Un algorithme efficace pour une généralisation du Pooling Problem

Paulin Jacquot [1] Maxime Fender [2]

[1]EDF R&D, Inria Saclay and CMAP, École Polytechnique, France

[2]Artelys, France

24 Février 2017

ROADEF 2017, Metz, France

Problème - contaminants

- L'eau dans le réseau est contaminée avec un ensemble de contaminants $\mathcal{C}$.
- Chaque $c \in \mathcal{C}$ est présent avec une concentration ρ_c.

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\text{min}}_{ws}, \varphi^{\text{max}}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\text{min}}_{ws}, \varphi^{\text{max}}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\text{min}}_{wd}, \varphi^{\text{max}}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\text{max}}_{c,wd}$,

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},
- unités *processing* (PU), $\rightarrow$ flux $\varphi^{\text{in}} \in [\varphi^{\min}_{pu}, \varphi^{\max}_{pu}]$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,pu}$
  $\rightarrow$ rejettent de l'eau $\varphi^{\text{out}}_{pu} = \varphi^{\text{in}}_{pu} + \Delta_{pu}$,

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},
- unités *processing* (PU), $\rightarrow$ flux $\varphi^{\text{in}} \in [\varphi^{\min}_{pu}, \varphi^{\max}_{pu}]$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,pu}$
  $\rightarrow$ rejettent de l'eau $\varphi^{\text{out}}_{pu} = \varphi^{\text{in}}_{pu} + \Delta_{pu}$,
- unités *buffer* (BU), $\rightarrow$ stockent une quantité d'eau $\leq C_{bu}$, coût c_{bu},

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},
- unités *processing* (PU), $\rightarrow$ flux $\varphi^{\text{in}} \in [\varphi^{\min}_{pu}, \varphi^{\max}_{pu}]$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,pu}$
  $\rightarrow$ rejettent de l'eau $\varphi^{\text{out}}_{pu} = \varphi^{\text{in}}_{pu} + \Delta_{pu}$,
- unités *buffer* (BU), $\rightarrow$ stockent une quantité d'eau $\leq C_{bu}$, coût c_{bu},
- *tuyaux* (s, e) reliant une sortie d'unité s à une entrée e d'unité, avec un diamètre $d \in \{d_1, ..d_K\}$ et $\varphi_{s,e} \leq \pi(d/2)^2.$, coût c_d.

Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},
- unités *processing* (PU), $\rightarrow$ flux $\varphi^{\text{in}} \in [\varphi^{\min}_{pu}, \varphi^{\max}_{pu}]$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,pu}$
  $\rightarrow$ rejettent de l'eau $\varphi^{\text{out}}_{pu} = \varphi^{\text{in}}_{pu} + \Delta_{pu}$,
- unités *buffer* (BU), $\rightarrow$ stockent une quantité d'eau $\leq C_{bu}$, coût c_{bu},
- *tuyaux* (s, e) reliant une sortie d'unité s à une entrée e d'unité, avec un diamètre $d \in \{d_1, ..d_K\}$ et $\varphi_{s,e} \leq \pi(d/2)^2.$, coût c_d.

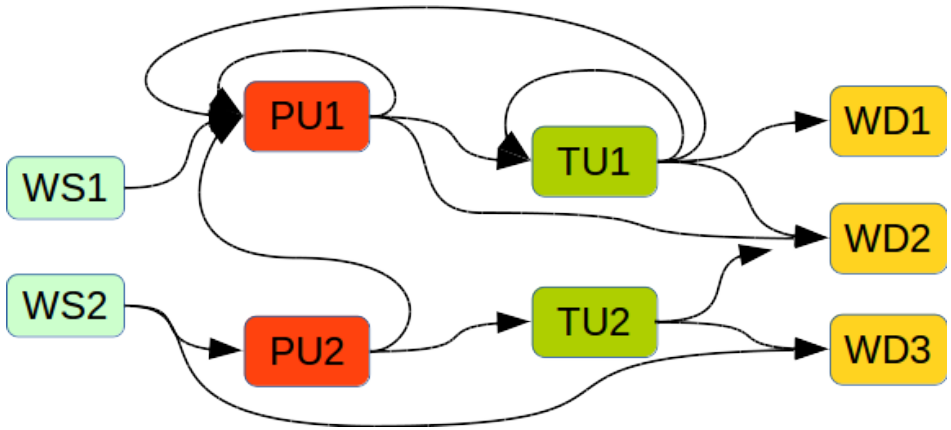
Problème - Réseau

Un réseau de traitement est composé de différentes unités :

