Multi-objective branch-and-cut algorithm and multi-modal traveling salesman problem

Nicolas Jozefowiez[1], Gilbert Laporte[2], Frédéric Semet[3]

1. LAAS-CNRS, INSA, Université de Toulouse, Toulouse, France, nicolas.jozefowiez@laas.fr

2. CIRRELT, HEC, Montréal, Canada, gilbert@crt.umontreal.ca

3. LAGIS, Ecole Centrale de Lille, Villeneuve d'Ascq, France, frederic.semet@ec-lille.fr

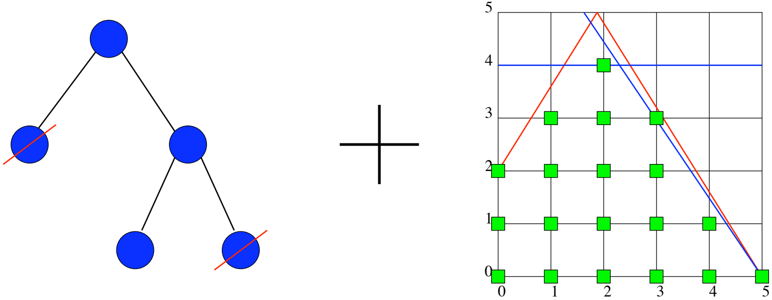
Outlines

- Branch-and-cut algorithm
- Multi-objective optimization
- A multi-objective branch-and-cut algorithm
- The multi-modal traveling salesman problem

Branch-and-cut algorithm

A method to solve integer programs:

$$\begin{aligned} \min \quad & cx \\ Ax \geq\ & b \\ x \geq\ & 0 \text{ and integer} \end{aligned}$$



Branch–and–bound algorithm

Cutting plane method

Explicit enumeration

Build an exploration tree → at each node, branching on a variable

Keep the best found feasible solution (the upper bound ub)

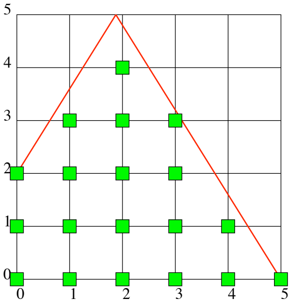
Implicit enumeration

At each node, a lower bound lb is computed

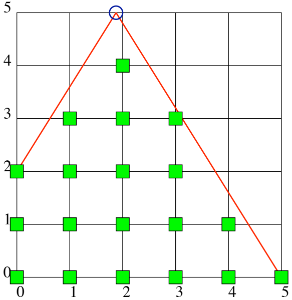
A node can be pruned if given the branching choice:

1. the problem is infeasible (pruned by infeasiblity)
2. the solution is feasible (pruned by optimality)
3. $lb \geq ub$ (pruned by bound)

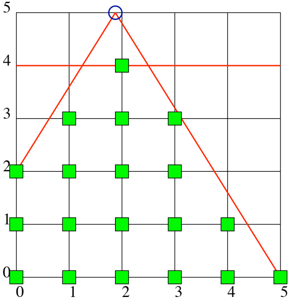
Cutting plane method



$$
\begin{aligned}
\min \quad & -1.00x_1 - 0.64x_2 \\
50x_1 + 31x_2 &\leq 250 \\
3x_1 - 2x_2 &\geq -4 \\
x_1, x_2 &\geq 0 \text{ and integer}
\end{aligned}
$$

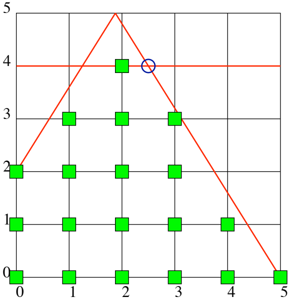


$$\begin{aligned} \min \quad & -1.00x_1 - 0.64x_2 \\ 50x_1 + 31x_2 & \leq 250 \\ 3x_1 - 2x_2 & \geq -4 \\ x_1, x_2 & \geq 0 \text{ and integer} \end{aligned}$$

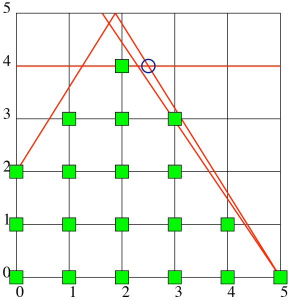


$$\begin{aligned} \min \quad & -1.00x_1 - 0.64x_2 \\ 50x_1 + 31x_2 &\leq 250 \\ 3x_1 - 2x_2 &\geq -4 \\ x_2 &\leq 4 \\ \\ x_1, x_2 &\geq 0 \text{ and integer} \end{aligned}$$

Cutting plane method

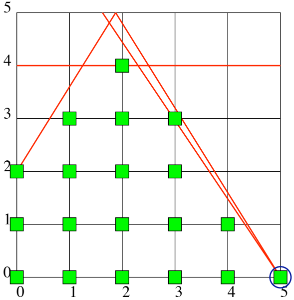


$$\begin{array}{rcl} & \min & -1.00x_1 - 0.64x_2 \\ 50x_1 + 31x_2 & \leq & 250 \\ 3x_1 - 2x_2 & \geq & -4 \\ x_2 & \leq & 4 \\ \\ x_1, x_2 & \geq & 0 \text{ and integer} \end{array}$$



$$\begin{aligned} \min \quad & -1.00x_1 - 0.64x_2 \\ 50x_1 + 31x_2 &\leq 250 \\ 3x_1 - 2x_2 &\geq -4 \\ x_2 &\leq 4 \\ 3x_1 + 2x_2 &\leq 15 \\ x_1, x_2 &\geq 0 \text{ and integer} \end{aligned}$$

Cutting plane method



$$\begin{aligned} \min \quad & -1.00x_1 - 0.64x_2 \\ 50x_1 + 31x_2 &\leq 250 \\ 3x_1 - 2x_2 &\geq -4 \\ x_2 &\leq 4 \\ 3x_1 + 2x_2 &\leq 15 \\ x_1, x_2 &\geq 0 \text{ and integer} \end{aligned}$$

