Rapport TP
Optimisation - OMA

Baptiste Doyen

October 22, 2018

Abstract

Rapport du TP d'optimisation continue (séance 1) et discrète (séance 2 et 3).

1 Séance 1 : optimisation continue et optimisation approchée

1.1 Optimisation sans contrainte

1.1.1 Méthode de gradient

1. **Essai avec différentes valeurs de ρ**

   Certaines valeurs de ρ assurent la convergence de la méthode du gradient. Empiriquement, il semble que pour les valeurs inférieures ou égales à 0.022, la convergence est assurée tandis que pour les valeurs supérieures ou égales à 0.023, il n'y a plus convergence.

2. **Méthode du gradient avec choix adaptatif**

   À chaque itération on choisit un pas qui annule le terme d'ordre 2.
   Calcul du pas : $d_k = \frac{||\nabla f(x_k)||^2}{\nabla f(x_k)^T A \nabla f(x_k)}$

3. **Comparaison des résultats**

   La valeur minimale trouvée de -1.8369 est la même. On remarque néanmoins que plus ρ devient petit, plus la méthode avec choix adaptif devient rapide (plus de deux fois plus rapide pour $\rho = 0.001$ par exemple).
   Pour des valeurs de ρ plus grande (de l'ordre de 0.01), les temps d'exécutions sont similaires en revanche.

1.1.2 Méthode de Quasi-Newton

Il existait un autre moyen d'obtenir ce résultat : il s'agit de la méthode à pas optimal choisi précédemment. Ici la hessienne de f_1 se calcule simplement, on a donc pas besoin de l'approximer avec une méthode de Quasi-Newton.

1.2 Optimisation sous contraintes

Dans cette partie on suppose que $U \in \mathcal{U}_{ad} = [0; 1]^5$.

1.2.1 Optimisation à l'aide de routines Matlab

On utilise ici l'algorithme SQP (Sequential Quadratic Programming).

(i) Résultats pour f_1

Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
0	6	0.000000e+00	0.000e+00	1.000e+00	0.000e+00	3.500e+00
1	20	-5.005467e-02	0.000e+00	5.765e-02	9.985e-02	6.590e+00
2	29	-7.313055e-02	0.000e+00	3.430e-01	9.696e-02	3.229e+00
3	35	-1.302938e-01	0.000e+00	1.000e+00	6.214e-02	7.129e-01
4	41	-1.327878e-01	0.000e+00	1.000e+00	5.254e-02	6.543e-01
5	47	-1.385285e-01	0.000e+00	1.000e+00	2.859e-02	3.977e-02
6	53	-1.385316e-01	0.000e+00	1.000e+00	5.595e-04	6.658e-03
7	59	-1.385317e-01	0.000e+00	1.000e+00	8.968e-05	4.680e-06

Figure 1: Résultats de l'algorithme SQP pour f_1

(ii) Résultats pour f_2

Iter	Func-count	Fval	Feasibility	Step Length	Norm of step	First-order optimality
0	6	8.172101e+01	0.000e+00	1.000e+00	0.000e+00	7.475e+01
1	12	1.393828e+01	0.000e+00	1.000e+00	1.743e+00	9.465e+01
2	18	0.000000e+00	0.000e+00	1.000e+00	1.000e+00	1.452e+01
3	24	0.000000e+00	0.000e+00	1.000e+00	0.000e+00	6.661e-16

Figure 2: Résultats de l'algorithme SQP pour f_2

Commentaire : ici l'algorithme converge vers 0, ce qui est bien la valeur du minimum que l'on aurait pu attendre pour f_2 compte tenu des contraintes. Afin de pouvoir réaliser des itérations ailleurs qu'au point nul, on a initialisé l'algorithme de convergence en un vecteur aléatoire. L'algorithme converge tout de même rapidement vers le vecteur nul.

