Finding all efficient solutions to a bi-objective combinatorial optimization problem

S.L. Gadegaard

Aarhus University
Denmark

2018 RAMOO Nantes

The problem

- We want to compute all optimal solutions to the problem

$$\max C^1 x$$
$$\max C^2 x$$
$$\text{s.t.: } x \in \mathcal{X}$$

  where $\mathcal{X} \subseteq \{0,1\}^n$.
- “compute all optimal solutions” $\sim$ *generate all solutions in* $\mathcal{X}_E$
- Develop a decision space-based algorithm
- Use the the machinery lying around from single objective optimization.

Motivation

- Most effort is on generating a minimal complete set of efficient solutions.
- Often, an optimal solution is not really "optimal" for a decision maker
  - The outcome might not be as important as the solution itself (as long as it is a "good" solution).
  - Some "hidden" utility function must be optimized over the set of $\mathcal{X}_E$
  - Alternative optimal solutions tell about the problem and its structure
- Some special functions attain their optimum over the efficient set of multi-objective combinatorial optimization problem.

Bi-objective branch and bound

A B&B algorithm for bi–objective optimization can be outlined as follows

Initialization: Initialize a stack, T, of branching nodes with a root node and initialize a lower bound set, L, consisting of feasible tuples $\{x, z\}$

Node selection: Pick a node, $\eta \in T$, from the stack of branching nodes

Upper bound set: Generate an upper bound set, U_η, of all efficient solutions contained in η

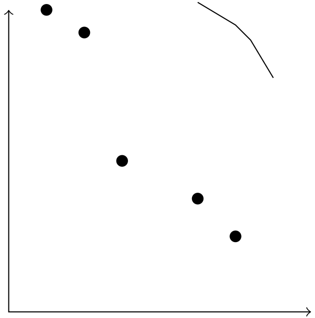
Lower bound set: Update the lower bound set, L, if new yet non-dominated solutions have been found

Pruning: If $U_\eta \preceq L$, no non–dominated solutions exists in subsequent nodes. Go to Node selection.

Varaible selction: Pick a variable x_i to branch on.
Set $\eta_0 = \eta \cap \{x_i = 0\}$, $\eta_1 = \eta \cap \{x_i = 1\}$, $T = (T \cup \eta_0 \cup \eta_1) \setminus \eta$.
Go to Node selection

Two new attempts – Upper bound set

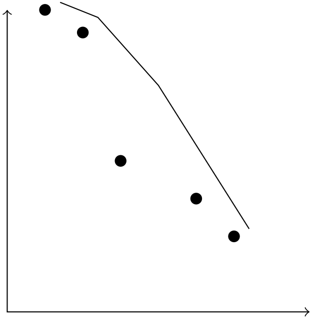
- At each branching node, compute bi-objective LP relaxation



S.L Gadegaard, L.R.Nielsen and M. Ehrgott
Bi-objective branch-and-cut algorithms based on LP relaxation and bound sets
To appear in *INFORMS Journal on Computing* (Accepted 2018)

Two new attempts – Upper bound set

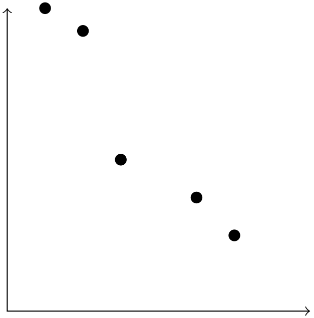
- At each branching node, compute bi-objective LP relaxation
  $\rightarrow$ Add cuts



S.L Gadegaard, L.R.Nielsen and M. Ehrgott
Bi-objective branch-and-cut algorithms based on LP relaxation and bound sets
To appear in *INFORMS Journal on Computing* (Accepted 2018)

