Branch and bound for multi-objective optimization

Overview and future challenges

S.L. Gadegaard

Department of Economics and Business Economics
Aarhus BSS, Aarhus University

September 13, 2024

For the hands on part:

1. Make sure you have a working, updated installation of Python
2. Make sure you have a solver installed (CBC, cplex, Gurobi)
3. I can recommend the PyCharm Community Edition IDE
4. Visit https://github.com/SuneGadegaard/RAMOO2024

Outline

The single objective case

Given is a (feasible) combinatorial optimization problem

$$\min f(x)$$
$$\text{s.t.: } x \in \mathcal{X} \subseteq \{0,1\}^n$$

- $f : \mathcal{X} \to \mathbb{R}$
- x^* is an optimal solution
- $z^* = f(x^*)$ is the optimal objective function value.
- We will assume the problem is linear: $f(x) = c^T x$ and $\mathcal{X} = \{x \in \{0,1\}^n \ : \ Ax \geq b\}$

Implicit enumeration and divide and conquour

- Solving $\min\{f(x) \ : \ x \in \mathcal{X}\}$ may be too computationally hard
- Given a collection $\{\eta_i\}_{i=1}^K$, with $\mathcal{X}(\eta_i) \subseteq \mathcal{X}$ and $\mathcal{X} = \bigcup_{i=1}^K \mathcal{X}(\eta_i)$
- Let $z^*(\eta_i) = \min\{f(x) \ : \ x \in \mathcal{X}(\eta_i)\}$ for $i = 1, ..., K$
- Then $z^* = \min\{z^*(\eta_i) : i = 1, ..., K\}$
- Suppose an *upper bound* on z^* is known: $z^* \leq U$
- Suppose a *lower bound* on $z^*(\eta_i)$ is known: $L(\eta_i) \leq z^*(\eta_i)$
- If $L(\eta_i) > U$, then we do not need to solve $\min\{f(x) \ : \ x \in \mathcal{X}(\eta_i)\}$

A branch-and-bound algorithm

- Use the convention that a lower bound on an infeasible problem is ∞.

Branch-and-bound algorithm Bertsimas and Tsitsiklis (1997)

Step 0: Initialize η_0 with $\mathcal{X}(\eta_0) = \mathcal{X}$. Set $T = \{\eta_0\}$, $U = \infty$ and $x^* = \texttt{null}$.

Step 1: If $T = \emptyset$, return (x^*, U) as an optimal solution/solution value.
Else, pop a subproblem η from T.

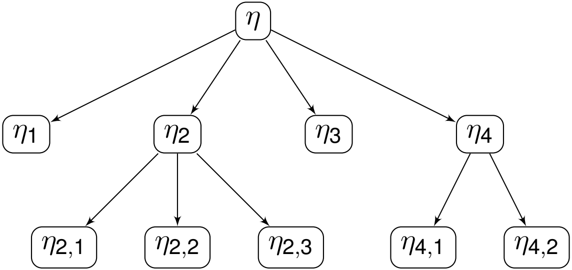
Step 2: Obtain a lower bound $L(\eta)$ of the subproblem η. If $L(\eta) \geq U$ go back to Step 1. Else continue to Step 3.

Step 3: *If possible*, obtain an optimal solution $x^*(\eta) \in \arg\min\{f(x) \ : \ x \in \mathcal{X}(\eta)\}$. Update (x^*, U) if necessary. Go back to Step 1. *Else* go to Step 4

Step 4: Split η into smaller problems $\{\eta_i\}_i$: $\mathcal{X}(\eta_i) \subset \mathcal{X}(\eta)$ and $\bigcup_i \mathcal{X}(\eta_i) = \mathcal{X}(\eta)$. Set $T = T \cup (\bigcup_i \eta_i)$ and go back to Step 1.

Branch-and-bound and tree search

- A natural way to represent the subproblems is a tree - hence, *tree search*



LP-based branch-and-bound

1. Many successful implementations use linear programming for lower bounding
2. Solving an LP relaxation gives
   1. Proof that subproblem infeasible, or
   2. A lower bound if LP-feasible, and
   3. An optimal solution to subproblem if integer-feasible

LP-based branch-and-bound - pruning by infeasibility

- Given a subproblem, η, one starts by solving its LP-relaxation.
  - If the LP-relaxation is infeasible, so is the subproblem
- Hence, we can remove the subproblem, as it contains no solutions

LP-based branch-and-bound - pruning by optimality

- Let x^{LP} be an optimal LP solution to a subproblem.
- If $x^{LP} \in \{0, 1\}^n$, then x^{LP} is optimal for the subproblem
- No need to explore that subproblem any further
- Update the global upper bound: $U = \min\{U,\ c^T x^{LP}\}$

