Approximate Uni-directional Benders Decomposition

Christina Burt
Nir Lipovetzky, Adrian Pearce, Peter Stuckey

25th January 2015

THE UNIVERSITY OF MELBOURNE

Decomposition for Planning subproblems

Suppose we have a complex real world problem to solve, and we cannot readily solve it using available methods.

One thing we can do is decompose the problem so that we can get good quality solutions, e.g. using Benders decomposition or Logic-based Benders.

However, these methods are limited to approaches that can generate no-good and optimality cuts.

In this talk, we look for a general framework for producing good quality solutions where no such cuts are available from the subproblem.

General Problem

$$
\begin{aligned}
\min \quad & f(x, y) \\
\text{s. t.} \quad & g_k(x, y) \leq 0 \quad \forall\ k \in \{1, \ldots, m_1\}, \\
& g_k(x, y) \leq 0 \quad \forall\ k \in \{m_1 + 1, \ldots, m\}, \\
& x_i \in \mathcal{R}, \qquad \forall\ i \in \mathcal{I}_X, \\
& y_i \in \mathbb{Z}, \qquad \forall\ i \in \mathcal{I}_Y.
\end{aligned}
$$

where f and g are general functions.

General Problem

Suppose the problem can be decomposed into two problems, such that there exists efficient algorithms for either problem.

$$
\begin{aligned}
\min \quad & f(x,y) & \\
\text{s. t.} \quad & g_k(x,y) \leq 0 & \forall\ k \in \{1,\ldots,m_1\}, \\
& g_k(x,y) \leq 0 & \forall\ k \in \{m_1+1,\ldots,m\}, \\
& x_i \in \mathcal{R}, & \forall\ i \in \mathcal{I}_X, \\
& y_i \in \mathbb{Z}, & \forall\ i \in \mathcal{I}_Y.
\end{aligned}
$$

Suppose also that we want to use methods that cannot easily talk to each other, e.g., via no-good or optimality cuts.

Related work

- Susana Fernández and Daniel Borrajo, *Solving clustered oversubscription problems for planning e-courses*, Proceedings of SPARK Workshop, vol. 9, 2009.
- José E Flórez, Alvaro Torralba Arias de Reyna, Javier García, Carlos Linares López, Angel García Olaya, and Daniel Borrajo, *Planning multi-modal transportation problems.*, Proceedings of ICAPS, 2011, pp. 66–73.
- Nir Lipovetzky, Christina N. Burt, Adrian R. Pearce, and Peter J. Stuckey, *Planning for mining operations with time and resource constraints*, Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS-14, 2014.

In this talk. . .

- a naive decomposition
- why the naive approach doesn't work (all the time)
- what does work and why
- example of application

Naive Approach

Suppose the problem can be decomposed into two problems, such that there exists efficient algorithms for either problem.

$$
\begin{aligned}
\min \quad & f(x, y) \\
\text{s. t.} \quad & g_k(x, y) \leq 0 && \forall\ k \in \{1, \ldots, m_1\}, \\
& g_k(x, y) \leq 0 && \forall\ k \in \{m_1 + 1, \ldots, m\}, \\
& x_i \in \mathcal{R}, && \forall\ i \in \mathcal{I}_X, \\
& y_i \in \mathbb{Z}, && \forall\ i \in \mathcal{I}_Y.
\end{aligned}
$$

Naive Approach

Data: Problem to solve
Result: Heuristic solution to problem

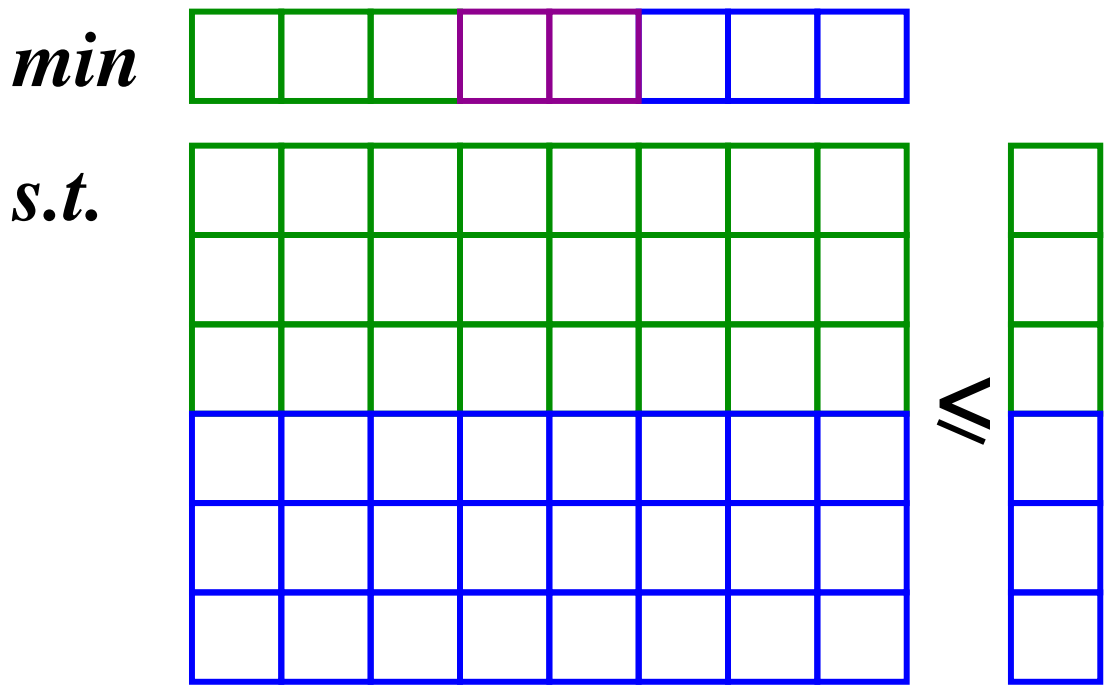
1 **begin**
2 | partition P into two subproblems, $P1$ (*master*) and $P2$
3 | solve $P1$ ignoring variables from $P2$, obtain solution x
4 | substitute solution, $\bar{x}$, into $P2$
5 | solve $P2$
6 **end**

