UNIVERSOTE DE TOULOUSE 2 –JEAN JAURES

Département de Mathématiques et Informatique

Projet de stage

Présenté par :

Achraf Talby

MASTER 1 ISMAG

Modélisation d'un problème de réaffectation des véhicules entres plusieurs centres de secours d'un SDIS

Sous la direction de :

OUSSAMA BEN AMMAR
ROMAIN GUILLAUME

Soutenu le 11 SEPTEMBRE 2014

Devant le jury composé de :

- CAROLINE THIERRY
- CLAUDIE CHABRIAC
- OUSSAMA BEN AMMAR
- ROMAIN GUILLAUME

Remerciements

Au terme de ce travail, j'ai l'honneur d'exprimer mes vifs remerciements à Mr. Oussama BEN AMMAR et Mr. Romain GUILLAUME encadrants au sein du département Mathématique Informatique pour leur soutien, leurs remarques pertinentes et leur engagement sans mesure qui devenait ma source d'inspiration ainsi la confiance qui m'ont accordé, en me proposant ce projet de stage.

Je saisis aussi l'occasion pour remercier tout le personnel du département pour son soutien et sa générosité, surtout Mme. Caroline THIERRY la responsable du MASTER ISMAG, qui m'accompagner tout au long de ce stage, et qui a été généreuse en m'offrant de son temps pour les réunions, ainsi que les conseils importants qui me donnait.

Résumé

J’ai effectué mon stage de Master1, d’une durée de 2 mois, au sein du département Mathématiques-Informatique de l’Université de Toulouse 2.La tâche qui m’a été confié fait partie du projet des MASTER2 : « Optimisation sous incertitude de la politique de réaffectation des véhicules entre plusieurs centre d’un SDIS ». Il s’agit d’implémenter en C++ des méta-heuristiques pour le problème de réaffectation des Véhicules de Secours et d'Assistance aux Victimes d'un Service Départemental d'Incendie et de Secours sur différents périmètres géographiques. L'efficacité de ces algorithmes sera évaluée sur des données de taille réelle. Deux méta-heuristiques couplées avec l’algorithme de Charnes et Cooper seront notamment testées : un algorithme génétique et la méthode tabou. Les approches testées seront mises en perspectives par rapport à l'état de l'art du domaine.

Table des matières

1. Introduction :

1. A propos de ce document :

Le présent document est le résumé du travail effectué entre mai 2014 et juillet 2014 dans le cadre du stage du Master1. Il présente la programmation en C++ de quelques algorithmes d'optimisation. Ce rapport est axé sur quatre parties qui sont organisés comme suit :

La première partie définit le contexte général et l'objectif du projet, ainsi qu'une présentation du problème de transport.

La deuxième partie intitulée Balas-Hammer, présente une méthode d'optimisation qui permet de trouver une solution initiale.

La troisième partie présente l'algorithme de Charnes et Cooper et explique son fonctionnement.

La dernière partie est dédiée à une fonction que j'ai dû mettre en œuvre pour rendre admissible les solutions de Balas-Hammer non adapté à Charnes et Cooper.

A la fin du rapport, présente quelques résultats et les différentes annexes traitant les programmes réalisés durant mon stage.

2. Contexte

Les services départementaux de secours et d'incendies, ont pour missions principales, en plus de la gestion des incendies, les secours d'urgence aux personnes victimes d'accidents, de sinistres ou de catastrophes et leur évacuation vers les hôpitaux. Les services départementaux de secours et d'incendies sont constitués de centre de secours, au moins deux centres de secours sont rattachés à une commune. Les premiers, dits « centres de 1er appel », sont ceux qui sont normalement appelés à intervenir sur le territoire de la commune. Les seconds, dits « centres de 2ème appel », sont ceux qui sont appelés en cas d'indisponibilité du centre de 1er appel ou qui peuvent être appelés en renfort en cas de sinistre important. L'objectif est de pouvoir répondre entre 10 et 20 minutes. Afin d'atteindre cet objectif dans un contexte non contraint en ressources, il faut avoir suffisamment de véhicules dans chaque centre de secours, pour répondre à la demande en interventions des communes les plus proches en temps du lieu de l'accident à ce centre de secours. Sous contrainte de moyens, l'objectif de couverture opérationnelle peut être formulé de la façon suivante: Minimiser le nombre moyen d'interventions des véhicules provenant de centres de secours n'étant pas centre de premier appel. Ces données constituent un problème de transport.

3. Problème de transport :

Le problème de transport est un problème linéaire qui peut être représenté sous forme de graphe, il est présenté comme suivant:

- Il y a m casernes et n points d'alerte.
- La caserne i offres a_i VSAV.
- Le point j réclame b_j VSAV.
- Soit $\boldsymbol{c_{ij}}$ le cout du transport d'une unité de i vers j.

L'objectif est de satisfaire la demande en minimisant le coût total de transport.