A simple branch-and-cut algorithm

STEP 1 (Root of the tree)
Generate an initial upper bound ub
Define a first sub-problem
Insert the sub-problem in a list L

STEP 2 (Stopping criterion)
If $L = \emptyset$ then STOP, else choose a sub-problem from L and remove it from L

STEP 3 (Sub-problem solution)
Solve the sub-problem to obtain the lower bound lb

STEP 4 (Constraint generation)
if there is no solution or $lb \geq ub$ **then**
  Go to STEP 2.
else if the solution is integer **then**
  $ub \leftarrow lb$ and go to STEP 2.
else if violated constraints are identified **then**
  Add them to the model and go to STEP 3.
else
  Go to STEP 5.
end if

STEP 5 (Branching)
Branch on variable and introduce new sub-problems in L. Go to STEP 2.

Multi-objective optimization problem

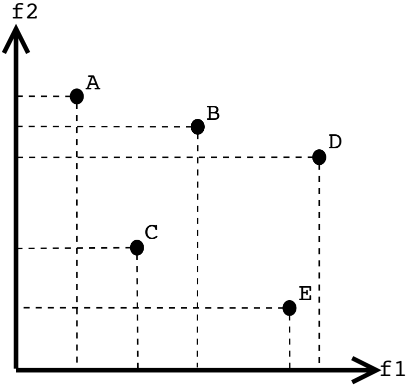
$$(PMO) = \begin{cases} min\ F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ s.t.\ x \in \Omega \end{cases}$$

with:

- $n \geq 2$: number of objectives
- $F = (f_1, f_2, \ldots, f_n)$: vector of functions to optimize
- $\Omega \subseteq \mathbb{R}^m$: set of feasible solutions
- $x = (x_1, x_2, \ldots, x_m) \in \Omega$: a feasible solution
- $\mathcal{Y} = F(\Omega)$: objective space
- $y = (y_1, y_2, \ldots, y_n) \in \mathcal{Y}$ avec $y_i = f_i(x)$: a point in the objective space

Pareto dominance relation

A solution x dominates ($\preceq$) a solution y if and only if
$\forall i \in \{1, \ldots, n\}, f_i(x) \leq f_i(y)$ and $\exists i \in \{1, \ldots, n\}$ such that $f_i(x) < f_i(y)$.



Exact algorithms for MOP

	$n = 2$	$n \geq 2$
Iteration	Two-Phase method PPM	K-PPM
Multi-objective method		[Sourd, Spanjaard, 2008] (*)

(*) does not work if the aggregated problem is NP-hard

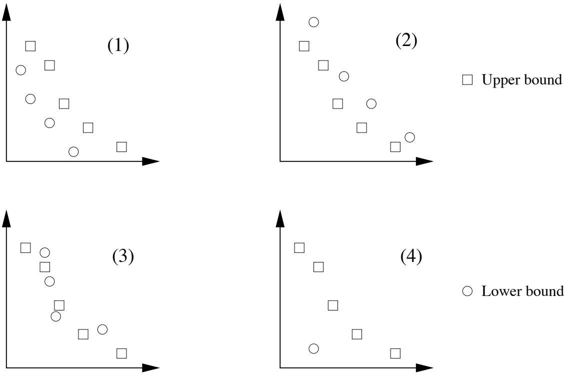
$\Rightarrow$ a multi-objective branch-and-cut algorithm for multi-objective integer programs

Lower bound $\Leftrightarrow$ multi-objective linear program

Possibility to use scalar techniques to solve it to optimality (or a subset that can be extended)

Upper bound = set of non-dominated solutions found during the search

Lower bound = set of non-dominated points in the objective space such that all feasible solutions are dominated by these points



A multi-objective branch-and-cut algorithm

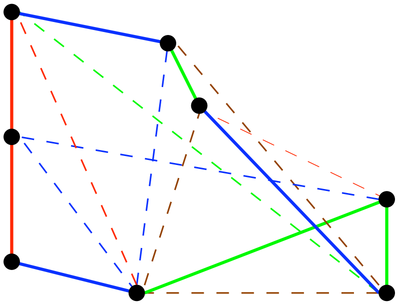
STEP 1 (Root of the tree)
Generate an initial upper bound ub
Define a first sub-problem
Insert the sub-problem in a list L

STEP 2 (Stopping criterion)
If $L = \emptyset$ then STOP, else choose a sub-problem from L and remove it from L

STEP 3 (Sub-problem solution)
Solve the sub-problem to obtain the lower bound lb

STEP 4 (Constraint generation)
Try to insert integer solutions from lb in ub
if $lb = \emptyset$ or $ub \preceq lb$ **then**
  Go to STEP 2.
else if violated constraints are identified for $\{x \in lb | \nexists y \in ub, y \preceq x\}$ **then**
  Add them to the model and go to STEP 3.
else
  Go to STEP 5.
end if

STEP 5 (Branching)
Branch on variable and introduce new sub-problems in L. Go to STEP 2.