Two new attempts – Lower bound set

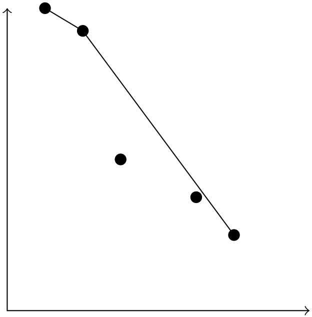
- At each branching node, compute extreme supported IP solutions



S.N Parragh and F. Tricoire
Branch-and-bound for bi-objective integer programming
optimization-online.org (2015)

Two new attempts – Lower bound set

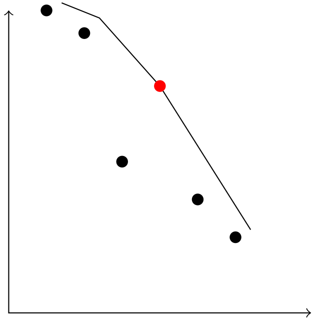
- At each branching node, compute extreme supported IP solutions
  $\rightarrow$ Much stronger bound, but harder to compute



S.N Parragh and F. Tricoire
Branch-and-bound for bi-objective integer programming
optimization-online.org (2015)

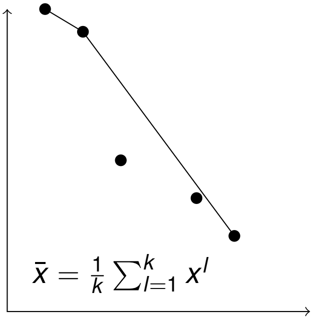
Two new attempts – Branching

- Branch on a variable that separates the current node
  - Branch on one of the extreme points of the bi-objective LP relaxation
  - Branch on variable with fractional average value of IP extreme supported solutions.
  - Branch in objective space



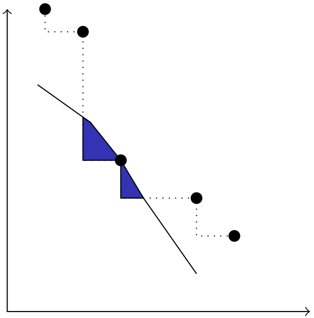
Two new attempts – Branching

- Branch on a variable that separates the current node
  - Branch on one of the extreme points of the bi-objective LP relaxation
  - Branch on variable with fractional average value of IP extreme supported solutions.
  - Branch in objective space



Two new attempts – Branching

- Branch on a variable that separates the current node
  - Branch on one of the extreme points of the bi-objective LP relaxation
  - Branch on variable with fractional average value of IP extreme supported solutions.
  - Branch in objective space



What seems to work best?

- Branching in objective space
  - Cuts out a large part of the feasible space
  - Can be implemented without removing equivalent solutions
- Upper bound set from IP is much stronger
  - Much better pruning potential
  - Time consuming if no good algorithm is known
- Objective space cuts ruin structure!

What seems to work best?

- Branching in objective space
  - Cuts out a large part of the feasible space
  - Can be implemented without removing equivalent solutions
- Upper bound set from IP is much stronger
  - Much better pruning potential
  - Time consuming if no good algorithm is known
- Objective space cuts ruin structure!

What seems to work best?

- Branching in objective space
  - Cuts out a large part of the feasible space
  - Can be implemented without removing equivalent solutions
- Upper bound set from IP is much stronger
  - Much better pruning potential
  - Time consuming if no good algorithm is known
- Objective space cuts ruin structure!

What I want to do

- Utilize problem specific solvers as much as possible

What I want to do

- Utilize problem specific solvers as much as possible
  - Weighted sum scalarizations
  - No complicating constraints

Some trivial observations

Branching on single variables, does (usually) not ruin structure of a problem

Some trivial observations

Branching on single variables, does (usually) not ruin structure of a problem

- For the linear assignment problem: If $x_{ij} = 1$ remove job i and agent j and add c^1_{ij} and c^2_{ij} to objectives. If $x_{ij} = 0$ set $c_{ij} = \infty$

