

## Rappel

Le problème du contrebandier en ECLIPSE CLP :

```
:- lib(ic).
```

```
contrebandier(W,P,C) :-
```

```
    [W,P,C] #:: [0..9],
```

```
    4*W + 3*P + 2*C #=< 9,
```

```
    15*W + 10*P + 7*C #>= 30,
```

```
    labeling([W,P,C]).
```

# Optimisation

- Souvent, un problème est modélisé par des contraintes, et on ne veut pas seulement une solution mais la “meilleure”.
- C’est un **problème d’optimisation**.
- On définit une **fonction objective** (une fonction des solutions vers les réels) qui permet de classer les solutions.
- Un **problème d’optimisation**  $(C, f)$  est une contrainte  $C$  et une fonction objective  $f$ .
- Une affectation  $\theta_1$  est meilleure qu’une affectation  $\theta_2$ , si  $f(\theta_1) < f(\theta_2)$ .
- Une **solution optimale** est une solution de  $C$  telle qu’il n’y a pas de meilleure solution de  $C$ .

# Exemple

- $C : X + Y \geq 4$
- $f(X, Y) := X^2 + Y^2$
- Solution optimale  $\{X \leftarrow 2, Y \leftarrow 2\}$
- Il y a des problèmes d'optimisation qui n'ont pas de solution, pour deux raisons :
  - ▶ la contrainte n'a pas de solution:  $(X \geq 2 \wedge X \leq 0, X^2)$
  - ▶ le problème n'a pas d'optimum:  $(X \leq 0, X)$
- La solution optimale, quand elle existe, n'est pas toujours unique.

# Deux approches à l'optimisation

- Chercher une solution optimale par un *programme*, en se servant d'un solveur de contraintes.  
Exemple : solutions optimales pour des contraintes sur domaine fini.
- Intégration de l'optimisation dans le solveur de contraintes.  
Exemple : l'algorithme simplexe pour les contraintes et fonctions objectives linéaires sur  $\mathbb{R}$ .

# Optimisation pour CSP sur domaine fini

Puisque les domaines sont finis, on peut facilement utiliser un solveur pour construire un optimiseur.

Idée naïve :

- Résoudre la contrainte  $C$ .
- Évaluer la fonction objective  $f$  sur toutes les solutions de  $C$ .
- Renvoyer la solution de  $C$  avec la valeur minimale pour  $f$ .

Inefficace quand  $C$  a beaucoup de solutions.

# Optimisation en réessayant

- Soit  $intsolv(C, D)$  un solveur qui rend soit un domaine simple qui est une solution, soit *faux*.
- Chercher une solution avec valeur minimale de l'expression  $f$ .
- *meilleure* contient la meilleure solution jusqu'à présent.
- $reessayeintopt(C, D, f, meilleure)$ 
  - ▶  $D_2 := intsolv(C, D)$
  - ▶ Si  $D_2$  est un domaine faux, retourne *meilleure*
  - ▶ Soit  $sol$  une solution qui correspond à  $D_2$
  - ▶ retourne  $reessayeintopt(C \wedge f < f(sol), D, f, f(sol))$

## Exemple optimisation en réessayant

- Sac du contrebandier.
- $4W + 3P + 2C \leq 9 \wedge 15W + 10P + 7C \geq 30$  et  $D(W) = [0..9]$ ,  
 $D(P) = [0..9]$ ,  $D(C) = [0..9]$
- En général : on cherche à *minimiser* une fonction. Donc ici : fonction objective : - Profit.
- Le contrebandier veut minimiser sa perte  $-15W - 10P - 7C$ .
- On utilise la recherche par retour en arrière avec bornes consistances : première solution :  $D(W) = \{0\}$ ,  $D(P) = \{1\}$  et  $D(C) = \{3\}$ . Perte de  $-31$  ou profit de 31
- Nouveau problème en ajoutant  $-15W - 10P - 7C < -31$

## Exemple suite

- On recommence la recherche par retour en arrière
- D'abord  $D(W) = [0..2], D(P) = [0..3], D(C) = [0..4]$
- Mais maintenant le choix de  $W = 0$  donne un domaine faux par propagation
- On essaie  $W = 1$  et ça donne  $D(W) = \{1\}, D(P) = \{1\}$  et  $D(C) = \{1\}$  avec une perte de  $-32$
- On ajoute  $15W - 10P - 7C < -32$  et on recommence: pas de solution, donc 32 est le meilleur profit possible.

