A combinatorial branch-and-bound algorithm for box search

Quentin Louveaux and Sébastien Mathieu

Department of Electrical Engineering and Computer Science
University of Liège, Belgium
Visiting Professor at the University of California, Davis

September 2013

A practical problem posed by the steel industry

A problem inspired by data mining

Galvanization consists in applying a protective zinc coating on steel.
This is a very complicated process depending on many parameters.

Idea of the problem : Based on the data gathered in the past, find a set of parameters that lead to an average high quality product.

Or, based on a set of runs, find the parameters that explain why these runs led to bad quality products.

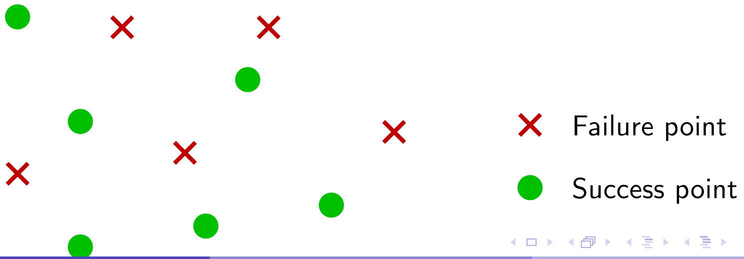
A mathematical statement

This can be transformed into a discrete optimization problem.

Data

- D-dimension normalized space (e.g. $[0,1]^D$)
- M points characterized by :
  - coordinates $\mathbf{x}^i$
  - classification :

$$c^i = \begin{cases} \text{success} = 1 \\ \quad \textit{or} \\ \text{failure} = \text{-}1 \end{cases} \qquad \text{or} \qquad y^i \in \mathbb{R}$$



A mathematical statement

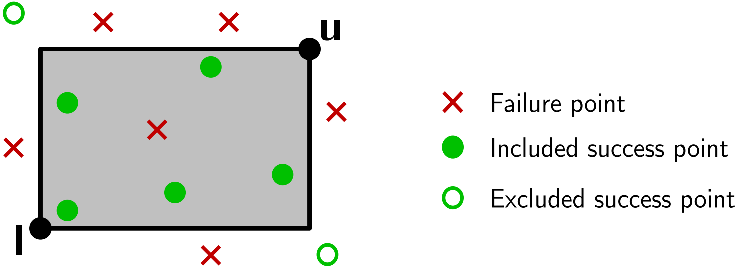
Definition

A box S is defined by

- two extreme points : $\mathbf{l}$ and $\mathbf{u}$

Implicitly it defines

- the sets of included success and failure points
- the sets of non-included success and failure points



A mathematical statement

Definition

The quality or value of a box is defined as the number of success points minus the number of failure points included in the box or more precisely

$$f(B) = \sum_{i \in B} c^i,$$

where c^i is the value of point i.

Problem statement

Find the box B with the maximal score

Alternative (easier) problem

Find the box B with the maximal number of success points without any failure points.
Tackled by [Eckstein, Hammer, Liu, Nediak, Simeone 2002]

A mathematical statement

Definition

The quality or value of a box is defined as the number of success points minus the number of failure points included in the box or more precisely

$$f(B) = \sum_{i \in B} c^i,$$

where c^i is the value of point i.

Problem statement

Find the box B with the maximal score

Alternative (easier) problem

Find the box B with the maximal number of success points without any failure points.
Tackled by [Eckstein, Hammer, Liu, Nediak, Simeone 2002]

A mathematical statement

Definition

The quality or value of a box is defined as the number of success points minus the number of failure points included in the box or more precisely

$$f(B) = \sum_{i \in B} c^i,$$

where c^i is the value of point i.

Problem statement

Find the box B with the maximal score

Alternative (easier) problem

Find the box B with the maximal number of success points without any failure points.
Tackled by [Eckstein, Hammer, Liu, Nediak, Simeone 2002]

Some comments

Why boxes ?

Because they are easy to interpret and to use for the operator !

Easy problem ?