1.2.2 Optimisation sous contraintes et pénalisation

1. **Fonction de pénalisation** Soit U un vecteur de $\mathbb{R}^N$.
   On définit le vecteur U^+ par :

   $$\forall i \in [0, N], (U^+)_i = max(0, (U)_i)$$

   De sorte que : $||U^+||^2 = 0 \Longleftrightarrow \forall i \in [1; N], (U)_i \leq 0$

   Soit $\beta : \mathbb{R}^5 \to \mathbb{R}$ la fonction définie par :

   $$\beta(U) = ||(-U)^+||^2 + ||(U - [\mathbf{1}])^+||^2$$

   où $[\mathbf{1}] \in \mathbb{R}^5$ désigne le vecteur colonne ne contenant que des 1.

   Ainsi, $\beta(U) = 0 \Longleftrightarrow (-U)^+ = 0$ et $(U - [\mathbf{1}])^+ = 0$
   $\Longleftrightarrow \forall i \in [0, N], -(U)_i \leq 0$ et $(U)_i - 1 \leq 0 \Longleftrightarrow U \in \mathcal{U}_{ad}$.

   De plus, β est continue et pour $u \notin \mathcal{U}_{ad}, \beta(u) > 0$. Enfin, f_1 est continue, et coercive ($f_1(u) \underset{||u|| \to +\infty}{\to} +\infty$ car $U^T S U$ est quadratique en U et domine donc $B^T U$ en l'infini).

Elle est donc inf-compacte (cas de la dimension finie) et est de plus bornée inférieurement (en effet elle est convexe - sa hessienne vaut $2A^TA \in S_n^+$ - sur $\mathcal{U}_{ad}$ convexe et admet un minimum local autour de 0 qui est donc un minimum global par convexité). On peut donc légitimement appliquer la méthode de pénalisation pour f_1.

On vérifie que c'est également le cas avec f_2 (elle est continue, coercive et bornée inférieurement car $x \mapsto xexp(x)$ l'est aussi et $U^TSU \geq 0$).

(on remarque que β définie plus haut est différentiable ($||U||^2 = U^TU$), ce qui sera utile dans la mise en oeuvre de cette méthode de pénalisation).

2. **Mise en oeuvre de la méthode de pénalisation**

   On obtient la même valeur minimale qu'avec la méthode SQP vue précédemment soit -0.13853. Cette méthode a néanmoins un temps d'exécution plus long que l'algorithme SQP (4 fois plus long environ).

   Pour f_2, une tolerance beaucoup plus petit doit être appliquée si l'on souhaite une solution très petite proche de 0. Par exemple on doit fixer tolerance $= 10^{-9}$ pour avoir un minimum de l'ordre de 10^{-9} très proche de 0. Le coût calculatoire est du coup impacté et le temps d'éxécution plus long (une demi-seconde sur PC)

1.2.3 Méthodes duales pour l'optimisation sous contraintes

1. **Écriture du lagrangien**

   Soit $\mathcal{L}_1(U, \lambda)$ le lagrangien associé à la fonction f_1 et aux contraintes définissant $\mathcal{U}_{ad}$:

$$\mathcal{L}_1(U, \lambda) = f_1(U) + \sum_{i=0}^{p} \lambda_i g_i(U)$$

   où : $p = 10$ et $g_i(U) = -u_i$ si $i \leq 5$ et $g_i(U) = u_i - 1$ sinon.

   On vérifie l'existence d'un point selle pour le Lagrangien :

   (i) f_1 et $(g_i)_i$ sont continûment différentiables
   (ii) Les contraintes sont qualifiées : $\forall u \in \mathcal{U}_{ad}, \nabla g_i(u) = +/- e_i$ où e_i désigne le $ième$ vecteur élémentaire de $\mathbb{R}^p$
   (iii) Le problème de minimisation sous contraintes admet une solution : f_1 est inf-compacte sur $\mathcal{U}_{ad}$ fermé.

   On peut donc légitimement appliquer l'algorithme d'Uzawa pour résoudre le problème d'optimisation sous-contraintes.

2. **Mise en oeuvre de l'algorithme d'Uzawa**

   Ici l'espace dual Λ est $\mathbb{R}_+^p$, projeter un vecteur v de l'espace sur Λ revient donc à déterminer v^+.

