LE BARS Yannis
TRAVERS Clément
30/03/17
601A

Rapport Recherche Opérationnelle
BladeFlyer II : conquest of water

Contenu

Introduction

Dans le cadre du module X6I0030 Recherche opérationnelle, nous avons réalisé un problème de tournée de véhicule avec capacités et profits. L'objectif du projet était de combiner l'algorithmique fait en C au solveur GLPK afin de répondre efficacement au sujet.

Un drone, partant d'une base, devait parcourir différents points de pompages afin d'y récupérer l'eau présente sur place sans dépasser la capacité de stockage du drone. Nous devions résoudre le problème en recherchant la plus petite distance parcourue par le drone qui respecte les contraintes énoncées plus tôt.

Nous avons donc créé différents algorithmes comme l'algorithme d'énumération des regroupements réalisables aux points de pompages ou l'algorithme d'énumération des tournées tout en essayant d'utiliser les meilleurs structures de données possibles.

I – Résumé du problème

Les données du problème :

L'ensemble des lieux de 0 à n, n étant le nombre de points de pompages et 0 représentant la base.

Le distancier de chaque lieu i vers un lieu j noté cij, i et j appartenant à l'ensemble des lieux.

La capacité du drone Ca.

La quantité d'eau di disponible en chaque point de pompage i de 1 à n, supposée inférieure à la capacité du drone.

L'autonomie de vol du drone n'est pas pris en compte même si ce serait facile à l'intégrer.

Les variables de décision :

Soit xi = 1 si et seulement si la tournée i est choisit où i est l'ensemble créé suite à l'algorithme de regroupement.

Xi = 0 sinon.

La fonction objectif :

Si l'on note li la longueur de la plus courte tournée visitant chaque point de pompage d'un regroupement, la fonction objectif devient :

min z = sum {de 1 à |i|} li*xi

Les contraintes :
Chaque point de pompage est visité une seule fois.
Pour chaque tour, la quantité d'eau de chaque point de pompage additionnée ne doit pas dépasser la capacité du drone.
Pour chaque regroupement, il faut trouver la combinaison qui renvoit la plus petite longueur.

II – Description de l'algorithme d'énumération des regroupements possibles des points de pompages

```
fonction regroupement (Pointeur sur donnees p, Entiers i, pos, k , Tableau d'Entiers L, t,
Entier r ) : Entier
    Variables
        Entiers j, l, j1, quantite
        Tableau de (p→nblieux-1) Entiers t2
        Tableau d'Entiers tab
    Début
        quantite ← 0
        si (k=i)
          alors p→regroupe.longueur ← réallouer(pos+1)
                p→regroupe.n ← réallouer(pos+1)
                p→regroupe.tab ← réallouer(pos+1)
                p→regroupe.n[pos] = i
                p→regroupe.tab[pos] ← allouer(i)
                pour j de 0 à i-1 faire
                    p→regroupe.tab[pos][j] ← L[j]
                    quantite ← quantite + p→demande[L[j]]
                finpour
                si (quantite ≤ p→capacite)
                  alors tab ← allouer (p→regroupe.n[pos])
                        p→regroupe.longueur[pos] ← combinaisons (p→regroupe.tab[pos],
                                                                 tab, p→regroupe.n[pos], 0, p)
                        libérer tab
                        pos ← pos + 1
                  sinonlibérer p→regroupe.tab[pos]
                finsi
          sinon      pour l de 0 à r-1 faire
                     L[k] ← t[l]
                     j1 ← 0
                     pour j de l+1 à r-1 faire
                         t2[j1] ← t[j]
                         j1 ← j1 + 1
                     finpour
                     pos ← regroupement(p , i, pos, k+1, L, t2, j1)
                finpour
        finsi
        retourner pos
    Fin
```

La fonction regroupement est une fonction récursive sur le nombre de lieux. Elle va mettre dans un tableau tous les regroupements qui commencent par 1, ensuite tous ceux qui

commencent par 2 et cetera jusqu'à ce qu'il n'y en ai plus.

Lorsque la condition du si est validée, on réalloue d'un chaque paramètre de la variable regroupe faisant partie des données. En effet, l'ensemble des parties d'un ensemble vaut 2^n, n étant le nombre d'éléments de l'ensemble. Dans notre cas, comme nous ne comptons pas l'ensemble vide cela fait 2^n-1. Nous nous sommes rendu compte que l'allocation dynamique utilisant uniquement malloc() créait une erreur de segmentation lorsque nous avions beaucoup de points de pompages : pour pouvoir donc utiliser notre programme sur de grosses valeurs, il fallait pouvoir faire des réallouages dynamique.