Finding the best homogeneous box (without any failure) is NP-hard which implies that our problem is NP-hard.

Some comments

Why boxes ?

Because they are easy to interpret and to use for the operator !

Easy problem ?

Finding the best homogeneous box (without any failure) is NP-hard which implies that our problem is NP-hard.

A mixed-integer programming formulation

Variables

$l_d \in [0,1]^D$		The lower bounds of the box
$u_d \in [0,1]^D$		The upper bounds of the box
$z^i \in \{0,1\}$	$= 1$	if i belongs to the box
	$= 0$	if i does not belong to the box
$v^i_t \in \{0,1\}$	$= 1$	if failure point i satisfies the lower bound in dim t
	$= 0$	otherwise
$w^i_t \in \{0,1\}$	$= 1$	if failure point i satisfies the upper bound in dim t
	$= 0$	otherwise

A mixed-integer programming formulation

maximize $\sum_{i=1}^{M} c^i z^i$ subject to

for $d = 1, ..., D$:

$$l_d \le u_d \quad (1)$$

for i success point, $d = 1, ..., D$:

$$l_d \le x_d^i + (1 - z^i) \quad (2)$$

$$x_d^i z^i \le u_d \quad (3)$$

for i failure point, $d = 1, ..., D$:

$$v_d^i \ge (x_d^i - l_d) + \epsilon \quad (4)$$

$$w_d^i \ge (u_d - x_d^i) + \epsilon \quad (5)$$

for i failure point :

$$z^i \ge \sum_{d=1}^{D} (v_d^i + w_d^i) - 2D + 1/2 \quad (6)$$

A mixed-integer programming formulation

maximize $\sum_{i=1}^{M} c^i z^i$ subject to

for $d = 1, ..., D$:

$$l_d \leq u_d \tag{1}$$

for i success point, $d = 1, ..., D$:

$$l_d \leq x_d^i + (1 - z^i) \tag{2}$$

$$x_d^i z^i \leq u_d \tag{3}$$

for i failure point, $d = 1, ..., D$:

$$v_d^i \geq (x_d^i - l_d) + \epsilon \tag{4}$$

$$w_d^i \geq (u_d - x_d^i) + \epsilon \tag{5}$$

for i failure point :

$$z^i \geq \sum_{d=1}^{D} (v_d^i + w_d^i) - 2D + 1/2 \tag{6}$$

A mixed-integer programming formulation

maximize $\sum_{i=1}^{M} c^i z^i$ subject to

for $d = 1, ..., D$:

$$l_d \leq u_d \tag{1}$$

for i success point, $d = 1, ..., D$:

$$l_d \leq x_d^i + (1 - z^i) \tag{2}$$

$$x_d^i z^i \leq u_d \tag{3}$$

for i failure point, $d = 1, ..., D$:

$$v_d^i \geq (x_d^i - l_d) + \epsilon \tag{4}$$

$$w_d^i \geq (u_d - x_d^i) + \epsilon \tag{5}$$

for i failure point :

$$z^i \geq \sum_{d=1}^{D} (v_d^i + w_d^i) - 2D + 1/2 \tag{6}$$

A mixed-integer programming formulation

maximize $\sum_{i=1}^{M} c^i z^i$ subject to

for $d = 1, ..., D$:

$$l_d \leq u_d \tag{1}$$

for i success point, $d = 1, ..., D$:

$$l_d \leq x_d^i + (1 - z^i) \tag{2}$$

$$x_d^i z^i \leq u_d \tag{3}$$

for i failure point, $d = 1, ..., D$:

$$v_d^i \geq (x_d^i - l_d) + \epsilon \tag{4}$$

$$w_d^i \geq (u_d - x_d^i) + \epsilon \tag{5}$$

for i failure point :

$$z^i \geq \sum_{d=1}^{D} (v_d^i + w_d^i) - 2D + 1/2 \tag{6}$$

Is it a good formulation ?