1.3 Optimisation non-convexe - Recuit simulé

1. **Fonction f_3**

   Soit $U \in \mathbb{R}^5, f_3(U) := f_1(U) + 10 * sin(2f_1(U))$.
   On calcule $\nabla f_3(U) = \nabla f_1(U) + 20 * \nabla f_1(U) * cos(2f_1(U))$

2. **Avec une méthode d'optimisation classique**

   On applique la méthode de descente de gradient avec ρ constant ainsi qu'une routine MATLAB du type Quasi-Newton.
   On converge parfois vers un minimul local de valeur $-4,5147$ avec la méthode de Quasi-Newton.
   De manière surprenante, l'algorithme de descente de gradient avec ρ constant mais très faible (10^{-3}) permet de converger parfois vers le minimum global mais converge aussi vers un minimum global de valeur -7.6563 selon l'initialisation.
   Ces méthodes sont assez sensibles au choix du point d'initialisation.

3. **Avec la méthode du Recuit simulé**

   On fixe les paramètres Température initiale à 100 et Nombre de transformations par palier à 100 aussi.
   Pour ces paramètres là, l'algorithme converge toujours vers la valeur minimale de -10.7979.

1.4 Application : synthèse d'un filtre à réponse impulsionnelle finie

1. **Optimisation sans contraintes**

   On choisit une méthode de discrétisation à pas constant et régulière par rapport aux deux intervalles : l'intervalle $[0, 0.1]$ est subdivisé en M sous -intervalles égaux et de même pour l'intervalle $[0.15, 0.5]$ avec $M = 1000$. On applique la méthode de Quasi-Newton au problème d'optimisation.

2. **Optimisation sous contraintes**

   Sans contrainte le problème est le suivant :

   $$\min_{h \in \mathbb{R}^{30}} \max_{1 \leq j \leq 30} |H_0(\nu_j) - H(\nu_j)|$$

   ce qui est équivalent à ce problème avec des contraintes d'inégalités :

   $$\begin{array}{l} \min\limits_{t \in \mathbb{R}_+, h \in \mathbb{R}^{30}} t \\ t \geq |H_0(\nu_j) - H(\nu_j)| \ \ (1 \leq j \leq 30) \end{array}$$

   Le problème formulé ainsi est alors un problème d'optimsation linéaire (la fonction objectif t est linéaire en t) sous contraintes.

2 Séance 2 et 3 : optimisation discrète et optimisation multi-objectif

2.1 Rangement d'objets (optimisation combinatoire)

1. **Question préliminaire**

   Soit $i, j \in [|1; n|]$, puisque $x_{ij} = 0$ ou $x_{ij} = 1$, la boîte i contient un objet et un seul ssi $\sum_{j=1}^{n} x_{ij} = 1$ et l'objet j contient un objet et un seul ssi $\sum_{i=1}^{n} x_{ij} = 1$

2. **PLNE**

   Soit $c \in \mathbb{R}^{n^2 \times 1}$ défini par $c_{ij} := ||O_j - B_i||$ et $x := (x_{ij})_{ij}$

   La fonction coût liée au coût de déplacement est la somme totale des déplacements effectués pour ranger tous les objets dans toutes les boîtes. Comme la décision de ranger l'objet j dans la boîte i est encodée par la variable binaire x_{ij}, ce coût s'exprime par :

$$\sum_{j=1}^{n} \sum_{i=1}^{n} x_{ij} ||O_j - B_i|| = c^T x$$

   Pour les contraintes : il s'agit de contraintes d'égalité énoncées dans la question préliminaire et que l'on exprime par $Ax = b$ avec :
   $\mathrm{A} = \left(\begin{array}{c|c|c|c} A_1 & A_2 & ... & A_n \\ \hline I_n & I_n & I_n & I_n \end{array}\right) \in \mathbb{M}_{2n,n^2}$
   où $A_i = e_i[\mathbf{1}]^T \in \mathbb{M}_{n,n}$, $[\mathbf{1}]^T = (1, 1, ..., 1)$ et $b = [\mathbf{1}]$

   On en déduit ainsi la formulation du problème sous le format PLNE :

$$\begin{array}{l} \min\limits_{x \in \mathbb{R}^n} c^T x \\ Ax = b \\ x_{ij} \in \{0; 1\} \end{array}$$

   Résolution numérique : La valeur optimale du coût de déplacement fournie par la méthode *intlinprog* de MATLAB est environ de 15.3776

   On peut représenter le vecteur x solution par une matrice avec le numéro des lignes j qui correspond au numéro des objets et le numéro des colonnes i qui correspond au numéro des boîtes.

