Large-scale parallelisation for the Benders' decomposition framework in SCIP

Stephen J. Maher

University of Exeter,

@sj_maher

s.j.maher@exeter.ac.uk

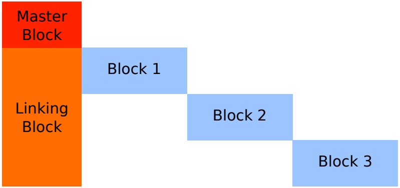
October 1, 2021

Structured mixed integer programming

Basic idea: Minimise a linear objective function over a set of solutions satisfying a structured set of linear constraints.

$$
\begin{aligned}
\min \quad & c^\top x + d^\top y, \\
\text{subject to} \quad & Ax \geq b, \\
& Bx + Dy \geq g, \\
& x \geq 0, \\
& y \geq 0, \\
& x \in \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}, \\
& y \in \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}.
\end{aligned}
$$

Decomposition methods for mixed integer programming



Linking variables

- **Variable decomposition**
  - Existence of a set of linking variables
  - Exploits property of restriction, i.e. blocks are "easy" to solve after fixing variables
  - **Parallelisation:** each block can be solved in parallel.

Benders' decomposition

Original problem

$$
\begin{aligned}
\min \quad & c^\top x + d^\top y, \\
\text{subject to} \quad & Ax \geq b, \\
& Bx + Dy \geq g, \\
& x \geq 0, \\
& y \geq 0, \\
& x \in \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}, \\
& y \in \mathbb{R}^{n_2}.
\end{aligned}
$$

Benders' decomposition

$$
\begin{aligned}
\min \quad & c^\top x + f(x), \\
\text{subject to} \quad & Ax \geq b, \\
& x \geq 0, \\
& x \in \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}.
\end{aligned}
$$

where

$$
f(x) = \min_{y \geq 0} \{ d^\top y \mid Bx + Dy \geq g, \; y \in \mathbb{R}^{n_2} \}
$$

Benders' decomposition

$$\begin{aligned} \min \quad & c^\top x + f(x), \\ \text{subject to} \quad & Ax \geq b, \\ & x \geq 0, \\ & x \in \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}. \end{aligned}$$

where

$$f(x) = \min_{y \geq 0} \{ d^\top y \mid Bx + Dy \geq g,\ y \in \mathbb{R}^{n_2} \}$$

equivalently, using the dual formulation we can define

$$f'(x) = \max_{u \geq 0} \{ u^\top (g - Bx) \mid D^\top u \geq d^\top,\ u \in \mathbb{R}^{m_2} \}$$

$$(f'(x) = f(x))$$

Benders' decomposition

Using the dual formulation of $f(x)$, given by

$$f'(x) = \max_{u \geq 0} \{u^\top (g - Bx) \mid D^\top u \geq d^\top,\ u \in \mathbb{R}^{m_2}\}$$

let

- $\mathcal{O}$ be the set of all extreme points of $f'(x)$
- $\mathcal{F}$ be the set of all extreme rays of $f'(x)$

an equivalent formulation of the original problem is

$$\begin{aligned} \min \quad & c^\top x + \varphi, \\ \text{subject to} \quad & Ax \geq b, \\ & \varphi \geq u^\top (g - Bx) \quad \forall u \in \mathcal{O} \\ & 0 \geq u^\top (g - Bx) \quad \forall u \in \mathcal{F} \\ & x \geq 0, \\ & x \in \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}. \end{aligned}$$

Benders' decomposition

- The sets $\mathcal{O}$ and $\mathcal{F}$ are exponential in size
- The reformulated original problem becomes intractable

Benders' decomposition

- The sets $\mathcal{O}$ and $\mathcal{F}$ are exponential in size
- The reformulated original problem becomes intractable
- **Need to use a delayed constraint generation algorithm**

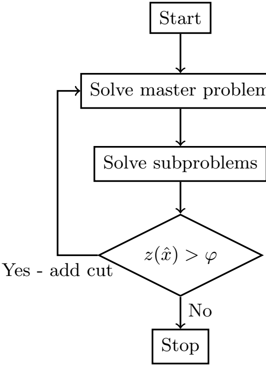
Benders' decomposition

- The sets $\mathcal{O}$ and $\mathcal{F}$ are exponential in size
- The reformulated original problem becomes intractable
- **Need to use a delayed constraint generation algorithm**