Data:

$G = (V, E)$: an undirected valuated graph
C is a set of colors
Each $e \in E$ has a color $k \in C$

Goal:

Find a Hamiltonian cycle
Two objectives:

1. Minimize the total length of the cycle
2. Minimize the number of colors appearing on the cycle

Variables

$$x_e = \begin{cases} 1 & \text{if } e \in E \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

$$u_k = \begin{cases} 1 & \text{if } k \in C \text{ is used,} \\ 0 & \text{otherwise.} \end{cases}$$

Constants and notations

$$\forall e \in E,\ \delta(e) = k \in C \text{ the color of } e$$

$$\forall k \in C,\ \zeta(k) = \{e \in E | \delta(e) = k\}$$

$$\forall S \subset V,\ \omega(S) = \{e = (i,j) \in E | i \in S \text{ and } j \in V \setminus S\}$$

Integer program

Objective functions

$$\min \sum_{e \in E} c_e x_e$$

$$\min \sum_{k \in C} u_k$$

Constraints

$$\sum_{e \in \omega(\{i\})} x_e = 2 \qquad \forall i \in V$$

$$\sum_{e \in \omega(S)} x_e \geq 2 \qquad \forall S \subset V, 3 \leq |S| \leq |V| - 3$$

$$x_e \leq u_{\delta(e)} \qquad \forall e \in E$$

$$x_e \in \{0,1\} \qquad \forall e \in E$$

$$u_k \in \{0,1\} \qquad \forall k \in C$$

$$u_k \leq \sum_{e \in \zeta(k)} x_e \qquad \forall k \in C$$

$$\sum_{k \in C} \gamma_i^k u_k \geq 2 \qquad \forall i \in V$$

$$\sum_{k \in C} \lambda_k(S) u_k \geq 2 \qquad \forall S \in V, 3 \leq |S| \leq |V| - 3$$

with

$$\gamma_i^k = \begin{cases} 0 & \text{if } \nexists e \in \omega(\{i\}), e \in \zeta(k), \\ 1 & \text{if } \exists! e \in \omega(\{i\}), e \in \zeta(k), \\ 2 & \text{otherwise.} \end{cases} \qquad \lambda_k(S) = \begin{cases} 0 & \text{if } \nexists e \in \omega(S), e \in \zeta(k), \\ 1 & \text{if } \exists! e \in \omega(S), e \in \zeta(k), \\ 2 & \text{otherwise.} \end{cases}$$

Computation of the lower bound

Initial sub-problem :

$$\min \quad \sum_{e \in E} c_e x_e$$

$$\min \quad \sum_{k \in C} u_k$$

$$\begin{aligned}
\sum_{e \in \omega(\{i\})} x_e &= 2 && \forall i \in V \\
x_e &\leq u_{\delta(e)} && \forall e \in E \\
u_k &\leq \sum_{e \in \zeta(k)} x_e && \forall k \in C \\
\sum_{k \in C} \gamma_i^k u_k &\geq 2 && \forall i \in V
\end{aligned}$$

$$\begin{aligned}
0 \leq x_e &\leq 1 && \forall e \in E \\
0 \leq u_k &\leq 1 && \forall k \in C
\end{aligned}$$

Computation of the lower bound

Solve the following problem for different values of ϵ

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e + m \sum_{k \in C} u_k \\
& \sum_{e \in \omega(\{i\})} x_e = 2 && \forall i \in V \\
& x_e \leq u_{\delta(e)} && \forall e \in E \\
& u_k \leq \sum_{e \in \zeta(k)} x_e && \forall k \in C \\
& \sum_{k \in C} \gamma_i^k u_k \geq 2 && \forall i \in V \\
& \sum_{k \in C} u_k \leq \epsilon \\
& 0 \leq x_e \leq 1 && \forall e \in E \\
& 0 \leq u_k \leq 1 && \forall k \in C
\end{aligned}
$$

After founding non-dominated solution for a given ϵ, identify violated constraints and add them

Computation of the lower bound

ub is the upper bound. Set $L_{\text{tabu}} \leftarrow \emptyset$ and *continue* $\leftarrow$ TRUE

while *continue* is TRUE **do**
  continue $\leftarrow$ FALSE
  pruned $\leftarrow$ TRUE
  Set $\epsilon \leftarrow \alpha$ with α an integer such that $\alpha \notin L_{\text{tabu}}$ and $\nexists \beta \notin L_{\text{tabu}}$ such that $\alpha < \beta \leq |C|$.

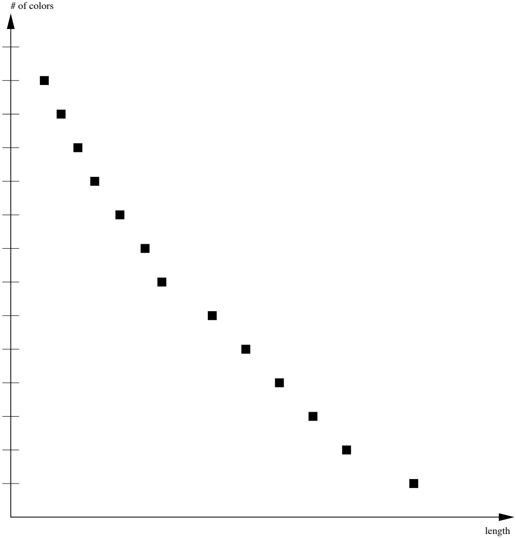
  while $\epsilon \neq 0$ **do**
    Solve the linear program. Let (x^*, u^*) be the optimal solution and l^* the length of the solution and o^* the number of colors used.

    if a solution is found **then**
      if the solution is feasible and integer or the solution is dominated by *ub* **then**
        if the solution is feasible and integer **then**
          Try to add it in *ub* and update *ub* if necessary
        end if
        $L_{\text{tabu}} \leftarrow \{\lceil o^* \rceil \dots \epsilon\}$
      else
        pruned $\leftarrow$ FALSE
        if constraints violated by (x^*, u^*) are identified **then**
          Stock them
        end if
      end if
    else
      $L_{\text{tabu}} \leftarrow \{1 \dots \epsilon\}$
    end if