Ensuite, les quantités d'eau disponibles à chaque point de pompage du regroupement en cours sont additionnées ensembles puis on vérife que cette somme ne dépasse pas la capacité maximale de stockage du drone. Si tel est le cas, on lance l'algorithme combinaison que nous allons expliquer dans la suite de ce rapport.

III – Description de l'algorithme d'énumération des tournées

```
fonction combinaisons ( Tableau d'Entiers tab, tab2, Entiers taille1, taille2,
                        Pointeur sur donnees p) : Entier
    Variables
        Entiers i, j, longueur, min
        Tableau d'Entiers tab3
    Début
        min ← 0 ; longueur ← 0
        si (taille1 ≤ taille2)
            alors pour i de 0 à taille1-1
                        si (i=0)
                            alors longueur ← longueur + p→C[0][tab2[i]]
                            sinon longueur ← longueur + p→C[tab2[i-1]][tab2[i]]
                        finsi
                  finpour
                  longueur ← longueur + p→C[tab2[taille1-1]][0]
                  retourner longueur
            sinon pour i de 0 à (taille1-taille2)-1
                        tab3 ← allouer(taille1-taille2)
                        tab2[taille2] ← tab[i]
                        pour j de 0 à i-1
                            tab3[j] ← tab[j]
                        finpour
                        pour j de i à (taille1-taille2)-2
                            tab3[j] ← tab[j+1]
                        finpour
                        longueur ← combinaisons( tab3, tab2, taille1, taille2+1, p)
                        si (i = 0)
                            alors min ← longueur
                            sinon si (longueur < min)
                                        alors min ← longueur
                            finsi
                        finsi
                        libérer tab3
                  finpour
                  retourner min
        finsi
    Fin
```

Cette fonction recherche la combinaison de points de pompage où la longueur du tour est la plus petite possible. Elle est également récursive où sa récursivité e fait sur le nombre d'éléments restant pour un regroupement.

Dans un tableau, nous allons entrer les différentes combinaisons réalisables en retirant les possibles doublons comme 1,1 et 3,3 pour le regroupement 1,3 par exemple. Puis nous rappelons la fonction pour effectuer toutes les autres combinaisons et retournons la longueur minimal à chaque rappel.

Dans la condition d'arrêt, nous additionnons les différentes distances de la combinaisons pour ensuite renvoyer la longueur la plus petite.

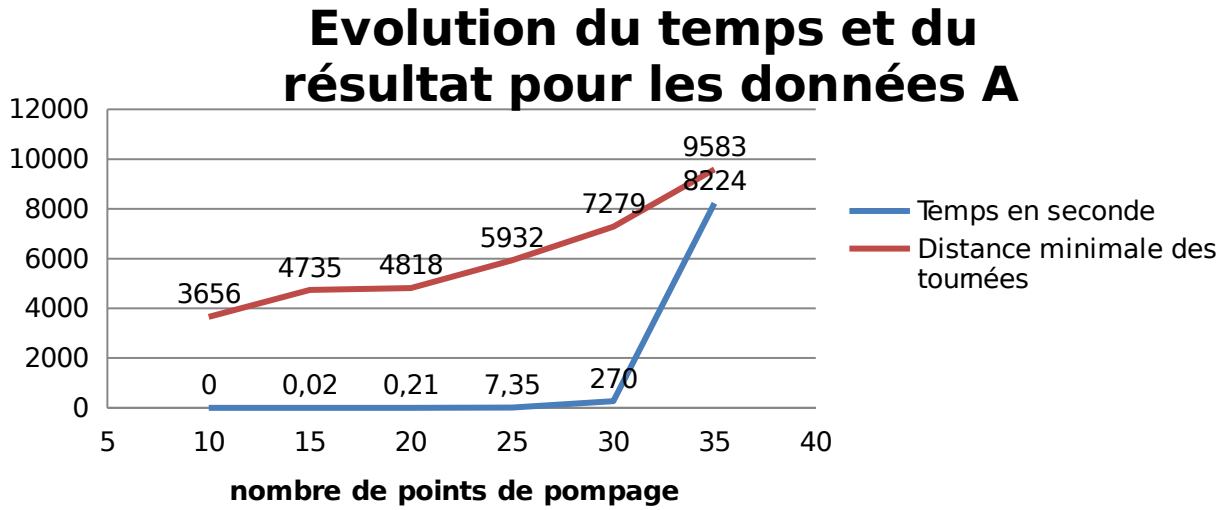
IV – Description des structures de données

