Continuous and Discrete Optimization

OMA Labs 2018-2019

Table of contents:

1. Optimisation continue et optimisation approchée

1.1. Optimisation sans contraintes

1.1.1. Méthode du gradient

a)

b)

c)

1.1.2. Méthode BFGS

1.2. Optimisation sous contraintes

1.2.1. Optimisation à l'aide de routines

1.2.2. Optimisation sous contraintes et pénalisation

1)

2) ### 1.2.3. Méthodes duales pour l'optimisation sous contraintes

1)

2)

1.3. Optimisation non convexe - Recuit simulé

a)

b)

c)

1.4. Application à la synthèse à réponse impulsionnelle infinie

---

2. Optimisation discrète et optimisation multiobjectif

2.1. Rangement d'objets

1) Comment traduire mathématiquement que la boîte i contient un objet et un seul et que l'objet j se trouve dans une boîte et une seule ?

On note $O_j \to B_i$ la relation *"l'objet j est dans la boite i"*

Alors,

$$\forall i \ \exists! j \ / \ O_j \to B_i \Leftrightarrow \forall i \ \sum_j x_{i,j} = 1$$

Et de même,

$$\forall j \ \exists! i \ / \ O_j \to B_i \Leftrightarrow \forall j \ \sum_i x_{i,j} = 1$$

2) Formulation du PLNE et mise en oeuvre de sa résolution

Le problème de programmation linéaire en nombre entiers s'écrit :

$$(\mathcal{P}) : \min_x \sum_{i,j} x_{i,j} \|O_j - B_i\| \quad \text{s.c} \begin{array}{l} \forall i \ \sum_j x_{i,j} = 1 \\ \forall j \ \sum_i x_{i,j} = 1 \end{array}$$

Résultat : 15.3776

3) Objet 1 à gauche directe de l'Objet 2

On formule cette nouvelle contrainte par :

$$\forall i \in [\![2, n-1]\!] \ x_{i,1} - x_{i+1,2} = 0$$

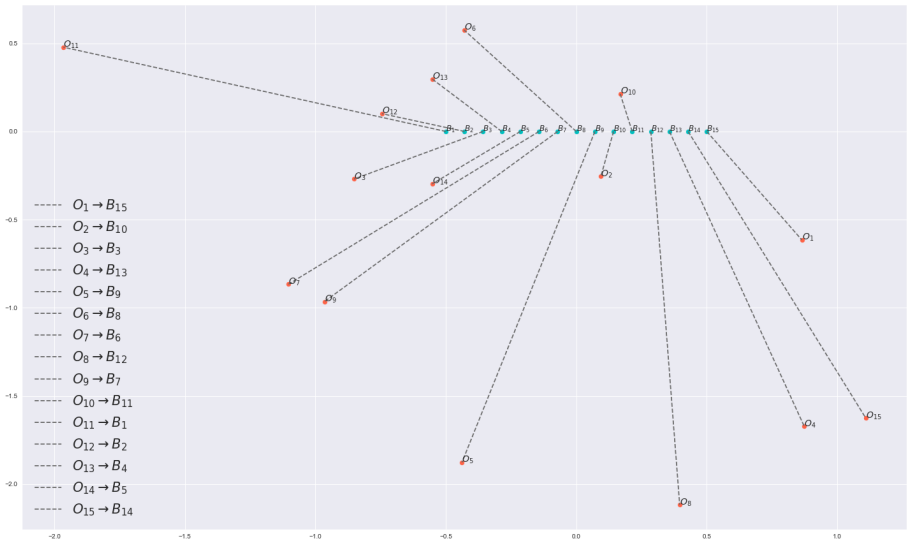

Figure 1: Problème initial, résultat : 15.3776

De plus, cette contrainte impose également que la relation $O_1 \to B_n$ est impossible car O_2 ne peut alors être droite de O_1.