Problème: On refait des calculs plusieurs fois !

## Contrebandier en réessayant en ECLIPSE CLP

```
contrebandier(W,P,C,Profit) :- retry(0,0,0,0,W,P,C,Profit).

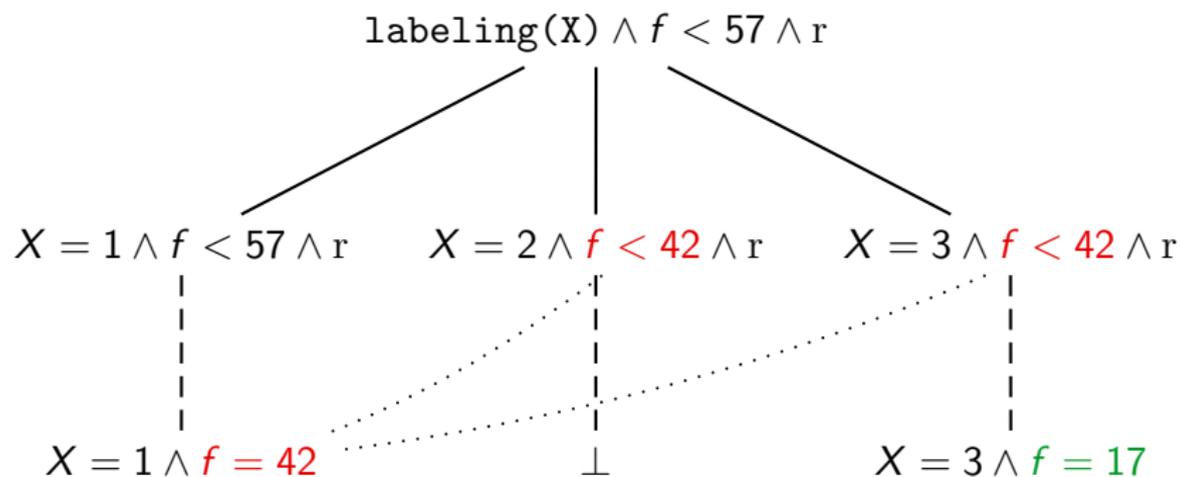
% cherche la meilleure solution pour W,P,C,Profit
% sachant que WOld, POld, COld, ProfitOld est déjà une solution
retry(WOld,POld,COld,ProfitOld,W,P,C,Profit) :-
    cb(ProfitOld,WBetter,PBetter,CBetter,ProfitBetter) ->
    retry(WBetter,PBetter,CBetter,ProfitBetter,W,P,C,Profit) ;
    W=WOld, P=POld, C=COld, Profit=ProfitOld.

% cherche une solution t.q. Profit est mieux que ProfitOld
cb(ProfitOld,W,P,C,Profit) :-
    [W,P,C] #:: [0..9],
    4*W + 3*P + 2*C #=< 9,
    15*W + 10*P + 7*C #= Profit,
    Profit #> ProfitOld,
    labeling([W,P,C]).
```

# Optimisation par retour en arrière

- On combine retour en arrière avec optimisation.
- À chaque étape dans le parcours de l'arbre de recherche, si *meilleure* est la meilleure solution jusqu'à présent, on ajoute la contrainte  $f < meilleure$ .
- Cela évite de refaire tout le parcours de l'arbre de recherche jusqu'à l'endroit où on avait trouvé *meilleure*.

# Optimisation par retour en arrière



## Exemple : contrebandier

- La première solution trouvée est :  $\{W \leftarrow 0, P \leftarrow 1, C \leftarrow 3\}$  avec perte de  $-31$ . On mémorise cette solution comme meilleure jusqu'à présent
- Le retour en arrière revient sur  $D(W) = \{0\}$ ,  $D(P) = [1..3]$ ,  $D(C) = [0..4]$  et on ajoute la nouvelle contrainte  $-15W - 10P - 7C < -31$ .
- On essaie  $P = 2$  et  $P = 3$ . Cela donne un domaine faux
- On revient sur  $D(W) = [0..2]$ ,  $D(P) = [1..3]$ ,  $D(C) = [0..4]$  et on essaie  $W = 1$ .
- Propagation donne  $D(W) = \{1\}$ ,  $D(P) = \{1\}$ ,  $D(C) = \{1\}$  avec perte  $-32$ . Donc nouvelle meilleure solution.

