Optimisation Combinatoire - TSP

Quentin Baert - quentin.baert@etudiant.univ-lille1.fr

4 janvier 2016

L'objectif du projet est d'explorer et d'analyser des heuristiques pour résoudre un problème multi-objectifs. Nous avons étudiés le problème TSP qui consiste à trouver un chemin optimal qui passe par plusieurs villes en passant une seule fois par chacune d'elles. Nous avons chercher à minimiser deux objectifs (que l'on peut voir comme la distance et le temps par exemple). Une solution d'une instance de mTSP (TSP multi-objectifs) consiste à un ordonnancement des villes. L'ordonnancement doit être fait de sorte à minimiser les deux objectifs.

Toutes les solutions ont été évaluée avec la même fonction d'évaluation (que l'on note c). Si on note $X(v_i, v_j)$ le coût de l'objectif X entre les villes v_i et v_j, on a pour deux objectifs A et B, et pour une solution s qui contient n villes, le calcul suivant :

$$c(s) = \Big(\sum_{i=1}^{n-1} A(v_i, v_{i+1}) + B(v_i, v_{i+1}) \Big) + A(v_n, v_1) + B(v_n, v_1)$$

1 Filtres

La première étape du projet fut d'implémenter des filtres. Le but d'un filtre est de parcourir une population de solutions pour ne garder que celles qui **ne sont pas dominées**. On dit qu'une solution est dominée s'il existe une autre solution dont le coût est meilleur (inférieur pour nous) pour tous les objectifs (2 pour nous). Ces solutions non dominées constituent le **front Pareto** de l'ensemble de solution.

1.1 Filtre off-line

Le filtre off-line construit le front Pareto en parcourant toutes les solutions et en comparant la solution courante à l'ensemble des autres solutions que l'on veut filtrer. Si la solution courante n'est dominé par aucune des autres solutions, elle est ajoutée au front Pareto.

Résultats

Voici les performance en temps du filtre off-line implémenté :

Nombre de solutions à filtrer	**Temps (s)**
1000	0,63
10000	5,3
100000	541

la FIGURE 1 présente un front Pareto mis en avant par le filtre off-line. Les points violets correspondent aux point du front pareto.