On impose donc $x_{n,1} = 0$ (il s'ensuit par les contraintes précédentes que $x_{1,2} = 0$)

Résultat : 15.5651

4) Objet 3 à droite de l'Objet 4

Montrons que $x_{i,3} + x_{i+k,4} \leq 1,\ \forall i,\ \forall k > 0 \Leftrightarrow O_3$ à droite de O_4

($\Rightarrow$) > Raisonnons par l'absurde et supposons O_3 à gauche de O_4

Notons i_3 l'indice de la boite contenant O_3. Alors, la boite contenant O_4 étant à droite de B_{i_3}, $\exists k > 0\ \ / O_4 \to B_{i_3+k}$

$\Rightarrow x_{i_3,3} = 1$ et $x_{i_3+k,4} = 1$

$\Rightarrow x_{i_3,3} + x_{i_3+k,4} > 1$ ce qui est impossible

($\Leftarrow$)

Notons i_3 l'indice de la boite contenant O_3

Comme O_4 à gauche de O_3, $\forall k > 0,\ x_{i_3+k,4} = 0$

$\Rightarrow x_{i_3,3} + x_{i_3+k,4} \leq 1$

On remarque que cette contrainte peut se réécrire de façon plus succinte selon :

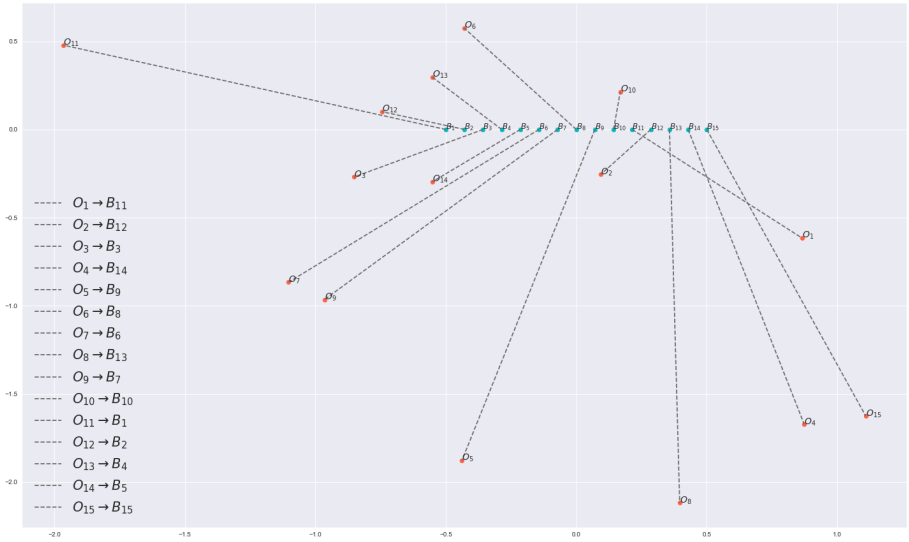

Figure 2: Objet 1 à gauche directe de l'Objet 2, résultat : 15.5651

$$\forall i \in [\![1, n-1]\!],\ x_{i,3} + \sum_{k=i+1}^{n} x_{k,4} \leq 1$$

En effet, pour un i donné, si $O_3 \to B_i$ et que O_3 à droite de O_4, alors toutes les boites d'indice supérieur ne peuvent contenir O_4, c'est à dire $\forall k \in [\![i+1, n]\!],\ x_{k,4} = 0$.

Cette contrainte englobe également les cas limite :

- Si $i = 1$, O_4 ne peut être dans B_1 aussi selon les contraintes précédentes et la nouvelle contrainte lui interdit d'être à droite de O_3, c'est donc un cas impossible
- So $i = n$, O_4 ne peut être dans B_n aussi selon les contraintes précédentes et est donc nécessairement à gauche de O_3

Résultat : 15.9014

5) Objet 7 à côté de l'Objet 9

Il suffit ici de symétriser la contrainte énoncée à la question 3 pour autoriser d'être à droite ou à gauche

$$\forall i \in [\![1, n-1]\!]\ \ x_{i,7} - x_{i+1,9} = 0 \quad \text{ou} \quad x_{i+1,7} - x_{i,9} = 0$$