Cut generating LP $\Leftrightarrow$ Benders' subproblem

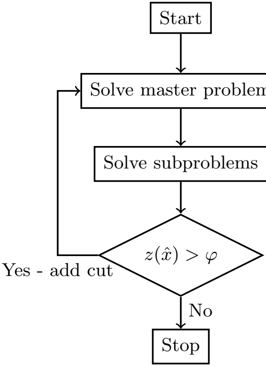
$$\begin{aligned} z(\hat{x}) = \min \quad & d^\top y, \\ \text{subject to} \quad & Dy \geq g - B\hat{x}, \\ & y \geq 0, \\ & y \in \mathbb{R}^{n_2}. \end{aligned}$$

Standard Benders' implementation



- Easy to understand and simple to implement.
- Not always effective, large overhead in repeatedly solving master problem.

Standard Benders' implementation



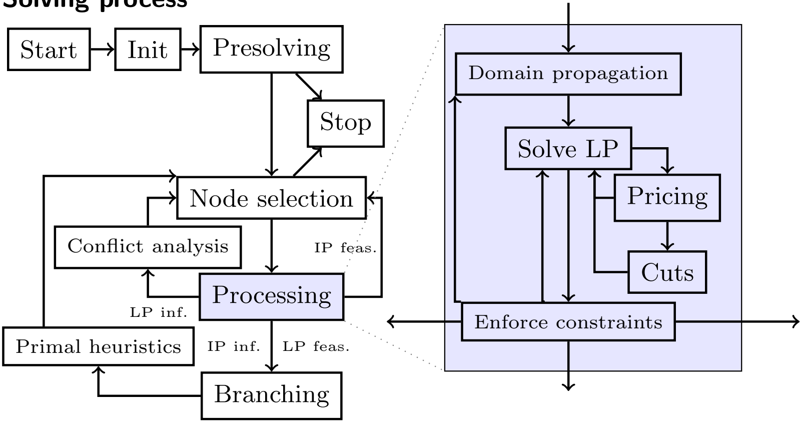
- Easy to understand and simple to implement.
- Not always effective, large overhead in repeatedly solving master problem.

- Easily parallelisable. All subproblems can be solved in parallel.
- Not always efficient—master problem is still solved sequentially.

Branch-and-cut

- Modern solvers pass through a number of different stages during node processing.
- Some of these stages can be used to generate Benders' cuts.
- By interrupting node processing, Benders' cuts are generated during the tree search.

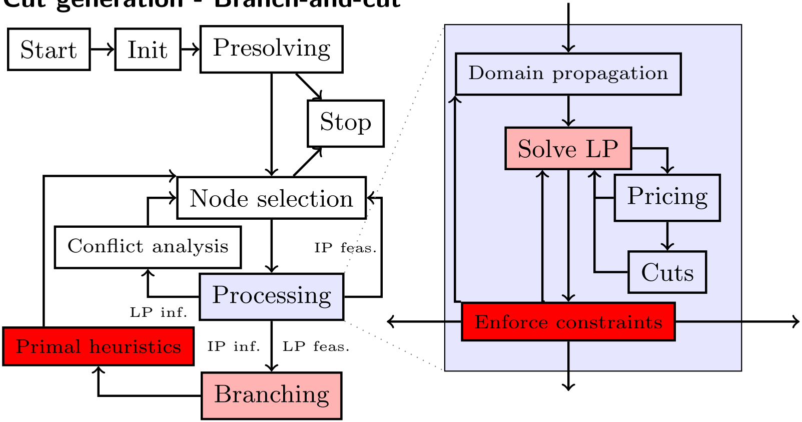
Solving process



Branch-and-cut

- Modern solvers pass through a number of different stages during node processing.
- Some of these stages can be used to generate Benders' cuts.
- By interrupting node processing, Benders' cuts are generated during the tree search.

Cut generation - Branch-and-cut



Branch-and-cut – parallelisation

- Many software implementations for the parallelisation of branch-and-cut, such as the UG framework.
- Useful if solving the subproblems sequentially is not time consuming.
- Addresses issues related to difficult master problem.

Hybrid parallelisation

- Parallelise the branch-and-cut tree search using the UG framework (distributed memory).
- Parallelise the solving of the subproblems using OpenMP (shared memory).

Hybrid parallelisation

- Parallelise the branch-and-cut tree search using the UG framework (distributed memory).
- Parallelise the solving of the subproblems using OpenMP (shared memory).