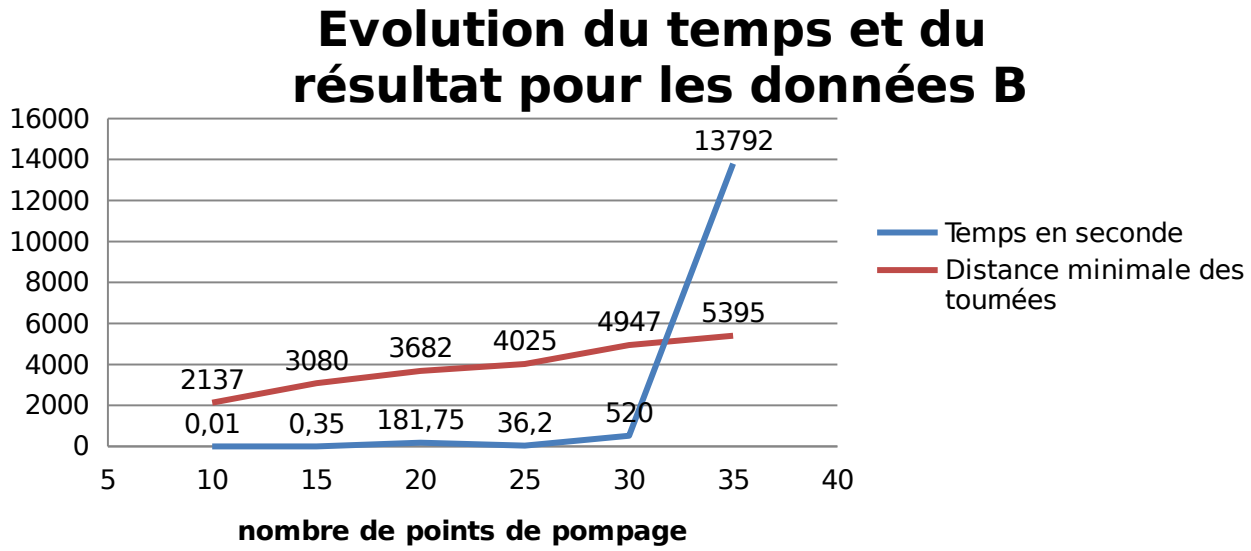
```
typedef struct {
	int *longueur; // Longueur du minimal tour du regroupement i
	int **tab; // Regroupement
	int *n; // Nombre  d'éléments dans le regroupement i
} groupe;

typedef struct {
	int nblieux; // Nombre de lieux (incluant le dépôt)
	int capacite; // Capacité du véhicule de livraison
	int *demande; // Demande de chaque lieu (la case 0 est inutilisée car le dépôt n'a aucune demande à voir satisfaire)
	int **C; // distancier (les lignes et colonnes 0 correspondent au dépôt)
	int nbcontr; // Nombre de contraintes
	int nbvar; // Nombre de variables
	groupe regroupe;
} donnees;
```

Nous avons utilisé principalement des tableaux dynamiques et ceux pour plusieurs raisons. La première est que nous faisons uniquement des insertions en fin de tableau ce qui est peu coûteux. Lorsque nous supprimons le contenu d'un tableau, nous le faisons pour tout le tableau, donc pas de déplacement de valeur pour « boucher » le trou dans le tableau dû à la suppression d'un élément.
La deuxième raison est que même si nous parcourrons à chaque fois tous les éléments des différents tableaux, donc même si nous utilisons une autre structure comme une liste chainée, une file ou une pile, se ne serait pas plus coûteux.
Et enfin la dernière raison et pas la moindre, l'accès des données d'un tableau en mémoire cache est beaucoup plus efficace car les données séquencées.

En plus de la structure de données fournies, nous avons rajouté une structure pour les regroupements.
Cette structure est composée d'un tableau contenant les longueurs minimales des regroupements, un tableau de regroupement de tableaux, et un tableau du nombre d'éléments contenu dans ces tableaux de ragroupement. Ce dernier champs a été rajouté pour récupérer la taille du tableau.