Pro

A quite compact formulation :

- $M + 2D + 2|\text{failure}|$ variables
- $(2|\text{success}| + 1)D + |\text{failure}|(1 + 2D)$ constraints

Cons

- The linear relaxation is very weak
- The formulation is numerically unstable

Is it a good formulation ?

Pro

A quite compact formulation :

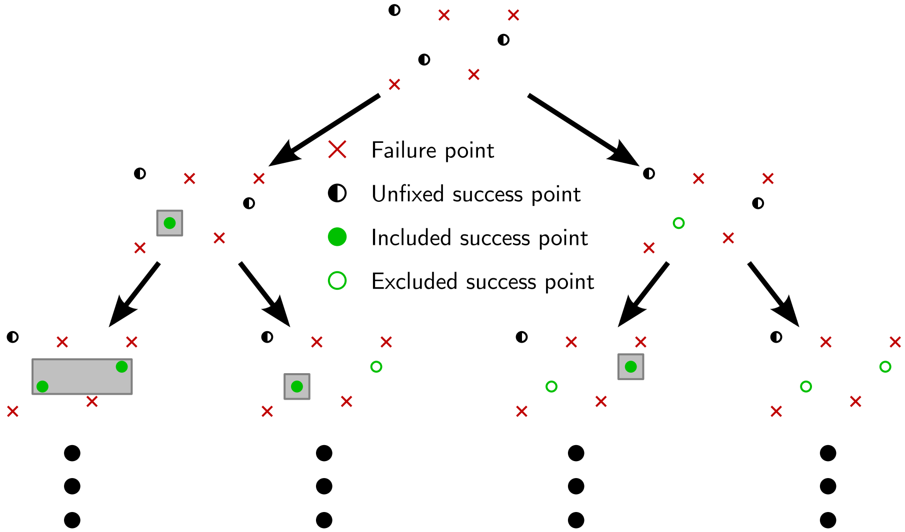
- $M + 2D + 2|\text{failure}|$ variables
- $(2|\text{success}| + 1)D + |\text{failure}|(1 + 2D)$ constraints

Cons

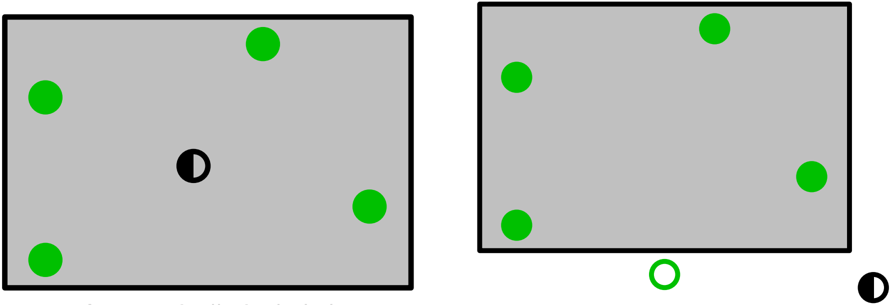
- The linear relaxation is very weak
- The formulation is numerically instable

A combinatorial branch-and-bound

We propose to branch on the decision :"is a point included or excluded of the box" ?