- unités *sources* (WS), $\rightarrow$ débit $\varphi^{\text{out}} \in [\varphi^{\min}_{ws}, \varphi^{\max}_{ws}]$ et conc. $\rho_{ws,c}$ fixes,
- unités *demande* (WD), $\rightarrow$ flux , $\varphi^{\text{in}} \in [\varphi^{\min}_{wd}, \varphi^{\max}_{wd}]$,conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,wd}$,
- unités *traitement* (TU) $\rightarrow$ flux $\varphi^{\text{in}} \leq \varphi^{\max}_{tu}$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,tu}$,
  $\rightarrow$ abaissent ρ_c à un seuil ou d'un facteur λ , coût d'instal. c_{tu},
- unités *processing* (PU), $\rightarrow$ flux $\varphi^{\text{in}} \in [\varphi^{\min}_{pu}, \varphi^{\max}_{pu}]$, conc. $\rho^{\text{in}}_{c} \leq \rho^{\max}_{c,pu}$
  $\rightarrow$ rejettent de l'eau $\varphi^{\text{out}}_{pu} = \varphi^{\text{in}}_{pu} + \Delta_{pu}$,
- unités *buffer* (BU), $\rightarrow$ stockent une quantité d'eau $\leq C_{bu}$, coût c_{bu},
- *tuyaux* (s, e) reliant une sortie d'unité s à une entrée e d'unité, avec un diamètre $d \in \{d_1, ..d_K\}$ et $\varphi_{s,e} \leq \pi(d/2)^2.$, coût c_d.

Design Fonctionnement

Problème- Exemple



Objectif: trouver un réseau fonctionnel qui minimise les coûts (install. + fonctionnement).

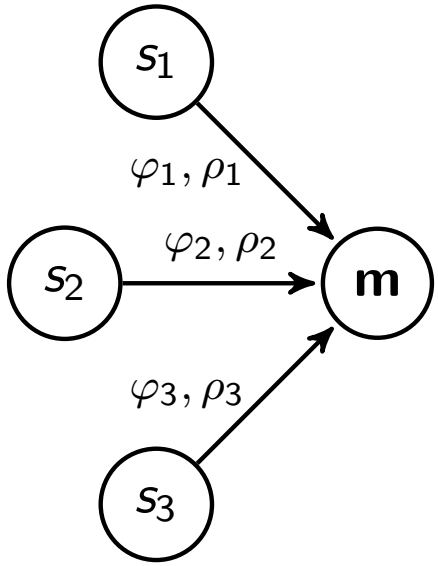
Relaxation (MILP)

- Linéarisation des contraintes d'unités TU : $\rho^{out} = \min(\rho^{in}, \mu)$:

$$\begin{cases} s^{in}, s^{\mu} \geq 0 \\ \eta \in 0, 1 \\ \rho^{in} = \rho^{out} + s^{in} \\ \mu = \rho^{out} + s^{\mu} \\ s^{in} \leq \overline{\rho} \cdot (1 - \eta) \\ s^{\mu} \leq \mu \cdot \eta \end{cases} \tag{1}$$

- Linéarisation de toutes les contraintes bilinéaires:

$$\rho_{m,c}\varphi_m = \sum_{(s,m)} \rho_{sm,c}\varphi_{sm}$$



Relaxation McCormick

Égalités bilinéaires $\rightarrow$ relaxations de McCormick ([5]).
Sous les contraintes $\varphi \in [\varphi^{LB}, \varphi^{UB}]$
et $\rho \in [\rho^{LB}, \rho^{UB}]$, les produits :

$$w = \varphi \cdot \rho \qquad (2)$$

sont remplacés par les 4 équations :