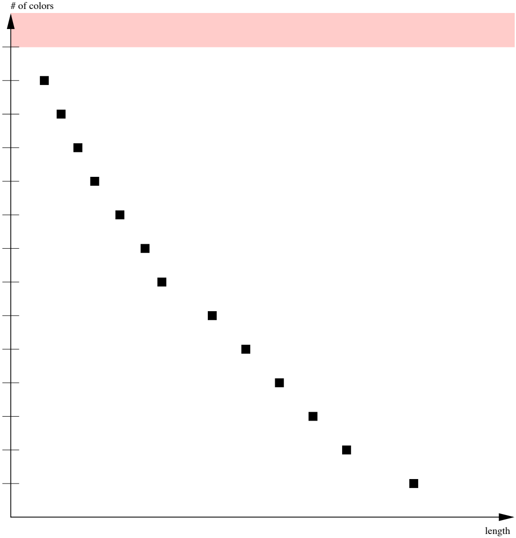
    Set $\epsilon \leftarrow \alpha$ with α an integer such that $\alpha \notin L_{\text{tabu}}$ and $\nexists \beta \notin L_{\text{tabu}}$ such that $\alpha < \beta < o^*$.
  end while

  if violated constraints have been found **then**
    Add them to the model
    continue $\leftarrow$ TRUE
  end if
end while