Sometimes this approach can work really well!

But sometimes, it can produce bad quality solutions. In the worst case, it can return *infeasible even when a solution exists*.

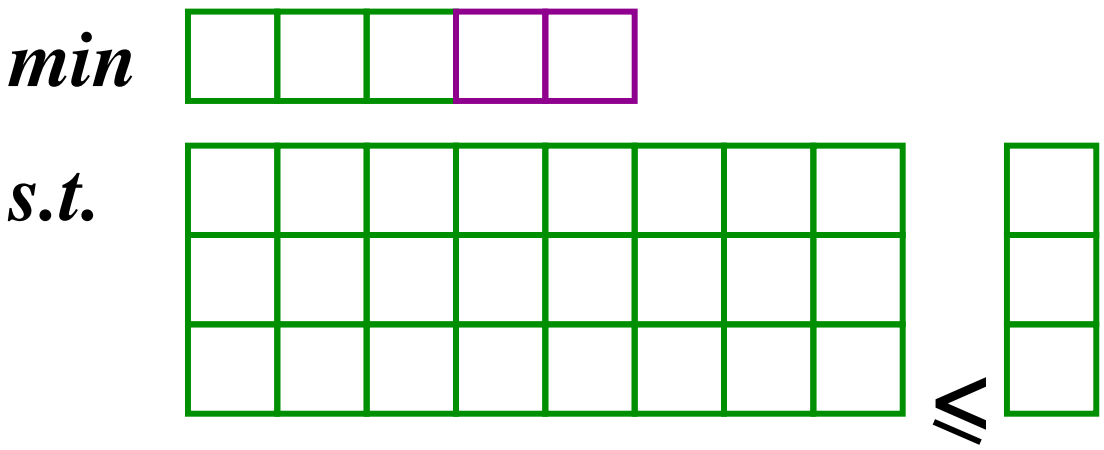
Naive Approach

Partition the problem into two subproblems



Naive Approach

Solve the first subproblem, obtain the solution x.

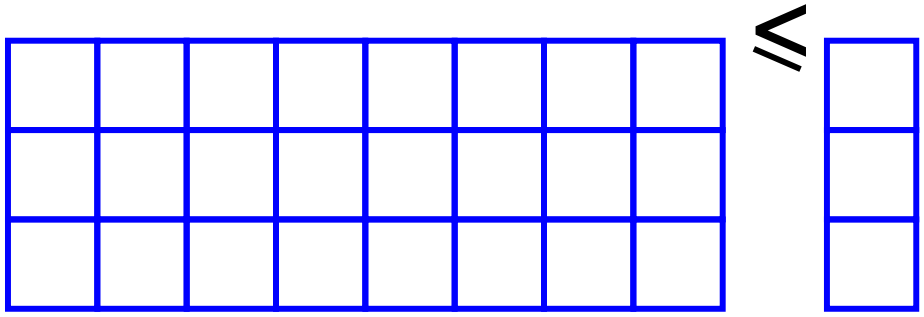


Naive Approach

Fix the solution x in the second subproblem, and solve.