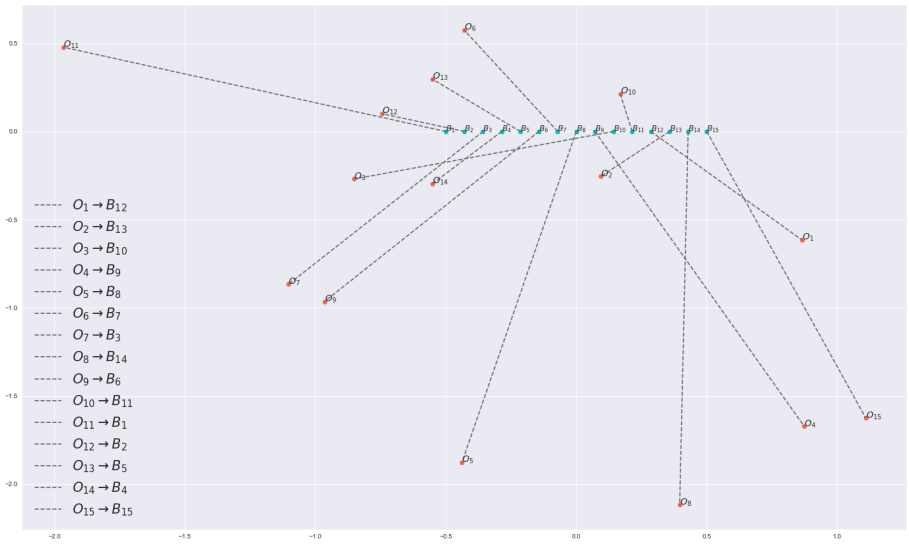

Figure 3: Objet 3 à droite de l'Objet 4, résultat : 15.9014

$$\Rightarrow \forall i \in [\![1, n-1]\!] \ (x_{i,7} - x_{i+1,9})(x_{i+1,7} - x_{i,9}) = 0$$

Le problème d'optimisation n'est alors plus linéaire.

Une façon de contourner ce problème consiste à résoudre le problème en imposant O_7 à gauche de O_9, puis celui imposant O_7 à droite de O_9, et de garder la solution minimisant la distance totale

Résultat : 15.9048

6) Unicité de la solution

On remarque que les objets 13 et 14 sont symétriques par rapport à l'axe des abscisses. Donc en principe, échanger leurs attributions mutuelles ne devrait pas impacter la distance.

En imposant la contrainte $x_{4,13} = 1$ on retrouve la meme distance que précédemment, il n'y a donc pas unicité de la solution. –>

2.2. Communication entre espions

1) Modélisation du problème et démarche de résolution

On pose $G = [\![1, n]\!]$ l'ensemble des espions et $E = \bigcup_{i \in G} \{(i, s), s \in S_i\}$ les voies de communications entre les espions.

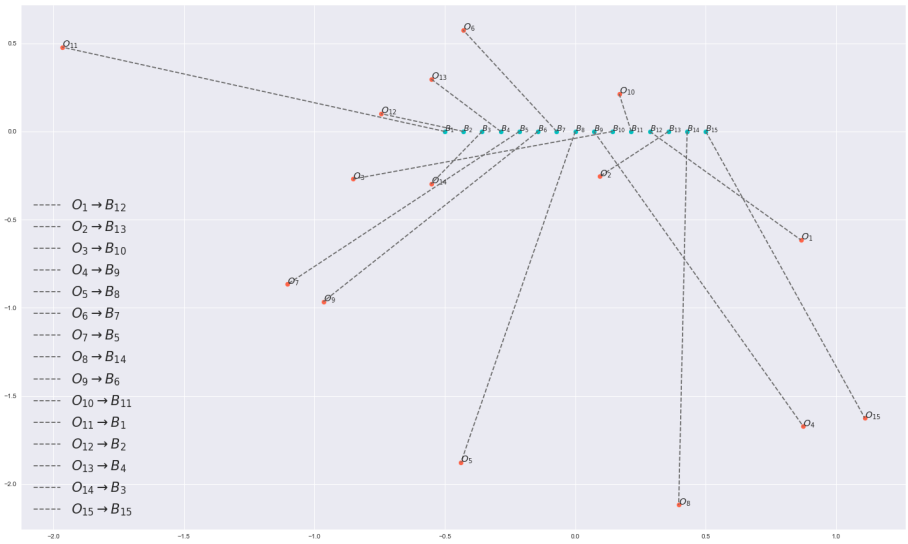

Figure 4: Objet 7 à côté de l'Objet 9, résultat : 15.9048

L'objectif étant de minimiser la probabilité d'interception, on peut réexprimer le problème comme la maximisation de la probabilité de non-interception.