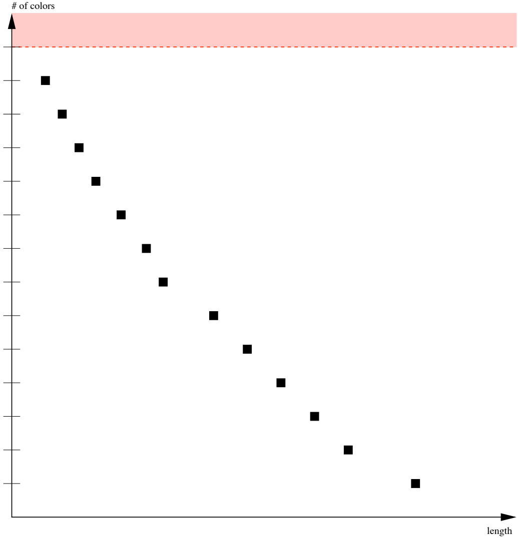
Computation of the lower bound



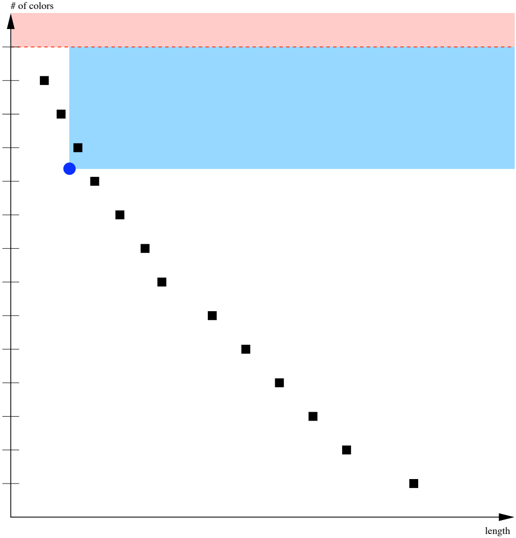
Computation of the lower bound



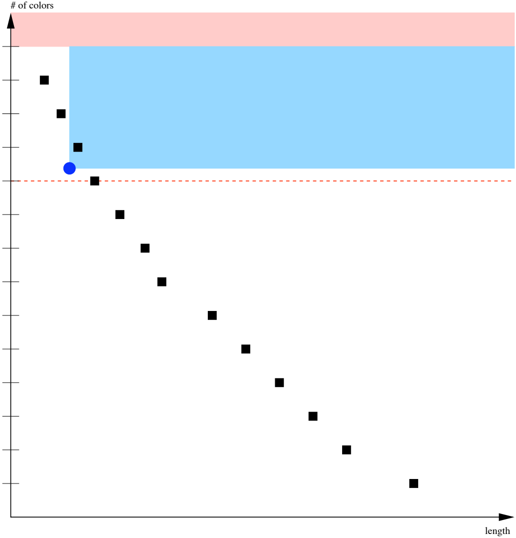
Computation of the lower bound



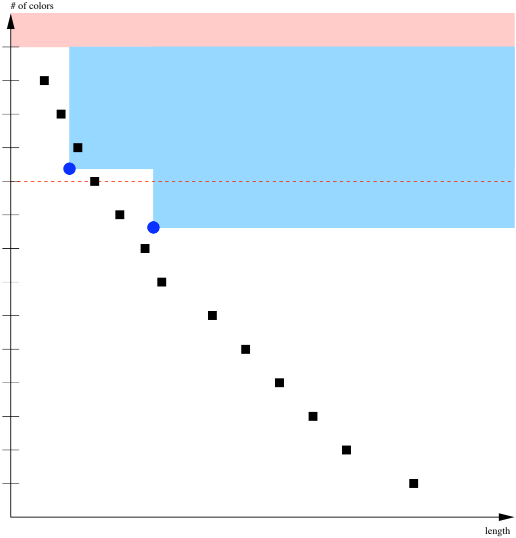
Computation of the lower bound



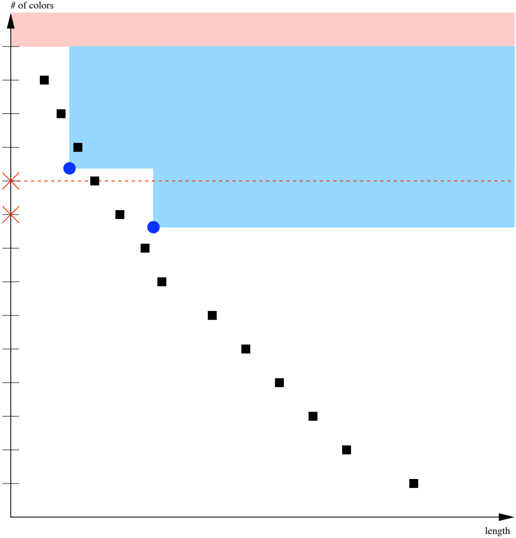
Computation of the lower bound



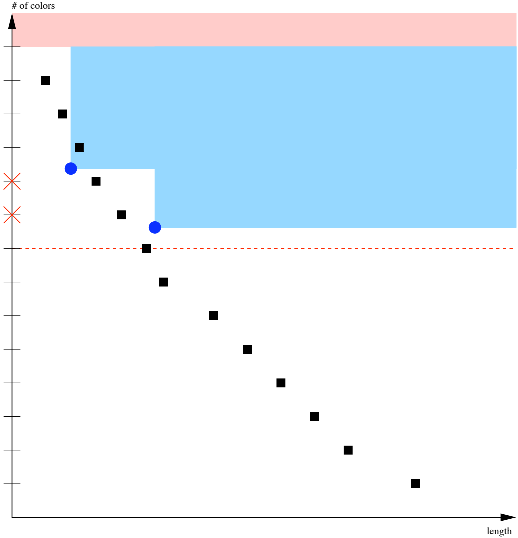
Computation of the lower bound



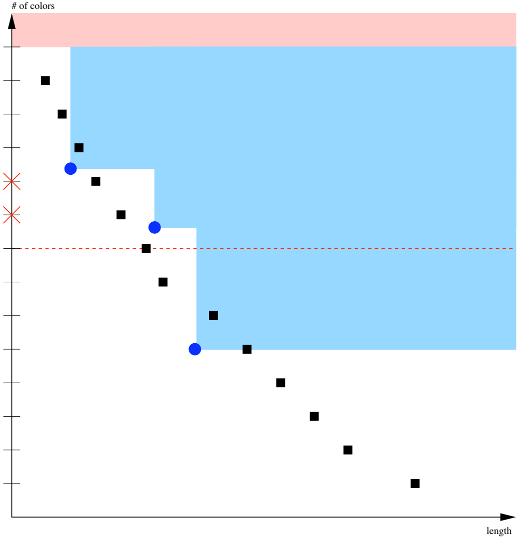
Computation of the lower bound



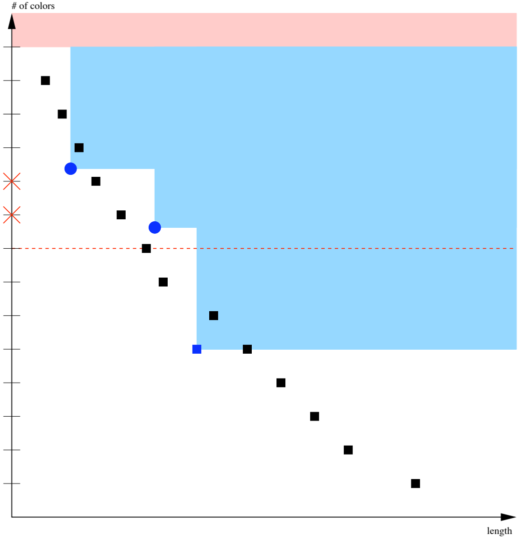
Computation of the lower bound



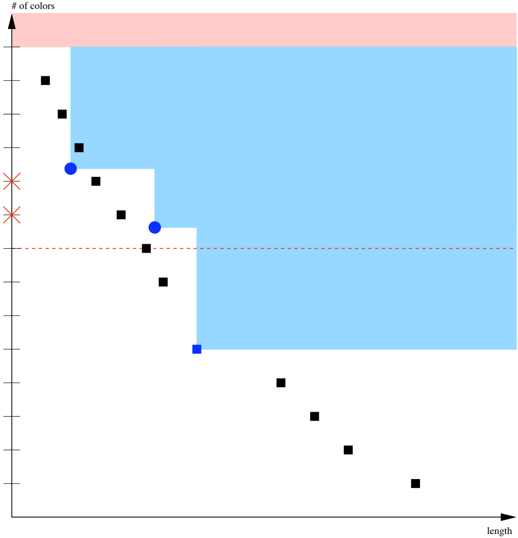
Computation of the lower bound



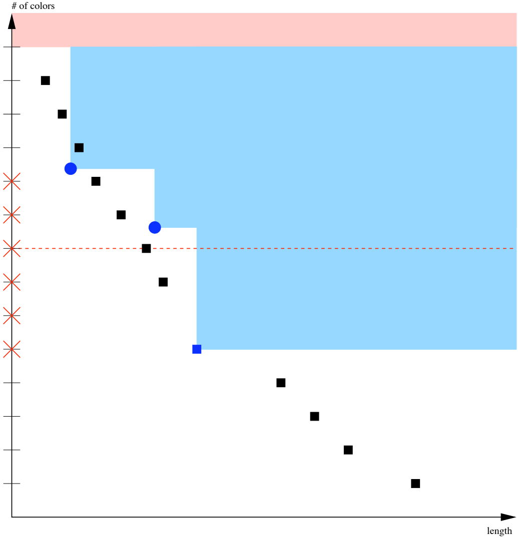
Computation of the lower bound



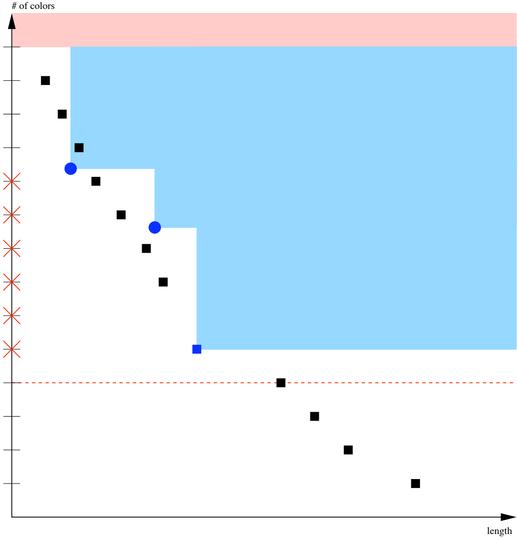
Computation of the lower bound



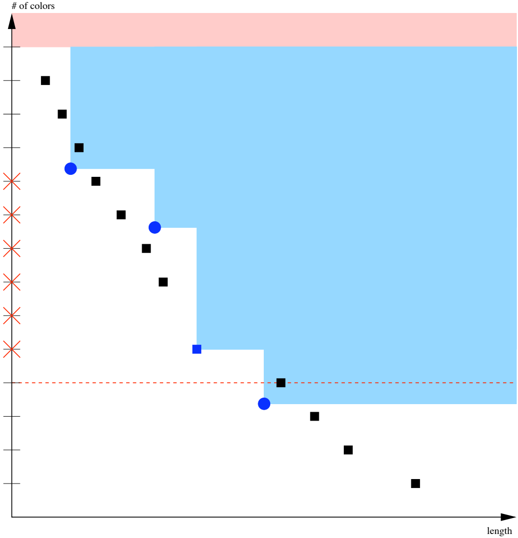
Computation of the lower bound