## En ECLIPSE CLP

```
:- lib(ic).
contrebandleropt(W,P,C,Profit) :-
    [W,P,C] #:: [0..9],
    4*W + 3*P + 2*C #=< 9,
    15*W + 10*P + 7*C #= Profit, Profit #> 30, Perte #= -Profit,
    branch_and_bound:minimize(labeling([W,P,C]),Perte).
```

```
[eclipse]: contrebandleropt(W,P,C,Profit).
```

```
Found a solution with cost -31
```

```
Found a solution with cost -32
```

```
Found no solution with cost -88.0 .. -33.0
```

```
W = 1
```

```
P = 1
```

```
C = 1
```

```
Profit = 32
```

```
Yes
```

# L'algorithme simplex

- Optimise une fonction linéaire par rapport à des contraintes linéaires
- L'algorithme d'optimisation le plus couramment utilisé
- Du à George Dantzig (1914-2005) en 1947.



- Lié à l'élimination Gauss-Jordan pour la résolution de contraintes linéaires

# L'algorithme simplex

Un problème d'optimisation  $(C, f)$  est en **forme simplex**, si

- $C$  est une conjonction de  $C_e$  et  $C_i$
- $C_e$  est une conjonction d'équations linéaires
- $C_i$  est la conjonction des inéquations

$$X_1 \geq 0 \wedge \dots \wedge X_n \geq 0$$

où  $variables(C_e) \subseteq \{X_1, \dots, X_n\}$ .

Donc : toutes les variables **non négatives**.

- $f$  est une expression linéaire sur des variables de  $C$

Exemple : Minimiser  $3X + 2Y - Z + 1$  par rapport à

$$\begin{aligned} X + Y &= 3 \\ -X - 3Y + 2Z + T &= 1 \\ X \geq 0, Y \geq 0, Z \geq 0, T \geq 0 \end{aligned}$$

# L'idée de l'algorithme simplexe

- Les contraintes  $C_e \wedge C_i$  définissent un *simplexe* :



- Tous les points avec la même valeur pour  $f$  forment un plan.
- L'optimum se trouve sur un coin du simplexe.
- On commence dans un coin, et puis on continue sur les arrêtes dans une direction où  $f$  décroît.

## 4 classes de problèmes

Forme générale : équations et inéquations quelconques sur  $\mathbb{R}$



Forme simplex



Forme simplex de base (voir plus tard)



Forme simplex résolue (voir plus tard)

# Mettre en forme simple

Un problème peut être mis en forme simple en

- remplaçant des variables  $X$  par  $X^+ - X^-$
- remplaçant des inéquations  $e \leq r$  par  $e + s = r$  où  $s$  est une nouvelle variable (angl: *slack variable*)
- mettant  $X \geq 0$  pour toute variable  $X$

## Exemple mise en forme simple

Minimiser  $Y + Z$  par rapport à

$$X = 10 + Y$$

$$Y \geq Z$$

Minimiser  $Y^+ - Y^- + Z^+ - Z^-$  par rapport à

$$X^+ - X^- = 10 + Y^+ - Y^-$$

$$Y^+ - Y^- = Z^+ - Z^- + R$$

$$X^+ \geq 0, X^- \geq 0, Y^+ \geq 0, Y^- \geq 0, Z^+ \geq 0, Z^- \geq 0, R \geq 0$$

## Rappel : système d'équations en forme résolue

$$\begin{aligned}x_1 &= \beta_1 + \alpha_{1,1}y_1 + \dots + \alpha_{1,m}y_m \\ &\vdots \\ x_n &= \beta_n + \alpha_{n,1}y_1 + \dots + \alpha_{n,m}y_m\end{aligned}$$

- *paramètres* :  $y_1, \dots, y_m$
- *non-paramètres* :  $x_1, \dots, x_n$
- $\{x_1, \dots, x_n\} \cap \{y_1, \dots, y_m\} = \emptyset$ .

# Forme simplex de base

Un problème d'optimisation simplex est en forme de base, si

- Toutes les équations sont en forme résolue
- Chaque constante à droite est non-négative
- Seulement des variables paramètres apparaissent dans la fonction objective

On obtient une **solution de base** en instanciant chaque paramètre avec 0 et chaque non-paramètre avec la constante dans son équation (attention, c'est une solution de la contrainte, mais pas forcément optimale!)

