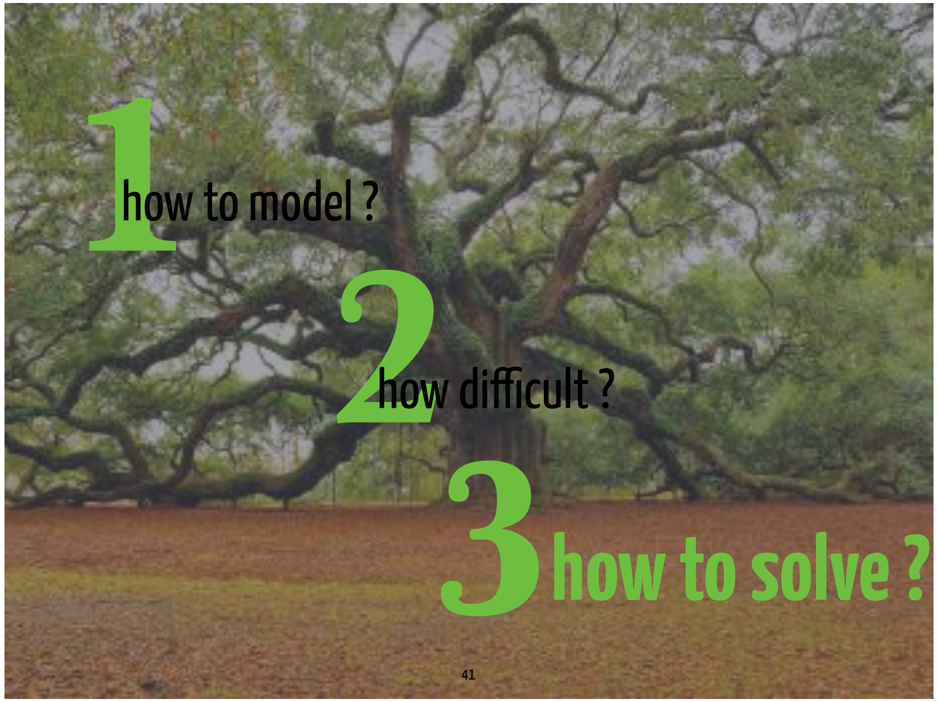
Proposition

A is TU $\iff$ A^t is TU $\iff$ (A, I_m) is TU
where A^t is the transpose matrix, I_m the identitiy matrix

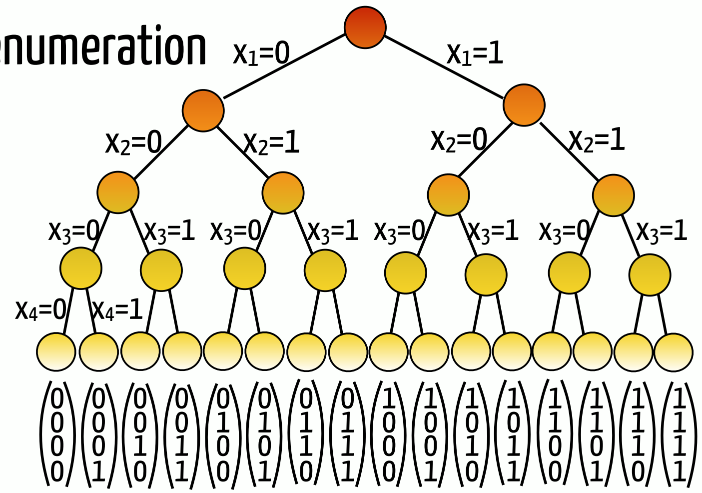
Interlude

Show that the **Transhipment** ILP is **ideal**

Show that the **Scheduling** ILP is **NOT ideal**



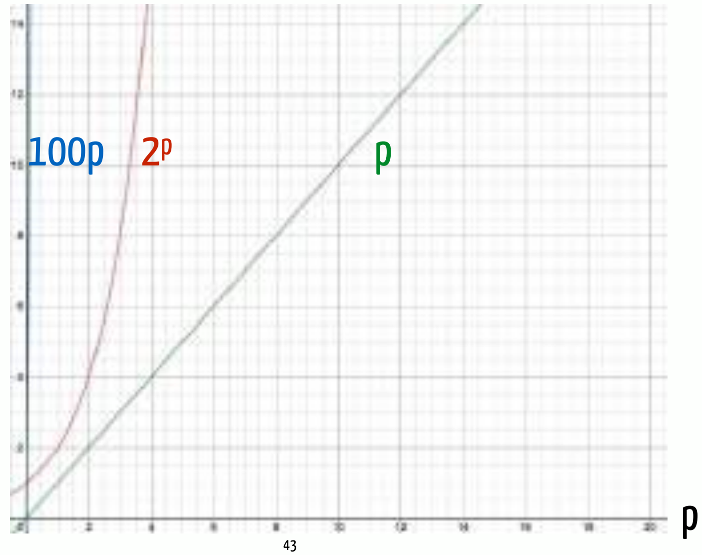
Complete enumeration



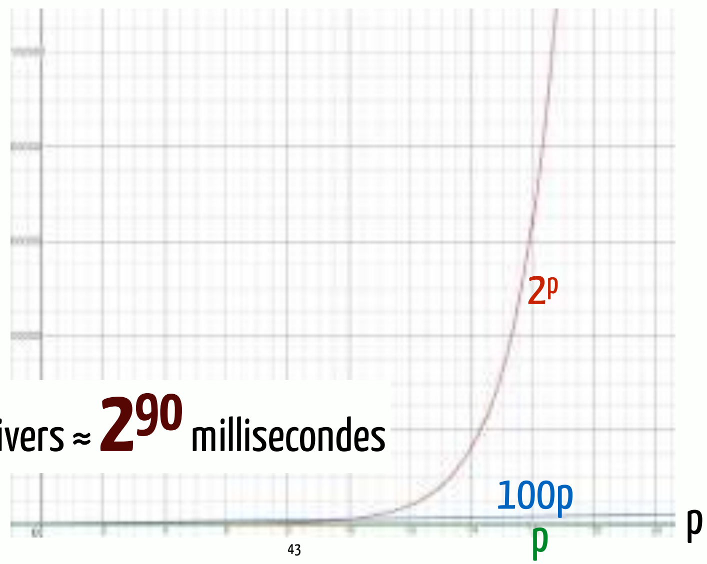
MILP with p binaries

$\min\{cx \mid Ax \geq b, x \in \{0,1\}^p \times \mathbb{R}^{n-p}\}$ = $\mathbf{2^p}$ LPs to solve