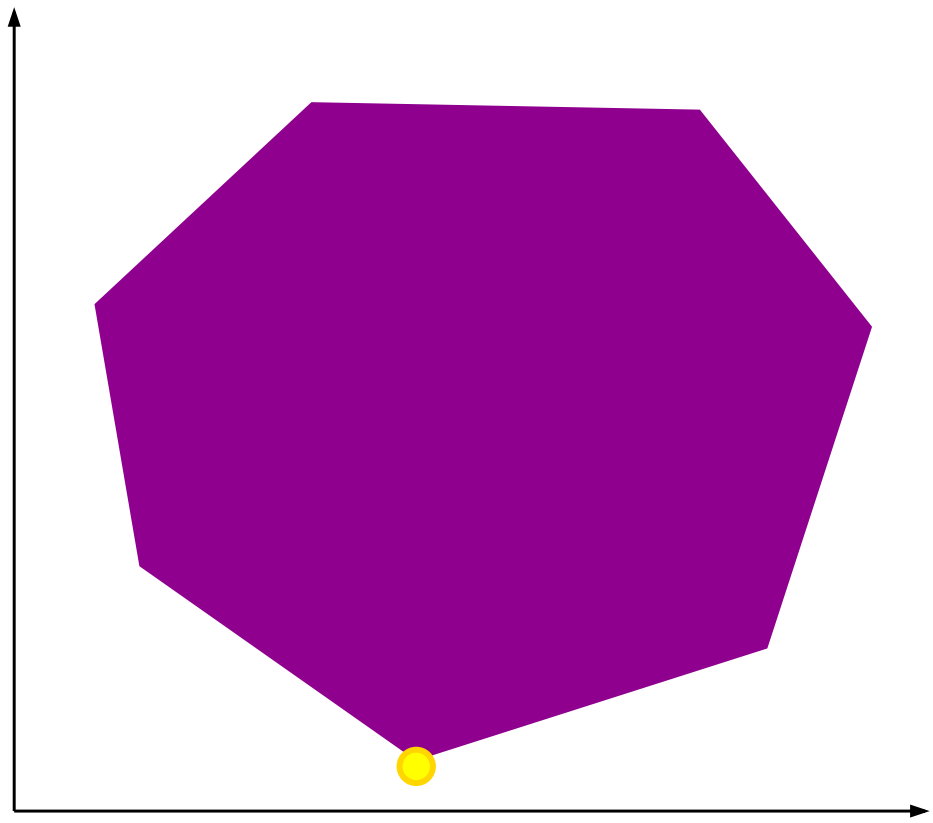
min

s.t.



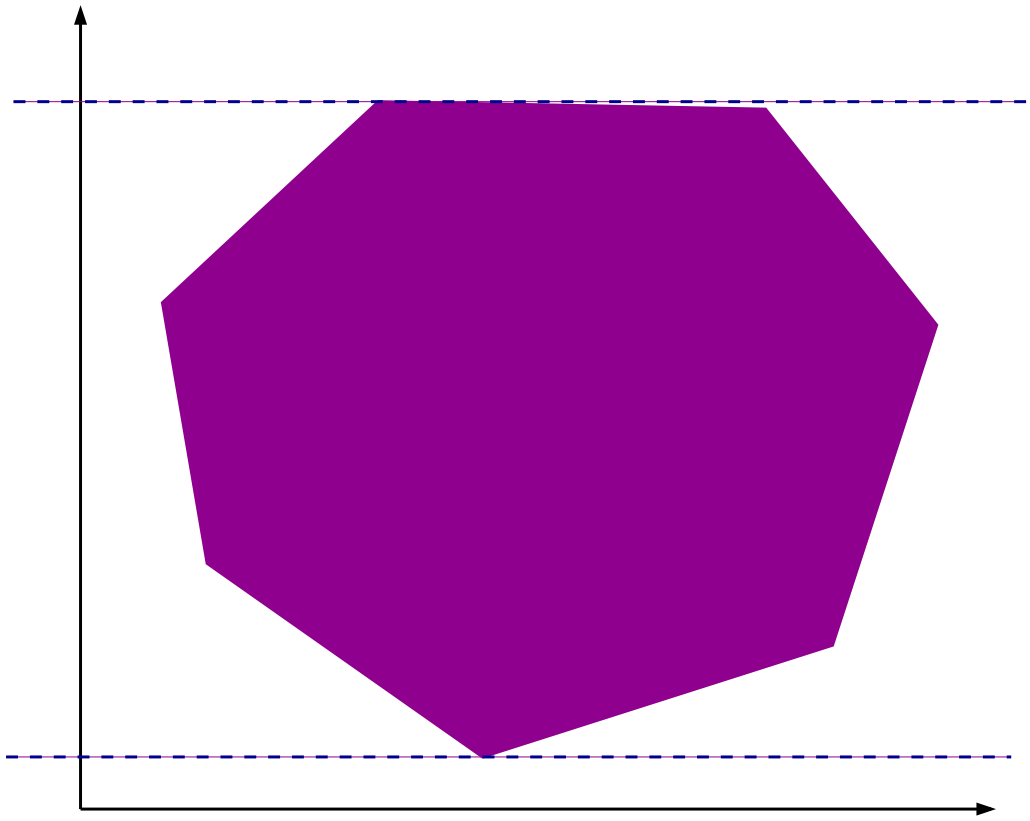
Why the Naive Approach Doesn't Work (all the time)

Consider the **solution space**, here for a 2D problem.