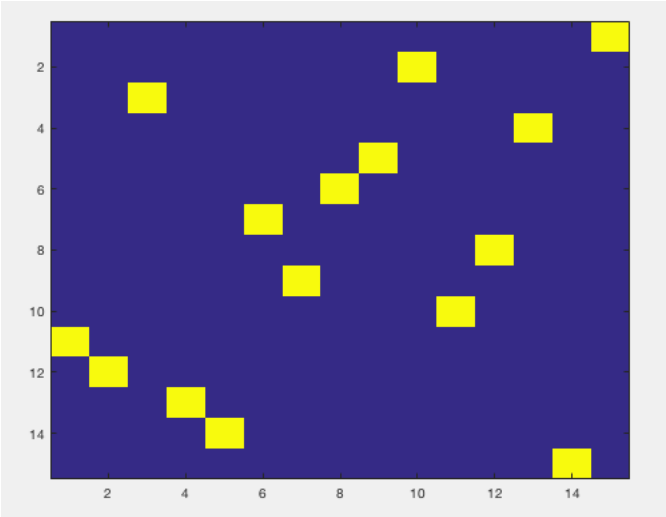

Figure 3: Matrice de la solution au PLNE - l'objet 11 est dans la boîte 1

3. **Ajout contrainte** 1 **au PLNE**

   La contrainte "*l'objet 1 doit se situer dans la boîte située juste à gauche de la boîte contenant l'objet 2*" équivaut à "*si la boîte* $i+1$ *contient l'objet 2, alors la boîte* i *contient l'objet 1 et sinon, la boîte* i *ne contient pas l'objet 1*". On obtient ainis la condition :

$$x_{i,1} = x_{i+1,2} \quad (2 \leq i \leq n)$$

   Il en résulte l'ajout d'une contrainte d'inégalité au PLNE précédent.

   Résolution numérique : La valeur optimale est environ de 15.5651.

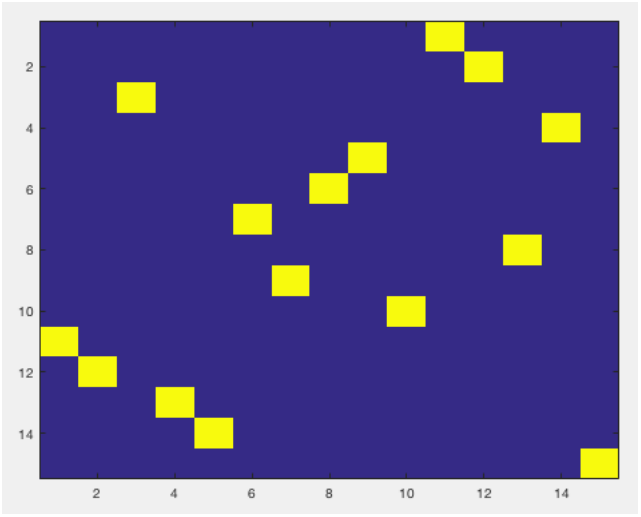

Figure 4: Matrice de la solution au PLNE + contrainte 1 - l'objet 1 est en 11 juste à gauche de l'objet 2 qui est en 12

4. **Ajout contrainte 2 au PLNE**

   On raisonne par contraposée : $\exists i_0, \exists k_0 > 0, x_{i_0,3} = 1$ et $x_{i_0+k_0,4} = 1$ équivaut à dire que "*l'objet 4 se situe à droite de l'objet 3*" ssi "*l'objet 3 se situe à gauche de l'objet 4*" ssi **non**("*l'objet 3 se situe à droite de l'objet 4*").
   Ce qui démontre l'équivalence des deux propositions contraires.

   Résolution numérique : La valeur optimale est environ de 15.9014.

