Sorbonne Université
Master 2 - Androide

Emmanuelle AMANN
Hector KOHLER

Modèles et algorithmes pour la décision multicritère et collective

Elicitation incrémentale et recherche locale pour le problème du sac à dos multi-objectifs

But du projet :

Développer des procédures de résolution du problème, basées sur l'élicitation incrémentale, et comparer leur efficacité

23 Janvier 2022

1 Introduction - Le problème du sac à dos

Le problème du sac à dos multi-objectifs peut être formulé comme suit :

$$\begin{aligned} \max z_k(x) &= \sum_{i=1}^{n} c_k^i x_i && k = 1, \ldots, p \\ \text{subject to } & \sum_{i=1}^{n} w_j^i x_i \leq W && j = 1, \ldots, m \\ & x_i \in \{0, 1\} && i = 1, \ldots, n \end{aligned}$$

$z_k(x)$ représente la valeur du critère k de la solution x. W est la capacité totale du sac à dos. On cherche donc un ensemble de solutions efficaces X_e tel que pour tout $x \in X_e$ on ait le vecteur $z(x)$ non-dominé au sens de Pareto. On peut obtenir un ensemble X_e approché $\hat{X}_e$ avec la Recherche Locale de Pareto (PLS). Cet ensemble approché servira de point de départ pour certaines procédures d'élicitaion incrémentale étudiées.

1.1 Implémentation de PLS

On implemente PLS selon l'algorithme suivant :

Algorithm 1 Recherche Locale Pareto (PLS)

Paramètres $\downarrow$: une population initiale P_0, une fonction de voisinage $V(x)$.
Paramètres $\uparrow$: une approximation $\widetilde{X}_E$ de l'ensemble des solutions efficaces X_E.

--| Initialisation de $\widetilde{X}_E$ et d'une population P avec la population initiale P_0
$\widetilde{X}_E \leftarrow P_0$
$P \leftarrow P_0$
--| Initialisation d'une population auxiliaire P_a
$P_a \leftarrow \varnothing$
while $P \neq \varnothing$ **do**
  --| Génération de tous les voisins p' de chaque solution $p \in P$
  for all $p \in P$ **do**
    for all $p' \in V(p)$ **do**
      --| si p' n'est pas dominé par p
      if $f(p) \not\preceq f(p')$ **then**
        if `MiseAJour`$(\widetilde{X}_E \updownarrow, p' \downarrow)$ **then**
          `MiseAJour`$(P_a \updownarrow, p' \downarrow)$
  --| P est composée des nouvelles solutions potentiellement efficaces qui viennent d'être trouvées
  $P \leftarrow P_a$
  --| Réinitialisation de P_a
  $P_a \leftarrow \varnothing$

FIGURE 1 – Algortihme PLS

La procédure de mise à jour de la liste des solutions mutuellement non-dominées est un parcours complet de la liste X afin de déterminer si la solution x doit être ajoutée ou pas. De plus, si la solution x est ajoutée, toutes les solutions de X dominées par x sont retirées de la liste. Cette procédure renvoie un booléan égal à Vrai si la solution a été ajoutée dans la liste.

L'initialisation de la population se fait aléatoirement parmi les solutions réalisables.

La fonction de voisinage utilisée est la fonction des échanges 1-1 (pour générer les voisins d'une solution, on teste toutes les combinaisons de solutions en remplaçant un seul objet). Une étude plus détaillée de PLS et des ses variantes est fournie en Annexes dans le contexte bi-objectifs.

2 Première procédure de résolution

Pour la première procédure de résolution nous nous sommes basés sur la partie 3 de [1]. Cette procédure repose sur le principe de minMax regret pour un ensemble de solution possible. Pour cela nous nous basons sur un ensemble de solutions non-dominées $\mathcal{X}$, appartenant donc au front de Pareto, que nous avons générés à l'aide de la méthode PLS décrite ci-dessus.

Un vecteur de poids w est choisit aléatoirement pour simuler les préférences du décideur auquel nous poserons des questions.