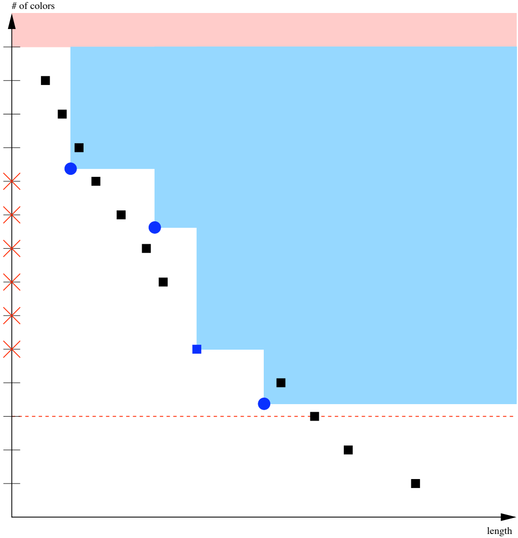
Computation of the lower bound



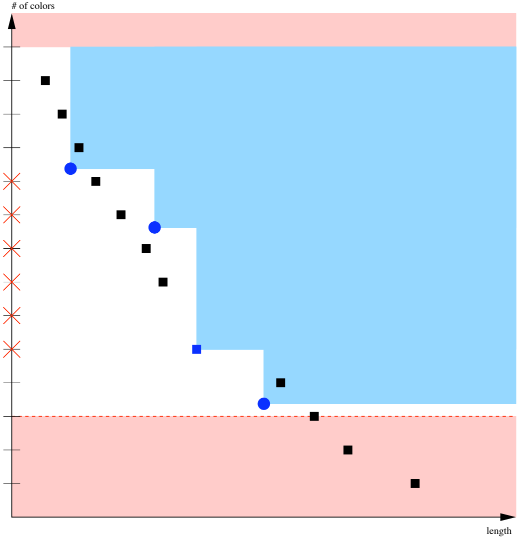
Computation of the lower bound



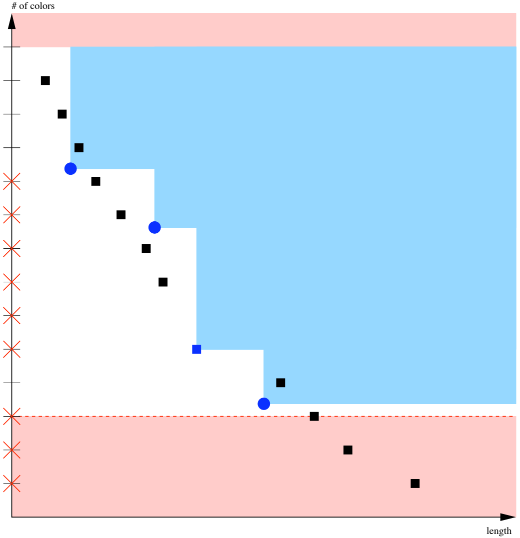
Computation of the lower bound



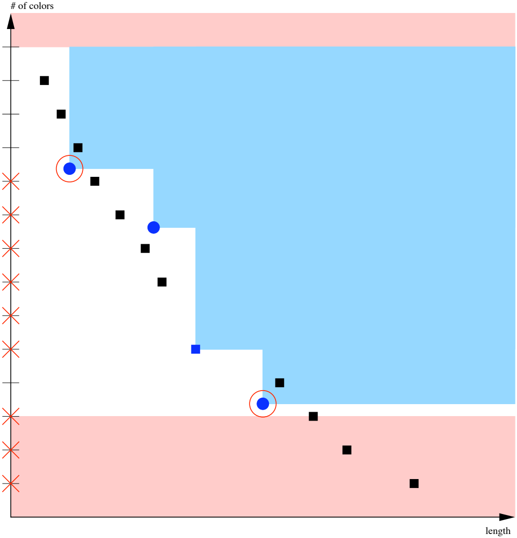
Computation of the lower bound



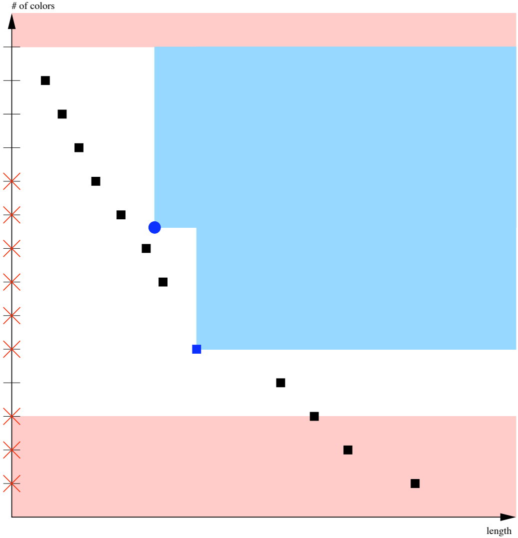
Computation of the lower bound



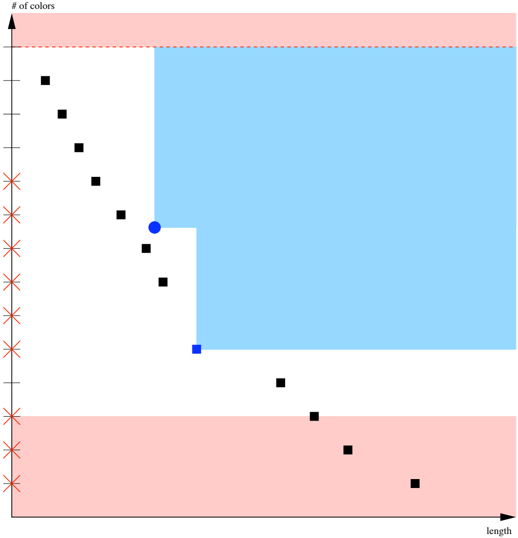
Computation of the lower bound



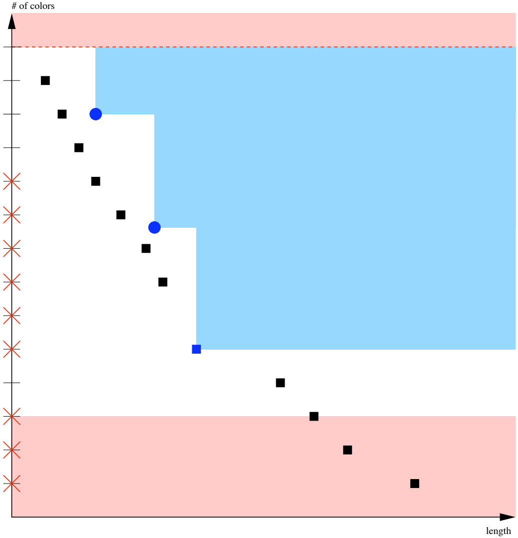
Computation of the lower bound



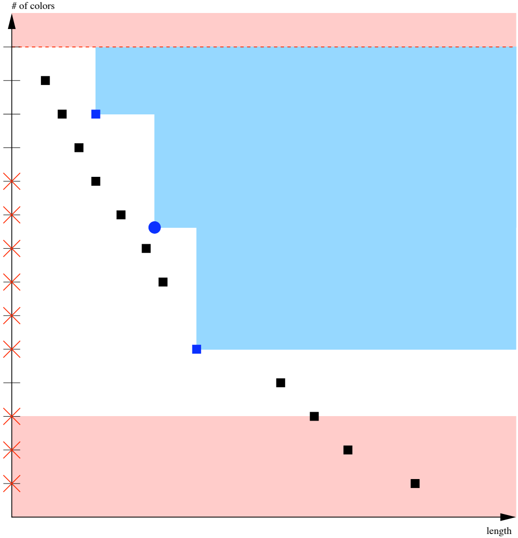
Computation of the lower bound



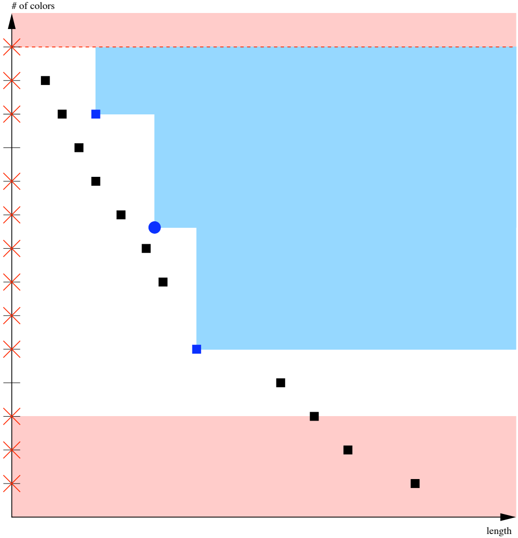
Computation of the lower bound



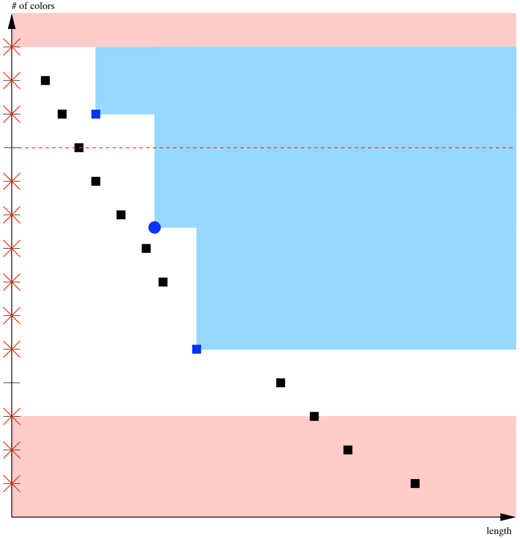
Computation of the lower bound



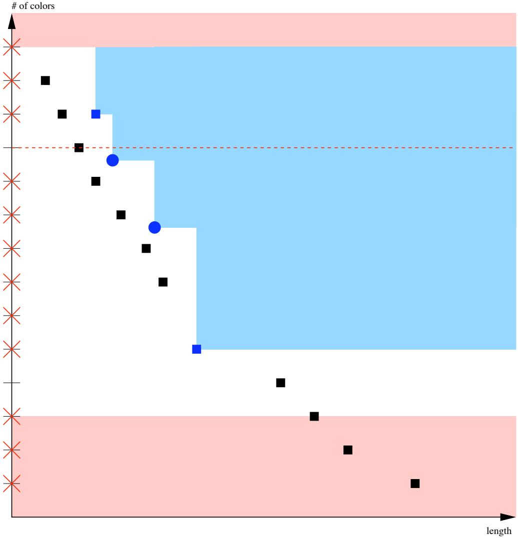
Computation of the lower bound



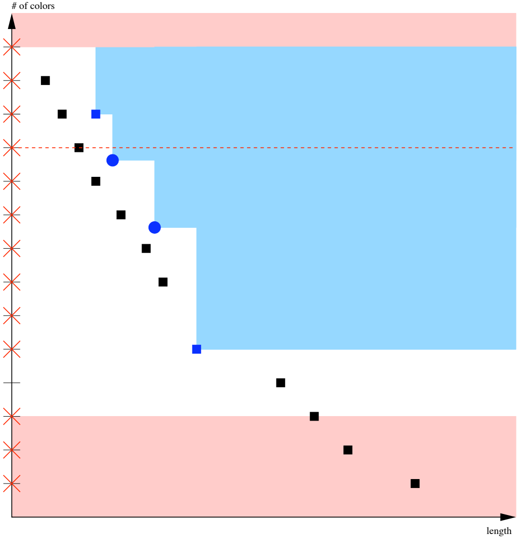
Computation of the lower bound



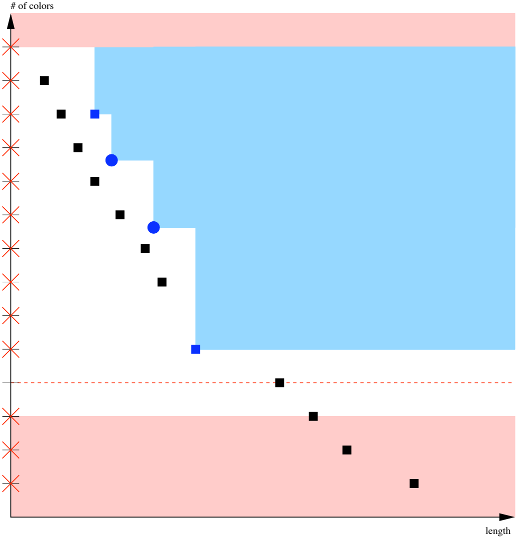