LP-based branch-and-bound - pruning by bound

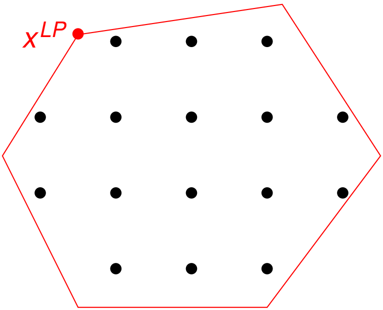
- Let $x^{LP} \notin \{0, 1\}^n$ be an optimal LP solution to a subproblem.
- Assume we have an upper bound on z^*, say U.
- If $L(\eta) := c^T x^{LP} \geq U$, we can prune η
- The node η will not contain improving solutions.

LP-based branch-and-bound - branching

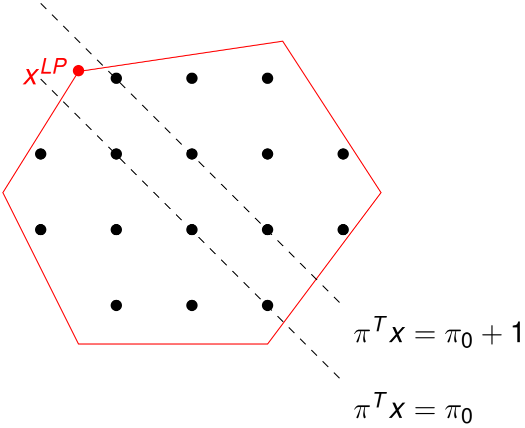
- Let x^{LP} be an optimal LP solution to subproblem η.
- Assume that $x^{LP} \notin \{0, 1\}^n$ and η cannot be pruned.
- We need to split η into smaller subproblems.
- Ideally, x^{LP} should not be included in any new subproblems
- To keep the problems linear, we may find a pair $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ such that

$$\pi_0 < \pi^T x^{LP} < \pi_0 + 1$$

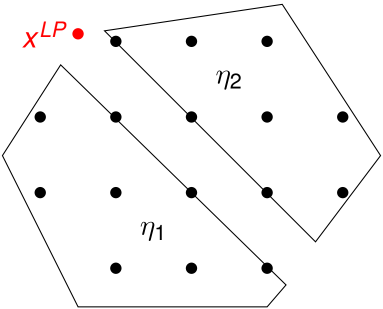
Lp-based branch-and-bound - branching



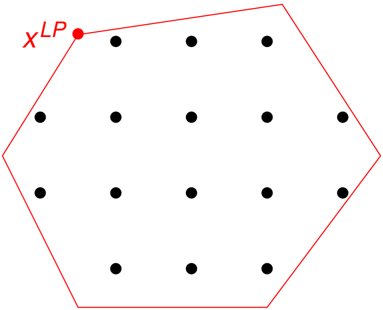
Lp-based branch-and-bound - branching



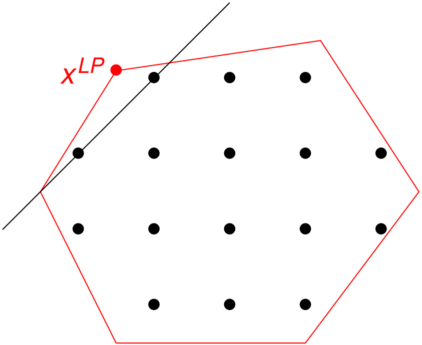
Lp-based branch-and-bound - branching



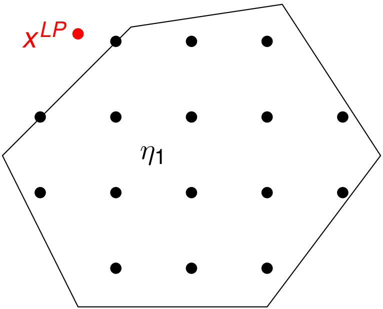
Lp-based branch-and-bound - branching



Lp-based branch-and-bound - branching



Lp-based branch-and-bound - branching



LP-based branch-and-bound - branching

$$
\begin{tikzcd}[column sep=small]
 & \eta \arrow[dl] \arrow[dr] & & \\
\eta_1 := \eta \cap \{\pi^T x \leq \pi_0\} & & \eta_2 := \eta \cap \{\pi^T x \geq \pi_0 + 1\} \arrow[dl] \arrow[dr] & \\
 & \eta_{2,1} := \eta_2 \cap \{\tilde{\pi}^T x \leq \tilde{\pi}_0\} & & \eta_{2,2} := \eta_2 \cap \{\tilde{\pi}^T x \geq \tilde{\pi}_0 + 1\}
\end{tikzcd}
$$