A partir de l'ensemble des solutions du front de Pareto, on cherche le vecteur de poids associé à un couple (x, y) de solutions qui minimisera le plus grand regret que le décideur pourrait avoir si l'on se trompait dans ses préférences. Pour chaque couple de préférences $x \succsim y$ on cherche donc le vecteur de poids qui maximisera le regret si l'on venait à choisir y au lieu de x.

Parmi tous les couples de préférence on garde celui minimisant le plus grand regret et on demande au décideur qu'elles sont ses préférences pour ce couple (x, y). A partir de sa réponse, on garde en mémoire sa préférence et l'on cherche un nouveau couple (x', y') par la même procédure.

Cette méthode se base sur une différence pondérée des vecteur des solutions, c'est la méthode de la somme pondérée *Weighted Sum*. La méthode *OWA* s'effectue de la même manière, seulement elle trie les critères de chaque solution $x \in \mathcal{X}$ dans l'ordre croissant pour ainsi donner plus de poids aux critères les moins intéressants. On fournit en Annexe un tableau regroupant les principales méthodes de la procédure.

3 Deuxième procédure de résolution

La deuxième procédure de résolution étudiée est le Regret-Based Local Search [2]. C'est un algorithme mêlant recherche locale de solutions et élicitation incrémentale. En particulier, la recherche locale se fait a partir d'une solution générée de manière gloutonne et maximisant la somme pondérée $1/n \sum_{j=1}^{n} y_j(i)$: la moyenne des valeurs des objectifs de l'objet i .

On génère ensuite les voisins de la solution courante par échanges 1-1 (un objet sort du sac et un objet rentre dans le sac). On ne garde que les voisins non-dominés. C'est sur l'ensemble des voisins non-dominés que l'on applique la procédure d'élicitation incrémentale décrite plus haut. L'algorithme est décrit sur la figure 2.

Algorithm 1 RBLS

IN ↓ P: a multi-objective knapsack problem; f_ω: a family of rank-dependent operators; Θ: a set of statements; δ: a positive threshold.
- -| Initialization:
$\Omega_\Theta \leftarrow \{\omega : \forall (a,b) \in \Theta, f_\omega(a) \geq f_\omega(b)\}$
$x^* \leftarrow$ `ComputeInitialSolution`(P)
$it \leftarrow 0$
improve ← **true**
- -| Local Search:
while improve and $(it < max\_it)$ **do**
  - -| Generation of neighbors:
  $X^* \leftarrow$ `ComputeAllSwaps`(x^*)
  - -| Standard regret-based elicitation:
  while $MMR(X^*, \Omega_\Theta) > \delta$ **do**
    - -| Ask the DM to compare two solutions:
    $(a,b) \leftarrow$ `AskQuery`(X^*, Ω_Θ)
    - -| Update preference information:
    $\Theta \leftarrow \Theta \cup \{(a,b)\}$
    $\Omega_\Theta \leftarrow \{\omega : \forall (a,b) \in \Theta, f_\omega(a) \geq f_\omega(b)\}$
  end while
  - -| Move to another solution:
  if $MR(x^*, X^*, \Omega_\Theta) > \delta$ **then**
    $x^* \leftarrow$ `Select`$(\arg\min_{x \in X^*} MR(x, X^*, \Omega_\Theta))$
    $it \leftarrow it + 1$
  else
    improve ← **false**
  end if
end while
return x^*

FIGURE 2 – Algorithme RBLS

4 Résultats expérimentaux

Dans cette partie nous présentons des résultats expérimentaux sur les procédures décrites. Le processus expérimental est simple. L'instance du problème du sac à dos est commune à toutes les expériences : elle contient 20 objets et 3, 4 ou 5 critères à agréger.

4.1 Processus expérimental

. Dans un premier temps, on génère avec PLS un ensemble de solutions X pour chaque sous-instance de 20 objets et p critères. Cet ensemble X servira d'entrée à la méthode d'élicitation incrémentale directe.