Some trivial observations

Branching on single variables, does (usually) not ruin structure of a problem

- For the 0-1 knapsack problem: If $x_i = 1$ remove item i, reduce capacity by w_i, and add p_i^1 and p_i^2 to objectives. If $x_i = 0$, remove item i

Some trivial observations

Branching on single variables, does (usually) not ruin structure of a problem

- For facility location: If $y_i = 1$ add f_i^1 and f_i^2 to the objective functions and set $f_i^1 = f_i^2 = 0$. If $y_i = 0$ set $f_i = \infty$

Some trivial observations

Branching on single variables, does (usually) not ruin structure of a problem

- $\rightarrow$ We can use customized solvers, if we branch on single variables

Some trivial observations

For any efficient solution, $\tilde{x}$, to

$$\begin{aligned} &\max\ C^1 x \\ &\max\ C^2 x \\ &\text{s.t.: } x \in \mathcal{X} \end{aligned}$$

there exists a $0 < \lambda < 1$, $\mathcal{I} \subseteq \{1, \dots, n\}$, and $\tau \in \{0,1\}^{\mathcal{I}}$ such that

$$\tilde{x} \in \arg\max\{(\lambda C^1 + (1-\lambda)C^2)x \ : \ x \in \mathcal{X}, x_i = \tau_i \ \forall i \in \mathcal{I}\}$$

Basic algorithm design

Initialization: Initialize a stack, T, of branching nodes with a root node and initialize a lower bound set, L, consisting of feasible tuples $\{x, z\}$

Node selection: Pick a node, $\eta \in T$, from the stack of branching nodes

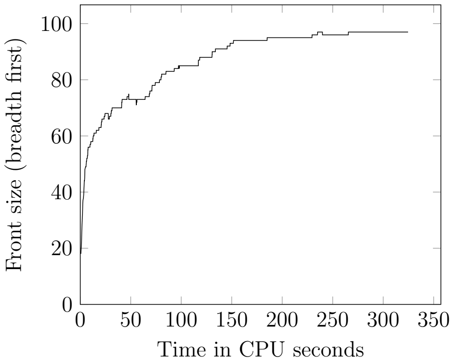
Upper bound set: Generate a solution for each (extreme) supported non-dominated outcome of η, say $\{x^1, \ldots, x^k\}$. Set $U_\eta = \left(\text{conv}(\{x^1, \ldots, x^k\})\right)_N$

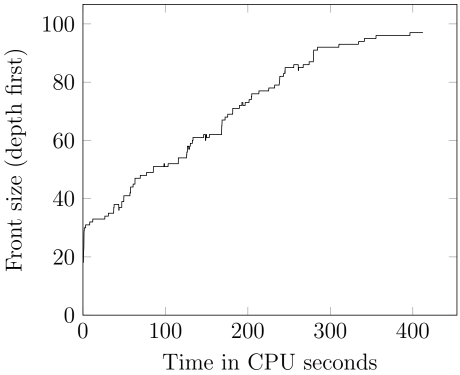
Lower bound set: Add $\{x^1, \ldots, x^k\}$ to L and filter for dominated solutions

Pruning: If $U_\eta \preceq L$, no non–dominated solutions exists in subsequent nodes. Go to Node selection.

Varaible selction: Pick a variable x_i to branch on.
Set $\eta_0 = \eta \cap \{x_i = 0\}$, $\eta_1 = \eta \cap \{x_i = 1\}$, $T = (T \cup \eta_0 \cup \eta_1) \setminus \eta$.
Go to Node selection