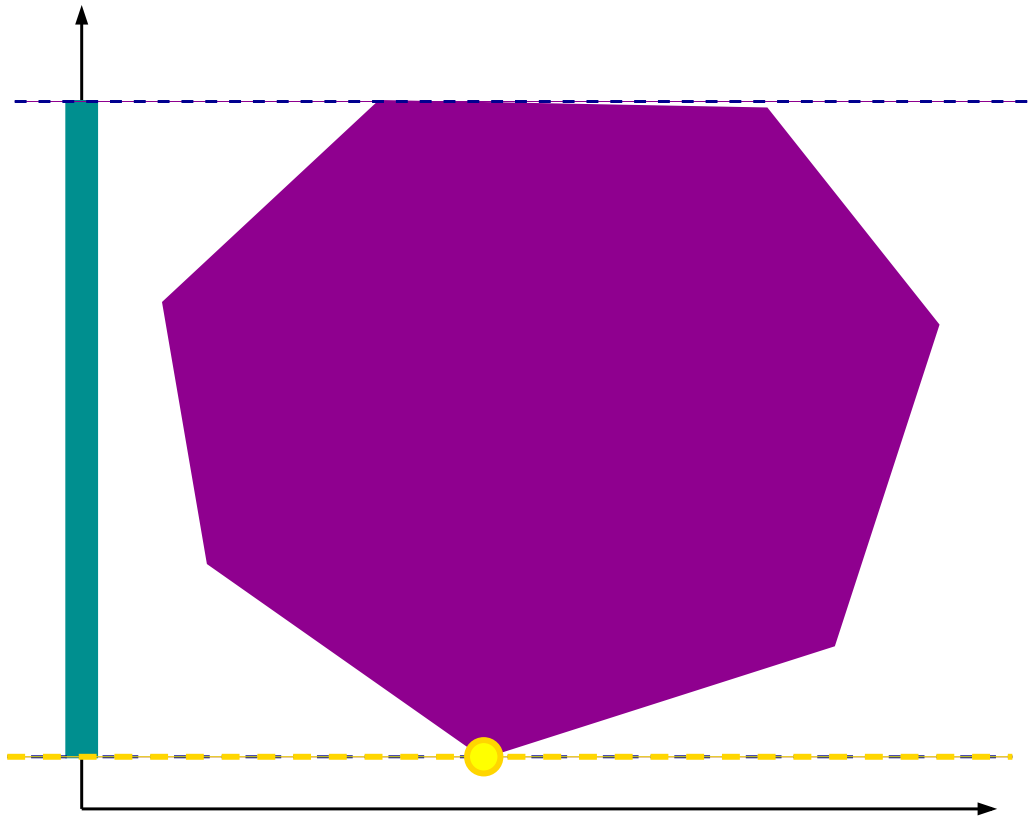
Why the Naive Approach Doesn't Work (all the time)

In the decomposition approach, we calculate the **projection**.



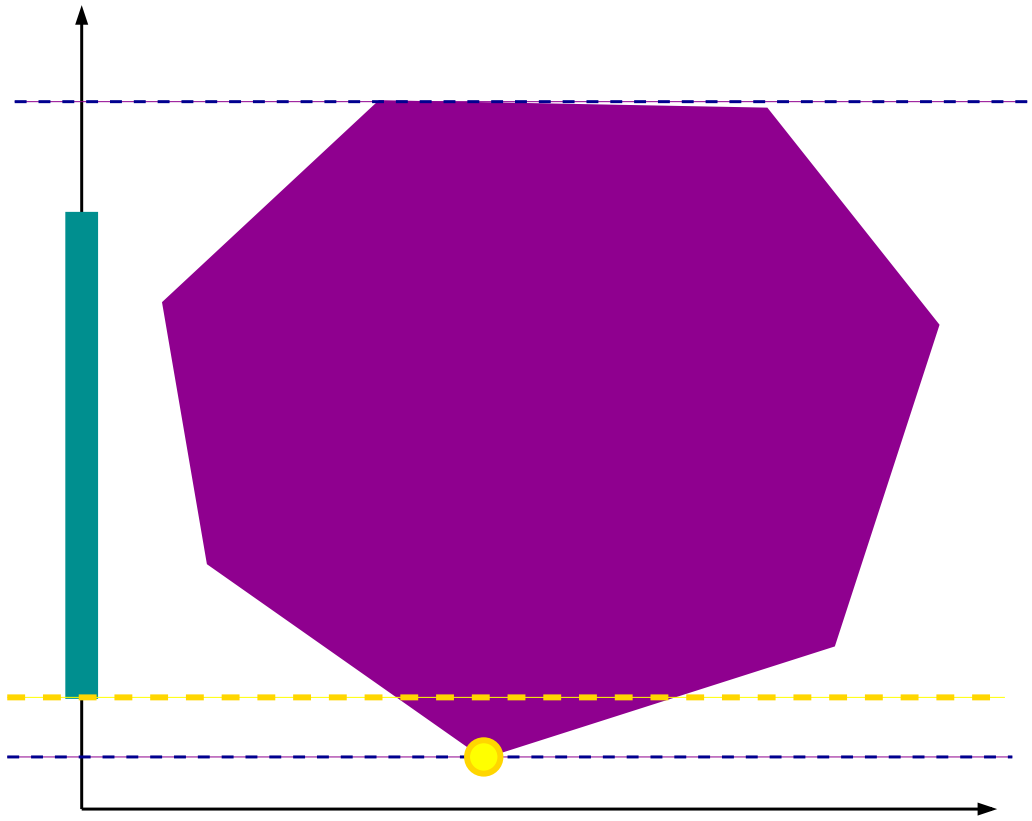
Why the Naive Approach Doesn't Work (all the time)

The **projection** is a representation of the problem in lower dimension. The **correct** projection looks like this.



