Using MW for Mixed-Integer Nonlinear Problems

G. Nannicini[1], P. Belotti[2], J. Lee[3],
J. Linderoth[4], F. Margot[5], A. Wächter[6]

[1] Singapore University of Technology and Design, Singapore and
Sloan School of Management, MIT, Cambridge, MA
[2] Dept. of Mathematical Sciences, Clemson University, Clemson, SC
[3] Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI
[4] Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI
[5] Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA
[6] Industrial Engineering and Management Sciences, Northwestern University, Chicago, IL

May 2, 2012

Summary of Talk

1. Introduction
2. Parallel Branch-and-Bound solver: Coupe
3. Computational experiments

1. Introduction

2. Parallel Branch-and-Bound solver: Couenne

3. Computational experiments

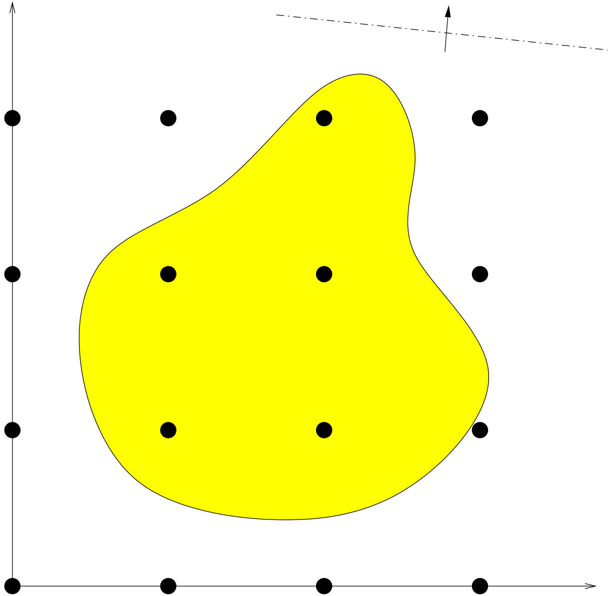
Nonconvex MINLP

- Consider a mathematical program of this form:

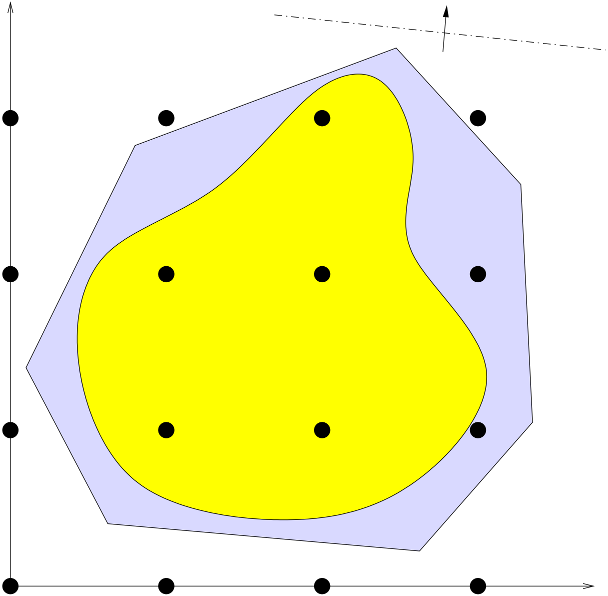
$$\left.\begin{array}{lrl} \min & f(x) & \\ s.t. & g_j(x) \leq 0 & \forall j \in M \\ & x_i^L \leq x_i \leq x_i^U & \forall i \in N \\ & x_i \in \mathbb{Z} & \forall i \in N_I, \end{array}\right\} \qquad \mathcal{P}$$

  with $N = \{1, \ldots, n\}$, $M = \{1, \ldots, m\}$, $x^L \in (\mathbb{R} \cup \{-\infty\})^n$, $x^U \in (\mathbb{R} \cup \{+\infty\})^n$
- The functions f, g_j's need not be convex: nonconvex MINLP
- Very expressive class of mathematical programs, but difficult to solve
- Applications *everywhere*