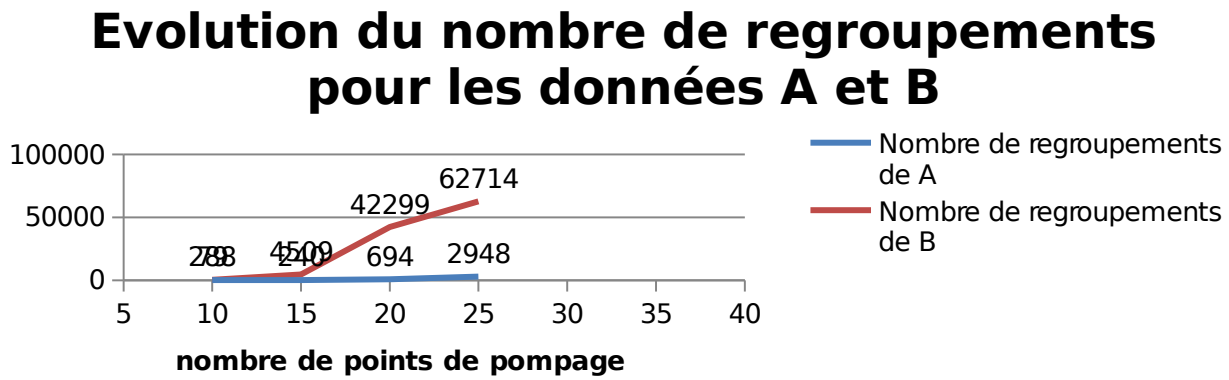
V – Analyse des résultats



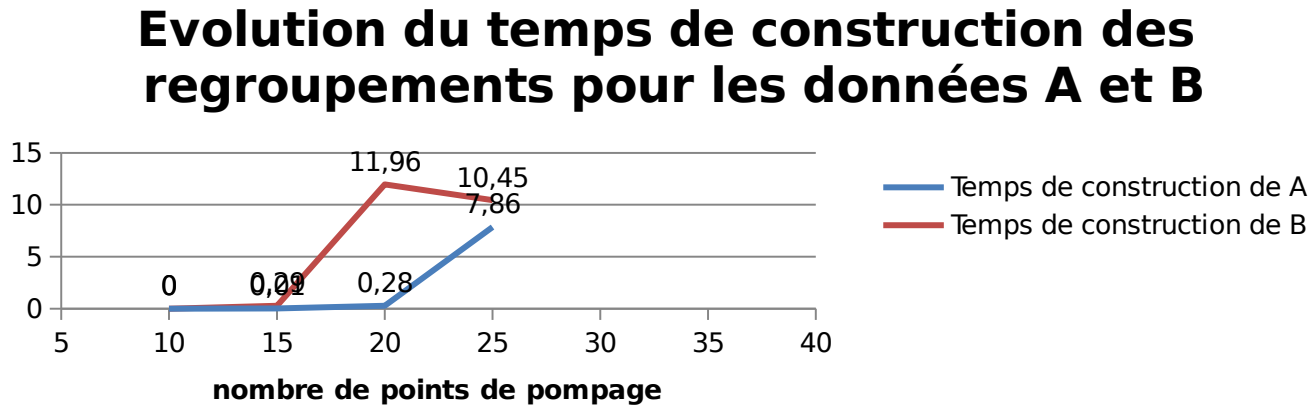


Nous constatons que le temps en seconde est exponentionnel à partir de 25 points de pompage que ce soit pour les données A ou pour les données B.

Dans les graphiques suivant, pour les données B il y a beaucoup plus de regroupements que pour les données A. En effet, la capacité des deux drones sont équivalentes mais les quantités d'eau disponibles aux points de pompage sont légèrement différentes : pour les données B ces quantités ont de plus petites valeurs que pour les données A. Ce qui explique que les temps de constructions des regroupements des données B soient élevés.



Nous remarquons également que sur les données A, GLPK est très rapide contrairement au données B où il met un peu plus de temps.



VI –Améliorations possibles

La fin de notre projet, nous avons pensé à une idée d'amélioration au niveau de l'algorithme de regroupement.
Dans l'état actuel du code, nous calculons la quantité d'eau sur tous les regroupements possibles alors que nous pourrions tester cette quantité d'eau avant la récursive. Ce qui permetterait d'améliorer le temps et creuser l'écart de temps entre les données A et B.
Nous pouvons imaginer comme développement de ce projet l'ajout de nouvelles données, comme une autonomie du drone, exprimée en distance maximale parcourue avant de devoir retourner à la base, ou de nouvelles contraintes, comme l'impossibilité d'effectuer un trajet entre deux stations particulières du graphe.

Conclusion

Pour ce projet, nous devions résoudre un problème de tournée de véhicule avec capacité et profits. Un drone devait visiter des points de pompage afin de s'approvisionner en eau. Nous devions utiliser pour la résolution de ce problème le solveur GLPK, ainsi que de l'algorithmique C.
Au cours de l'élaboration de ce projet nous avons du écrire différents algorithmes permettant de connaître le chemin le plus court entre les différentes stations. Les structures utilisées pour stocker les données en mémoire sont des tableaux dynamiques, qui simplifie l'ajout de données sans impacter le coût mémoire de l'algorithme.

Nous avons pu observer que le coût temporel de notre algorithme augmente de manière significative pour un nombre de lieux supérieur à 25, le temps d’attente devenant très long passé ce nombre. Ce projet nous a donc sensibilisés sur les besoins d'optimisation mémoire et temporelle du domaine de la recherche opérationnelle.