LP-based branch-and-bound - branching

- If $x_i \not\in \{0, 1\}$, then $\pi = e_i$ and $\pi_0 = 0$ is a natural choice.
  - Variable branching

LP-based branch-and-bound - branching

- If $x_i \notin \{0, 1\}$, then $\pi = e_i$ and $\pi_0 = 0$ is a natural choice.
  - Variable branching
- An obvious question is: What fractional variable should we branch on?
  - Excellent treatment of this problem in Achterberg, Koch, and Martin (2005)
  - Learning based strategies are treated in Khalil et al. (2016)

LP-based branch-and-bound - node selection

- After branching, a new node must be selected for processing
- Influences how fast good solutions are found and problems are solved
- Several strategies have been proposed
  1. Depth first search
  2. Breadth first search
  3. Best bound search
  4. ML based strategies

Outline

Branch-and-bound for MOCO problems - The problem

- We will now add at least one more *linear* objective function

$$\begin{aligned} &\min\ Cx \\ &\text{s.t.: } Ax \geq b \\ &\qquad x \in \{0,1\}^n \end{aligned}$$

- C is now a matrix of size $p \times n$.
- As before $\mathcal{X} = \{x \in \{0,1\}^n \ : \ Ax \geq b\}$
  - feasible set in *decision space*
- Now, $\mathcal{Y} = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \mathcal{X}\}$
  - feasible set in *objective space*
- This is a Multi-Objective Combinatorial Optimization (MOCO) problem

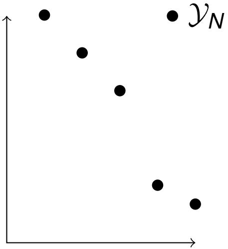
Branch-and-bound for MOCO problems - Optimality

We consider the Pareto principle

- For $y^1, y^2 \in \mathbb{R}^p$, $y^1 \leqq y^2$ if $y_i^1 \leq y_i^2$ for $i = 1, ..., p$
- For $y^1, y^2 \in \mathbb{R}^p$, $y^1 \leqslant y^2$ if $y^1 \leqq y^2$ and $y^1 \neq y^2$
  - If $y^1 \leqslant y^2$, y^1 *dominates* y^2
- For $y^1, y^2 \in \mathbb{R}^p$, $y^1 < y^2$ if $y_i^1 < y_i^2$ for $i = 1, ..., p$
- For a set $\mathcal{Y} \subseteq \mathbb{R}^p$, we let $\mathcal{Y}_N = \{y \in \mathcal{Y} \ : \nexists \tilde{y} \in \mathcal{Y} \text{ with } \tilde{y} \leqslant y\}$
- If $x \in \mathcal{X}$ and $Cx \in \mathcal{Y}_N$, then x is *Pareto optimal*
- The set of Pareto optimal solutions is $\mathcal{X}_E$ and $\mathcal{Y}_N = C\mathcal{X}_E$

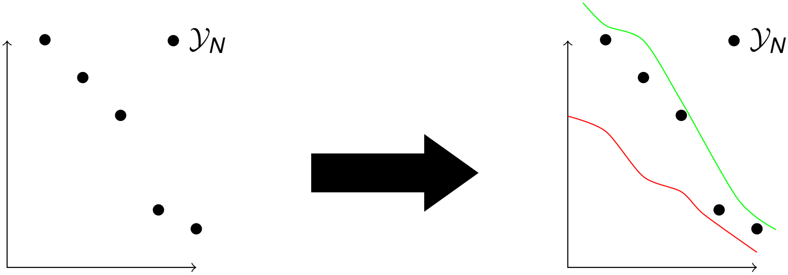
Branch-and-bound for MOCO problems - bounding

- The *solution* is now a tuple of *sets*, $(\mathcal{Y}_N, \bar{\mathcal{X}}_E)$, of non-dominated vectors a pre-images
  - $\bar{\mathcal{X}}_E \subseteq \mathcal{X}_E$, such that $\mathcal{Y}_N = C\bar{\mathcal{X}}_E$
- Hence, bounding the solution, is done using upper and lower bound *sets*



Branch-and-bound for MOCO problems - bounding

- The *solution* is now a tuple of *sets*, $(\mathcal{Y}_N, \bar{\mathcal{X}}_E)$, of non-dominated vectors a pre-images
  - $\bar{\mathcal{X}}_E \subseteq \mathcal{X}_E$, such that $\mathcal{Y}_N = C\bar{\mathcal{X}}_E$
- Hence, bounding the solution, is done using upper and lower bound *sets*