Dans un second temps, pour chaque sous-instance, on génère 20 Decision Makers fictifs par tirage uniforme et normalisation. On rappelle qu'un DM est un modèle linéaire f_w.

Enfin, on fait tourner les boucles principales des méthodes étudiées en enregistrant la valeur du regret minimax par *query* . A noter que chaque méthode est étudiée pour apprendre soit un agrégateur *OWA* (somme pondérée des valeurs ordonnées des critères), soit un agrégateur *WS* (somme pondérée des valeurs des critères). On enregistre aussi le temps nécessaire pour exécuter la boucle principale.

Comme on a 20 DM fictifs par sous-instance, tous les résultats présentés sont des moyennes.

4.2 Résultats

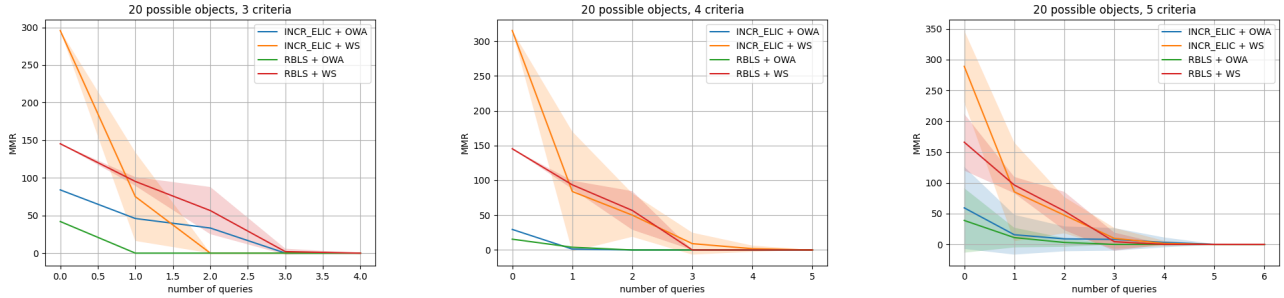

FIGURE 3 – Evolution of MMR per query

	3 criteria	4 criteria
Elicit + WS	1 ±0.1	65.5 ±12.9
Elicit + OWA	1.4 ±0.1	42.8 ±0.4
RBLS + WS	0.7 ±0.1	1.4 ±0.2
RBLS + OWA	0.3 ±0.1	1.2 ±0.2

TABLE 1 – Resolution time in seconds

Avant tout, merci de noter que par manque de temps, les résultats obtenus sur les sous-instances de $p = 5$ critères sont moyennés sur 5 DM différents seulement (d'où les écart-types importants sur la figure 3 pour 5 critères) et que les résultats de temps d'exécution n'ont pas pu être enregistrés.

Le résultat principal de ce rapport est la meilleure performance, quelque soit la méthode utilisée ou bien le nombre de critère de la sous-instance, issue de l'utilisation d'un agrégateur *OWA*. On voit sur le figure 3 que quelque soit le nombre de critères, la variante *OWA* d'une méthode réduit le MMR en moins de *queries* que la variante *WS*.

Un autre résultat est l'invariance des performances face au nombre de critères des méthodes apprenant un agrégateur *WS* comme vu sur la figure 3.

Par contre, plus le nombre de critères augmente, plus le MMR initial semble être faible pour les méthodes apprenant un agrégateur *OWA* comme vu sur la figure 3. Cela reste à confirmer en générant les résultats sur la sous-instance à 5 critères sur plus de DM.

Enfin, de façon générale, d'après le tableau 1, la méthode RBLS est plus rapide que la méthode par élicitation directe. De façon générale, le temps d'exécution augmente avec le nombre de critères et c'est la méthode par élicitation directe pour qui cela est le plus flagrant (un critère en plus dans la sous-instance multiplie le temps d'exécution par 50).

Références

[1] Nawal Benabbou, Christophe Gonzales, Patrice Perny, Paolo Viappiani *Minimax Regret Approaches for Preference Elicitation with Rank-Dependent Aggregators*, 2015.