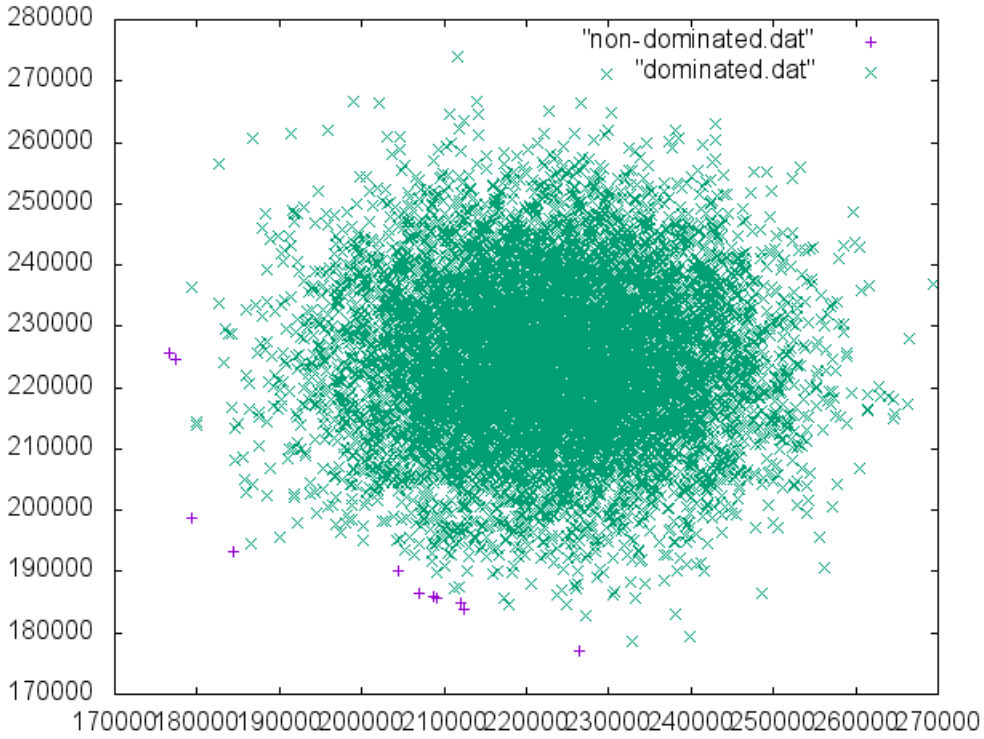

Figure 1: Front Pareto obtenue avec le filtre off-line sur 10000 solutions

1.2 Filtre on-line

Le filtre on-line construit le front Pareto en maintenant une archive des solutions non dominées. Les solutions sont donc traitées séquentiellement et comparées à toutes les solutions de l'archive. Si la solution courante domine une solution de l'archive, la solution dominée est exclue de l'archive. Si la solution courante n'est dominée par aucune solution de l'archive, elle est ajoutée à l'archive.

Résultats

Voici les performance en temps du filtre on-line implémenté :

Nombre de solutions à filtrer	**Temps (s)**
1000	0,595
10000	58,5
100000	>> 500

On peut constater que les performances sont bien moins bonnes que le filtre off-line alors que l'ajout d'une archive devrait les améliorer. Ce défaut de performance est sûrement dûe à une implémentation non optimisée : le filtre implémenté donne lieux à de nombreux appels récursifs et gère l'ancienne archive et la nouvelle archive à chacun de ses appels.

Comme le montre la FIGURE 2, le filtre on-line fait tout de même correctement sa tâche.

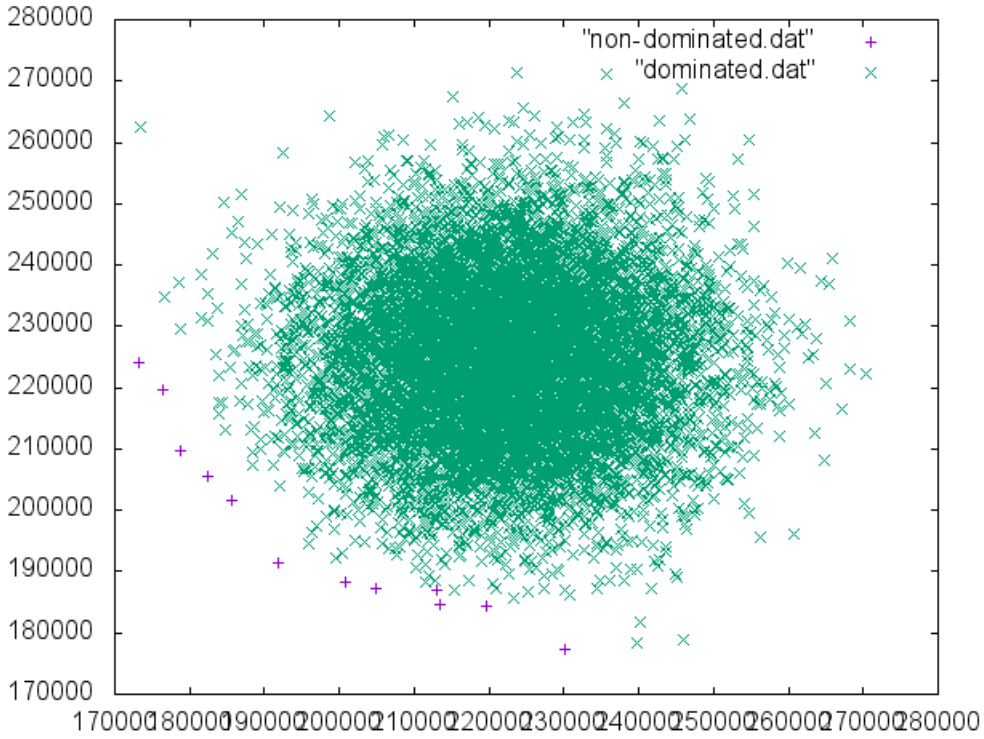

Figure 2: Front Pareto obtenue avec le filtre on-line sur 10000 solutions

2 Algorithmes

L'objectif est ici d'implémenter deux algorithmes qui permettent d'obtenir une approximation du front Pareto. Pour cela, on se base sur des objectifs dont on connait la meilleure approximation du front Pareto et on cherche à s'approcher le plus possible de cette meilleure approximation.

2.1 Approche scalaire

L'approche scalaire consiste à définir des vecteurs de poids qui servent à définir une nouvelle fonction de coût pour ensuite faire une recherche local sur une solution à l'aide de cette fonction de coût.