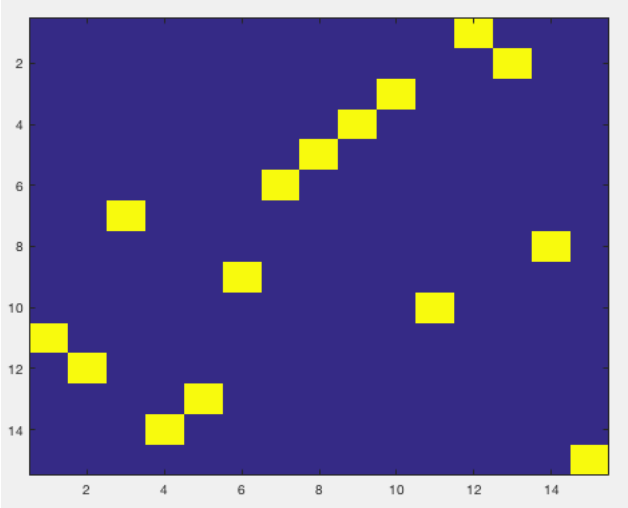

Figure 5: Matrice de la solution au PLNE + contrainte 2 - l'objet 3 est en 11 à droite de l'objet 4 qui est en 10

5. **Ajout contrainte 3 au PLNE**

   La condition "*l'objet 7 se situe à côté de l'objet 9*" équivaut à dire que "*l'objet 7 est à gauche de l'objet 9 ou à droite de l'objet 9*" ssi $x_{i,7} = x_{i+1,9}$ ou $x_{i,7} = x_{i-1,9}$.
   Or, comme $\sum_{i=1}^{n} x_{i9} = 1$, $\exists!\ i_0$ / $x_{i_0,9} = 1$. La condition de contrainte équivaut donc à :

$$x_{i,7} = x_{i+1,9} + x_{i-1,9} \quad (2 \leq i \leq n-1)$$

   Résolution numérique : La valeur optimale est environ de 15.9828.

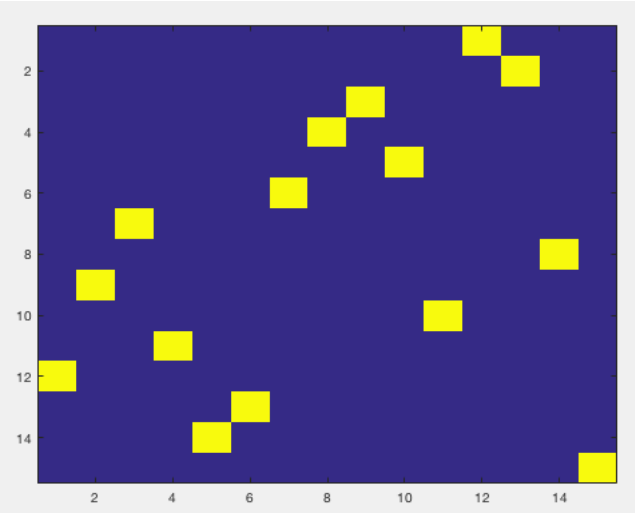

Figure 6: Matrice de la solution au PLNE + contrainte 3 - l'objet 7 est en 3 à droite de l'objet 9 qui est en 2

6. **Optimalité de la solution**

   Pour vérifier l'uncité de la solution, on peut reprendre le problème PLNE + contrainte 3, précédent et y intégrer cette contrainte : $v^T(x - x_{opt}) \leq \delta$ avec $v \in \mathbb{R}^n$ et $\delta < 0$. Si x satisfait ces conditions et $c^T x$ (fonction coût) a la même valeur, cela prouve que la solution x_{opt} obtenue n'est pas la seule solution optimale.
   Vérification numérique : pour $\delta = -0.01$ et v un vecteur aléatoire, on obtient une solution au problème avec la nouvelle contrainte et ayant la même valeur pour la fonction coût.

