Rapport projet Recherche Opérationnelle, 2020-2021

T.Bersoux,N.Compère,Gr.681

Table des matières

1 Introduction

Dans le cadre de l'UE "Recherche Opérationnelle", nous avons eu à résoudre un problème mélangeant partitionnement, couverture d'ensembles et plus court chemin. Pour cela, deux méthodes étaient proposées : Une première exacte avec solveur, une seconde approchée. Le but était donc d'implémenter ces deux approches et de les tester et de les comparer, afin de savoir s'il est possible de toujours utiliser la méthode exacte, et si la résolution approchée est suffisamment précise dans le cas contraire.

2 Implémentation

2.1 Utilitaires

Ce fichier regroupe quelques fonctions utiles pour les deux approches :

2.1.1 regroupeDistance

Cette structure s'est avérée très pratique dans notre implémentation, puisqu'elle nous permet de réunir en une variable un regroupement, la distance minimale pour relier toutes les villes de ce regroupement, et le cycle qui correspond à cette distance minimale.

2.1.2 poidsTot

Cette fonction permet de calculer le poids total des demandes d'un ensemble de villes. Cela permet de savoir si l'on va, ou non, dépasser la capacité d'un drone de livraison.

2.1.3 estDansEns

Cette fonction renvoie 1 pour la présence d'un élément dans un ensemble (tous deux donnés en paramètre). Le choix de renvoyer 1 ou 0, et non un booléen, n'est pas anodin : Cela permet d'utiliser le retour de cette fonction dans des calculs pour les contraintes de la résolution exacte.

2.2 resolutionExacte

2.2.1 partitionner

On crée les partitions de l'ensemble de villes. Pour cela (Sauf dans le cas où l'on ajoute l'élément seul dans son propre ensemble), on prend un ensemble de villes E et un élément j. Si le poids total de $E \cup \{j\}$ ne dépasse pas la capacité du drones, alors on ajoute $E \cup \{j\}$ à notre ensemble de partitions, et on tente de créer un ensemble $E \cup \{j, j+1\}$ en rappelant récursivement la fonction. Cela retourne bien à la fin l'ensemble partitionné.

2.2.2 regroupement

On appelle juste la résolution du problème du voyageur de commerce sur chaque ensemble crée dans partitionner afin d'associer ces ensembles à un cycle et une plus petite distance. C'est là que la structure regroupeDistance à son intérêt.

2.2.3 resolution

Classique, comme on a fait en TP. Il faut sélectionner des ensembles dans S, l'ensemble de regroupements, afin de parcourir la plus petite distance, tout en s'assurant que chaque ville soit visité une seule fois, ce qui donne le problème linéaire :

$$min\, z = \sum_{i \in S} distanceMin[i] * x_i$$

$$s.c \quad \sum_{i \in S} x_i * y_{ij} = 1 \; pour\, j \in [1, nbVilles],$$

$$et\, y_{ij}\, la\, presence\, de\, la\, ville\, j\, dans\, la\, tournee\, i$$

2.3 resolutionApprochee

Cette méthode construit les tournées au fur et à mesure, en faisant en sorte de fusionner des tournées afin de maximiser les gains de la fusion (gains de réduction de distance)

2.3.1 initApprochee

Construit les tournées initiales, c'est à dire 1→i→1 pour $i \in [1, nbVilles]$ (On supprime plus tard la tournée 1->1->1)

2.3.2 construireMatGains

À partir du distancier, construit la matrice de gains avec le calcul donné dans le sujet.

2.3.3 paires

À partir de la matrice de gains, on fait une liste des paires (i,j) avec $i \in [2, nbVilles]$, $j \in [i, nbvilles]$, $i \neq j$, liste décroissante par rapport au gain apporté par la fusion des tournées contenant i, j.

2.3.4 fusion

Deux étapes de l'algorithme :

— La première étape consiste à fusionner les deux tournées données en paramètre, la paire donnant les deux villes qui réalisent la "jonction" de cette fusion. On travaille sur des copies des tournées afin de faciliter la lecture (Autre choix : utiliser reverse() et récupérer le retour).
Pour une question de cohérence, on a choisi de faire en sorte que cette jonction se fasse toujours au milieu : Ainsi, on a juste à s'assurer que l'élément de jonction de la première tournée se situe à la fin de cette dernière, et que l'élément de jonction de la seconde tournée se situe à son début. Ensuite il suffit de supprimer les 1 de 1→i→j→1 en centre de tournée.