[2] Nawal Benabbou and Cassandre Leroy and Thibaut Lust, *Regret-Based Elicitation for Solving Multi-Objective Knapsack Problems with Rank-Dependent Aggregators*, 2020.

5 Annexes

5.1 PLS

Dans le TME sur les méta-heuristiques nous avons étudié différentes variantes de la méthode PLS. Appelons PLS1 la version basique de PLS définie dans la figure 1.

PLS2 correspond alors à PLS1 avec une méthode de mise à jour des solutions mutuellement non-dominées plus efficace (X trié selon un objectif).

PLS3 correspond à PLS2 avec une population initiale de meilleurs qualité.

PLS4 correspond à PLS3 avec une méthode de mis à jour des solutions mutuellement non-dominées encore plus efficace.

5.1.1 Résultats Expérimentaux

On utilise une instance du problème du sac à dos bi-objectifs avec 100 objets disponibles. On fournit une visualisation des fronts de Pareto approches obtenus avec PLS1 PLS2 PLS3 et PLS4 après 5 mise à jour de l'ensemble des solutions mutuellement non-dominées. De plus on visualise aussi le front de Pareto exact fourni avec l'instance. Enfin, on présente un tableau des temps de résolution.

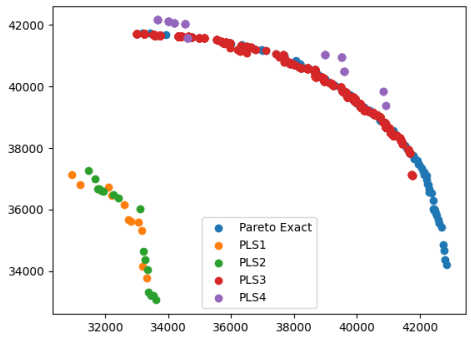

FIGURE 4 – Fronts de Pareto approchés et exact

	Times
PLS1	40
PLS2	33
PLS3	3246
PLS4	3

TABLE 2 – Execution times in seconds PLS variants

Les résultats coïncident avec ceux attendus dans le TME. A savoir, PLS2 est plus rapide que PLS1, et PLS3 et PLS4 approchent mieux le front exact. Cependant, le temps de résolution de PLS3 est grand et on a observe que la taille de la population courante explosait au fil des itérations. Enfin il semblerait que PLS4 découvre des nouvelles solutions non-dominées Pareto-optimales.

5.2 Elicitation incrémentale : principales méthodes

nom	arguments	retourne	pseudo code
`main_(X)`	`X` : solutions d'un front de Pareto	le couple (x, y) minimisant le regret maximal, le dernier vecteur de poids w trouvé qui minimisera le regret maximal des préférences de DM	récupère le couple (x, y), le regret minMax, le vecteur de poids w associé et la préférence z renvoyé par le **Decision Maker**
`CSS(X,P)`	`X` et `P` : ensemble des préférences (x, y) connues	le couple de préférence (x, y), la valeur, le vecteur de poids w et la préférence z de DM	récupère la solution de `MMR(X,P)` et demande la préférence du **DM** pour le couple (x, y)
`MMR(X,P)`	les ensembles `X` et `P`	le couple de préférence (x, y), la valeur minimale $min\_$ et le vecteur de poids w	recherche de la solution $x \in \mathcal{X}$ minimisant la valeur retournée par `MR(x,X,P)` = y, valeur, w
`MR(x,X,P)`	`X`, `P` et `x` : une solution de $\mathcal{X}$	solution y, la valeur $max\_$ et le vecteur de poids w	recherche de la solution $y \in \mathcal{X}$ maximisant la valeur retournée par `PMR_owa(x,y,P)` = valeur, w
`PMR_owa(x,y,P)`	`P`, `x` et `y` : une solution de $\mathcal{X}$, $x \neq y$	`PMR_ws(x_sort, y_sort)`	tri décroissant des critères de x et y, appel à `PMR_ws` avec les solutions x et y triées
`PMR_ws(x,y,P)`	`P` et 2 solutions `x` et `y`	valeur objective du PL, vecteur poids w	