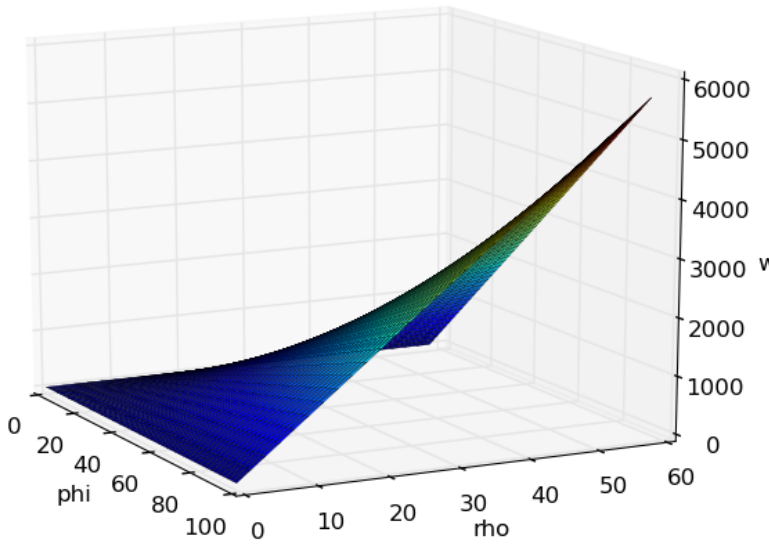


Figure: Graph of $w = \varphi \cdot \rho$

Relaxation McCormick

Égalités bilinéaires $\rightarrow$ relaxations de McCormick ([5]).
Sous les contraintes $\varphi \in [\varphi^{LB}, \varphi^{UB}]$
et $\rho \in [\rho^{LB}, \rho^{UB}]$, les produits :

$$w = \varphi \cdot \rho \qquad (2)$$

sont remplacés par les 4 équations :

$$\varphi^{LB} \cdot \rho + \varphi \cdot \rho^{LB} \leq w + \varphi^{LB} \cdot \rho^{LB}$$
$$\varphi^{UB} \cdot \rho + \varphi \cdot \rho^{UB} \leq w + \varphi^{UB} \cdot \rho^{UB}$$
$$\varphi^{LB} \cdot \rho + \varphi \cdot \rho^{UB} \geq w + \varphi^{LB} \cdot \rho^{UB}$$
$$\varphi^{UB} \cdot \rho + \varphi \cdot \rho^{LB} \geq w + \varphi^{UB} \cdot \rho^{LB}$$

$\rightarrow (w, \rho, \varphi)$ relâché dans l'enveloppe convexe de $\{(w, \rho, \varphi) \setminus w = \varphi \cdot \rho\}$.

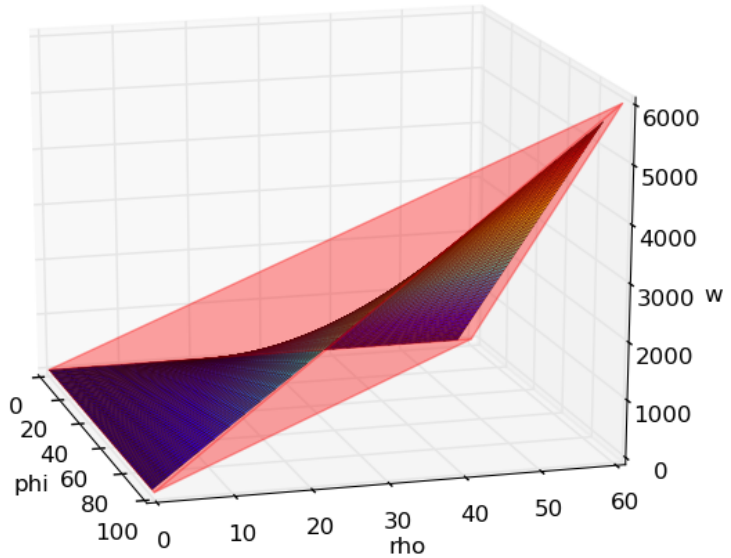


Figure: Convex Hull of $w = \varphi \cdot \rho$

Relaxation entière

Relaxations plus précises avec variables entières : avec les contraintes

$$\rho_{m,c}\varphi_m = \sum_{(s,m)} \rho_{sm,c}\varphi_{sm}, \ \forall c \text{ contaminant}$$