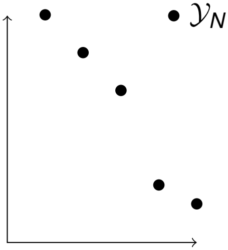
Branch-and-bound for MOCO problems - lower bound sets

- Formally, we use the definition of bound sets by Ehrgott and Gandibleux (2007)

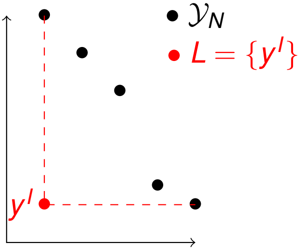
Lower bound set Ehrgott and Gandibleux (2007)

A lower bound set $L \subseteq \mathbb{R}^p$ for a set $\mathcal{Y} \subseteq \mathbb{R}^p$ is an $\mathbb{R}^p_{\geqq}$-closed and $\mathbb{R}^p_{\geqq}$-bounded set such that $\mathcal{Y} \subseteq L + \mathbb{R}^p_{\geqq}$ and $L \subseteq (L + \mathbb{R}^p_{\geqq})_N$

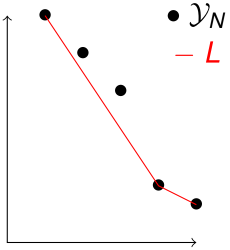
Branch-and-bound for MOCO problems - lower bound sets



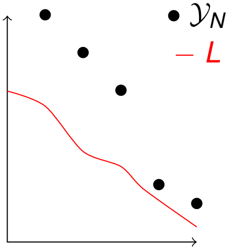
Branch-and-bound for MOCO problems - lower bound sets



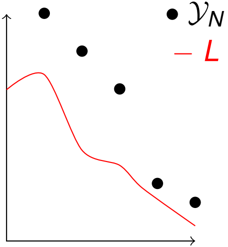
Branch-and-bound for MOCO problems - lower bound sets



Branch-and-bound for MOCO problems - lower bound sets



Branch-and-bound for MOCO problems - lower bound sets



Not a lower bound set

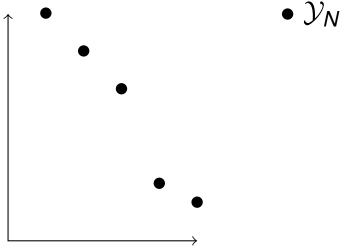
Branch-and-bound for MOCO problems - Upper bound sets

Upper bound sets Ehrgott and Gandibleux (2007)

An upper bound set U for a set $\mathcal{Y} \subseteq \mathbb{R}^p$ is an $\mathbb{R}^p_{\geqq}$-closed and $\mathbb{R}^p_{\geqq}$-bounded set such that $\mathcal{Y} \subseteq \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right]$ and $U \subseteq (U + \mathbb{R}^p_{\geqq})_N$.

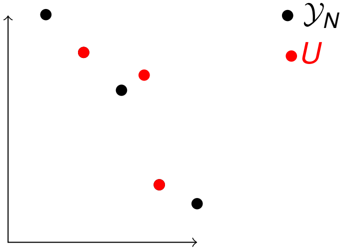
Branch-and-bound for MOCO problems - Upper bound sets

- A natural choice, often used, is images of feasible solutions
- Letting $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$



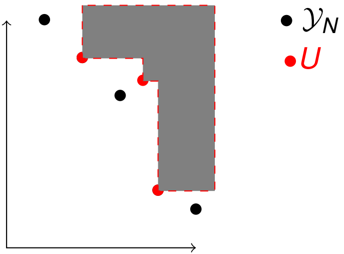
Branch-and-bound for MOCO problems - Upper bound sets

- A natural choice, often used, is images of feasible solutions
- Letting $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$



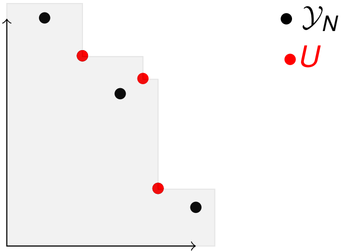
Branch-and-bound for MOCO problems - Upper bound sets

- A natural choice, often used, is images of feasible solutions
- Letting $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$



Branch-and-bound for MOCO problems - Upper bound sets

- A natural choice, often used, is images of feasible solutions
- Letting $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \; : \; y = Cx, \; x \in \bar{\mathcal{X}}\}$



$$\mathcal{A}(U) = \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right]$$

Local upper bound sets

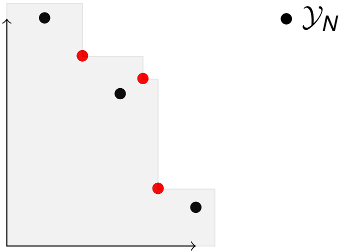
Local upper bound sets Klamroth, Lacour, and Vanderpooten (2015)