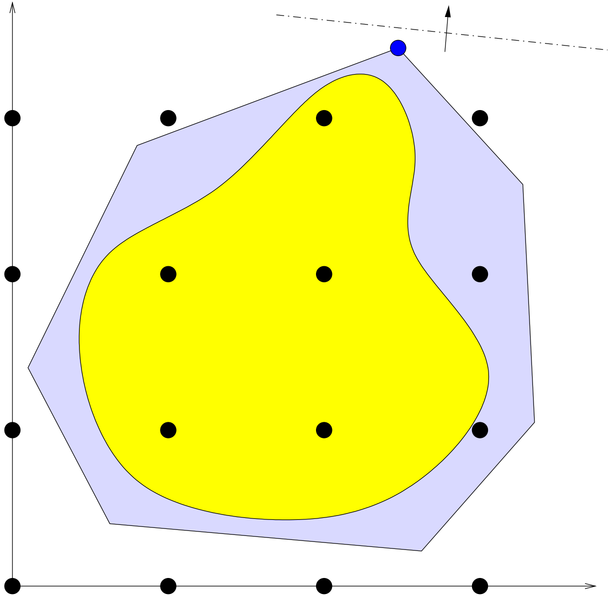
LP-based Branch-and-Bound



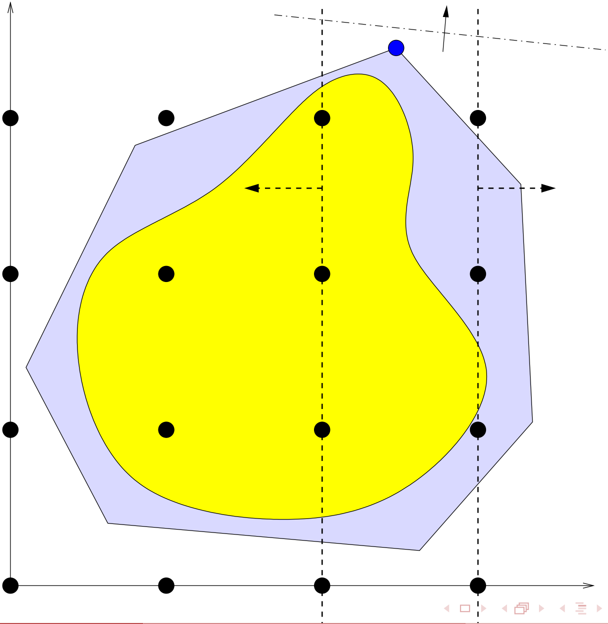
LP-based Branch-and-Bound



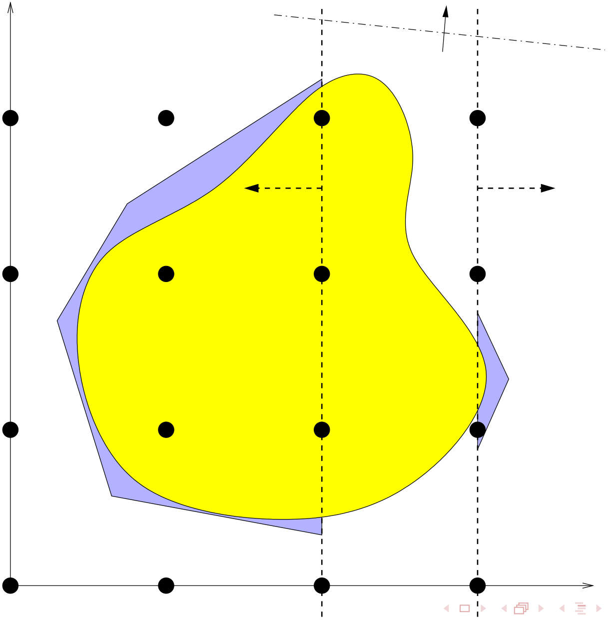
LP-based Branch-and-Bound



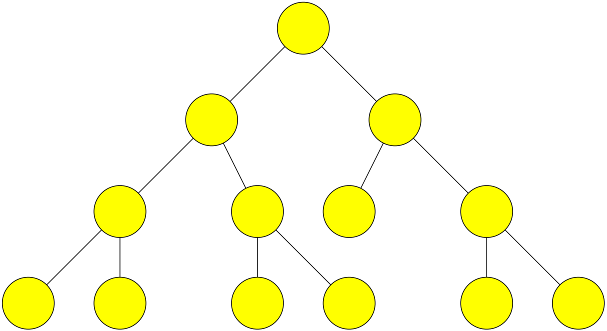
LP-based Branch-and-Bound



LP-based Branch-and-Bound



LP-based Branch-and-Bound



1 Introduction

2 Parallel Branch-and-Bound solver: Coupe

3 Computational experiments

Motivation

- Many problems cannot be solved with current technology
- A general-purpose brute-force solver can be used to certify optimality of solutions and facilitate comparisons
- Software is available, such as `Couenne`: an open-source solver for nonconvex MINLPs
- `Coupe` (`COUenne Parallel Extension`): a solver that runs on Condor and uses COIN-OR `Couenne` as main Branch-and-Bound code (for convexification, heuristics, etc.)