$\rightarrow$ $[\varphi^{\text{LB}}, \varphi^{\text{UB}}]$ découpé en K sous-segments:

$$\varphi^L = \bar{\varphi}_0 < \bar{\varphi}_1 < \ldots < \bar{\varphi}_K < \bar{\varphi}_{K+1} = \varphi^{\text{UB}}$$

$$\bar{\varphi}_k \cdot \rho + \varphi \cdot \rho^{\text{LB}} \leq w + \bar{\varphi}_k \cdot \rho^{\text{LB}} + (1 - b_k) \cdot M, \quad (3)$$

avec $b_k \in \{0, 1\}$ et $\sum_{k=0}^{K} b_k = 1$.

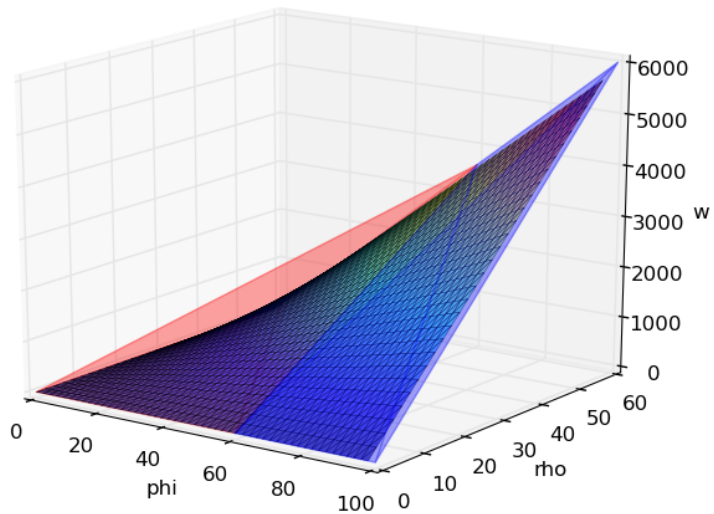


Figure: 2 Convex Hulls for $w = \varphi \cdot \rho$

Problème NLP

variables entières (réseau) fixées $\longrightarrow$ prolème (NLP)
Résolution locale avec algorithme "active set" (SL-EQP) (solveur non linéaire Knitro)

Start from $x_0 = x^{\text{start}}$;
while $k < maxIt$ *or* $err > tol$ **do**

Problème NLP

variables entières (réseau) fixées $\longrightarrow$ prolème (NLP)
Résolution locale avec algorithme "active set" (SL-EQP) (solveur non linéaire Knitro)

Start from $x_0 = x^{\text{start}}$;
while $k < maxIt$ *or* $err > tol$ **do**

1. LP point: solve the 1^{st} order (LP) problem with trust region Δ_{LP}^k;
Get direction d_k^{LP} and working set of constraints W_k ;

Problème NLP

variables entières (réseau) fixées $\longrightarrow$ prolème (NLP)
Résolution locale avec algorithme "active set" (SL-EQP) (solveur non linéaire Knitro)

Start from $x_0 = x^{\text{start}}$;
while $k < maxIt$ *or* $err > tol$ **do**

1. LP point: solve the 1^{st} order (LP) problem with trust region Δ_{LP}^k;
Get direction d_k^{LP} and working set of constraints W_k ;
2. Cauchy point: Compute $\alpha_k^{LP} \in \underset{\alpha \in (0,1]}{\text{argmin}}\ q(\alpha d_k^{LP})$;
Set $d_k^C = \alpha_k^{LP} d_k^{LP}$ and $x_k^C = x_k + d_k^C$;

Problème NLP

variables entières (réseau) fixées $\longrightarrow$ prolème (NLP)
Résolution locale avec algorithme "active set" (SL-EQP) (solveur non linéaire Knitro)