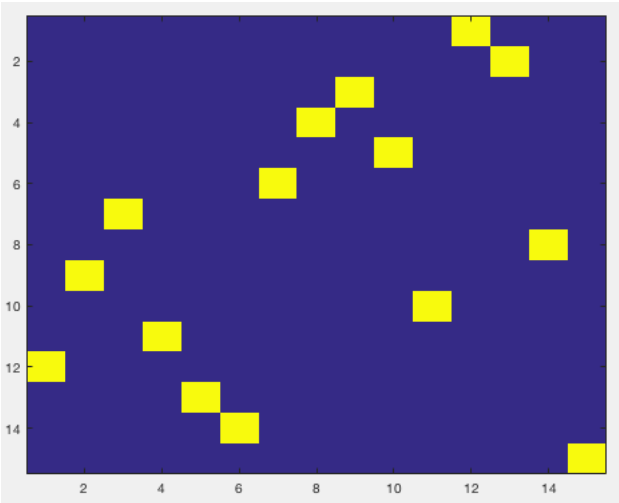

Figure 7: Matrice de la solution au PLNE + contrainte 3 et 4 - cette solution a intervertit les objets 13 et 14 de place par rapport à x_{opt} précédent

2.2 Communication entre espions (optimisation combinatoire)

1. **Modélisation**

   Soient deux agents indexés par i et j. La probabilité d'interception de la communication entre ces deux agents est notée p_{ij}. On cherche à minimiser la probabilité totale d'interception d'un message entre tous les agents. Pour cela on peut maximiser la probabilité de non-interception d'un message. Cette probabilité vaut :

   $$\prod_{i,j}(1 - p_{ij})$$

   En prenant le *log* de cette quantité on se ramène ainsi à maximiser une certaine somme puis en prenant l'opposé cela revient à minimiser une somme de coûts c_{ij} : $\sum_{i,j} c_{ij}$ avec $c_{ij} = -log(1 - p_{ij})$

   On propose alors de représenter le problème sous forme d'un graphe $\mathcal{G}$. Les données du graphes sont les suivantes :
   (i) Les noeuds du graphes sont indexées par les numéros $i \in [|1; n|]$ des agents.
   (ii) Pour chaque arrête (i, j) du graphe, on associe un coût qui vaut c_{ij} défini plus haut (par symétrie on a $c_{ij} = c_{ji}$).

   Le problème peut alors être formulé comme la recherche d'un arbre recouvant minimal sur $\mathcal{G}$.

2. **Résolution**

   Pour cela nous utilisons la méthode *graphminspantree* de la toolbox $Bioinformatics$ de MATLAB.
   Numériquement, la probabilité d'interception vaut $0,5809$
   Et la visualisation de l'arbre recouvrant minimal est la suivante :