Let $\mathcal{C}(u) := u - \mathbb{R}^p_{\geqq}$ be a *search cone* with respect to $u \in \mathbb{R}^p$.
The set of local upper bounds with respect to U, $\mathcal{N}(U)$, is a unique discrete set of points in $\mathbb{R}^p$ satisfying

1. $\mathcal{A}(U) = \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right] = \bigcup_{u \in \mathcal{N}(U)} \mathcal{C}(u)$.
2. $\mathcal{N}(U)$ is minimal: There is no two $u^1, u^2 \in \mathcal{N}(U)$, with $u^1 \neq u^2$ such that $\mathcal{C}(u^1) \subseteq \mathcal{C}(u^2)$.

Branch-and-bound for MOCO problems - Local upper bound sets

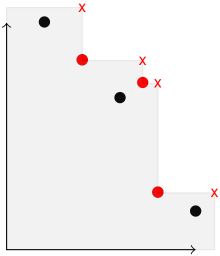
- Let $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$
- In $\mathbb{R}^2$ the set $\mathcal{N}(U)$ is the set of local Nadir points



$$\mathcal{A}(U) = \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right]$$

Branch-and-bound for MOCO problems - Local upper bound sets

- Let $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$
- In $\mathbb{R}^2$ the set $\mathcal{N}(U)$ is the set of local Nadir points

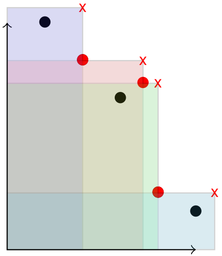


$\bullet$ $\mathcal{Y}_N$
x $\mathcal{N}(U)$

$$\mathcal{A}(U) = \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right]$$

Branch-and-bound for MOCO problems - Local upper bound sets

- Let $\bar{\mathcal{X}} \subseteq \mathcal{X}$, $U = \{y \in \mathbb{R}^p \ : \ y = Cx, \ x \in \bar{\mathcal{X}}\}$
- In $\mathbb{R}^2$ the set $\mathcal{N}(U)$ is the set of local Nadir points



$\bullet\ \mathcal{Y}_N$

$x\ \mathcal{N}(U)$

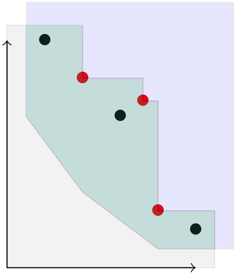
$$\mathcal{A}(U) = \text{cl}\left[\mathbb{R}^p \setminus \left(U + \mathbb{R}^p_{\geqq}\right)\right] = \bigcup_{u \in \mathcal{N}(U)} \mathcal{C}(u)$$

Branch-and-bound for MOCO problems - The search region

The search region Forget, G., et al. (2022)

Given an upper bound set U and a lower bound set L for $\mathcal{Y}_N$, the search region is given by

$$\mathcal{A}(U, L) = \left(L + \mathbb{R}^p_{\geqq}\right) \cap \left(\mathcal{N}(U) - \mathbb{R}^p_{>}\right)$$



- $\mathcal{Y}_N$
- U

Branch-and-bound for MOCO problems - recap of notation

η: A branching node

$\mathcal{X}(\eta)$: Set of feasible solutions in decision space at node η

$\mathcal{Y}_N(\eta)$: Set of feasible solutions in outcome space at node η: $\mathcal{Y}_N(\eta) = (C\mathcal{X}(\eta))_N$

U: (Global) upper bound set, i.e. $\mathcal{Y}_N \subseteq \mathcal{N}(U) - \mathbb{R}^p_{\geqq}$

$L(\eta)$: A lower bound set for $\mathcal{Y}_N(\eta)$

Convention 1: If $\mathcal{X}(\eta) = \emptyset$ then $L(\eta) = \emptyset$.

Convention 2: We update U *before* calculating $\mathcal{A}(U, L(\eta))$.

Branch-and-bound for MOCO problems - The search region

MOCO Branch-and-bound algorithm

Step 0: Initialize η_0 with $\mathcal{X}(\eta_0) = \mathcal{X}$. Set $T = \{\eta_0\}$, $U = \{\}$ and $\bar{\mathcal{X}}_E = \{\}$.

Step 1: If $T = \emptyset$, return $(\bar{\mathcal{X}}_E, U)$ as optimal.
Else, pop a subproblem η from T.

Step 2: Obtain a lower bound *set* $L(\eta)$ of the subproblem η. *If* any feasible solutions $x^i \in \mathcal{X}(\eta)$ was found, update $(\bar{\mathcal{X}}_E, U)$. Go back to Step 3.