Issues when implementing on Condor

- Each machine could disappear at any moment: cannot rely on completing a specific computation in a timely fashion!
- No shared memory
- Slow communication (TCP/IP)

MW: Master/Worker

- Master/worker paradigm: one machine “knows” everything and dispatches tasks to the workers, then puts together the results
- The master should do as little work as possible (besides managing the workers)
- We should minimize the number of messages exchanged between the master and the workers: a worker should be able to work on its own for a few minutes
- Cannot expect workers to complete their tasks in a specific order
- Implemented through the `MW` library: deals with managing the machines, communicating results

Structure

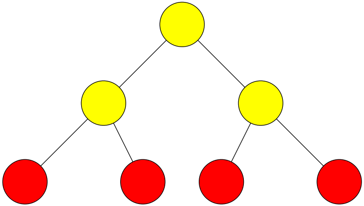
- The master reads the problem, computes the convexification, and sets up tasks for the workers
- Tasks:
  - Branch-and-Bound
  - Bound tightening
  - Heuristics
- All these things can be done in any order, and the master takes care of putting together the results
- The master decides the number of workers, overall strategy, deals with ramp-up and ramp-down, ...
- Suitable for problems with easy LP but huge enumeration tree

Branch-and-Bound and tree search strategy

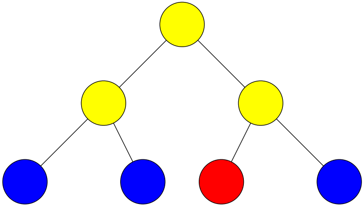
- Branch-and-Bound task: the worker receives a node, performs Branch-and-Bound for some time, sends back all remaining active nodes
- In other words, each worker explores a sub-tree of the full Branch-and-Bound tree
- If idle workers: best bound search at the workers, short time limit (ramp-up)
- If all workers have tasks: depth-first search at the workers with long time limit, while the master still dispatches Branch-and-Bound tasks in a best bound fashion
- If master out of memory: depth-first search at the workers and master