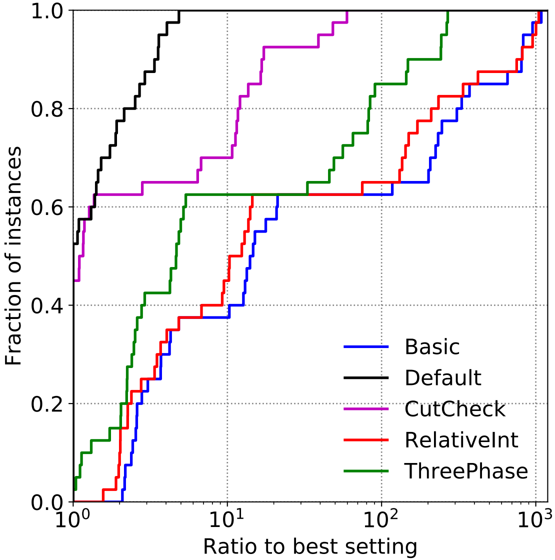
Benefits

- Parallelisation of tree search avoids waiting for difficult master problem solves.
- Solving the subproblems in parallel takes advantage of available cores at each node.
- Able to balance the effort between master problem tree search and subproblem solving.

Current status and features

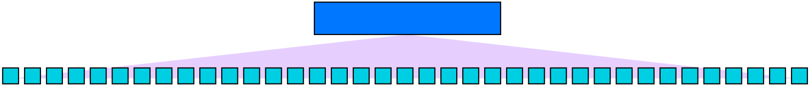
- Both tree search and subproblem parallelisation available.
  - Tree search parallelisation activated by enabling Benders' framework.
  - Subproblem parallelisation enabled by setting the number of threads. SCIP must be built with OpenMP.
- Customisable sorting of subproblems for load balancing
  - Prioritise subproblems with less calls, then by average number of LP iterations.
- Parallelisation relies on transfer of Benders' cuts between *solvers*. By calling `SCIPstoreBendersCut` in custom Benders' cuts plugins, custom Benders' decomposition implementations can be parallelised.

Current challenges



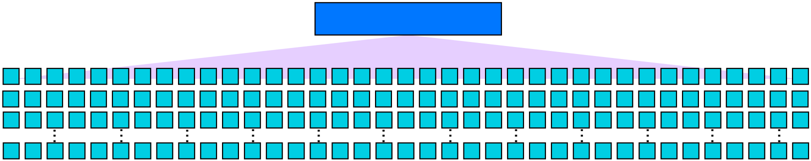
Stochastic Server Location Problem

Design Issues



- Each Benders' subproblem is implemented as a SCIP instance.
- With a large number of subproblems memory consumption can be very high.
- When parallelising the tree search, the subproblems must be *copied* to every solver.
    - Large number of solvers and large number of subproblems results in a very high memory consumption.

Design Issues



- Each Benders' subproblem is implemented as a SCIP instance.
- With a large number of subproblems memory consumption can be very high.
- When parallelising the tree search, the subproblems must be *copied* to every solver.
  - Large number of solvers and large number of subproblems results in a very high memory consumption.

Memory saving mode

- Subproblems, especially in the context of stochastic programming, may have very similar structures.
  - differences only in the constraint matrix or objective function coefficients or in the RHS.
- Create a SCIP instance per each thread. Using a subproblem difference create each subproblem on the fly.

Disadvantage

- Only applicable when subproblems have similar structures
- Does not benefit from warm starting between subproblem solves
- Creating and destroying subproblems is time consuming

Load balancing

- Current best balance of shared and distributed memory unclear
- Benders' cuts are not generated at all nodes in the tree, so reserving threads for subproblem solving may be inefficient.
- Automatic load balancing would allow for idle threads to be used for alternative purposes.

Partial node processing

- Only solve a subset of subproblems at each node in the tree. Delaying the complete processing of the node.
- The node is not fully evaluated, but enough cuts may be generated to improve the bound.
- Identify the balance of the number of subproblems to solve to gain sufficient bound improvement.

Key points

- Hybrid parallel implementation of Benders' decomposition available in SCIP
- Current version is available to solve smaller scale problems.
- Future development will reduce the memory consumption of the Benders' framework, enabling its use on large problems and large computational resources.
- Improved load balancing and partial node processing will be investigated.