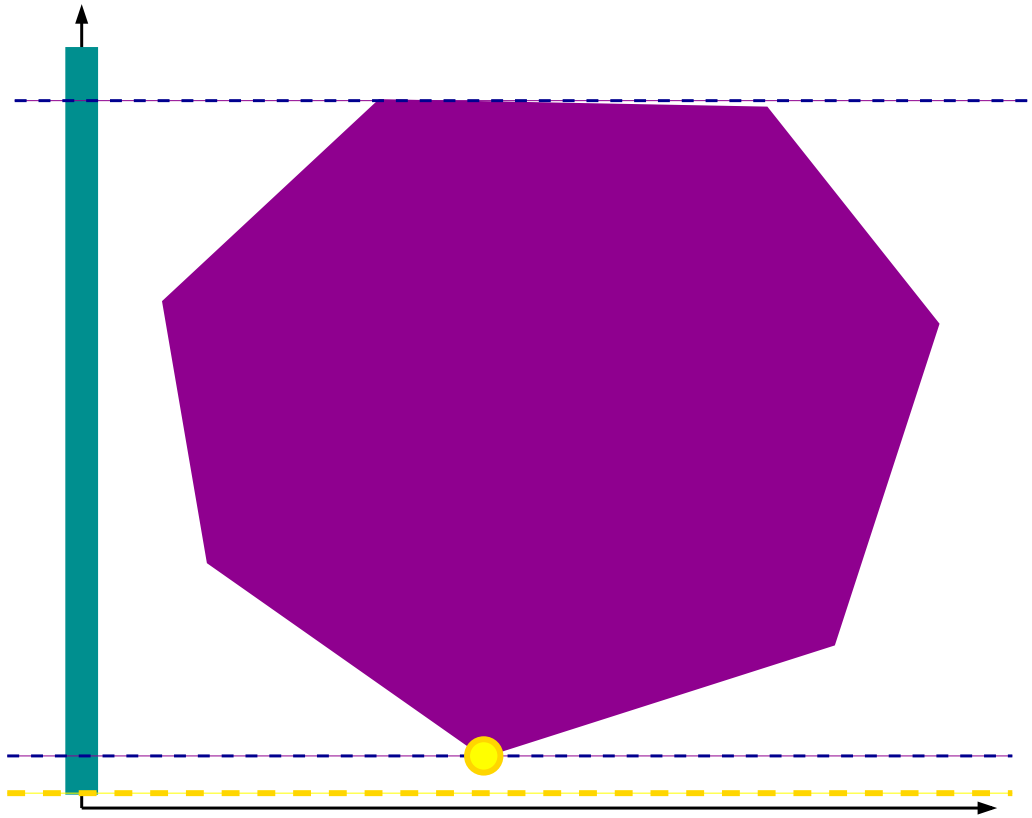
Why the Naive Approach Doesn't Work (all the time)

However, the naive approach can find this.



Why the Naive Approach Doesn't Work (all the time)

Or this.



What is Projection?

Projection is a procedure that takes a problem with n variables, and through substitution, 'projects out' variables.

The result is a problem in **lower dimension** that still tells us something about the full problem.

Gaussian elimination is projection for equality systems.

What is Projection?

For example, consider the system

$$x_1 + 2x_2 = 3 \tag{1}$$
$$3x_1 + x_2 = 4 \tag{2}$$

To 'project' out x_1 we can perform the operation $3 \times \mathrm{row}_1 - \mathrm{row}_2$. This gives

$$5x_2 = 5 \tag{3}$$

We determine that $x_2 = 1$. Substituting this solution into any equation, we determine that $x_1 = 1$.

Observation

Observe that when we solved the subproblem involving x_2, that the coefficients and constants had changed from the original equations.

If we simply *ignore* the projected variable, we would get:

$$2x_2 = 3$$

$$x_2 = 4$$

However, there is a **residual** in the correct projection.

$$5x_2 = 5$$

The naive approach is missing this residual.

The residual is **characterised** by the variables that have been *projected out*.

Approximating the Residual

Therefore, it is possible to **approximate the residual** using an approximation of the variables that have been projected out (i.e., the second subproblem).

Suppose that we can **approximate the second subproblem** by only using variables from the first subproblem (and ancillary variables).

Then this approximation is also an **approximation of the residual**.

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

begin
  partition P into two partitions, $P1$ (*master*) and $P2$
  create an approximation for $P2$, using $P1$- and anc- vars
  add the approximation to the master
  solve the master to obtain partial solution, x
  substitute solution, $\bar{x}$, into $P2$
  solve $P2$
end

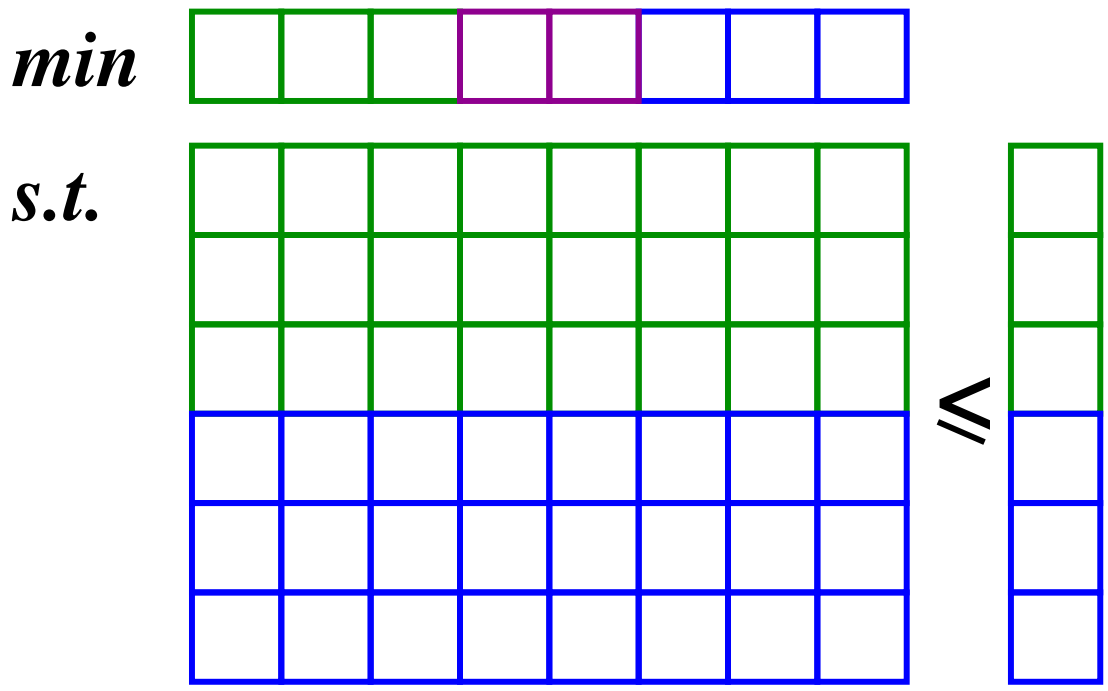
---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