Inference of the inclusion/exclusion of points

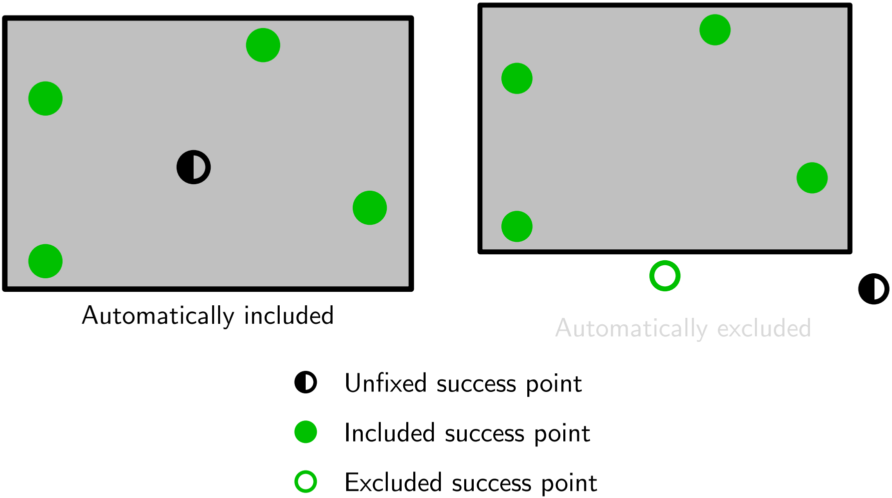


Automatically included

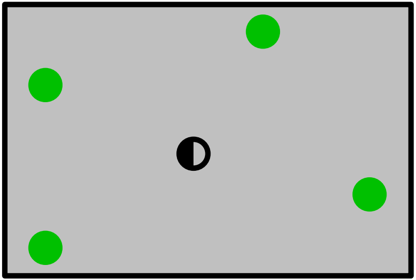
Automatically excluded

- Unfixed success point
- Included success point
- Excluded success point

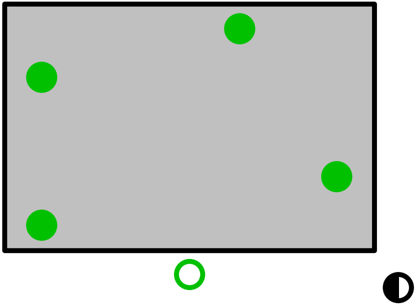
Inference of the inclusion/exclusion of points



Inference of the inclusion/exclusion of points



Automatically included



Automatically excluded

Unfixed success point

Included success point

Excluded success point

Finding a primal bound

Definition

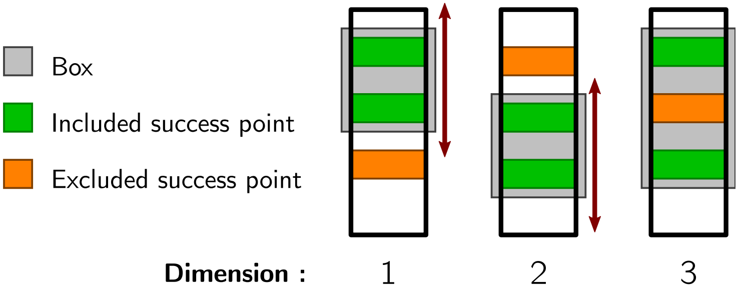
We define the operational box as a box with

$$l_t = \min_{i=1,\dots,M} x_t^i$$

$$u_t = \max_{i=1,\dots,M} x_t^i$$

Using the inference of included points, we can define a primal solution.

Finding an upper bound



The maximum number of success points that might be included is the size of the biggest **partition** (↕).

Definition

For each excluded point i and dimension t, we define Δ_t^i to be

- 0 if i is in the operational box for that specific dimension t
- the sum of the values of the positive points that are on the right side of i compared with the operational box for that specific dimension t

Lemma

Δ_t^i is an upper bound on the number of additional success points that can be included in the box if i is excluded in dimension t.

Lemma

For a specific fixing, the maximum number of additional success points that can be added to the box is given by

$$\min_{i \in mathcalE} \left\{ \max_{t=1,\ldots,D} \Delta_t^i \right\}$$

Definition

For each excluded point i and dimension t, we define Δ_t^i to be

- 0 if i is in the operational box for that specific dimension t
- the sum of the values of the positive points that are on the right side of i compared with the operational box for that specific dimension t

Lemma

Δ_t^i is an upper bound on the number of additional success points that can be included in the box if i is excluded in dimension t.

Lemma

For a specific fixing, the maximum number of additional success points that can be added to the box is given by

$$\min_{i \in mathcalE} \left\{ \max_{t=1,\dots,D} \Delta_t^i \right\}$$

Definition

For each excluded point i and dimension t, we define Δ_t^i to be

- 0 if i is in the operational box for that specific dimension t
- the sum of the values of the positive points that are on the right side of i compared with the operational box for that specific dimension t

Lemma

Δ_t^i is an upper bound on the number of additional success points that can be included in the box if i is excluded in dimension t.