Combinatorial explosion



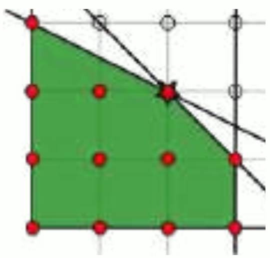
Combinatorial explosion



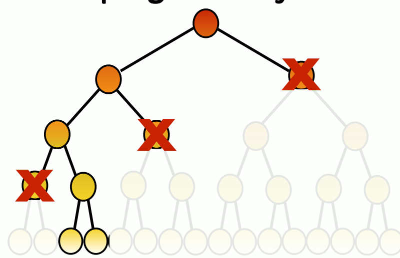
âge de l'univers ≈ $\mathbf{2^{90}}$ millisecondes

Two options

1 compute an ideal formulation



2 evaluate partial solutions progressively

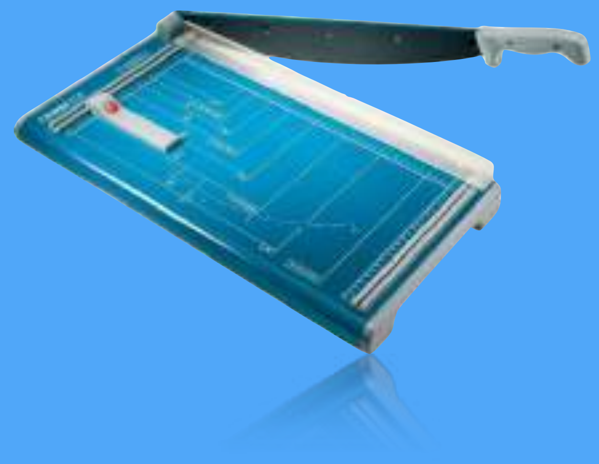


1 **Cut Generation** compute an ideal formulation

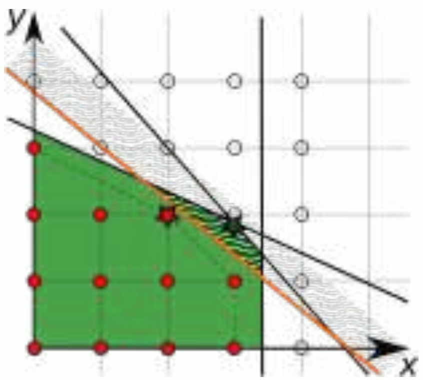
2 **Branch&Bound** evaluate partial solutions progressively

3 **modern Branch&Cut** mix up+presolve+heuristics

4 **decomposition methods** (Branch&Price, Lagrangian relaxation, Benders)



Cutting Plane Algorithm



Cut valid inequality that separates a relaxed LP solution

Farkas Lemma cuts are linear combinations of constraints

cutting plane algorithm

1. solve the LP relaxation of (P), get $\bar{\mathbf{x}}$
2. if $\bar{\mathbf{x}}$ is integral STOP: feasible then optimal for (P)
3. find cuts C for (P,$\bar{\mathbf{x}}$) from template T
4. add constraints C to (P) then 1.

separation subproblem

templates

general-purpose
ed integer rounding, split, Chvátal-Gomory

structure-based
clique, cover, flow cover, zero half

problem-specific
subtour elimination (TSP), odd-set (matc

ex 1 Chvátal-Gomory cuts

$(P) : \max\{cx \mid Ax \leq b, x \in \mathbb{Z}_+\}$

For any $u \in \mathbb{R}^m_+$ the following inequalities are valid:

1. surrogate: $\sum_j \sum_i u_i a_{ij} x_j \leq \sum_i u_i b_i$ $\quad (u \geq 0)$
2. round off: $\sum_j \lfloor \sum_i u_i a_{ij} \rfloor x_j \leq \sum_i u_i b_i$ $\quad (x \geq 0)$
3. Chvátal-Gomory: $\sum_j \lfloor \sum_i u_i a_{ij} \rfloor x_j \leq \lfloor \sum_i u_i b_i \rfloor$ $\quad (\lfloor uA \rfloor x \in \mathbb{Z})$

variants in the choice of u, ex: Gomory or MIR cuts

ex 2 Cover cuts

$$S = \{y \in \{0,1\}^7 | 11y_1 + 6y_2 + 6y_3 + 5y_4 + 5y_5 + 4y_6 + y_7 \leq 19\}$$

- (y_3, y_4, y_5, y_6) is a minimal cover for $11y_1 + 6y_2 + 6y_3 + 5y_4 + 5y_5 + 4y_6 + y_7 \leq 19$ as $6 + 5 + 5 + 4 > 19$ then $y_3 + y_4 + y_5 + y_6 \leq 3$ is a cover inequality
- we can derive a stronger valid inequality $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \leq 3$ by noting that y_1, y_2 has greater coefficients than any variable in the cover
- note furthermore that (y_1, y_i, y_j) is a cover $\forall i \neq j \in \{2,3,4,5,6\}$ then $2y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \leq 3$ is also valid

lifting

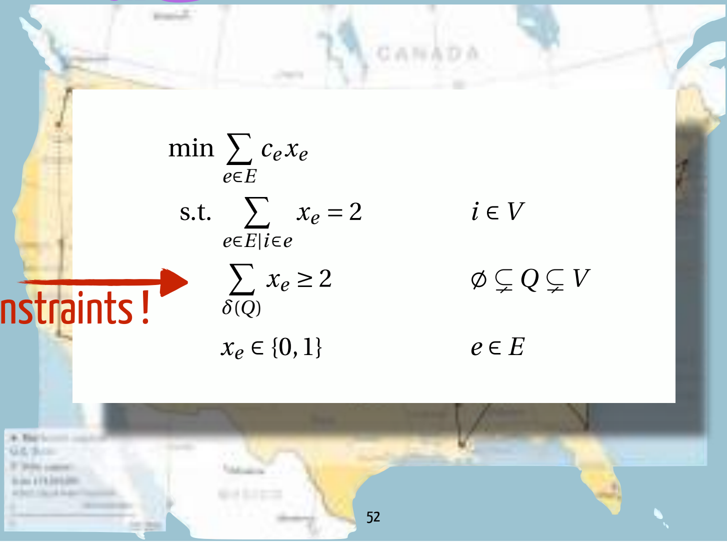
separation: solve knapsack $\min\{\sum_j (1 - \bar{y}_j) x_j \mid \sum_j a_j x_j \geq b + \epsilon, x \in \{0,1\}^n\}$

get coefficients x^* of the cover inequality $\sum x^*_j y_j \leq \sum x^*_j - 1$

if $\sum (1 - \bar{y}_j) x^*_j < 1$ then it is a cut (not satisfied by current LP solution $\bar{y}$)

ex 3 Subtour for TSP

ex 3 Subtour for TSP



$$\begin{aligned} \min \ & \sum_{e \in E} c_e x_e \\ \text{s.t.} \ & \sum_{e \in E | i \in e} x_e = 2 && i \in V \\ & \sum_{\delta(Q)} x_e \geq 2 && \emptyset \subsetneq Q \subsetneq V \\ & x_e \in \{0,1\} && e \in E \end{aligned}$$