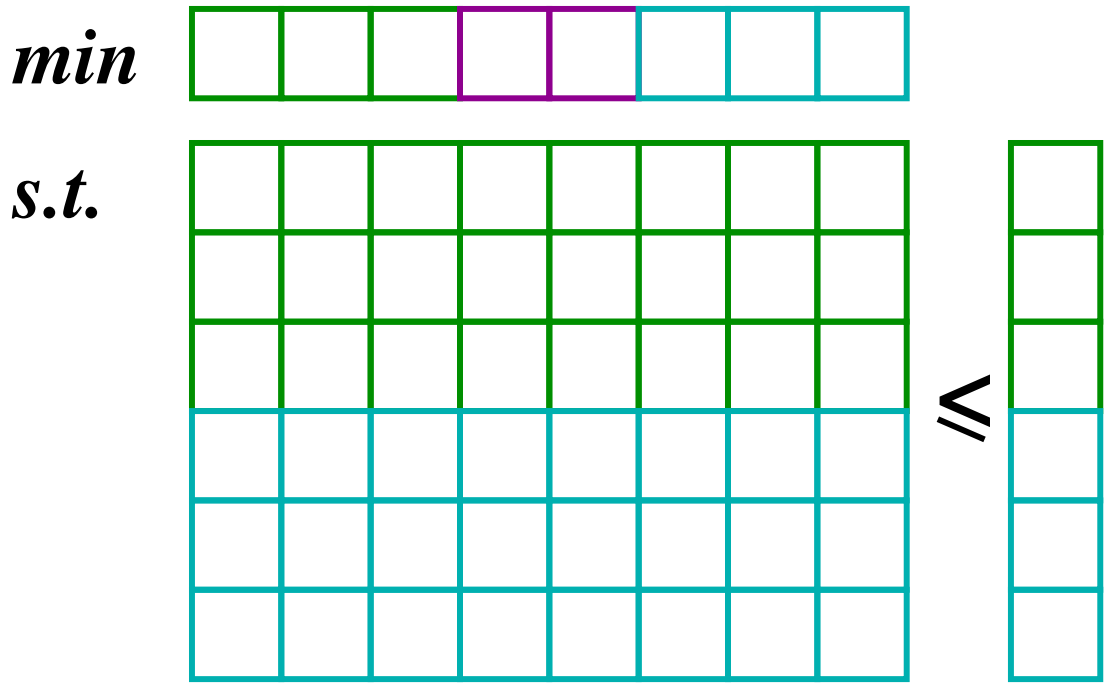
General Framework

Partition the problem into two subproblems



General Framework

Approximate the second subproblem using variables from the first. Call this the master. Solve the master, obtain the solution x.

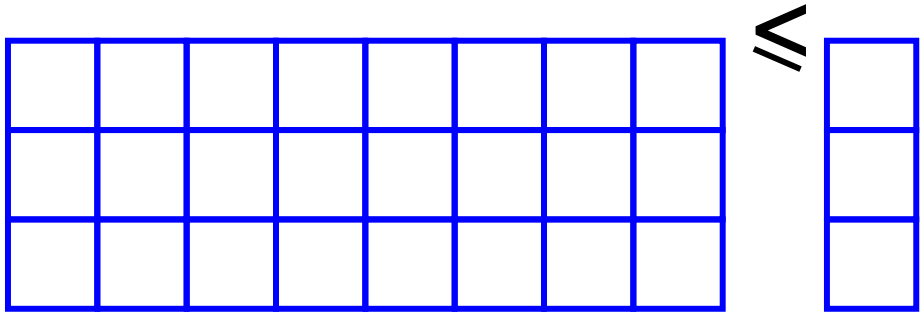


General Framework

Fix the solution x in the second subproblem, and solve.

min

s.t.



Example

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_{e} d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & & \\
\text{s.t.} \quad & z_{m,k} \leq C_{m,k} y_m & & \forall\ m \in N \backslash s, k\ , \\
& \sum_{m} z_{m,k} \geq D_k & & \forall\ k\ , \\
& y_s = 1\ , & & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \leq |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & & \forall\ S \subset N\ , \\
& y_m,\ x_e \in \{0,1\}\ , & & \\
& z_{m,k} \in \mathcal{Z}^+\ . & &
\end{aligned}
$$

The variables determine which markets, which route, and how many of each product.

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_{e} d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & & \\
\text{s.t.} \quad & z_{m,k} \leq C_{m,k} y_m & & \forall\ m \in N \backslash s, k\ , \\
& \sum_{m} z_{m,k} \geq D_k & & \forall\ k\ , \\
& y_s = 1\ , & & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \leq |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & & \\
& z_{m,k} \in \mathcal{Z}^+\ . & &
\end{aligned}
$$

We ensure that products cannot be purchased from markets that are not visited, and should not exceed the capacity of that market.