Step 3: If $\mathcal{A}(U, L) = \emptyset$ go back to Step 1. Else continue to Step 4.

Step 4: Split η into smaller problems: $\eta_i \subset \eta$ and $\bigcup_i \eta_i = \eta$. Set $T = T \cup (\bigcup_i \eta_i)$ and go back to Step 1.

B&B for MOCO problems - Is $\mathcal{A}(U, L) = \emptyset$?

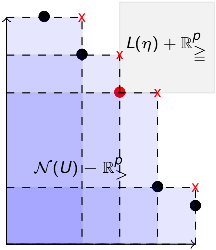
- We can prune a branching node, η, if $\mathcal{A}(U, L(\eta)) = \emptyset$
- We can use the same three rules as for single objective optimization

Infeasibility: If $\mathcal{X}(\eta) = \emptyset \Rightarrow \mathcal{A}(U, L(\eta)) = \emptyset$.

Optimality: If $L(\eta) = \{y\} \in \mathcal{Y}(\eta)$ then $\mathcal{A}(U, L(\eta)) = \emptyset$.

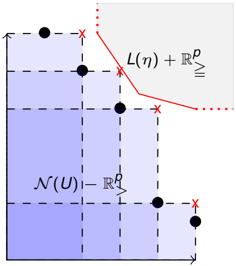
Bound: If $\left(L(\eta) + \mathbb{R}^p_{\geqq}\right) \cap \mathcal{N}(U) = \emptyset$, then $\mathcal{A}(U, L(\eta)) = \emptyset$.

Branch-and-bound for MOCO problems - Prune by optimality



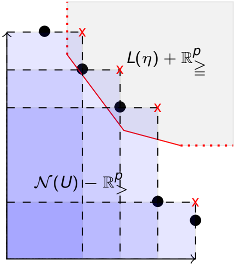
- U
- $L(\eta) \in \mathcal{Y}_N(\eta)$

Branch-and-bound for MOCO problems - Prune by bound



- U
- $L(\eta)$

Branch-and-bound for MOCO problems - Cannot prune



- U
- $L(\eta)$

Outline

1. Tree search for single objective optimization problems
2. Tree search for multi-objective optimization problems
   - Bound sets
   - Pruning
3. Components of a branch and bound algorithm for MOOPs
   - Selecting the next node to process
   - Obtaining a lower bound set
   - Obtaining an upper bound set
   - Branching by splitting a node
   - Speed-up techniques
4. Possible future directions
5. Hands on experience

Node selection strategies

Depth first: Pick the last node that was added to T

- Used in fx Kiziltan and Yucaoğlu (1983), Mavrotas and Diakoulaki (2005), Vincent et al. (2013), Przybylski (Yesterday)

Breadth first: Pick the active node closest to the root node

- Used in fx Visée et al. (1998), Parragh and Tricoire (2019)

Dynamic rules:

Best bound LP: Stidsen, Andersen, and Dammann (2014) and G, . Nielsen, and Ehrgott (2019)

Best bound ϵ-indicator: Jesus et al. (2021)

Best bound Hausdorff: Adelgren and Gupte (2022), Forget and Parragh (2024)

Best bound hyper volume: Bauß and Stiglmayr (2024)

Best bound scalarized: Forget and Parragh (2024)

Obtaining a lower bound set

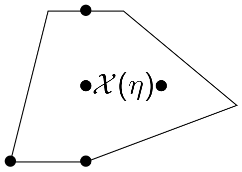
Utopian point	Klein and Hannan (1982), Kiziltan and Yucaoğlu (1983)
Ideal point	Ramos et al. (1998)
Linear relaxation	Belotti, Soylu, and Wiecek (2013), G, . Nielsen, and Ehrgott (2019), Adelgren and Gupte (2022)
Convex relaxation	Parragh and Tricoire (2019)
Non-convex relaxation	Bauß and Stiglmayr (2024)
Implicit	De Santis et al. (2020), G, . Nielsen, and Ehrgott (2019)

Upper bound set

- Use feasible solutions
- Keep track of local upper bounds/search cones (Dächert, Klamroth, et al. 2017)
- Solutions found at intermediate nodes