(Very) Preliminary results - Assignment problem

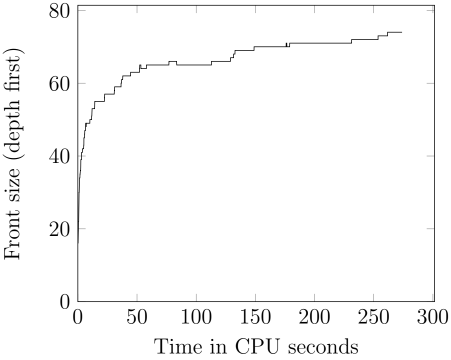


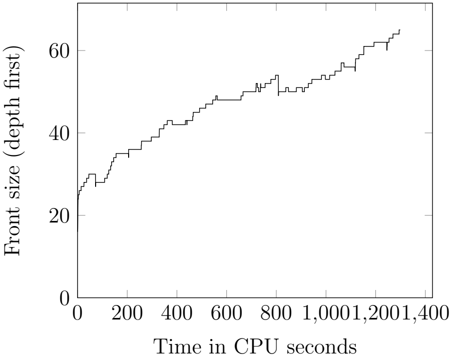


68,602 LPs
5063 Nodes
325 seconds
0.06 seconds per node

89,714 LPs
7221 Nodes
408 seconds
0.05 seconds per node

(Very) Preliminary results – Knapsack

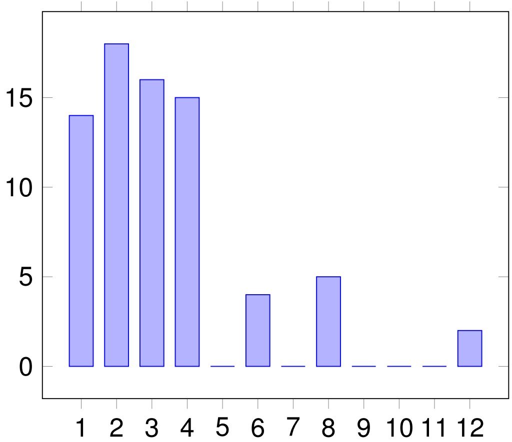




75,489 IPs
5940 Nodes
274 seconds
0.05 seconds per node

— IPs
28,031 Nodes
1,298 seconds
0.04 seconds per node

(Very) Preliminary results – Knapsack



Variable selection

- For the node η to be separated, we must have $\bar{x}_i := \frac{1}{k}\sum_{l=1}^{k} x_i^l \in (0, 1)$
- Many branching strategies could be used: most fractional, random, adapted versions of strong/pseudo cost/reliability branching.
- Must balance the effort of recomputing lower bound sets, selecting variables, impact on lower bound set a.s.o..

Variable selection

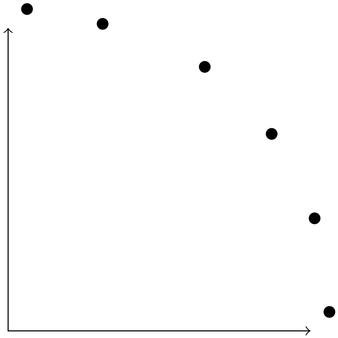
- For the node η to be separated, we must have $\bar{x}_i := \frac{1}{k}\sum_{l=1}^{k} x_i^l \in (0,1)$
- Many branching strategies could be used: most fractional, random, adapted versions of strong/pseudo cost/reliability branching.
- Must balance the effort of recomputing lower bound sets, selecting variables, impact on lower bound set a.s.o..

Variable selection

- For the node η to be separated, we must have $\bar{x}_i := \frac{1}{k}\sum_{l=1}^{k} x_i^l \in (0,1)$
- Many branching strategies could be used: most fractional, random, adapted versions of strong/pseudo cost/reliability branching.
- Must balance the effort of recomputing lower bound sets, selecting variables, impact on lower bound set a.s.o..