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & \\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m & \forall\ m \in N \backslash s, k\ , \\
& \sum_m z_{m,k} \ge D_k & \forall\ k\ , \\
& y_s = 1\ , & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & \\
& z_{m,k} \in \mathcal{Z}^+\ . &
\end{aligned}
$$

Purchased products from all markets should at least meet our demand.

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_{e} d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & \\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m & \forall\ m \in N \backslash s, k\ , \\
& \sum_{m} z_{m,k} \ge D_k & \forall\ k\ , \\
& y_s = 1\ , & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & \\
& z_{m,k} \in \mathcal{Z}^+\ . &
\end{aligned}
$$

The trip should start at the source node.

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & & \\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m & & \forall\ m \in N \backslash s, k\ , \\
& \sum_m z_{m,k} \ge D_k & & \forall\ k\ , \\
& y_s = 1\ , & & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & & \\
& z_{m,k} \in \mathcal{Z}^+\ . & &
\end{aligned}
$$

Each node should have an inward and outward arc (degree 2).

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & & \\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m & & \forall\ m \in N \backslash s, k\ , \\
& \sum_m z_{m,k} \ge D_k & & \forall\ k\ , \\
& y_s = 1\ , & & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & & \\
& z_{m,k} \in \mathcal{Z}^+\ . & &
\end{aligned}
$$

For each possible subset of selected markets, the number of edges should be 1 less than the number of markets (subtour elimination).

THE UNIVERSITY OF MELBOURNE

Example: Travelling Purchaser Problem

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & \\
\text{s.t.} \quad & z_{m,k} \leq C_{m,k} y_m & \forall\ m \in N \backslash s, k\ , \\
& \sum_m z_{m,k} \geq D_k & \forall\ k\ , \\
& y_s = 1\ , & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \leq |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & \forall\ S \subset N\ , \\
& y_m, \quad x_e \in \{0,1\}\ , & \\
& z_{m,k} \in \mathcal{Z}^+\ . &
\end{aligned}
$$

Our objective is to minimise the cost of the route and the cost of the products.

THE UNIVERSITY OF MELBOURNE

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

begin
  partition P **into two partitions,** $P1$ **(master) and** $P2$
  create an approximation for $P2$, using $P1$- and ancillary- variables
  add the approximation to the master
  solve the master to obtain partial solution, x
  substitute solution, $\bar{x}$, into $P2$
  solve $P2$
end

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

Example: Travelling Purchaser Problem

We partition the problem into two subproblems. The first subproblem is a facility location problem.

$$
\begin{aligned}
TPP: \quad \min \quad & \sum_e d_e x_e + \sum_{m,k} c_{m,k} z_{m,k} & & \\
\text{s.t.} \quad & z_{m,k} \le C_{m,k} y_m & & \forall\ m \in N \backslash s, k\ , \\
& \sum_m z_{m,k} \ge D_k & & \forall\ k\ , \\
& y_s = 1\ , & & \\
& \sum_{e \in \{(m,j) \forall\ j \in N \backslash m\}} x_e = 2 y_m & & \forall\ m \in N\ , \\
& \sum_{e \in S} x_e \le |S| - \frac{1}{n - |S|} \sum_{m \notin S} y_m & & \forall\ S \subset N\ , \\
& y_m, x_e \in \{0,1\}\ , & & \\
& z_{m,k} \in \mathcal{Z}^+\ . & &
\end{aligned}
$$

The second subproblem is just a TSP.

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

1 **begin**
2 | partition P into two partitions, $P1$ (*master*)) and $P2$
3 | **create an approximation for $P2$, using $P1$- and anc- vars**
4 | add the approximation to the master
5 | solve the master to obtain partial solution, x
6 | substitute solution, $\bar{x}$, into $P2$
7 | solve $P2$
8 **end**

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

THE UNIVERSITY OF MELBOURNE

Example: TPP Residual Approximation

We want to approximate $P2$ using only variables y_m (which markets) and $z_{m,k}$ (how much of each product from each market).

A simple approximation is to visit each market from the start node directly. Therefore, the return route should be counted.

i.e.

$$
\begin{aligned}
\min \quad & 2d_{s,m}y_m && && \min \quad \sum_e d_e x_e \\
& y_m \in \{0,1\}\ , && \approx && \text{s.t.} \sum_{e \in \{(m,j) \forall\ j \in N \setminus m\}} x_e = 2\bar{y}_m \qquad \forall\ m \in N\ , \\
& && && \sum_{e \in S} x_e \leq |S| - \frac{1}{n-|S|} \sum_{m \notin S} \bar{y}_m \qquad \forall\ S \subset N\ , \\
& && && x_e \in \{0,1\}.
\end{aligned}
$$

THE UNIVERSITY OF MELBOURNE

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

1 **begin**
2 | partition P into two partitions, $P1$ (*master*) and $P2$
3 | create an approximation for $P2$, using $P1$- and ancillary- variables
4 | **add the approximation to the master**
5 | solve the master to obtain partial solution, x
6 | substitute solution, $\bar{x}$, into $P2$
7 | solve $P2$
8 **end**