Lemma

For a specific fixing, the maximum number of additional success points that can be added to the box is given by

$$\min_{i \in mathcalE} \left\{ \max_{t=1,\dots,D} \Delta_t^i \right\}$$

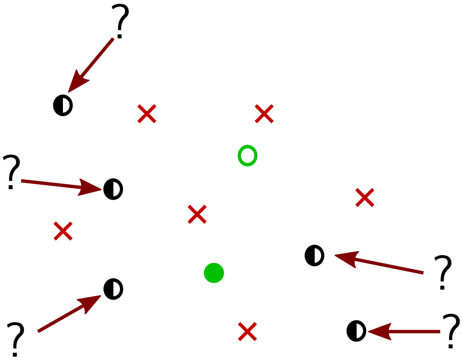
Branching strategies

What is branching in this case ?

Select the point that will be included/excluded.

$\Rightarrow$ **Aim :** compute the least number of nodes needed in the branch-and-bound tree

Which one should we choose ?

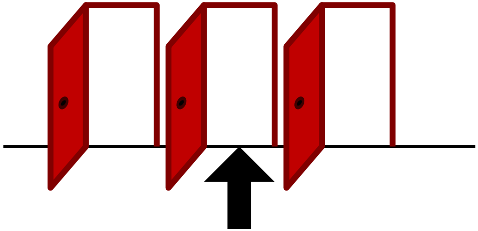


Branching candidates : unfixed points

Strong branching

Principle

- Try branching on every ◐
- Choose the greatest $\searrow$ of upper bound.



Results

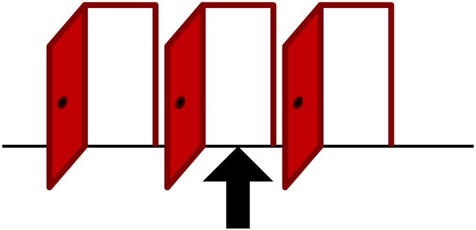
Low number of nodes, but high computational cost

$\Rightarrow$ like LP-based B&B

Strong branching

Principle

- Try branching on every ◐
- Choose the greatest ↘ of upper bound.



Results

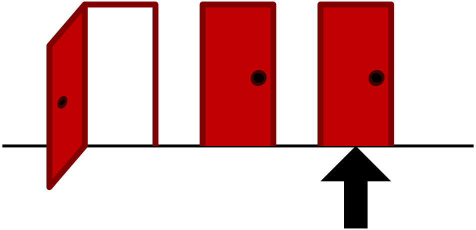
Low number of nodes, but high computational cost

$\Rightarrow$ like LP-based B&B

Reliability branching

Principle

- Introduced by *T. Achterberg, T. Koch, A. Martin* in 2004
- ◐ selected 5 times OK ⇒ globally OK
- ◐ branched less than 5 times ⇒ not **reliable**

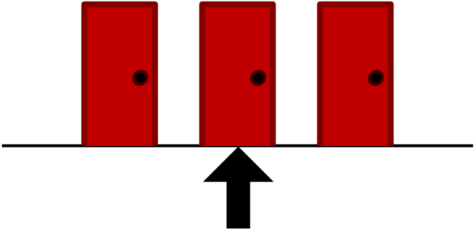


Scattered branching

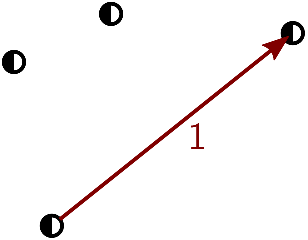
Principle

- Problem specific
- Build a *scattered order*
- Select the next in the line

  $\Rightarrow$ Cheap computational cost

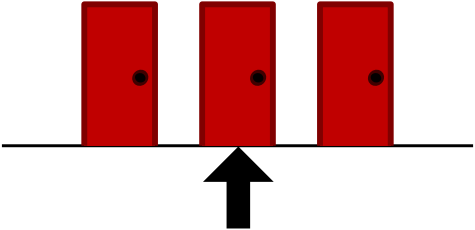


Scattered branching

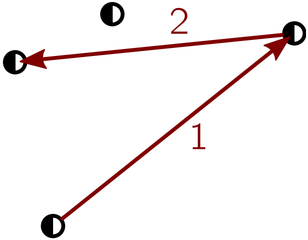


Principle

- Problem specific
- Build a *scattered order*
- Select the next in the line
  - $\Rightarrow$ Cheap computational cost