## Exemple

Un problème simplex en forme de base:

Minimiser  $10 - Y - Z$  par rapport à

$$\begin{aligned} X &= 3 - Y \\ T &= 4 + 2Y - 2Z \\ X \geq 0, Y \geq 0, Z \geq 0, T &\geq 0 \end{aligned}$$

On obtient une solution de base et sa valeur objective:

$$\{X \leftarrow 3, T \leftarrow 4, Y \leftarrow 0, Z \leftarrow 0\}$$

et  $f = 10$

# Forme simplex résolue

- Forme *résolue* : forme simplex de base où la fonction objective ne contient pas de variables avec coefficient négatif.
- Dans ce cas, on obtient la solution optimale en mettant tous les paramètres à la valeur 0.
- But : transformer une forme simplex de base en une forme simplex résolue équivalente !

# Algorithme simplex

On commence par un problème en forme de base

- Répéter
  - ▶ Choisir une variable  $y$  avec un coefficient négatif dans la fonction objective (la *variable de pivot*)
  - ▶ Choisir une équation  $x = b + cy + \dots$  avec  $c < 0$  (s'il y en a plusieurs : choisir tel que  $-b/c$  minimal)
  - ▶ Réécrire cette équation en  $y = -b/c + (1/c)x + \dots$  (pivoter)
  - ▶ Substituer  $-b/c + (1/c)x + \dots$  pour  $y$  dans toutes les autres équations et dans la fonction objective
- jusqu'à ce qu'il n'existe pas une telle variable  $y$  ou une telle équation
- Si un tel  $y$  n'existe pas une solution optimale est trouvée
- Sinon il n'y a pas de solution optimale

## Exemple

Minimiser  $10 - Y - Z$  par rapport à

$$X = 3 - Y$$

$$T = 4 + 2Y - 2Z$$

$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0$$

On choisit  $Y$  et la première équation.

## Exemple (suite)

Minimiser  $7 + X - Z$  par rapport à

$$\begin{aligned} Y &= 3 - X \\ T &= 10 - 2X - 2Z \\ X \geq 0, Y \geq 0, Z \geq 0, T &\geq 0 \end{aligned}$$

On choisit  $Z$  et la deuxième équation.

## Exemple (suite)

Minimiser  $2 + 2X + 0.5T$  par rapport à

$$\begin{aligned} Y &= 3 - X \\ Z &= 5 - X - 0.5T \\ X \geq 0, Y \geq 0, Z \geq 0, T \geq 0 \end{aligned}$$

On ne peut plus choisir de variable, valeur optimale: 2

# Les cas où le minimum n'existe pas

Minimiser  $1 - X - Y$  par rapport à

$$T = 1 + X + Y$$

$$U = 2 + X + Y$$

$$X \geq 0, Y \geq 0, T \geq 0, U \geq 0$$

- Les valeurs de  $X$  et de  $Y$  peuvent être arbitrairement grandes.
- La valeur de  $1 - X - Y$  peut être arbitrairement petit.

# Comment obtenir une solution de base ?

- en particulier: parfois (assez souvent en examen !) faire quelques réécritures simples des équations suffit pour obtenir une solution de base
- en général: résoudre un problème simple différent :
  - ▶ Ajouter des variables artificiellement pour obtenir des équations en forme de base
  - ▶ Minimiser la somme des variables artificielles
  - ▶ Si la somme est 0 on peut construire une forme résolue de base pour le problème initiale

## Exemple

Minimiser  $3X + 2Y - Z + 1$  par rapport à

$$X + Y = 3$$

$$-X - 3Y + 2Z + T = 1$$

$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0$$

On met toutes les équations dans la forme  $0 = \dots$  et on obtient:

$$0 = 3 - X - Y$$

$$0 = 1 + X + 3Y - 2Z - T$$

$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0$$

On met deux variables artificielles  $A_1$  et  $A_2$  à gauche à la place de 0 et on obtient:

## Exemple (suite)

Minimiser  $A_1 + A_2$  par rapport à

$$A_1 = 3 - X - Y$$

$$A_2 = 1 + X + 3Y - 2Z - T$$

$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0, A_1 \geq 0, A_2 \geq 0$$

On réécrit la fonction objective en substituant les variables artificielles :

Minimiser  $4 + 2Y - 2Z - T$

On choisit  $T$  et la deuxième équation :

## Exemple (suite)

Minimiser  $3 - X - Y + A_2$  par rapport à

$$\begin{aligned}A_1 &= 3 - X - Y \\T &= 1 + X + 3Y - 2Z - A_2 \\X \geq 0, Y \geq 0, Z \geq 0, T \geq 0, A_1 \geq 0, A_2 \geq 0\end{aligned}$$