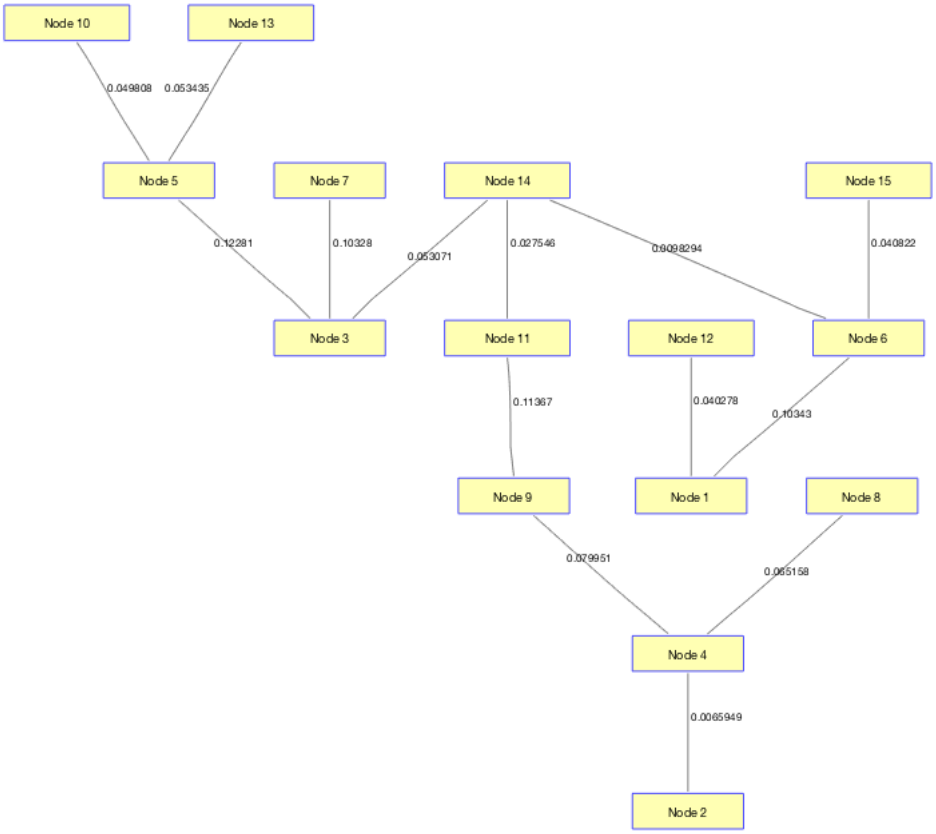

Figure 8: Résultat de l'algorithme Minimum Spanning Tree pour le problème des espions

2.3 Dimmensionnement d'une poutre (optimisation multiobjectif)

1. **Méthode Gloutonne**

   On génère aléatoirement N points (a, b) vérifiant les contraintes :

   $$0.02 \leq a \leq 1$$

   $$0 \leq b \leq a - 0.01$$

   Puis on représente sur un plan les points $(p(a, b), d(a, b))$. Cela nous permet ensuite de déterminer le front de Pareto de rang 1 de cet ensemble de points : il s'agit de l'ensemble des points "non-dominés" par un autre point. On les détermine en recherchant les points pour lesquels $p(a, b)$ et $d(a, b)$ ne peuvent pas être améliorés.

La figure qui suit a été réalisée pour $N = 10^4$. En revanche pour $N \geq 10^5$, la méthode gloutonne est inefficace (MATLAB sur un PC portable ne renvoit pas de résultat pour ces valeurs).
On obtient 340 points pour approximer le front de Pareto (de l'ordre de $\sqrt{N}$).