2^n constraints !

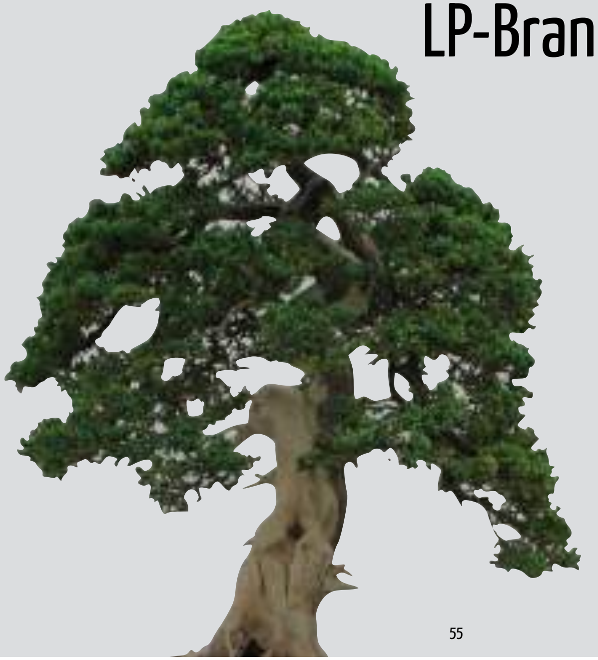
ex 3 Subtour for TSP

separation: solve min s-t cut in $(V, \overrightarrow{E}, \bar{x})$ for some fixed s and each $t \in V \backslash \{s\}$ to find a cutset $\delta(Q)$ of capacity < 2 or prove none exists

limits depending on the templates

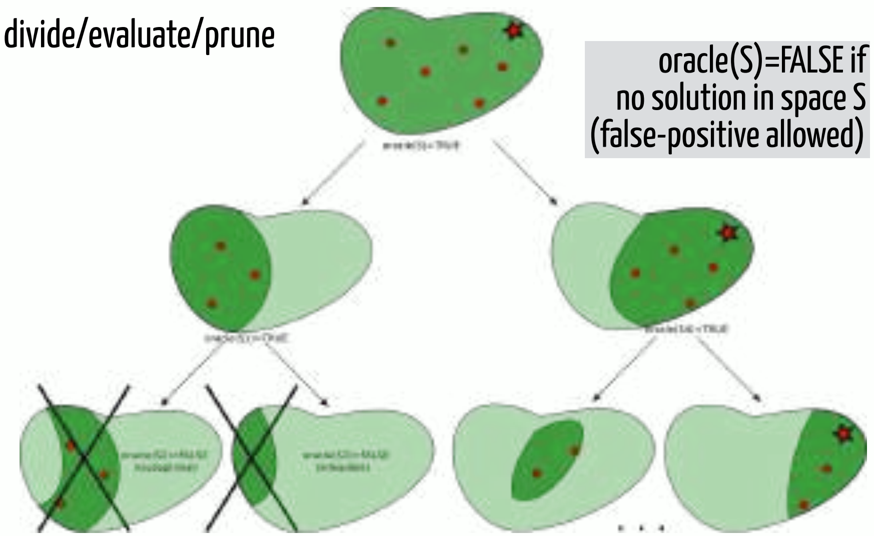
- the algorithm may stop prematurely
- the algorithm may not converge
- the algorithm may converge slowly
- the separation procedure may be NP-hard
- the LP relaxation grows
- the LP relaxation structure changes

LP-Branch and Bound



Search tree

divide/evaluate/prune



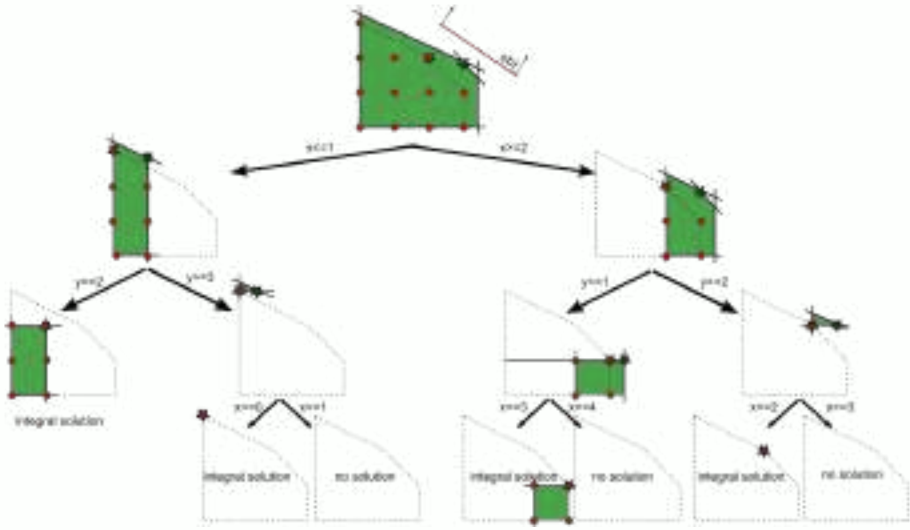
oracle(S)=FALSE if
no solution in space S
(false-positive allowed)

LP-based branch and bound

1. evaluate by solving the LP relaxation and compare bounds
2. divide with variable bounding (hyperplanes)

oracle(S) = FALSE if either:

- the LP relaxation is unfeasible on **S**
- the relaxed LP solution **x** is not better than the best integer solution found so far **x***
- **x** is integer (then update **x***)



branching

node selection

which order to visit nodes ?