— La seconde étape consiste à mettre à jour la liste des paires dont la fusion est encore possible, c'est à dire que les deux éléments de la paire ne sont pas déjà dans la même tournée, et les deux éléments sont accessibles. Pour cela, nous nous sommes rendus compte qu'il était plus simple/lisible de prendre la suppression à l'envers : Il vaut mieux ajouter dans une nouvelle liste les paires encore valides. Ainsi, il suffit de regarder que les éléments ne soit pas dans la même tournée, et qu'ils sont soit tous deux dans d'autres tournées, soit que l'un deux est dans la tournée et est accessible.

2.3.5 construireTournees

Il suffit d'appeler fusion jusqu'à ce que la liste de paires que l'on peut fusionner soit vide. On vérifie juste avant cela que la nouvelle tournée ne dépassera pas la capacité d'un drone.
Nous avons choisi de placer la fusion résultante dans la case qui contient la première tournée considérée pour la fusion, et de considérer que la seconde tournée est alors vide (Plus pratique lors de l'affichage). Il faut ainsi tenir à jour un tableau "emplacement", qui référence l'emplacement de chaque ville dans les tournées.
Nous avons pensé ensuite que le fonctionnement semblait un peu similaire aux classes-unions ! Mais ne sachant pas comment les implémenter en Julia et manquant de temps, nous conservons notre solution (Peut-être un peu moins lisible, mais aussi efficace.)

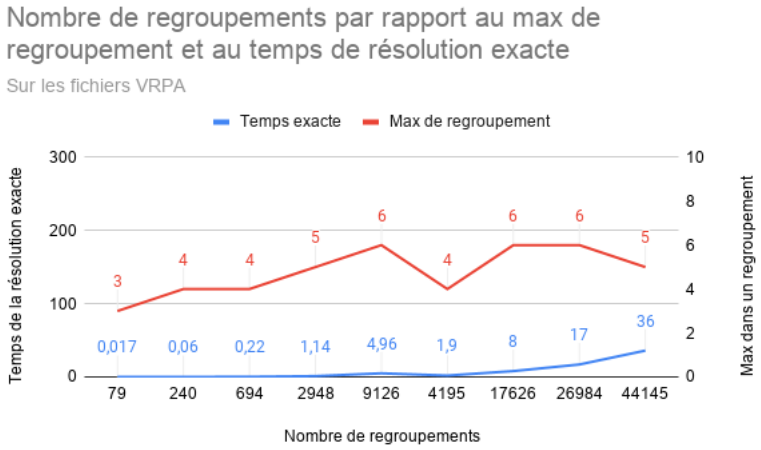
2.3.6 resolution

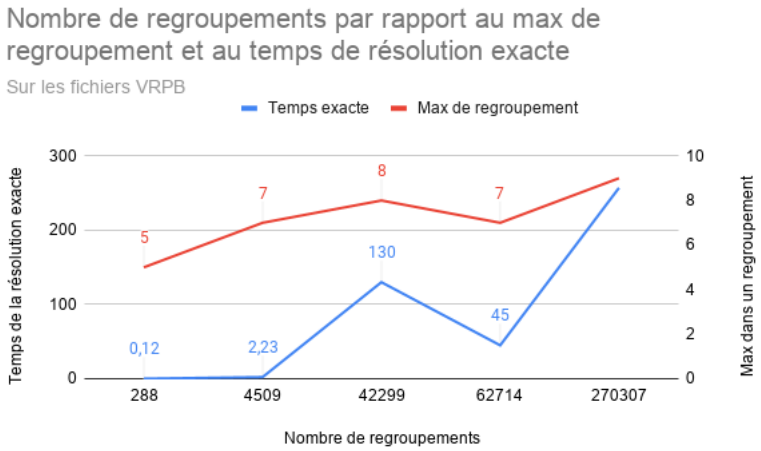
Il suffit d'appeler toutes les fonctions précédentes et d'afficher le résultat. On filtre lors de l'affichage les ensembles vides.

3 Tests

Nous obtenons les résultats attendus pour tous les fichiers, sauf VRPB à partir de 35 villes, où la résolution prend plus de 30min (Nous n'avons pas poussé au delà...).
Les consignes parlaient de donner le temps CPU requis pour le partionnement : Celui-ci étant toujours inférieur à 10^{-7}s, nous ne le considérons pas dans notre analyse.
Nous avons réalisé une fonction de test qui nous permet de lisser les calculs de temps et de nous donner plusieurs informations utiles :