Start from $x_0 = x^{\text{start}}$;
while $k < maxIt$ *or* $err > tol$ **do**

1. **LP point:** solve the 1^{st} order (LP) problem with trust region Δ_{LP}^k;
   Get direction d_k^{LP} and working set of constraints W_k ;
2. **Cauchy point:** Compute $\alpha_k^{LP} \in \underset{\alpha \in (0,1]}{\text{argmin}}\ q(\alpha d_k^{LP})$;
   Set $d_k^C = \alpha_k^{LP} d_k^{LP}$ and $x_k^C = x_k + d_k^C$;
3. **EQP point:** Compute x^{EQP} by solving an EQP with trust region Δ_k and constraints defined by W_k . Define $d_k^{CE} = x_k^{EQP} - x_k^C$;

Problème NLP

variables entières (réseau) fixées $\longrightarrow$ prolème (NLP)
Résolution locale avec algorithme "active set" (SL-EQP) (solveur non linéaire Knitro)

Start from $x_0 = x^{\text{start}}$;
while $k < maxIt$ *or* $err > tol$ **do**

1. LP point: solve the 1^{st} order (LP) problem with trust region Δ_{LP}^k;
Get direction d_k^{LP} and working set of constraints W_k ;
2. Cauchy point: Compute $\alpha_k^{LP} \in \underset{\alpha \in (0,1]}{\text{argmin}}\ q(\alpha d_k^{LP})$;
Set $d_k^C = \alpha_k^{LP} d_k^{LP}$ and $x_k^C = x_k + d_k^C$;
3. EQP point: Compute x^{EQP} by solving an EQP with trust region Δ_k and constraints defined by W_k . Define $d_k^{CE} = x_k^{EQP} - x_k^C$;
4. Trial point: Compute $\alpha_k \in [0, 1]$ as the (approximate) minimizer of $q(d_k^C + \alpha d_k^{CE})$. Set $d_k = d_k^C + \alpha_k d_k^{CE}$ and $x_k^T = x_k + d_k$;
Update: update $x_{k+1} \leftarrow x_k^T$ if x_k^T good enough, else reduce trust region : $\Delta_{k+1} < \Delta_k$;
$k \leftarrow k + 1$;

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;

while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;
while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**
  for $k_{it} = 0$ *to* N_{lim} **do**
    Solve **MILP** relaxation with LB (XPRESS) ;
    if $sol > LB$ **then**
      $LB \leftarrow sol$;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;
while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**
  for $k_{it} = 0$ *to* N_{lim} **do**
    Solve **MILP** relaxation with LB (XPRESS) ;
    if $sol > LB$ **then**
      $LB \leftarrow sol$;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  repeat
    Solve **MILP** relaxation with LB (XPRESS), get sol x^*_{LP};
    if $f(x^*_{LP}) > LB$ **then**
      $LB \leftarrow f(x^*_{LP})$;
    Find $\tilde{\varphi}$ with biggest slack: $argmax|\varphi\rho - w|$ and divide segment at $\tilde{\varphi}$;
  until $t \geq T^{lim}$;

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;
while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**
  for $k_{it} = 0$ *to* N_{lim} **do**
    Solve **MILP** relaxation with LB (XPRESS) ;
    if $sol > LB$ **then**
      $LB \leftarrow sol$;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  repeat
    Solve **MILP** relaxation with LB (XPRESS), get sol x^*_{LP};
    if $f(x^*_{LP}) > LB$ **then**
      $LB \leftarrow f(x^*_{LP})$;
    Find $\tilde{\varphi}$ with biggest slack: $argmax|\varphi\rho - w|$ and divide segment at $\tilde{\varphi}$;
    Solve the **NLP** starting from x^*_{LP};
    if *NLP feasible* (x^*_{NL} *local solution found*) **then**
      if $f(x^*_{NL}) < UB$ **then**
        update $UB \leftarrow f(x^*_{NL})$;
        update best sol $x^* \leftarrow x^*_{NL}$;
      Break repeat loop ;
  until $t \geq T^{lim}$;