Notons $K_{ij} = \{\text{interception de la commmunication entre } i \text{ et } j\}$, alors:

$$\max_{E' \subset E} \mathbb{P}\left(\bigcap_{(i,j) \in E'} \overline{K_{ij}} \right) \Leftrightarrow \max_{E' \subset E} \prod_{(i,j) \in E'} (1 - p_{ij}) \Leftrightarrow \min_{E' \subset E} \sum_{(i,j) \in E'} -\log(1 - p_{ij}) \quad \text{avec } E' \text{ connexe}$$

En pondérant les arrêtes de notre graphe par l'application symétrique $w = (i,j) \in E \longmapsto -\log(1 - p_{ij})$, le problème revient alors à résoudre un problème d'arbre recouvrant minimum sur (G, E, w),

$$i.e. \quad (\mathcal{P}) : \text{Trouver } (G, E') \quad \text{tq} \quad \begin{array}{l} \sum_{e' \in E'} w(e') \text{ est min} \\ (G, E') \text{ connexe} \end{array}$$

Des algorithmes polynomiaux de type Kruskal et Prim permettent de résoudre le problème de façon exacte.

2) Résultat

La probabilité d'interception est donnée par :

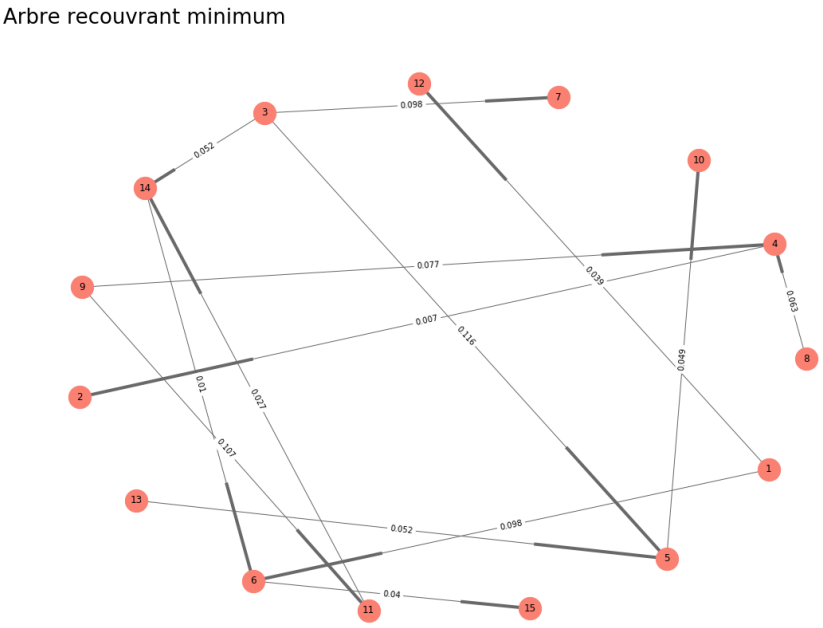

Figure 5: Arbre recouvrant minimum

$$\mathbb{P}(interception) = 1-\mathbb{P}(\overline{interception}) = \mathbb{P}\left(\bigcap_{(i,j)\in E'} \overline{K_{ij}}\right) = \prod_{(i,j)\in E'} \mathbb{P}(\overline{K_{ij}}) = \prod_{(i,j)\in E'} (1-p_{ij})$$

Où E' est la solution obtenu par l'algorithme de Kruskal

On obtient $\mathbb{P}(interception) = 0.5809$

2.3 Dimensionnement d'une poutre

1) Méthode gloutonne

Dans ce problème, il sera question d'optimisation multi-objectif puisque l'on cherche à minimiser à la fois le poids et la deflexion de la poutre sous certaines contraintes.