On choisit  $X$  et la première équation :

## Exemple (suite)

Minimiser  $A_1 + A_2$  par rapport à

$$\begin{aligned} X &= 3 - Y - A_1 \\ T &= 4 + 2Y - 2Z - A_1 - A_2 \\ X \geq 0, Y \geq 0, Z \geq 0, T \geq 0, A_1 \geq 0, A_2 \geq 0 \end{aligned}$$

On substitue 0 pour  $A_1$  et  $A_2$  et dans la fonction objective originale les non-paramètres:

## Exemple (suite)

Minimiser  $10 - Y - Z$  par rapport à

$$X = 3 - Y$$

$$T = 4 + 2Y - 2Z$$

$$X \geq 0, Y \geq 0, Z \geq 0, T \geq 0$$

# Comment construire la forme de base ?

En général, il y a trois cas:

- La valeur optimale de la fonction objective du problème modifié est  $> 0$ 
  - ▶ Le problème original est insatisfaisable.
- La valeur optimale est 0 et toutes les variables artificielles sont des paramètres
  - ▶ Après élimination des variables artificielles on obtient une forme de base
- La valeur optimale est 0, mais pas toutes les variables artificielles sont des paramètres
  - ▶ Si une des variables de la partie droite est une des variables originales, on peut réécrire l'équation avec cette variable à gauche et ainsi de suite.

# Cycles

- Une solution de base est dégénérée, si une variable non-paramètre prend la valeur 0.
- Dans ce cas, il y a un danger d'un cycle.
- Il y a des règles simples pour éviter des cycles, par exemple en ordonnant les variables : on choisit comme variable pivot toujours la variable la plus petite parmi les variables avec un coefficient négatif dans la fonction objective.

# Optimisation sur $\mathbb{Z}$

- L'algorithme simplex est défini sur des domaines réels.
- On peut l'utiliser aussi comme ingrédient de base pour des problèmes d'optimisation sur les entiers.
- Mais on doit résoudre *plusieurs* problèmes d'optimisation sur  $\mathbb{R}$ .

# Optimisation sur $\mathbb{Z}$

- Optimisation **branch and bound** pour  $(C, f)$  où  $C$  est une contrainte sur les entiers et  $f$  une fonction à minimiser
  - ▶ Utiliser simplex pour trouver un optimum réel
  - ▶ Si la solution est entière : la retourner
  - ▶ Sinon choisir une variable  $x$  avec une valeur optimale non entière  $d$  et considérer récursivement les problèmes
    - ★  $(C \wedge x \leq \lfloor d \rfloor, f)$
    - ★  $(C \wedge x \geq \lceil d \rceil, f)$
  - ▶ On garde en mémoire la meilleure solution entière trouvée jusqu'à présent
  - ▶ Si on a déjà trouvé une solution sur les entiers, alors tout problème avec une plus grande meilleure solution sur les réels peut être abandonné

## Utiliser la librairie clpr ou clpq

```
[eclipse]: lib(clpr).
```

```
[eclipse]: {X+Y = 3, -X-3*Y+2*Z+T = 1, X >= 0,  
           Y >= 0, Z >=0, T >= 0}, inf(3*X+2*Y-Z+1, Min).
```

```
... Min = 2 ...
```

```
[eclipse] : {X+Y = 3, -X-3*Y+2*Z+T = 1, X >= 0, Y >= 0,  
            Z >=0, T >= 0}, inf(3*X+2*Y-Z+1, 3*X+2*Y-Z+1).
```

```
X = 0 Y = 3 Z = 5 T = 0
```

```
[eclipse]: {Y <= X, Y <= 8-2*X, X-Y+1 = 3*Z}, inf(-Y, Min).
```

```
... Min = -2.6666666666666665 ...
```

```
[eclipse]: {Y<=X, Y<=8-2*X, X-Y+1=3*Z}, bb_inf([X, Y, Z], -Y, Min).
```

```
... Min = -1.00000000000000004 ...
```

```
[eclipse]: lib(clpq).
```

```
[eclipse]: {Y <= X, Y <= 8-2*X, X-Y+1 = 3*Z}, inf(-Y, Min).
```

```
... Min = -8 / 3 ...
```

```
[eclipse]: {Y<=X, Y<=8-2*X, X-Y+1=3*Z}, bb_inf([X, Y, Z], -Y, Min).
```

```
... Min = -1 ...
```