Computation of the lower bound



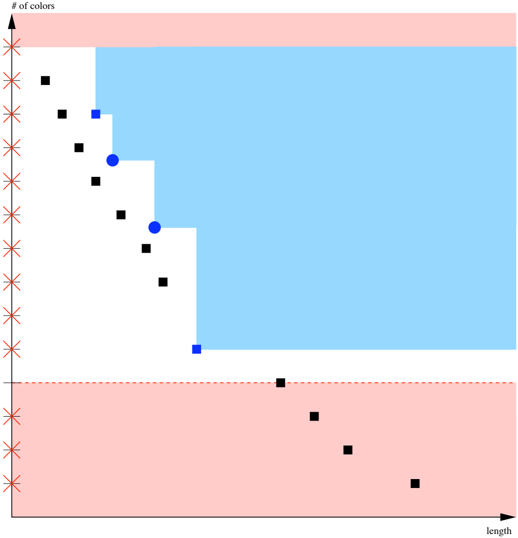
Computation of the lower bound



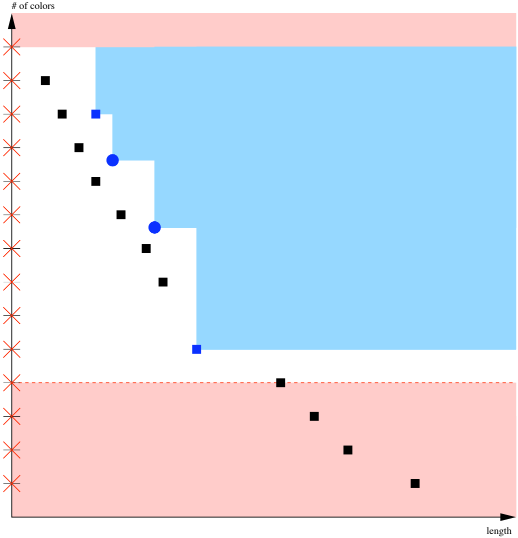
Computation of the lower bound



Computation of the lower bound



Computation of the lower bound



Constraint generation, cutting, and branching

Constraint generation

Connectivity constraints → min-cut problem

Call to a CONCORDE function [Padberg & Rinaldi, 1990]

Cutting

$\forall\epsilon$, the sub-problem is infeasible

$\forall\epsilon$, the solution is either feasible or dominated by ub

Branching

First on the u_k variables then on the x_e

Priority on the variable that is non integral for the most values of ϵ

Initial upper bound

ϵ-constraint method + MIP/CONCORDE

Computational results

$\|C\|$	$\|V\|$	#nodes	#u	#x	#cut	#Pareto	#Parub	#time
20	20	270.1	129.0	5.6	57.4	10.3	4.4	4.1
20	30	573.6	256.2	30.1	132.8	13.9	4.8	28.2