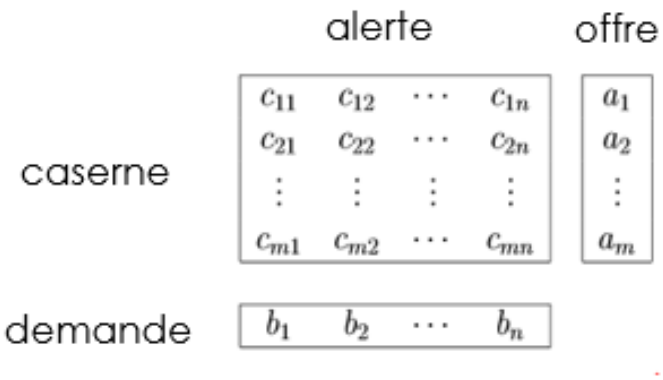


FIGURE 1 : SYSTÈME LINÉAIRE

Cette matrice peut être représenté sous forme de graphe avec les points de départs (les casernes) à gauche et les points d'arriver à droite (points d'alerte).

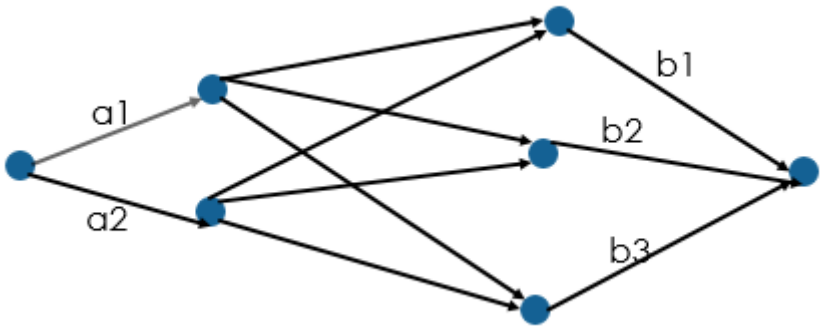


FIGURE 2 : GRAPHE DU SYSTÈME

Pour la résolution de ce problème, il y a certaine condition à respecter :

- La somme des offres doit être égale à celle des demandes $\sum a_i = \sum b_j$.
- si $\sum a_i \geq \sum b_j$, on introduit une case b_{n+1} qui reçoit la différence, avec $c_{i,n+1} = 0$.
- Si $\sum a_i \leq \sum b_j$, on introduit une case a_{m+1} qui reçoit la différence, avec $c_{m+1,j} = 10000$.

Le problème précédant est un problème linéaire :

– Fonction économique : $min \sum_{ij} c_{ij}\, x_{ij}$ (coût de transport à minimiser).

– Contrainte de production: $\sum_i x_{ij} \quad \forall\, i = 1, \ldots, m$

– Contraintes de consommation : $\sum_i x_{ij} = b_j \quad \forall\, j = 1, \ldots, n$

– Contraintes de positivité : $x_{ij} \geq 0 \quad \forall i, j$

La solution admissible est sous la forme :

FIGURE3 : SOLUTION INITIALE

4. Objectifs

La partie qui m'a été confié à pour objectif de corrigé des bugs dans des programmes de méthodes d'optimisation : Charnes et Cooper qui utilise la méthode de Balas-Hammer fournissant une solution initiale admissible.

Une fois ces bugs résolus, il faut tourner deux algorithmes ; l'algorithme génétique et la méthode tabou en intégrant les résultats précédant.

Il fallait corriger le programme de Balas-Hammer qui fournit la solution initiale, pour manque de commentaire sur le code il était difficile de comprendre sa logique, par conséquent le programmer à nouveau était la solution.

2. Balas-Hammer

L'algorithme de Balas-Hammer, appelé aussi méthode des différences maximales ou méthode des regrets, est une heuristique permettant d'optimiser un problème de transport et rendre le cout minimum.

Avant l'introduction de Balas-Hammer il faut que la condition sur les sommes de l'offre et la demande soit satisfaite. La somme des offres doit être égale à celle des demandes : $\sum a_i = \sum b_j$.

J'ai programmé quelques lignes de code qui calcule les sommes, font la soustraction entre eux, puis ajoutent une case à la demande b_{n+1} qui reçoit la différence si $\sum a_i \geq \sum b_j$ avec l'ajout d'une colonne de coûts nuls $c_{i,n+1} = 0$, mais si $\sum a_i \leq \sum b_j$, on introduit une case à l'offre a_{m+1} qui reçoit la différence, avec l'ajout d'une ligne de cout infini $c_{m+1,j}$= 10000.

1. Méthode

On calcule pour chaque rangée, ligne ou colonne, la différence entre le coût le plus petit et celui qui lui est immédiatement supérieur, ce qu'on appelle le regret. C'est une mesure de la priorité à accorder aux transports de cette ligne ou de cette colonne, car un regret important correspond à une pénalisation importante si on n'utilise pas la route de coût minimum.