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

1 **begin**
2 | partition P into two partitions, $P1$ (*master*) and $P2$
3 | create an approximation for $P2$, using $P1$- and ancillary- variables
4 | add the approximation to the master
5 | **solve the master to obtain partial solution,** x
6 | substitute solution, $\bar{x}$, into $P2$
7 | solve $P2$
8 **end**

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

1 **begin**
2 | partition P into two partitions, $P1$ (*master*) and $P2$
3 | create an approximation for $P2$, using $P1$- and ancillary- variables
4 | add the approximation to the master
5 | solve the master to obtain partial solution, x
6 | **substitute solution, $\bar{x}$, into $P2$**
7 | solve $P2$
8 **end**

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

General Framework

Data: Problem P
Result: 'Good quality'[a] feasible[b] solution for P

1 **begin**
2 | partition P into two partitions, $P1$ (*master*) and $P2$
3 | create an approximation for $P2$, using $P1$- and ancillary- variables
4 | add the approximation to the master
5 | solve the master to obtain partial solution, x
6 | substitute solution, $\bar{x}$, into $P2$
7 | **solve** $P2$
8 **end**

---

[a] preferably construct the *master* so that not all optimal solutions are excluded

[b] construct the *master* so that any solution is feasible for $P2$

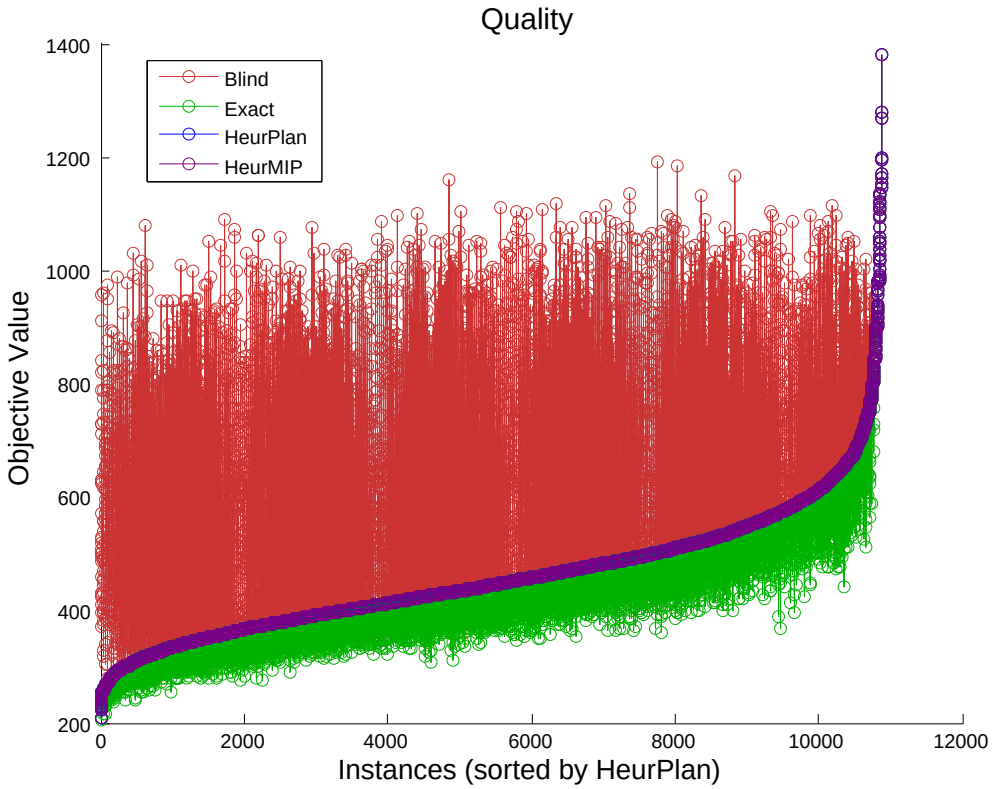
Example: TPP Experiments

We will run 14300 randomly generated instances between 5 and 30 markets.

We run the following experiments:

- Blind is the naive decomposition
- HeurPlan solves the first partition using MIP and the second using a Planner
- HeurMIP solves both partitions using MIP
- Exact solves the full problem using MIP

Example: TPP Experiments



The average quality of HeurMIP is **1.0598** times the optimal solution; Blind is **1.4268** times the optimal solution.

For 30 markets, the average times were blind took 0.19s, HMIP took 0.22s, Hplan took 0.46s and Exact took 7.48s.

THE UNIVERSITY OF MELBOURNE

Example: MOP Problem

This is an example using Planning as a temporal planing subproblem with state-dependent components, see *Lipovetzky et. al. 2014 ICAPS Applications Track*.

	Makespan [minutes]			
Inst.	P1/POPF	H1/POPF	P1/CPT	H1/CPT
(1)	12218	**9704**	—	—
(2)	8131	**7130**	7805	—
(3)	14906	9475	14659	**7835**
(4)	12171	7645	10438	**6021**
(5)	37405	**26761**	—	—

Conclusions

- we have described a **decomposition framework** for obtaining good quality feasible solutions,
- the framework allows for **any method** (e.g. Planning, MIP) to be used for either partition,
- the approach can therefore be used to **reduce the search space** Planning problems.

Approximate Uni-directional Benders Decomposition

Christina Burt
Nir Lipovetzky, Adrian Pearce, Peter Stuckey

25th January 2015

THE UNIVERSITY OF MELBOURNE