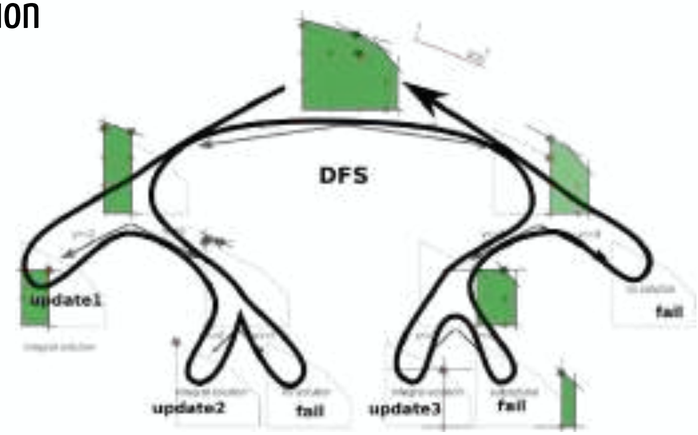
variable selection

how to separate nodes ?

constraint branching

versus variable branching

node selection

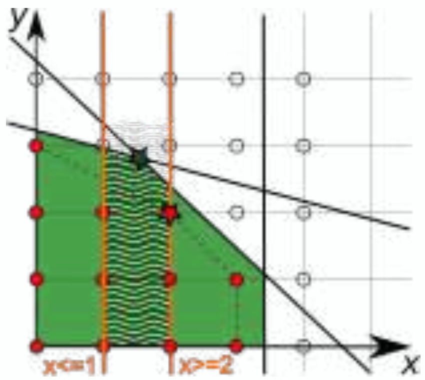


Best Bound First Search explore less nodes, manages larger trees

Depth First Search sensible to bad decisions at or near the root

DFS (up to n solutions) + **BFS** (to prove optimality)

variable selection



most fractional easy to implement but not better than random

strong branching best improvement among all candidates (impractical)

pseudocost branching record previous branching success for each var (inaccurate at root)

reliability branching pseudocosts initialised with strong branching

constraint branching

example: GUB dichotomy

- if (P) contains a GUB constraint $\sum_C x_i = 1, x \in \{0,1\}^n$
- choose $C' \subseteq C$ s.t. $0 < \sum_{C'} \bar{x}_i < 1$
- create two child nodes by setting either $\sum_{C'} x_i = 0$ or $\sum_{C'} x_i = 1$

- enforced by fixing the variable values
- leads to more balanced search trees

SOS1 branching in a facility location problem

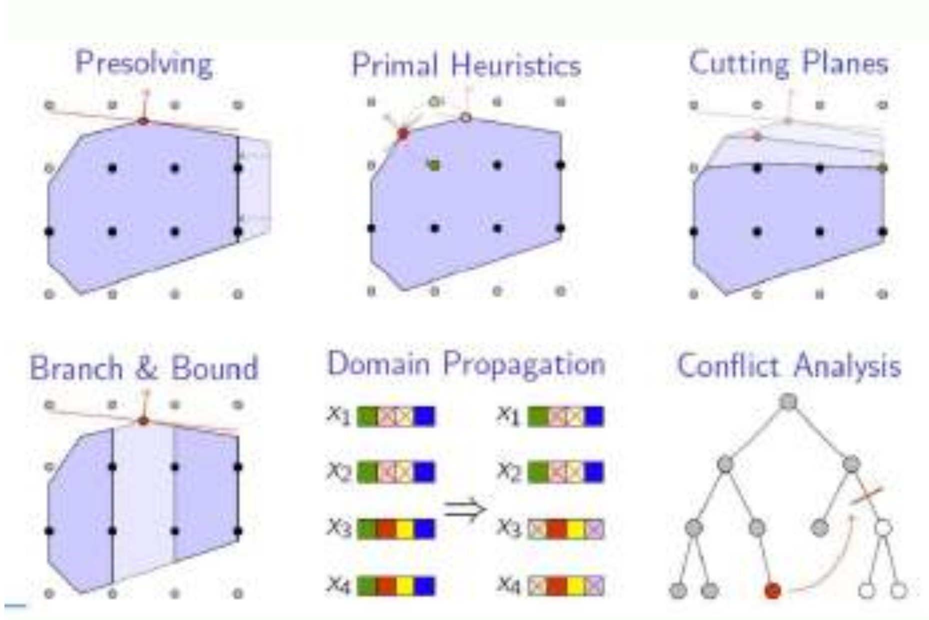
choose a warehouse depending on its size/cost:

$$\text{COST} = 100x_1 + 180x_2 + 320x_3 + 450x_4 + 600x_5$$
$$\text{SIZE} = 10x_1 + 20x_2 + 40x_3 + 60x_4 + 80x_5$$
$$\text{(SOS1)} \ : x_1 + x_2 + x_3 + x_4 + x_5 = 1$$

- let $\bar{x}_1 = 0.35$ and $\bar{x}_5 = 0.65$ in the LP solution then SIZE$= 55.5$
- choose $C' = \{1,2,3\}$ in order to model SIZE≤ 40 or SIZE≥ 60