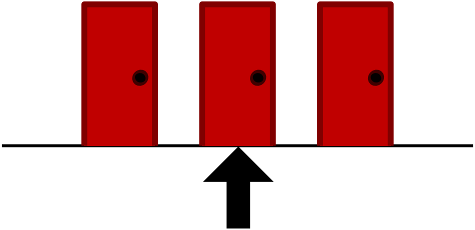


Scattered branching

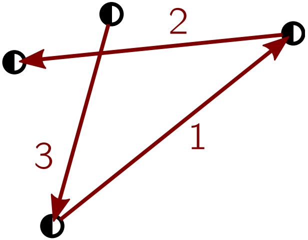


Principle

- Problem specific
- Build a *scattered order*
- Select the next in the line
  - $\Rightarrow$ Cheap computational cost

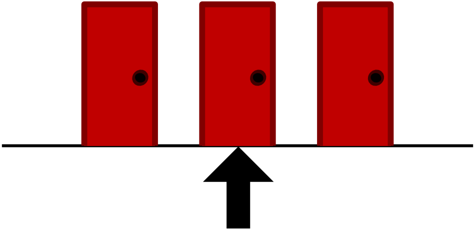


Scattered branching



Principle

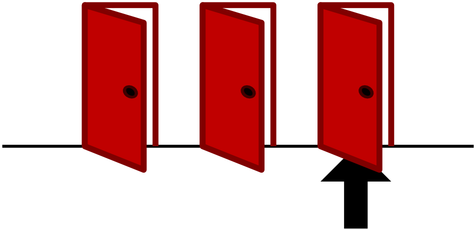
- Problem specific
- Build a *scattered order*
- Select the next in the line
  - $\Rightarrow$ Cheap computational cost



Least local branching

Principle

- $\forall$ ◐, compute an approximation of the $\searrow$ of upper bound.
  - Included : Compute the score of the new box
  - Excluded : Compute the partition for this point
- Branch on the greatest expected $\searrow$ of upper bound.



Computational results

D	M	PP-St *Gap [%]* *Time [s]* *Nodes*	PP-R *Gap [%]* *Time [s]* *Nodes*	PP-LL *Gap [%]* *Time [s]* *Nodes*	CPLEX *Gap [%]* *Time [s]* *Nodes*	BoP-R *Gap [%]* *Time [s]* *Nodes*	BoP-LL *Gap [%]* *Time [s]* *Nodes*
10	125	0 5.3 24	0 1.22 120	0 **1.07** 27	4.79 15.45 5	0 2.92 112	0 8.61 38
16	150	32.12 – 6865	0 1165.41 70547	0 **643.25** 10500	2.34 2706.25 585	0 952.89 23334	39.17 – 8314
20	150	28.58 – 7837	0 1849.1 107189	0 **1368.11** 21563	9.95 2235.64 433	0 2190.48 53885	37.77 – 6439
20	175	41.05 – 4414	40.92 – 159801	38.99 – 39609	**16.74** – 258	39.11 – 66184	38.89 – 4959
20	200	45.65 – 4294	40.84 – 155334	41.86 – 35540	**21.08** – 187	44.98 – 59910	48 – 5687

Conclusions

- The algorithm performs quite well on relatively small instances
- The algorithm is too slow for practical performances :
  We use a heuristic based on starting with the whole set and peeling the box in one dimension at each iteration.
- In practice, we like to define the box on few dimensions for it to be even more easy to interpret. $\rightarrow$ the problem is more complicated to formulate and solve