Un vecteur de coût est de la forme $(\alpha, \beta) \in \mathbb{R}^2$ (car nous avons deux objectifs) avec $0 \leq \alpha, \beta \leq 1$. À partir d'un tel vecteur, on peut créer une nouvelle fonction de coût $c_{\alpha,\beta}$ que l'on peut intépréter comme une fonction de coût mono-objectif qui offre un compromis entre les deux objectifs en fonction des poids α et β.

$$c_{\alpha,\beta}(s) = \Big(\sum_{i=1}^{n-1} \alpha A(v_i, v_{i+1}) + \beta B(v_i, v_{i+1}) \Big) + \alpha A(v_n, v_1) + \beta B(v_n, v_1)$$

L'idée est ensuite, pour chaque vecteur de poids, de générer une solution aléatoire et d'exécuter une recherche locale sur ce point à l'aide de la fonction de coût $c_{\alpha,\beta}$. Visuellement, la recherche locale rapproche le plus possible le point du meilleur front Pareto connu.

La dernière étape de l'approche scalaire est de filtrer l'ensemble des solutions obtenues après une recherche locale pour ne garder que celles non dominées.

Résultats

La FIGURE 3 présente le résultat obtenu sur l'instance *randomAB100* pour une approche scalaire avec 200 vecteurs de poids et une recherche locale effectuée avec un Hill Climbing qui utilise le voisinnage 2-opt et la méthode de sélection first improvment. Cette expérience s'est réalisée en 124 secondes.

On constate que les valeurs sont relativement bien étalées (bonne diversification) et que l'on s'approche fortement du front sans y être collé (intensification correcte).

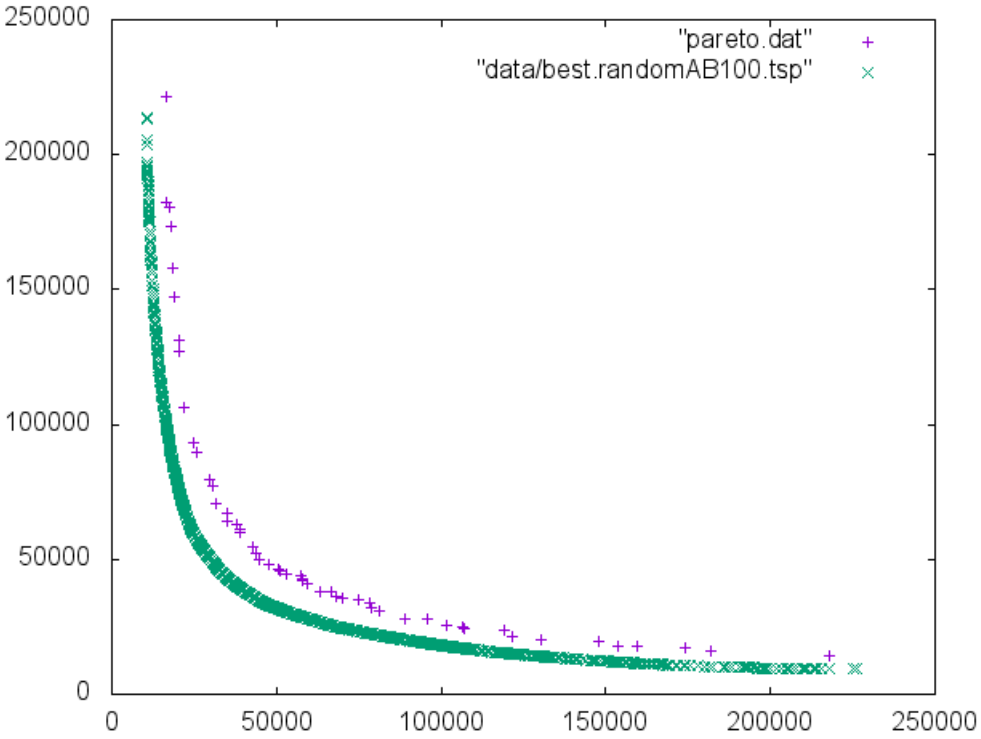

Figure 3: Approximation du front Pareto obtenue pour 200 vecteurs de poids

La FIGURE 4 présente le résultat obtenu dans les mêmes conditions mais avec 300 vecteurs de poids. Cette expérience s'est réalisée en 174 secondes.