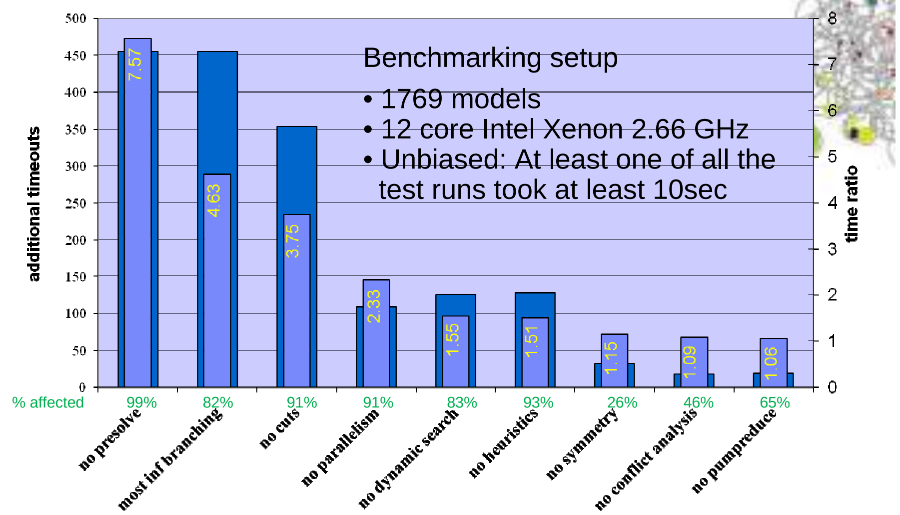
modern solvers

Simplex
var branching
Preprocessing
Branch & Cut
Heuristics
Parallelism



Slide from Martin Grötschel Co@W Berlin 2015

Component Impact CPLEX 12.5 Summary



CPLEX 12.7

- Boolean Quadric Polytope (BQP) cuts
- Clique cuts
- Cover cuts
- Disjunctive cuts
- Flow cover cuts
- Flow path cuts
- Gomory fractional cuts
- Generalized upper bound (GUB) cover cuts
- Implied bound cuts: global and local
- Lift-and-project cuts
- Mixed integer rounding (MIR) cuts
- Multi-commodity flow (MCF) cuts
- Reformulation Linearization Technique (RLT) cuts
- Zero-half cuts

GUROBI 7.5

CliqueCuts	Clique cut generation
CoverCuts	Cover cut generation
FlowCoverCuts	Flow cover cut generation
FlowPathCuts	Flow path cut generation
GUBCoverCuts	GUB cover cut generation
ImpliedCuts	Implied bound cut generation
MIPSepCuts	MIP separation cut generation
MIRCuts	MIR cut generation
StrongCGCuts	Strong-CG cut generation
ModKCuts	Mod-k cut generation
NetworkCuts	Network cut generation
ProjImpliedCuts	Projected implied bound cut generation
SubMIPCuts	Sub-MIP cut generation
ZeroHalfCuts	Zero-half cut generation
InfProofCuts	Infeasibility proof cut generation

Preprocessing

reduce size

remove redundancies $x+y\leq 3$, *binaries*

substitute variables $x+y-z=0$

fix variables by duality $c_j\geq 0,\ A_j\geq 0 \Rightarrow x=x_{min}$

fix variables by probing $x=1$ *infeas* $\Rightarrow x=0$

strengthen LP relaxation

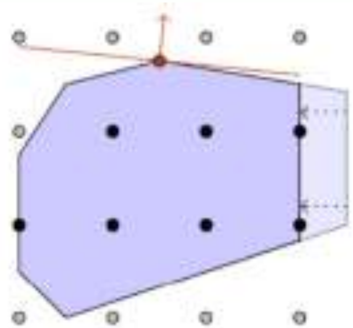
adjust bounds $2x+y\leq 1$, *binaries* $\Rightarrow x=0$

lift coefficients $2x-y\leq 1$, *binaries* $\Rightarrow x-y\leq 1$

identify/exploit properties

detect implied integer $3x+y=7$, x *int* $\Rightarrow$;

build the conflict graph
detect disconnected components
remove symmetries

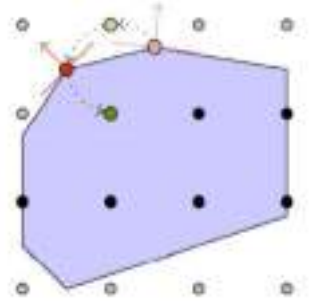


MIPLIB markshare_5_0

```
[...]$ gurobi.sh mymip.py markshare_5_0.mps.gz
Changed value of parameter Presolve to 0
   Prev: -1  Min: -1  Max: 2  Default: -1
Optimize a model with 5 rows, 45 columns and 203 nonzeros
Found heuristic solution: objective 5335
Variable types: 5 continuous, 40 integer (40 binary)

Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000  100%     -    0s

*62706364 28044              38       1.0000000    0.00000  100%   2.1 1241s

Explored 233048403 nodes (460515064 simplex iterations) in 3883.5 seconds
Thread count was 4 (of 4 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+00, best bound 1.000000000000e+00, gap 0.0%
Optimal objective: 1
```



```
[sofdem:~/Documents/Code/gurobi]$ gurobi.sh mymip.py markshare_5_0.mps.gz
Optimize a model with 5 rows, 45 columns and 203 nonzeros
Found heuristic solution: objective 5335
Presolve time: 0.00s
Presolved: 5 rows, 45 columns, 203 nonzeros
Variable types: 0 continuous, 45 integer (40 binary)

Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000  100%     -    0s
H    0     0                    320.0000000    0.00000  100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000  100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000  100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000  100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000  100%     -    0s
H    0     0                    239.0000000    0.00000  100%     -    0s
     0     0    0.00000    0    5  239.00000    0.00000  100%     -    0s
*   30     0             29      96.0000000    0.00000  100%   2.7    0s
*   99    32             34      58.0000000    0.00000  100%   2.1    0s
H  506   214                     53.0000000    0.00000  100%   1.9    0s
H30682   442                      1.0000000    1.00000  0.00%   2.1    0s

Cutting planes:
  Cover: 26

Explored 30682 nodes (65348 simplex iterations) in 0.70 seconds
Thread count was 4 (of 4 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+00, best bound 1.000000000000e+00, gap 0.0%
Optimal objective: 1
```

rounding LP solution
diving at some nodes
local search in the incumbent neighbourhood

Primal Heuristics

accelerate the search a little
appeal to the practitioner a lot

limits of branch&cut

- highly heuristic (branching decisions, cut generation)
- floating-point errors and optimality tolerance (0.01%)
- generic features
- less effective on general integers (ex: scheduling)
- hard to model (and solve) non-linear structures
- NP-hard

how to tune modern solvers

play with Gurobi


```
Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000   100%     -    0s
H    0     0                     320.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
H    0     0                     239.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    5  239.00000    0.00000   100%     -    0s
*   36     0              29      96.0000000    0.00000   100%   2.7    0s
*   99    32              34      58.0000000    0.00000   100%   2.1    0s
H  506   214                      53.0000000    0.00000   100%   1.9    0s
H30682   442                       1.0000000    1.00000  0.00%   2.1    0s
```

use as a heuristic

set a time limit

MIPFocus=1

ImproveStartGap=0.1


```
Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000   100%     -    0s
H    0     0                     320.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
H    0     0                     239.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    5  239.00000    0.00000   100%     -    0s
*   36     0              29      96.0000000    0.00000   100%   2.7    0s
*   99    32              34      58.0000000    0.00000   100%   2.1    0s
H  506   214                      53.0000000    0.00000   100%   1.9    0s
H30682   442                       1.0000000    1.00000  0.00%   2.1    0s
```