Les fonctions qui donnent le poids et la deflexion de la poutre, étant faciles à évaluer, nous pouvons utiliser la méthode gloutonne (gourmande) dans laquelle il sera question de générer 100000 points qui réalisent les contraintes physiques imposées et de tracer un front de Pareto.

La recherche des points Pareto Optimales se fait en 2 étapes:

- Pour chaque point généré, on stocke sous forme de liste l'ensemble des points situés en bas et à gauche de lui (ayant un poids et une deflexion inférieurs à lui)
- Si cette liste est vide, alors ce point est pareto-optimal. Dans le cas contraire, ce point n'est pas retenu

Résultat : 13.7482 s

2) Méthode plus sophistiquée

On transforme ici notre problème en problème mono-objectif:

$$\min_{(a,b)} : \lambda * p(a,b) + d(a,b)$$

On fait varier λ sur un intervalle afin de donner à chaque itération plus de poids à l'une des 2 fonctions que l'on cherche à minimiser et on résout plusieurs fois le problème de minimisation précédent.

On notera que le problème de minimisation sous contraintes se fera en utilisant une méthode de type quasi Newton (SLSQP) dans laquelle on définira des bornes et des contraintes. Aussi, comme les solutions obtenus dépendent de notre point d'initialisation, on définira à chaque itération un x0 différent que l'on générera de manière random.

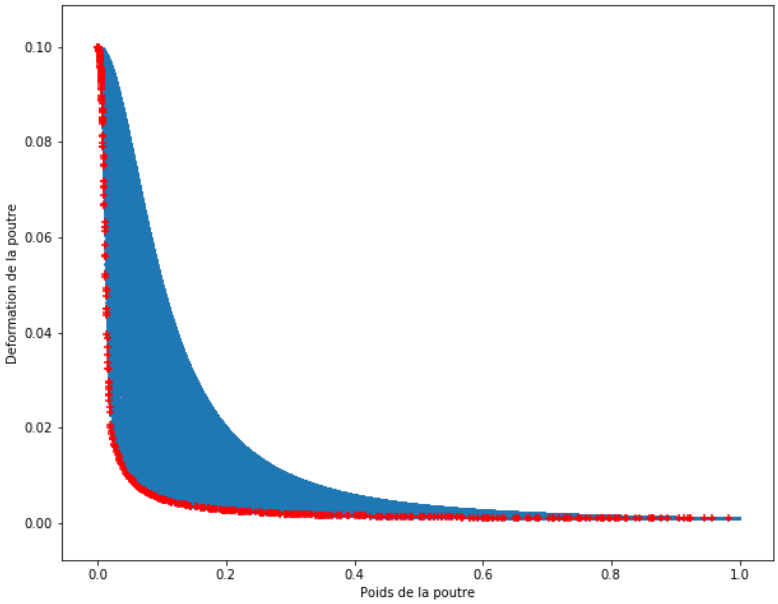

Figure 6: Front de pareto méthode gloutonne

Résultat : 2.4975 s

Comparaison :

Qualité de l'estimation: [Méthode Gloutonne : Très bonne estimation, Méthode Sophistiquée: Bonne estimation]

Nombre d'évaluations des objectifs: [Méthode Gloutonne: 200000, Méthode Sophistiquée: 10000]

Temps d'éxecution: [Méthode Gloutonne améliorée: 13.7s, Méthode Sophistiquée: 2.5s]

2.4 Approvisionnement d'un chantier

1) Modélisation du problème

Notons $\mathbf{a} = (a_t)_{1 \leq t \leq N}$ et $\mathbf{r} = (r_t)_{1 \leq t \leq N}$ les vecteurs designant respectivement le nombre de machine à ajouter et retirer au temps t.

Chaque semaine, l'entreprise paye donc $a_t p^{\text{init}}$ de frais de mise en service et $r_t p^{\text{fin}}$ de frais de restitution.