20	40	1002.9	410.2	90.8	225.6	15.9	5.3	100.8
20	50	1738.6	563.7	305.1	366.7	17.4	5.9	347.9
30	20	506.1	248.2	4.4	85.0	12.4	4.5	10.8
30	30	1284.6	592.5	49.3	193.4	16.4	4.5	96.7
30	40	2536.7	1084.2	183.6	352.3	18.8	4.2	484.3
30	50	5519.5	1880.9	878.4	590.8	21.7	5.0	1618.9
40	20	799.6	396.5	2.8	100.6	12.1	4.1	20.6
40	30	2241.6	1097.4	22.9	258.8	17.8	3.6	247.6
40	40	5684.9	2606.2	235.8	535.8	21.7	3.6	1852.6
40	50	14297.6	5321.8	1826.5	870.6	26.6	5.4	7200.4
50	20	804.0	400.4	1.1	98.8	12.4	4.6	21.4
50	30	3682.4	1806.6	34.1	336.4	18.8	3.7	541.4
50	40	10451.6	5053.2	172.1	748.6	23.9	4.2	5865.0
50	50	18975.5	8284.5	1202.8	1156.4	27.7	4.2	19942.8

Conclusions and perspectives

- Branch-and-cut algorithm able to solve a multi-objective problem in one run
- Identify new valid constraints $\rightarrow$ variables u_k
- Rules to choose on which variables to branch
- Progressive partition of the objective space