change the LP solver

if nbIteration(node) ≥ nbIteration(root)/2

NodeMethod=2

```
Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000   100%     -    0s
H    0     0                    320.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
H    0     0                    239.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    5  239.00000    0.00000   100%     -    0s
*   36     0              29     96.0000000    0.00000   100%   2.7    0s
*   99    32              34     58.0000000    0.00000   100%   2.1    0s
H  506   214                     53.0000000    0.00000   100%   1.9    0s
H30682   442                      1.0000000    1.00000  0.00%   2.1    0s
```

init with a feasible solution

if built-in heuristics fail

```
PumpPasses,MinRelNodes,ZeroObjNodes
model.read('initSol.mst')
model.cbSetSolution(vars, newSol)
```



```
Root relaxation: objective 0.000000e+00, 15 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0    5 5335.00000    0.00000   100%     -    0s
H    0     0                    320.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    6  320.00000    0.00000   100%     -    0s
     0     0    0.00000    0    5  320.00000    0.00000   100%     -    0s
H    0     0                    239.0000000    0.00000   100%     -    0s
     0     0    0.00000    0    5  239.00000    0.00000   100%     -    0s
*   36     0              29     96.0000000    0.00000   100%   2.7    0s
*   99    32              34     58.0000000    0.00000   100%   2.1    0s
H  506   214                     53.0000000    0.00000   100%   1.9    0s
H30682   442                      1.0000000    1.00000  0.00%   2.1    0s
```


tighten the model

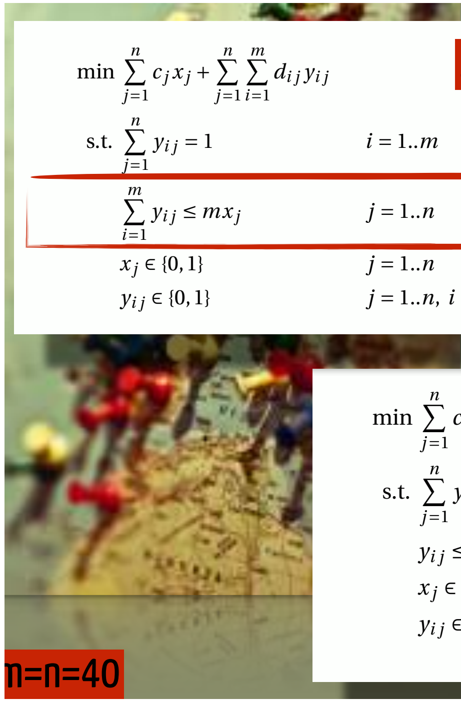
if the bound stagnates

```
Cuts=3
Presolve=3
model.cbCut(lhs, sense, rhs)
```


http://www.gurobi.com/

you know your problem better than your solver does

improve the model



$$\min \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij} y_{ij}$$

14 hours

$$\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{n} y_{ij} = 1 && i = 1..m \\
& \boxed{\sum_{i=1}^{m} y_{ij} \le m x_j} && j = 1..n \\
& x_j \in \{0,1\} && j = 1..n \\
& y_{ij} \in \{0,1\} && j = 1..n,\ i = 1..m
\end{aligned}$$

apacitated ty Location roblem

Input n facility locations, m customers

$$\min \sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij} y_{ij}$$

2 seconds

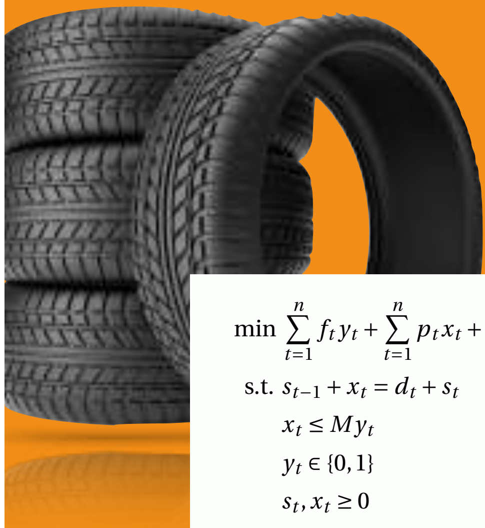
$$\begin{aligned}
\text{s.t.} \quad & \sum_{j=1}^{n} y_{ij} = 1 && i = 1..m \\
& y_{ij} \le x_j && j = 1..n,\ i = 1..m \\
& x_j \in \{0,1\} && j = 1..n \\
& y_{ij} \in \{0,1\} && j = 1..n,\ i = 1..m
\end{aligned}$$

n=n=40

Uncapacitated Lot Sizing Problem

Input n time periods, fixed production cost f_t, unit production cost p_t, unit storage cost h_t, demand d_t for each period t

Output a mimimum (production and storage) cost production plan to satisfy the demand



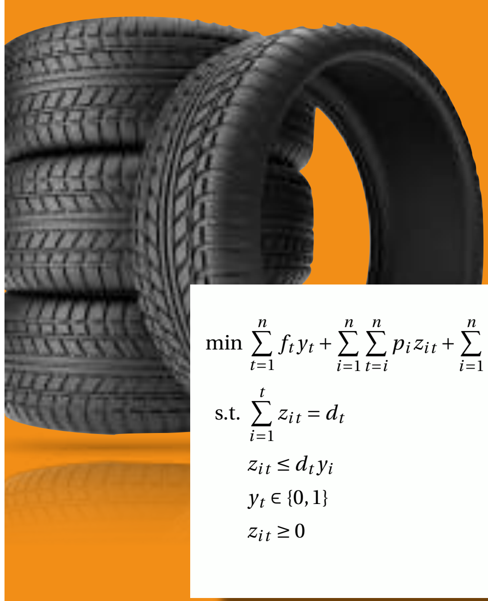
Uncapacitated Lot Sizing Problem

$$\min \sum_{t=1}^{n} f_t y_t + \sum_{t=1}^{n} p_t x_t + \sum_{t=1}^{n} h_t s_t$$

$$\begin{aligned}
\text{s.t.} \quad & s_{t-1} + x_t = d_t + s_t && t = 1..n \\
& x_t \le M y_t && t = 1..n \\
& y_t \in \{0,1\} && t = 1..n \\
& s_t, x_t \ge 0 && t = 1,\dots,n \\
& s_0 = 0
\end{aligned}$$

oduction ... , unit ... r each ... ion and ... to satisfy

z_{it} production in period i to satisfy demand of period t



Uncapacitated Lot Sizing Problem

$$\min \sum_{t=1}^{n} f_t y_t + \sum_{i=1}^{n}\sum_{t=i}^{n} p_i z_{it} + \sum_{i=1}^{n}\sum_{t=i+1}^{n}\sum_{j=i}^{t-1} h_j z_{it}$$

$$\begin{array}{lll} \text{s.t.} & \sum_{i=1}^{t} z_{it} = d_t & t = 1..n \\ & z_{it} \le d_t y_i & i = 1..n; t = i..n \\ & y_t \in \{0,1\} & t = 1..n \\ & z_{it} \ge 0 & i = 1..n; t = i..n \end{array}$$