3.1 Regroupements et influence sur la résolution



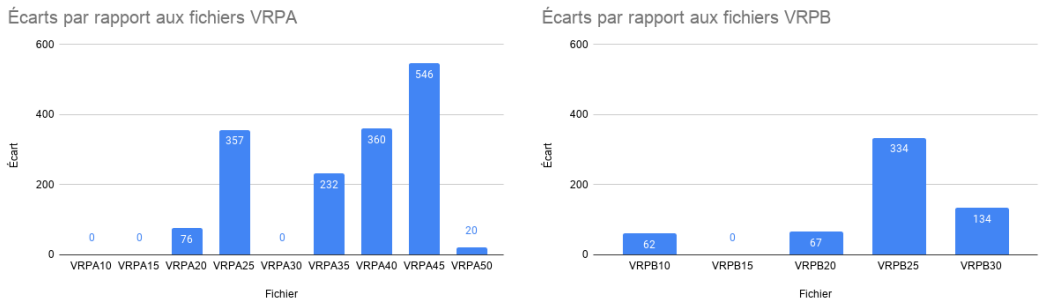


Sur ces deux figures, nous pouvons observer plusieurs choses : Premièrement, le temps de résolution (en secondes) ne dépend pas du maximum de villes dans les regroupements, mais du nombre total de regroupements.
De plus, le nombre de ville n'est pas tellement important, la différence se fait à cause du nombre de regroupements.
Néanmoins, comme on peut le constater en regardant les écarts entre les fichiers A et B, et aussi interne aux fichiers B entre 20, 25 et 30 villes, c'est surtout la disposition des villes et leurs demandes qui entraînent un temps de résolution particulièrement long pour la méthode exacte.
On suppose alors que TSP est surtout influencé par le nombre de regroupements et la taille de ces regroupements, et que la résolution du problème linéaire est surtout influencée par les faibles écarts entre divers ensembles.

3.2 Fiabilité de la résolution approchée et gain de temps

3.2.1 Fiabilité

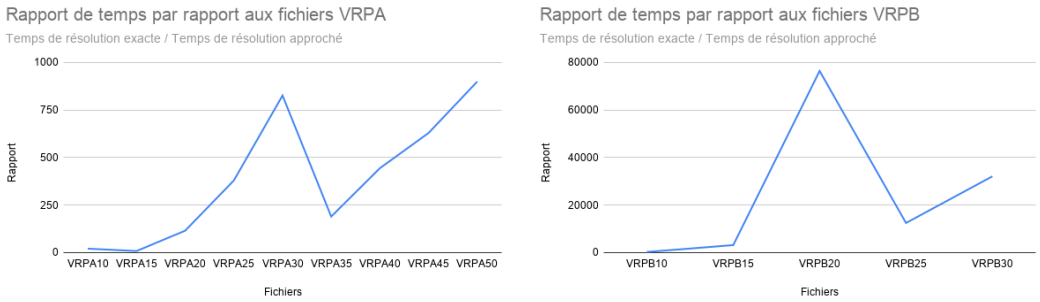
Pour comparer la fiabilité, nous avons tracé les écarts dans la longueur totale des solutions :



On observe que les écarts sont relativement faibles : En effet, ils ne dépassent pas une différence de plus de 5% (Le pire pourcentage, pour VRPA25), puisque les longueurs font plusieurs milliers et que l'écart ne fait pas plus d'un demi-millier... . Dans les cas VRPB avec plus de 30 villes, nous n'avons même pas d'autres choix que d'utiliser la méthode approchée puisque la méthode exacte est trop longue ! Nous n'avons alors pas d'informations sur l'écart, mais on peut supposer que celui-ci reste faible. On peut donc estimer que la méthode approchée est suffisamment fiable pour être utilisée constamment si elle permet un gain de temps important.

3.2.2 Gain de temps

La méthode approchée n'a jamais dépassé 0,04s, tandis que la méthode exacte prend au minimum 30 minutes sur les cas défavorables comme VRPB35, et dépasse souvent la seconde et même les dizaines de secondes. Afin de se rendre compte de l'écart de temps entre ces deux méthodes, nous avons réalisés les graphiques suivants :



Le rapport de temps atteint ainsi facilement plusieurs dizaines de milliers dans les cas les plus défavorables dans VRPB. La méthode approchée, elle, ne dépend que du nombres de villes, et est ainsi plus linéaire. On peut estimer que sur un nombre encore plus grand de villes, le temps de résolution approché reste raisonnable, tandis que le temps de résolution exacte deviendrai vite beaucoup trop grand (Puisque la tendance du rapport est linéaire).

4 Conclusion

Nous sommes satisfaits de notre travail. Le partitionnement est presque instantané et la méthode approchée est très rapide (Une poignée de millisecondes !). Nous obtenons les mêmes résultats que ceux attendus par le professeur, quels que soient les fichiers. Le problème peut ainsi être résolu efficacement : un protocole de résolution peut être de tenter la résolution exacte pendant quelques minutes, et si celle ci n'aboutit pas, d'utiliser la résolution approchée.