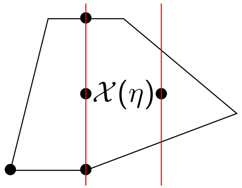
Splitting/Branching

- Splitting in decision space
- Splitting in objective space



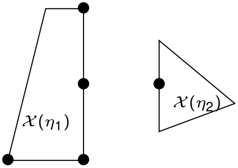
Splitting/Branching

- Splitting in decision space
- Splitting in objective space



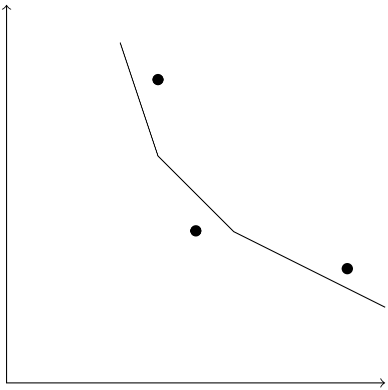
Splitting/Branching

- Splitting in decision space
- Splitting in objective space



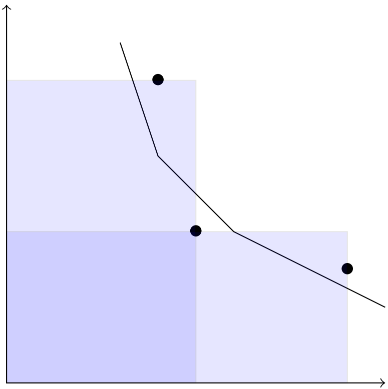
Splitting/Branching

- Splitting in decision space
- Splitting in objective space



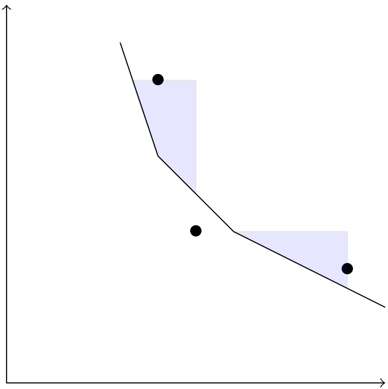
Splitting/Branching

- Splitting in decision space
- Splitting in objective space



Splitting/Branching

- Splitting in decision space
- Splitting in objective space



Suggestions for speed-ups

We may generally look into two directions for speed-ups

1. Process each node faster
2. Enumerate fewer nodes

Each node faster

Hyperplanes	Stidsen, Andersen, and Dammann (2014)
Updating	G, . Nielsen, and Ehrgott (2019)
Warm starting	Forget, G, and L. Nielsen (2022)
Separation test	Belotti, Soylu, and Wiecek (2013)
Probing	Forget and Parragh (2024)
Fast reoptimization	Przybylski (Yesterday)

Fewer nodes

Stronger upper bounds fast

Scalarized IPs	Parragh and Tricoire (2019), Bauß and Stiglmayr (2024)
Heuristics	Soylu (2015), An et al. (2024)
Managing UBs	Dächert and Klamroth (2015), Dächert, Klamroth, et al. (2017), Adelgren and Gupte (2022)
Node selection	Forget, G., et al. (2022), Bauß and Stiglmayr (2024), Jesus et al. (2021)

Fewer nodes

Stronger lower bounds

Cutting planes	G, . Nielsen, and Ehrgott (2019), Forget and Parragh (2024), Bökler et al. (2024)
Lagrangian	Brun et al. (2024), Dunbar, Sinha, and Schaefer (2024)
Preprocessing	Adelgren and Gupte (2022)

Outline

The largest speed-ups for $p = 1$

- Fast re-optimization of LPs
- Heuristics
- Cutting planes
- Preprocessing and probing

Necessary future directions

- Preprocessing of nodes
  - Bound and coefficient strengthening, variable elimination (Adelgren and Gupte 2022)
  - Clique generation, conflict graphs
- Probing
  - Variable fixing, dual information

Possible future directions

- Heuristics
  - MO-MILP based heuristics
  - Probably in parallel to main algorithm
- Cutting planes
  - Identical to $p = 1$ case
  - Compromise between LP relaxation and convex relaxation
- Stop! and resolve

Learning how to solve

- Learn how to select nodes
  - Start with depth first, switch to best bound (at the right time)
- Learn how to branch
  - Variable scores resembling reliability branching

Outline

Hands on experience

1. Navigate to https://github.com/SuneGadegaard/RAMOO2024
2. Read the `README.md` file
3. Download the four files
   - `BiObjectiveBnB.py`
   - `LowerBoundSets.py`
   - `helperStructures.py`
   - `main.py`
4. Download the files in the Instances folder.
5. In your favourite Python IDE, create a new project, and move the files to the project folder.
6. Solve the downloaded instances with different algorithm settings.

Achterberg, T., T. Koch, and A. Martin (2005). “Branching rules revisited”. In: *Operations Research Letters* 33.1, pp. 42–54.

Adelgren, N. and A. Gupte (2022). “Branch-and-bound for biobjective mixed-integer linear programming”. In: *INFORMS Journal on Computing* 34.2, pp. 909–933.