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;
while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**
  for $k_{it} = 0$ *to* N_{lim} **do**
    Solve **MILP** relaxation with LB (XPRESS) ;
    if $sol > LB$ **then**
      $LB \leftarrow sol$;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  repeat
    Solve **MILP** relaxation with LB (XPRESS), get sol x^*_{LP};
    if $f(x^*_{LP}) > LB$ **then**
      $LB \leftarrow f(x^*_{LP})$;
    Find $\tilde{\varphi}$ with biggest slack: $argmax|\varphi\rho - w|$ and divide segment at $\tilde{\varphi}$;
    Solve the **NLP** starting from x^*_{LP};
    if *NLP feasible* (x^*_{NL} *local solution found*) **then**
      if $f(x^*_{NL}) < UB$ **then**
        update $UB \leftarrow f(x^*_{NL})$;
        update best sol $x^* \leftarrow x^*_{NL}$;
      Break repeat loop ;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  until $t \geq T^{lim}$;

Our Algorithm

Set $LB \leftarrow 0$, $UB \leftarrow +\infty$;
while $t \leq T^{lim}$ & $|UB - LB| > \varepsilon^{lim}$ **do**
  for $k_{it} = 0$ *to* N_{lim} **do**
    Solve **MILP** relaxation with LB (XPRESS) ;
    if $sol > LB$ **then**
      $LB \leftarrow sol$;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  repeat
    Solve **MILP** relaxation with LB (XPRESS), get sol x^*_{LP};
    if $f(x^*_{LP}) > LB$ **then**
      $LB \leftarrow f(x^*_{LP})$;
    Find $\tilde{\varphi}$ with biggest slack: $argmax|\varphi\rho - w|$ and divide segment at $\tilde{\varphi}$;
    Solve the **NLP** starting from x^*_{LP};
    if *NLP feasible* (x^*_{NL} *local solution found*) **then**
      if $f(x^*_{NL}) < UB$ **then**
        update $UB \leftarrow f(x^*_{NL})$;
        update best sol $x^* \leftarrow x^*_{NL}$;
      Break repeat loop ;
    Find $\tilde{\varphi}$ associated with most violated constraint in **NLP** and divide segment at $\tilde{\varphi}$;
  until $t \geq T^{lim}$;
return best x^* where useless pipes with $\varphi_p < \varphi^{tol}$ are deleted;

Conclusion

- Méthode utilisant des relaxations MILP et NLP du problème.
- Pour plus d'efficacité :
  - des branches sont inutilement explorées durant chaque nouveau B&B $\rightarrow$ implémenter un B&B efficace et intelligent gagnerait beaucoup de temps (la résolution MILP prend rapidement un temps très long!),
  - travailler sur la formulation du problème pour réduire la dégénérescence du problème $\rightarrow$ résolution NLP plus efficace.

Conclusion

- Méthode utilisant des relaxations MILP et NLP du problème.
- Pour plus d'efficacité :
  - des branches sont inutilement explorées durant chaque nouveau B&B $\rightarrow$ implémenter un B&B efficace et intelligent gagnerait beaucoup de temps (la résolution MILP prend rapidement un temps très long!),
  - travailler sur la formulation du problème pour réduire la dégénérescence du problème $\rightarrow$ résolution NLP plus efficace.

THANK YOU !


Richard H Byrd, Nicholas IM Gould, Jorge Nocedal, and Richard A Waltz.
An algorithm for nonlinear optimization using linear programming and equality constrained subproblems.
Mathematical Programming, 100(1):27–48, 2003.


Richard H Byrd, Nicholas IM Gould, Jorge Nocedal, and Richard A Waltz.
On the convergence of successive linear-quadratic programming algorithms.
SIAM Journal on Optimization, 16(2):471–489, 2005.


Richard H Byrd, Jorge Nocedal, and Richard A Waltz.
Knitro: An integrated package for nonlinear optimization.
In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.


Artelys Knitro.
Knitro web page, 2016.


Garth P McCormick.
Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems.
Mathematical programming, 10(1):147–175, 1976.