LP=ILP

roduction
p_t, unit
r each

ion and
n to satisfy

z_{it} production in period i to satisfy demand of period t

Bin Packing Problem

Input n containers, m items, capacity c for all containers, weight w_j for each item j
Output a packing of all items in a mimimum number of containers

Bin Packing Problem

$$\min \sum_{i=1}^{n} y_i$$

$$\begin{array}{lll} \text{s.t.} & \sum_{j=1}^{m} w_j x_{ij} \le c y_i & i = 1..n \\ & \sum_{i=1}^{n} x_{ij} = 1 & j = 1..m \\ & x_{ij} \in \{0,1\} & i = 1..n;\ j = 1..m \\ & y_i \in \{0,1\} & i = 1..n \end{array}$$

ns, capacity c
w_j for each

ms in a
ainers

$\mathcal{S}$ all the possible arrangements of items in a bin

Bin Packing Problem

$$\min \sum_{s \in \mathcal{S}} x_s$$

$$\begin{array}{lll} \text{s.t.} & \sum_{s \in \mathcal{S}} a_{js} x_s = 1 & j = 1..n \\ & x_s \in \{0,1\} & s \in \mathcal{S} \end{array}$$

Dantzig-Wolfe decomposition

ns, capacity c
w_j for each

ms in a
ainers

$\mathcal{S}$ all the possible arrangements of items in a bin

how to manage the exponential number of variables ?

Bin Packing Problem

$$\min \sum_{s \in \mathcal{S}} x_s$$

$$\text{s.t.} \sum_{s \in \mathcal{S}} a_{js} x_s = 1 \qquad j = 1..n$$

$$x_s \in \{0,1\} \qquad s \in \mathcal{S}$$

ns, capacity c
w_j for each
ms in a
ainers

Dantzig-Wolfe decomposition

$\mathcal{S}$ all the possible arrangements of items in a bin

delayed column generation

$min\{c_B x_B + c_N x_N \,|\, A_B x_B + A_N x_N = b\}$ without (c_N, A_N) i.e. $x_N = 0$:

1/ solve the restricted LP with the primal simplex algorithm where the omitted columns N are implicitly non-basic
2/ find $j \in N$ that can profitably enter the basis $\bar{c}_j < 0$, stop if none

= dual cut generation: (cut separation = pricing problem)

$\min cx$			$\max ub$		
$A_i x \geq b_i,$	$\forall i$		$uA_j \leq c_j,$	$\forall j$	
$x_j \geq 0,$	$\forall j$		$u_i \geq 0,$	$\forall i$	

given a basic dual solution u find j such that $\bar{c}_j = c_j - uA_j < 0$

application to Bin Packing

$\mathcal{S} \subseteq 2^m$ all the possible arrangements of items in a bin
S a feasible subset (i.e. covering all the items)

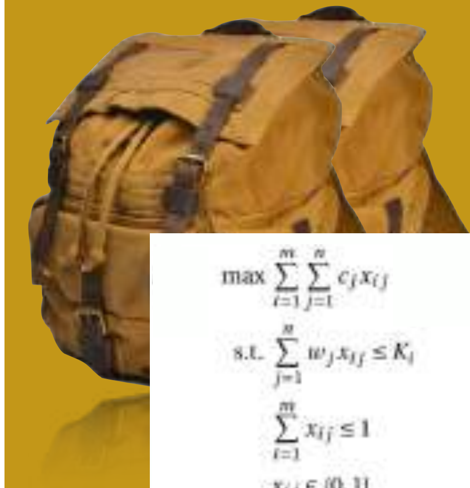
1. **solve the restricted LP:**
   $\min\{\sum_{s \in S} x_s \,|\, \sum_{s \in S} a_{js} x_s = 1 \ \forall j,\ x_s \geq 0 \ \forall s \in S\}$
   get the corresponding dual solution $\bar{u} \in \mathbb{R}^m$
2. **look for an improving basic direction** **= some**
   $s \in \mathcal{S} \backslash S$ **with** $\bar{c}_s = 1 - \sum_j a_{js}\bar{u}_j < 0$ **e.g. by solving**
   $\max\{\sum_j a_j \bar{u}_j \,|\, \sum_j w_j a_j \leq K, a \in \{0,1\}^m\}$
3. **if** $\sum_j a_j^* \bar{u}_j > 1$ **add column** $(1, a^*)$ **to** S **then** 1 **otherwise**
   STOP: $(\bar{x}_S, 0)$ **solves the LP-relaxation**

Branch-and-Price

- branch-and-bound for ILP with large number of variables where the LP relaxation is solved by column generation
- the branching strategy should keep the search tree balanced without altering the LP relaxation structure, ex (bin packing): branch by fixing to 0 either all $x_s \,|\, \{i,j\} \subseteq s$ or all $x_s \,|\, \{i,j\} \not\subseteq s$ for some pair of items (i,j) s.t. $0 < \sum_s a_{is} a_{js} x_s < 1$
- the pricing problem can be seen as an optimization problem but does not need to be solved at optimality, except for the convergence proof.
- convenient decomposition method when additional constraints only appear in the pricing problem, ex (conflicts in bin packing): $\sum_{j \in C} a_j \leq 1$

Multi 0-1 Knapsack Problem

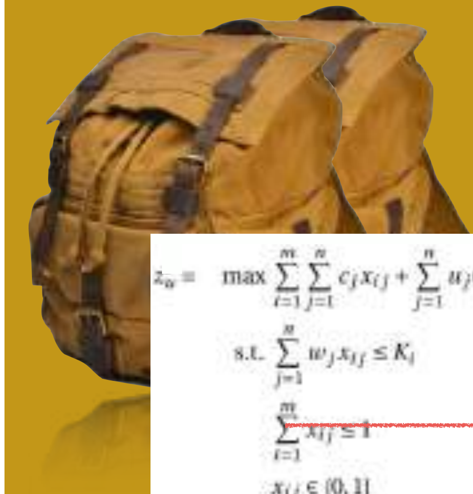
Input n items, m bins, value c_j and weight w_j for each item j, capacity K_i for each bin i.
Output a maximum value subset of items packed in the bins.