An, D. et al. (2024). “A matheuristic for tri-objective binary integer linear programming”. In: *Computers & Operations Research* 161, p. 106397.

Bauß, J. and M. Stiglmayr (2024). “Augmenting bi-objective branch and bound by scalarization-based information”. In: *Mathematical Methods of Operations Research*, pp. 1–37.

Belotti, P., B. Soylu, and M.M. Wiecek (2013). “A branch-and-bound algorithm for biobjective mixed-integer programs”. In: *Optimization Online*, pp. 1–29.

Bertsimas, D. and J.N. Tsitsiklis (1997). *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA.

Bökler, F. et al. (2024). “An outer approximation algorithm for generating the Edgeworth–Pareto hull of multi-objective mixed-integer linear programming problems”. In: *Mathematical Methods of Operations Research*, pp. 1–28.

Brun, M. et al. (2024). “On the strength of Lagrangian duality in multiobjective integer programming”. In: *Mathematical Programming*, pp. 1–33.

Dächert, K. and K. Klamroth (2015). “A linear bound on the number of scalarizations needed to solve discrete tricriteria optimization problems”. In: *Journal of Global Optimization* 61, pp. 643–676.

Dächert, K., K. Klamroth, et al. (2017). “Efficient computation of the search region in multi-objective optimization”. In: *European Journal of Operational Research* 260.3, pp. 841–855.

De Santis, M. et al. (2020). “Solving multiobjective mixed integer convex optimization problems”. In: *SIAM Journal on Optimization* 30.4, pp. 3122–3145.

Dunbar, A., S. Sinha, and A.J. Schaefer (2024). “Relaxations and duality for multiobjective integer programming”. In: *Mathematical Programming* 207.1, pp. 577–616.

Ehrgott, M. and X. Gandibleux (2007). "Bound sets for biobjective combinatorial optimization problems". In: *Computers & Operations Research* 34.9, pp. 2674–2694.

Forget, N., S.L. G, and L.R. Nielsen (2022). "Warm-starting lower bound set computations for branch-and-bound algorithms for multi objective integer linear programs". In: *European Journal of Operational Research* 302.3, pp. 909–924.

Forget, N., S.L. G., et al. (2022). "Branch-and-bound and objective branching with three or more objectives". In: *Computers & Operations Research* 148, p. 106012. ISSN: 0305-0548.

Forget, N. and S.N. Parragh (2024). "Enhancing branch-and-bound for multiobjective 0-1 programming". In: *INFORMS Journal on Computing* 36.1, pp. 285–304.

G, S.L., .R. Nielsen, and M. Ehrgott (2019). "Bi-objective branch-and-cut algorithms based on LP relaxation and bound sets". In: *INFORMS Journal on Computing* 31.4, pp. 790–804.

Jesus, A.D. et al. (2021). "On the design and anytime performance of indicator-based branch and bound for multi-objective combinatorial optimization”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 234–242.

Khalil, E. et al. (2016). “Learning to branch in mixed integer programming”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1.

Kiziltan, G. and E. Yucaoğlu (1983). “An algorithm for multiobjective zero-one linear programming”. In: *Management Science* 29.12, pp. 1444–1453.

Klamroth, K., R. Lacour, and . Vanderpooten (2015). “On the representation of the search region in multi-objective optimization”. In: *European Journal of Operational Research* 245.3, pp. 767–778. ISSN: 0377-2217.

Klein, D. and E. Hannan (1982). “An algorithm for the multiple objective integer linear programming problem”. In: *European journal of operational research* 9.4, pp. 378–385.

Mavrotas, G. and D. Diakoulaki (2005). “Multi-criteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming”. In: *Applied mathematics and computation* 171.1, pp. 53–71.

Parragh, S.N. and F. Tricoire (2019). “Branch-and-bound for bi-objective integer programming”. In: *INFORMS Journal on Computing* 31.4, pp. 805–822.

Ramos, R.M. et al. (1998). “The problem of the optimal biobjective spanning tree”. In: *European Journal of Operational Research* 111.3, pp. 617–628.

Soylu, B. (2015). “Heuristic approaches for biobjective mixed 0–1 integer linear programming problems”. In: *European Journal of Operational Research* 245.3, pp. 690–703.

Stidsen, T., K.A. Andersen, and B. Dammann (2014). “A branch and bound algorithm for a class of biobjective mixed integer programs”. In: *Management Science* 60.4, pp. 1009–1032.

Vincent, T. et al. (2013). “Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case”. In: *Computers & Operations Research* 40.1, pp. 498–509.

Visée, M. et al. (1998). “Two-phases method and branch and bound procedures to solve the bi–objective knapsack problem”. In: *Journal of Global Optimization* 12.2, pp. 139–155.