On remarque que le résultat est à peu près similaire. Le nombre de vecteurs de poids n'influence pas tellement la qualité de l'approximation du front.

On peut cependant supposer que si la recherche locale était effectuée avec un algorithme plus performant (comme ILS ou VNS), l'approximation serait bien meilleure.

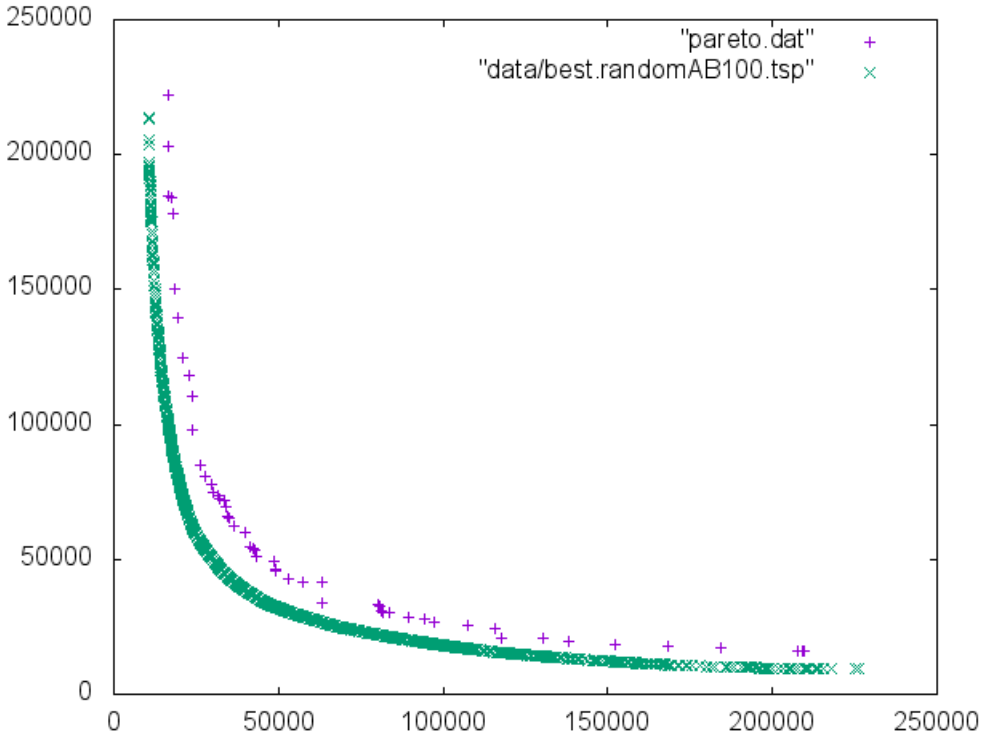

Figure 4: Approximation du front Pareto obtenue pour 300 vecteurs de poids

2.2 Approche Pareto

L'approche Pareto se rapproche plus d'un algorithme évolutionnaire. L'idée est ici d'avoir une population (appelé aussi génération) de solutions courante, de générer le voisinnage d'une ou plusieurs solutions de cette population et d'utiliser un filtre pour obtenir la nouvelle population courante. Chaque nouvelle population constitue donc le front Pareto de la génération précédente.

L'algorithme implémenté considère pour chaque génération un nombre donné de voisins à l'aide de l'opérateur de voisinnage 2-opt, et ce pour chacune des solutions. Le filtre s'effectue ensuite sur l'ancienne population plus l'ensemble de tous les voisins générés. Aux vues des performances des filtres, le filtre off-line est utilisé.

Résultats

La FIGURE 5 présente le résultat de l'approche Pareto implémentée pour 100 générations et 100 voisins par solutions sur l'instance *randomAB100*. Cette expérience s'est exécutée en 124 secondes.

On y constate que le front approximé est plutôt éloigné du meilleur front connu et que la diversification des points n'est pas très satisfaisante : les solutions ne couvrent pas tout l'amplitude du front.

La FIGURE 6 présente le résultat de l'approche Pareto, toujours sur 100 génération mais avec 200 voisins générés par solution. Cette expérience s'est exécutée en 43 minutes.

On remarque que le front obtenu est plus étendu que le précédent mais ne couvre toujours pas toute l'étendue du meilleur front. On peut logiquement supposé que le nombre de voisins par solution augmente la diversité de l'algorithme et permet d'étendre le front final.