Multi 0-1 Knapsack Problem

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} c_j x_{ij}$$
$$\text{s.t. } \sum_{j=1}^{n} w_j x_{ij} \le K_i \qquad i=1..m$$
$$\sum_{i=1}^{m} x_{ij} \le 1 \qquad j=1..n$$
$$x_{ij} \in \{0,1\} \qquad j=1..n, i=1..m$$

and
city K_i for
set of



Multi 0-1 Knapsack Problem

$$z_u = \max \sum_{i=1}^{m}\sum_{j=1}^{n} c_j x_{ij} + \sum_{j=1}^{n} u_j\left(1 - \sum_{i=1}^{m} x_{ij}\right) \qquad u \in \mathbb{R}^n_+$$
$$\text{s.t. } \sum_{j=1}^{n} w_j x_{ij} \le K_i \qquad i=1..m$$
$$\cancel{\sum_{i=1}^{m} x_{ij} \le 1 \qquad j=1..n}$$
$$x_{ij} \in \{0,1\} \qquad j=1..n, i=1..m$$

and
city K_i for
set of

find the smallest upper bound **lagrangian relaxation**

Lagrangian Relaxation

dualize the complicating or coupling constraints of an ILP:

$$(P): z = \max \sum_k c_k x_k$$
$$\sum_k D_k x_k \le e_k$$
$$A_k x_k \le b_k, \qquad \forall k$$
$$x_k \in \mathbb{Z}^p \times \mathbb{R}^n, \qquad \forall k$$

$$(D): w = \min_{u \ge 0} l(u)$$
$$l(u) = ue + \sum_k z_k^u$$
$$(P_u): z_u^k = \max c_k x_k - u D_k x_k$$
$$A_k x_k \le b_k$$
$$x_k \in \mathbb{Z}^p \times \mathbb{R}^n$$

(D) is the lagrangian dual problem
(P_u) is the lagrangian suproblem with multipliers u

Lagrangian Relaxation

dualize the complicating or coupling constraints of an ILP:

$$(P): z = \max \sum_k c_k x_k \qquad \sum_k D_k x_k \le e_k \qquad A_k x_k \le b_k, \ \forall k \qquad x_k \in \mathbb{Z}^p \times \mathbb{R}^n, \ \forall k$$

$$(D): w = \min_{u \ge 0} l(u) \qquad l(u) = ue + \sum_k z_k^u$$

$$(P_u): z_u^k = \max c_k x_k - u D_k x_k \qquad A_k x_k \le b_k \qquad x_k \in \mathbb{Z}^p \times \mathbb{R}^n$$

(D) is the lagrangian dual problem

(P_u) is the lagrangian suproblem with multipliers u

strong duality may not hold if p>0, ie the dual only provides an upper bound

$$w \ge z$$

solving the lagrangian dual

$$(P): z = \max \sum_k c_k x_k \qquad \sum_k D_k x_k \le e_k \qquad A_k x_k \le b_k, \ \forall k \qquad x_k \in \mathbb{Z}^p \times \mathbb{R}^n, \ \forall k$$

$$(D): w = \min_{u \ge 0} l(u) \qquad l(u) = ue + \sum_k z_k^u$$

$$(P_k^u): z_k^u = \max c_k x_k - u D_k x_k \qquad A_k x_k \le b_k \qquad x_k \in \mathbb{Z}^p \times \mathbb{R}^n$$

- function l is convex and a subgradient at $u \ge 0$ is $e - \sum D_k x_k^u$ where x_k^u an optimal solution of (P_k^u)
- minimize l with a subgradient, bundle, or cutting-plane method
- almost feasible solutions computed at each iteration: repair violations heuristically to get feasible solutions and lower bounds

performance
sophisticated algorithms

declarative
models, not algorithms

large-scale
decomposition methods

MILP perks

certification
primal-dual bounds

versatile
covers many problems

flexible
general-purpose solvers

logic & constraint programming

integer nonlinear programming

graph algorithms

combinatorial optimization beyond MILP

machine learning

dynamic programming

metaheuristics

Matteo Fischetti (2019) Introduction to Mathematical Optimization
Wolsey L. (1998) Integer Programming. Wiley
Bertsimas D., Tsitsiklis J. (1997) Introduction to Linear Optimization. Athena Scientific

Achterberg T., Berthold T., Hendel G. (**2012**) Rounding and propagation **heuristics** for MIP. In OR Proceedings 2011, 71-76.
Achterberg T., Bixby R., Gu Z., Rothberg E., Weninger D. (**2016**) **Presolve** reductions in MIP. ZIB-Report 16-44.
Bixby R. (**2012**) A brief **history** of LP and MIP computation. Documenta Mathematica, 107-121.
Cornuéjols G. (**2008**) **Valid inequalities** for MILP. Mathematical Programming, 112(1), 3-44.
Klotz E., Newman A. (**2013**) **Practical guidelines** for solving difficult MILP. Surveys in ORMS, 18(1), 18-32.
Linderoth J., Ralphs T. (**2005**) Noncommercial **software** for MILP. IP: theory and practice, 3, 253-303.
Linderoth J., Lodi A. (**2010**) MILP **software**. Wiley encyclopedia of ORMS.
Linderoth J., Savelsbergh M. (**1999**) A computational study of **search strategies** for MIP. INFORMS JoC, 11(2), 173-187.
Lodi A. (**2010**) MIP **computation**. In 50 Years of IP 1958-2008, 619-645.
Newman A., Weiss M. (**2013**) A survey of linear and mixed-integer optimization **tutorials**. INFORMS ToE, 14(1) 26-38.
Savelsbergh M. (**1994**) **Preprocessing** and probing techniques for MIP. ORSA Journal on Computing, 6(4), 445-454.
Vanderbeck F., Wolsey L. (**2010**) Reformulation and **decomposition** of IP. In 50 Years of IP 1958-2008, 431-502.