Tree search strategy

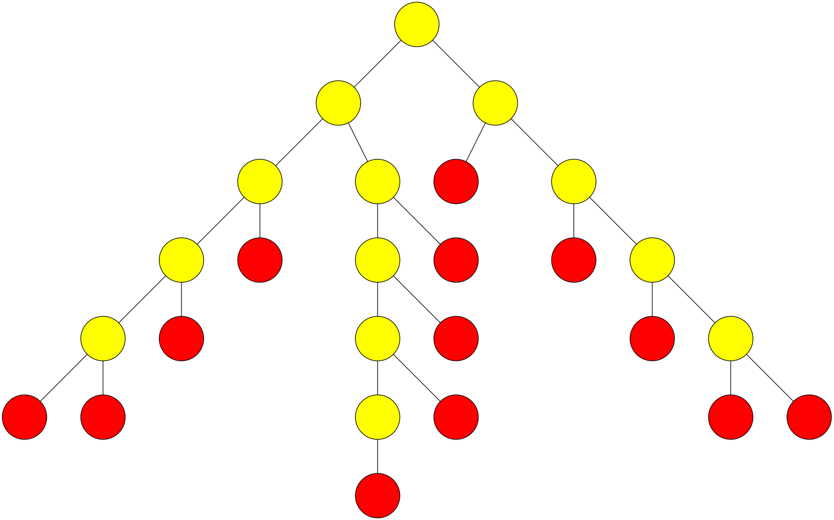
Tree search strategy



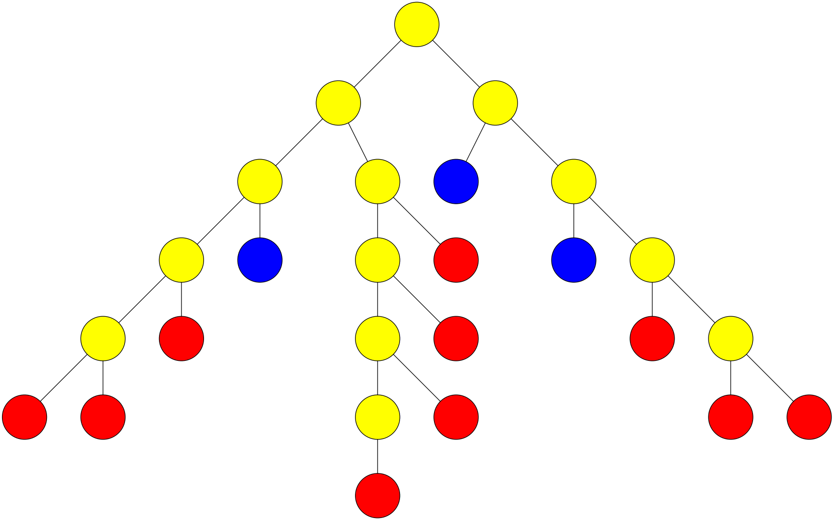
Tree search strategy



Tree search strategy



Tree search strategy



What if something goes wrong?

- The Branch-and-Bound library (`Couenne` and the underlying components: COIN-OR `Cbc` and `Clp`) sometimes incurs into problems
- It can happen, although rarely, that the solution process of one of the LPs cycles indefinitely

What if something goes wrong?

- The Branch-and-Bound library (`Couenne` and the underlying components: COIN-OR `Cbc` and `Clp`) sometimes incurs into problems
- It can happen, although rarely, that the solution process of one of the LPs cycles indefinitely
- Very rare event $\times$ 1 trillion trials $=$ sometimes it happens

What if something goes wrong?

- The Branch-and-Bound library (`Couenne` and the underlying components: COIN-OR `Cbc` and `Clp`) sometimes incurs into problems
- It can happen, although rarely, that the solution process of one of the LPs cycles indefinitely
- Very rare event $\times$ 1 trillion trials $=$ sometimes it happens
- Timeout mechanism:
  - Periodically check for machines that did not report back after the allotted time
  - Force-remove them
  - Reassign tasks

It sounds crazy, but. . .

- Having a huge availability of CPU power opens up new possibilities:
  - New branching schemes!
  - New bound tightening algorithms!
  - New heuristics!
  - . . .

It sounds crazy, but. . .

- Having a huge availability of CPU power opens up new possibilities:
  - New branching schemes!
  - New bound tightening algorithms!
  - New heuristics!
  - . . .
- . . . but so far we have only implemented a new bound tightening algorithm:
  - Use truncated Branch-and-Bound searches to eliminate small parts of the feasible space
  - Adaptive selection of the size of the eliminated parts
  - Very time consuming, but stronger than existing techniques
  - We call this new algorithm AGGRESSIVE PROBING

1 Introduction

2 Parallel Branch-and-Bound solver: Coupe

3 Computational experiments

Testing the parallel solver Coupe

- Setup:
  - Perform traditional bound tightening at the root
  - Apply Aggressive Probing at the root with a time limit of 3 minutes per variable bound, then switch to Branch-and-Bound
  - Periodically perform heuristics
  - Remaining tasks are Branch-and-Bound
- We solved two instances in the benchmark set MINLPLib for the first time: `space25a` and `waterx`

Testing the parallel solver Coupe

- `space25a`:
  - with Aggressive Probing: $3.6 \cdot 10^8$ nodes, 153 days of computing time, wall clock time 16 hours (298 average present workers, 75% utilization)
  - without Aggressive Probing: $9.5 \cdot 10^8$ nodes, 543 days of computing time, wall clock time 135 hours (133 average present workers, 70% utilization)
- `waterx`:
  - with Aggressive Probing: $2.0 \cdot 10^8$ nodes, 211 days of computing time, wall clock time 41 hours (199 average present workers, 60% utilization)
  - without Aggressive Probing: $2.6 \cdot 10^8$ nodes, 288 days of computing time, wall clock time 43 hours (227 average present workers, 69% utilization)

Conclusions

- Parallel solver that runs in an opportunistic environment and allows for a fast exploration of huge enumeration trees
- Simple but effective bound tightening algorithm that can be very time-consuming, suitable for parallel computing
- Found global optima for two instances for the first time