La FIGURE 7 présente le résultat de l'approche Pareto sur la même instance mais pour 300 générations et 400 voisins par solutions. Cette expérience s'est exécutée en 4 heures et 36 minutes.

Le résultat est clairement meilleur : les solutions obtenues sont plus proches du front et le couvre mieux. Cependant la partie du front qui minimise le plus les deux objectifs est la moins bien approximée. Cette partie est en fait à peine mieux approximée que pour le front précédent. Peut être qu'un changement de voisinnage

permettrait de passer outre cette limite et de trouver des nouveaux voisins non explorés lors de cette exécution.

On peut conclure de ces trois expériences que le nombre de voisins générés par solution a bien un impacte sur la diversité des solutions finales. On en déduit aussi, de manière assez évidente, que plus il y a de génération, plus l'on s'approche efficacement du meilleur front connu.

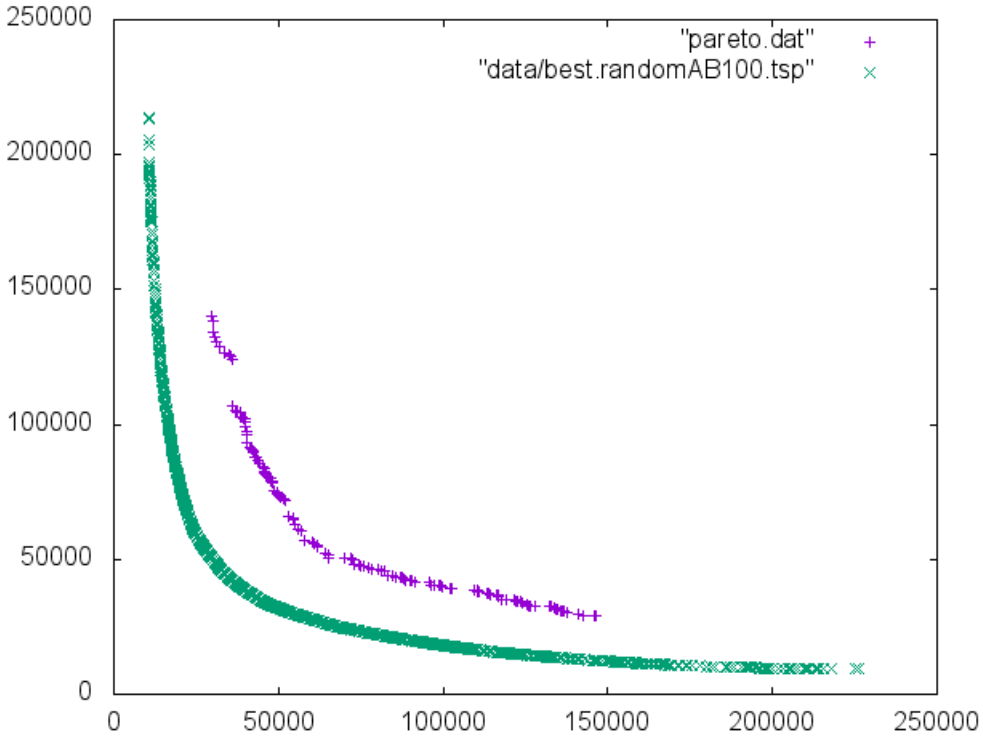

Figure 5: Approximation du front Pareto - 100 générations, 100 voisins

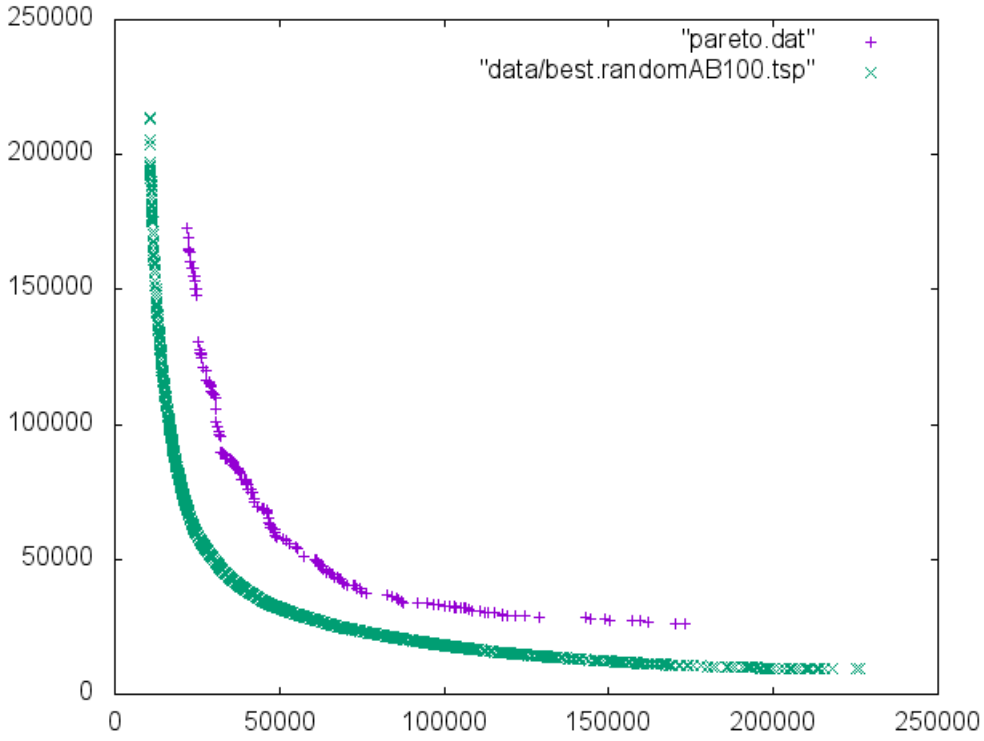

Figure 6: Approximation du front Pareto - 100 générations, 200 voisins

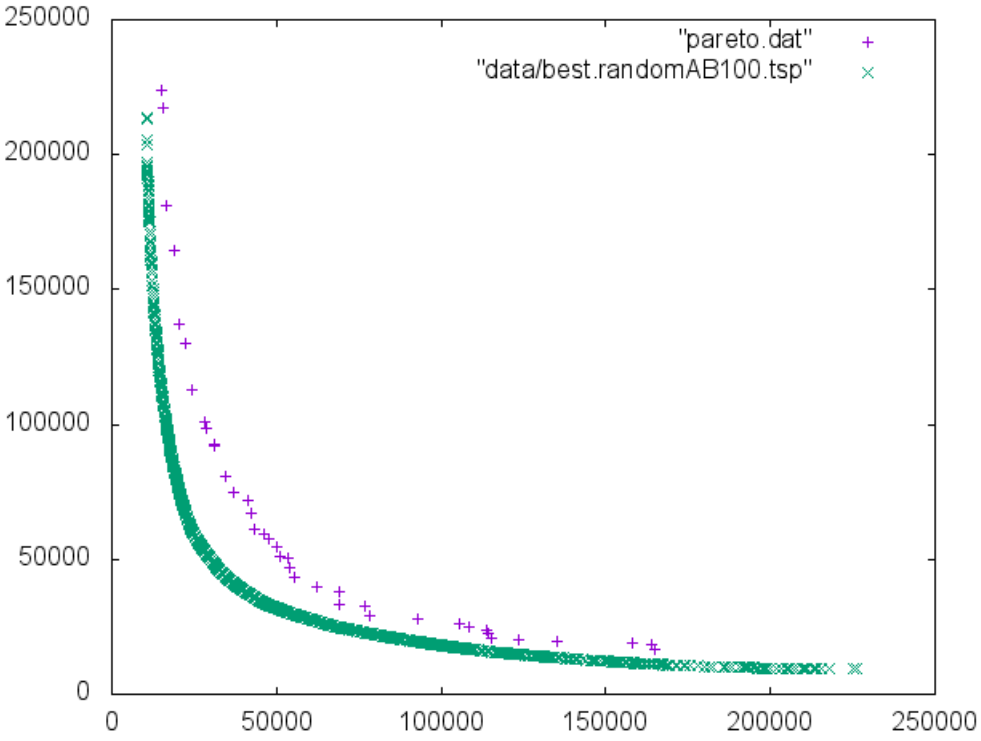

Figure 7: Approximation du front Pareto - 300 générations, 400 voisins

Conclusion

Les deux algorithmes vus ici semblent tous les deux efficaces pour approximer le front Pareto.

On préférera tout de même une approche scalaire avec une recherche locale élaborée tant pour son efficacité à trouver une bonne approximation du front Pareto que pour son temps d'exécution qui est beaucoup plus modéré que celui d'une approche scalaire.