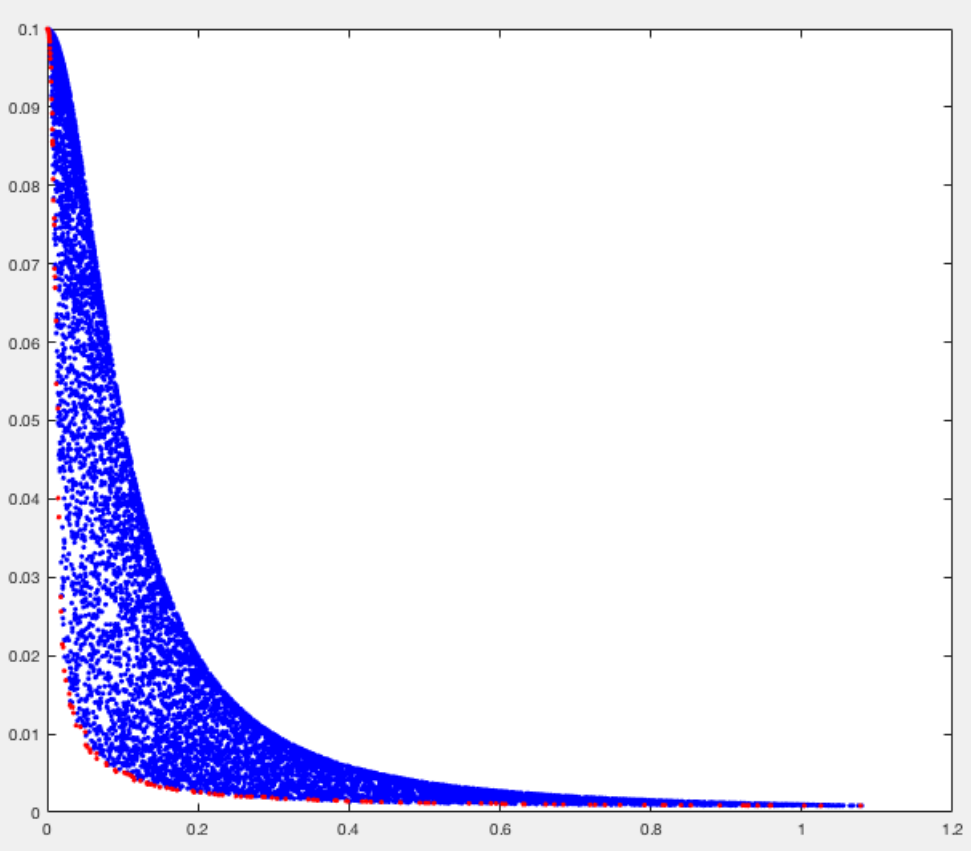

Figure 9: Front de Pareto de rang 1 (en rouge) des points (poids,déflexion) (en bleu)

2. **Méthode plus sophistiquée : par pondération**
   Pour $\alpha \geq 0$, on introduit le critère agrégé suivant :

$$J(a,b) = p(a,b) + \alpha d(a,b)$$

   À chaque valeur de α, on peut résoudre le problème de minimisation sous contraintes (sur un ensemble convexe) :

$$\begin{array}{l} \underset{(a,b)}{min}\ J(a,b) \\ 0.02 \leq a \leq 1 \\ 0 \leq b \leq a - 0.01 \end{array}$$

2.4 Approvisionnement d'un chantier (optimisation combinatoire)

On se propose de modéliser le problème d'approvisionnement à résoudre sous la forme d'un PLNE.
On note N la dimension du problème (le nombre de semaines). Pour chaque semaine i, le nombre $\hat{d}_i$ de tractopelles louées peut être décomposé en trois membres :

$$\hat{d}_i = loc_i + new_i - gone_i$$

avec loc_i le nombre de locations reconduites entre les semaines $i-1$ et i (coût $p_1 = 200$ *u.a.*) ; new_i le nombre de locations nouvelles entre les semaines $i-1$ et i (coût $p_2 = 1000$ *u.a.*) et $gone_i$ le nombre de locations arrêtées entre les semaines $i-1$ et i (coût $p_3 = 1200$ *u.a.*).

(i) Une des premières contraintes issue du problème est : $\hat{d}_i \geq d_i$ (stock requis à la semaine i).

Du fait de la modélisation choisie nous avons aussi les contraintes :
(ii) $loc_i \leq loc_{i-1} + new_{i-1}$ (*d'une semaine à l'autre on ne peut pas reconduire plus de contrats que ceux de la semaine précédente - idée de stock maximal disponible à la semaine i*)
(iii) $\sum_{i=1}^{N} gone_i = \sum_{i=1}^{N} new_i$ (*pendant les travaux, tous les tractopelles qui ont été louées doivent être remis à leur propriétaire à la fin et pas seulement ceux de la dernière semaine - idée de flux sortants qui sont égaux aux flux rentrants*)

On cherche donc un vecteur $x \in \mathbb{R}^{3N+3}$ représentant $\begin{pmatrix} loc_1 \\ new_1 \\ gone_1 \\ ... \\ loc_{N+1} \\ new_{N+1} \\ gone_{N+1} \end{pmatrix}$ avec les contraintes supplémentaires :

$$loc_1 = 0 \ ; \ gone_1 = 0 \ ; \ loc_{N+1} = 0 \ ; \ new_{N+1} = 0 \ ; \ gone_{N+1} = d_N$$

De plus, loc_i, new_i et $gone_i$ sont positifs et bornés supérieurement par $\sum_{i=1}^{N} d_i$ (au pire l'entreprise loue tous les tractopelles requis d'un coup)

Comme toutes les contraintes sont linéaires en x, on peut donc légitimement se ramener à PLNE où l'on cherche à minimiser la fonction coût $c^T x$ avec $c \in \mathbb{R}^{3N+3}$ et $c = (p_1 p_2 p_3 ... p_1 p_2 p_3)^T$

Résolution numérique : La valeur de la solution optimale fournie par MATLAB est de 3 311 200 *u.a.*