De plus, en prenant la somme sur les semaines passées des ajouts moins les retraits, on obtient le nombre de machines actuellement louées. On en déduit que l'entreprise paye également chaque semaine $\left(\sum_{i=0}^{t} a_i - r_i\right) p^{\text{loc}}$ de frais de location.

On peut ainsi définir une fonction de frais dépensé sur la durée du chantier par :

$$J(\mathbf{a}, \mathbf{r}) = \sum_{t=0}^{N} \left[a_t p^{\text{init}} + r_t p^{\text{fin}} + \sum_{i=0}^{t} (a_i - r_i) p^{\text{loc}} \right]$$

De plus, on doit vérifier les contraintes suivantes :

- Aucun retour de machine possible la première semaine $\Rightarrow r_1 = 0$
- Le nombre de machine louée sur une semaine donnée doit toujours être supérieur au nombre de machines requises $\Rightarrow \forall t \in [\![1, N]\!],\ \sum_{i=0}^{t} a_i - r_i \geq d_t$
- Tout est rendu et rien n'est ajouté lors de la dernière semaine $\Rightarrow \sum_{i=0}^{N} a_i - r_i = 0$

Finalement, les vecteurs $\mathbf{a}$ et $\mathbf{r}$ étant des vecteurs entiers de $\mathbb{N}^N$, en notant $x = (\mathbf{a}, \mathbf{r})$, on peut poser le problème comme le problème de programmation linéaire en nombre entier suivant :

$$(\mathcal{P}) : \min_{x \in \mathbb{N}^{2N}} J(x) \quad \text{s.c} \begin{array}{c} r_1 = 0 \\ \sum_{i=0}^{t} a_i - r_i \geq d_t \,, \forall t \\ \sum_{i=0}^{N} a_i - r_i = 0 \end{array}$$

2) Résultat

Coût optimal = 3304000

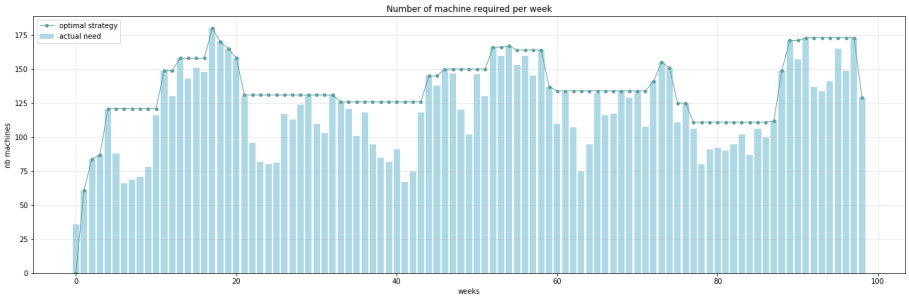

Figure 7: Besoin en machine vs stratégie optimale

3) Commentaires

On remarque que $p^{\text{init}} + p^{\text{fin}} = 10p^{\text{loc}}$. Ainsi, en terme de coût, les 2 process suivants sont équivalents :

$$\underbrace{\text{ajout machine}}_{p^{\text{init}}} \longrightarrow \underbrace{\text{location 10 semaines}}_{10p^{\text{loc}}} \longrightarrow \underbrace{\text{restitution}}_{p^{\text{fin}}}$$

$$\underbrace{\text{ajout machine}}_{p^{\text{init}}} \longrightarrow \underbrace{\text{restitution}}_{p^{\text{fin}}} \longrightarrow \underbrace{\text{ajout machine}}_{p^{\text{init}}} \longrightarrow \underbrace{\text{restitution}}_{p^{\text{fin}}}$$

On en déduit qu'une stratégie optimale intuitive consiste à ne conserver une machine que si sa durée de location n'excède pas 10 semaines, autrement il est plus intéressant de la restituer pour la relouer après.

Cette stratégie est cohérente avec la stratégie optimale observée entre les semaines 77 et 87 durant lesquelles le nombre de machines louées est supérieur au besoin car il coûterait plus cher de les restituer pour les relouer en vue de la semaine 87