Variable selection – an example

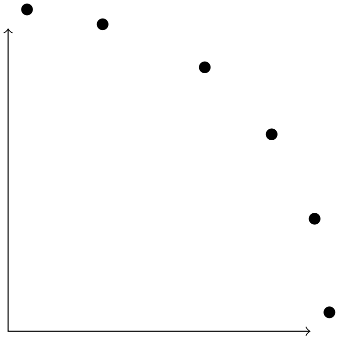
	x_1	x_2	x_3	x_4	x_5	x_6
Solution 1	1	1	1	0	0	1
Solution 2	1	0	1	0	1	1
Solution 3	0	0	1	1	0	1
Solution 4	1	1	0	0	0	1
Solution 5	1	0	0	1	0	1
Solution 6	0	1	0	1	0	0



Variable selection – an example

	x_1	x_2	x_3	x_4	x_5	x_6
Solution 1	1	1	1	0	0	1
Solution 2	1	0	1	0	1	1
Solution 3	0	0	1	1	0	1
Solution 4	1	1	0	0	0	1
Solution 5	1	0	0	1	0	1
Solution 6	0	1	0	1	0	0

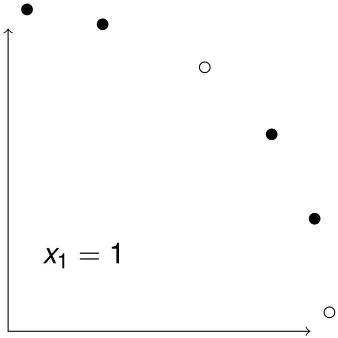
What happens if we branch on variable x_1?



Variable selection – an example

	x_1	x_2	x_3	x_4	x_5	x_6
Solution 1	1	1	1	0	0	1
Solution 2	1	0	1	0	1	1
Solution 3	0	0	1	1	0	1
Solution 4	1	1	0	0	0	1
Solution 5	1	0	0	1	0	1
Solution 6	0	1	0	1	0	0

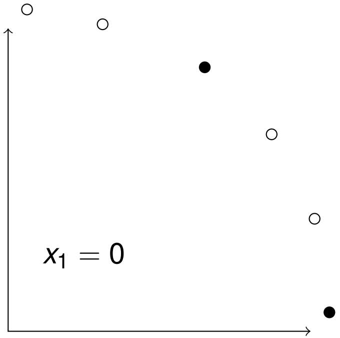
What happens if we branch on variable x_1?



Variable selection – an example

	x_1	x_2	x_3	x_4	x_5	x_6
Solution 1	1	1	1	0	0	1
Solution 2	1	0	1	0	1	1
Solution 3	0	0	1	1	0	1
Solution 4	1	1	0	0	0	1
Solution 5	1	0	0	1	0	1
Solution 6	0	1	0	1	0	0

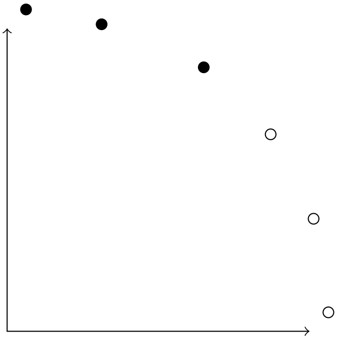
What happens if we branch on variable x_1?



Variable selection – an example

	x_1	x_2	x_3	x_4	x_5	x_6
Solution 1	1	1	1	0	0	1
Solution 2	1	0	1	0	1	1
Solution 3	0	0	1	1	0	1
Solution 4	1	1	0	0	0	1
Solution 5	1	0	0	1	0	1
Solution 6	0	1	0	1	0	0

What happens if we branch on variable x_3?



Lagrangean relaxation of no goods

- Given a feasible solution $\bar{x}$ a valid inequality is $\sum_{i:\bar{x}_i=0} x_i + \sum_{i:\bar{x}_i=1}(1-x_i) \geq 1$
- For a given $\lambda \in (0,1)$ let $\mu(\lambda)$ be an optimal dual multiplier to

$$\min_{\mu} \max \; (\lambda C^1 + (1-\lambda)C^2)x + \mu(\sum_{i:\bar{x}_i=0} x_i + \sum_{i:\bar{x}_i=1}(1-x_i) - 1)$$

$$\text{s.t.: } x \in \mathcal{X}$$

Questions?