Affecter à la relation de coût le plus petit correspondant à la rangée présentant le plus grand regret la quantité la plus élevée possible; ça veut dire le minimum entre l'offre et la demande correspondants. Ce qui sature une ligne ou une colonne. On répète le processus jusqu'à ce que toutes les rangées soient saturées.

2. Algorithme :

1. Trouver le coût minimal et celui immédiatement supérieur sur une ligne
2. Trouver le coût minimal et celui immédiatement supérieur sur une colonne
3. Calculer leur différence (les regrets) pour chaque ligne et colonne
4. Sélectionner la ligne ou la colonne ayant le regret maximum
5. Choisir dans cette ligne ou colonne le coût le plus faible

6. Attribuer à la relation (i,j) correspondante le minimum entre l'offre et la demande liées ; c'est au même temps le maximum possible pour saturer soit l'offre soit la demande.
7. Mettre à jour la demande et l'offre.
8. Eliminer la ligne ou la colonne ayant son offre ou sa demande satisfaite.
9. FIN

3. Exemple:

Trouver le coût minimum et celui qui lui est immédiatement supérieur pour chaque ligne et chaque colonne.

	D1	D2	D3	D4	regrets	offres
O1	11	12	10	10		60
O2	17	16	15	18		30
O3	19	21	20	22		90
regrets						
demandes	50	75	30	25		

TABLEAU 1 : CALCUL DES REGRETS

Dans la première ligne le plus petit élément est « 10 » et celui qui lui est immédiatement supérieur ou égal est « 10 ».

Calculer la différence entre ces deux éléments, (le regret).

On répète la même opération pour les rangers restants.

	D1	D2	D3	D4	regrets	offres
O1	11	12	10	10	0	60
O2	17	16	15	18	1	30
O3	19	21	20	22	1	90
regrets	6	4	5	8		
demandes	50	75	30	25		

TABLEAU 2 : CHOIX DU GRAND REGRET

Trouver le plus grand regret.

Dans ce cas, c'est 8. On a un regret de 8 si on ne choisit pas la colonne 3.

Trouver le plus petit coût correspondant à cette valeur.

	D1	D2	D3	D4	regret	offres
O1	11	12	10	10	0	60
O2	17	16	15	18	1	30
O3	19	21	20	22	1	90
regret	6	4	5	8		
demandes	50	75	30	25		

TABLEAU 3 : LE PLUS COUT LE PLUS PETIT

Le plus petit cout correspondant à la ranger de plus grand regret est « 10 ».

Remplir la case qui corresponde à ce coût par le minimum entre l'offre et la demande.

Eliminer la ligne ou la colonne pour laquelle la disponibilité ou la demande est satisfaite, et mettre à jour l'autre.

	D1	D2	D3	D4	regret	offres
O1	11	12	10	25	0	35
O2	17	16	15	18	1	30
O3	19	21	20	22	1	90
regret	6	4	5	8		
demandes	50	75	30	25		

TABLEAU 4 : LE MAXIMUM POSSIBLE

On avait une demande de 60 : $60 - 25 = 35$.

Répéter cet algorithme en mettant à jour à chaque fois l'offre et la demande jusqu'à écoulement de l'offre et satisfaction de la demande.

Solution initiale :

	D1	D2	D3	D4
O1	35	*	*	25
O2	*	30	*	*
O3	15	45	30	*

TABLEAU 5 : SOLUTION INITIALE

4. Conclusion :

Il existe d'autre méthode qui donne une solution initiale, sauf que Balas-Hammer fait intervenir les coûts dans les calculs. Le résultat fournit par Balas-Hammer est proche de la solution optimale, ce qui réduit le temps de calcul en minimisant le nombre d'itérations. En cas de données de petite taille, la solution initiale de Balas-Hammer est souvent solution optimale.

3. Charnes et Cooper :

L'algorithme de Charnes et Cooper est une variante de l'algorithme primal du simplexe adapté aux problèmes de transport. Il utilise une solution initiale admissible, telle que la solution de Balas-Hammer.

Comme dans l'algorithme du simplexe il faut à chaque étape exprimer la fonction économique en fonction des variables hors base :

$min\ z\ - \delta = \sum i, j$ hors base $\bar{C}_{ij}\, x_{ij}$.

Une variable pour laquelle $\bar{C}_{ij}$ ≤ 0 peut entrer dans le base, puisque l'augmentation de celle-ci conduit à une diminution de la valeur de la fonction économique (ici on veut minimiser le coût du transport). L'entrée en base de cette variable fait apparaitre dans le graphe associé un cycle et un seul.

On doit alors modifier les quantités transportées pour rééquilibrer le transport et avoir toujours une solution admissible.

1. Théorème 1 :

$$\bar{C}_{ij} = c_{ij} - (v_j - u_i) \quad \forall\, i,j$$

Ainsi, comme $\bar{C}_{ij}$ = 0 pour les variables de base, on a $c_{ij} = v_j - u_i$ pour les variables de base. Posons v_1= 0, on peut en déduire u_i et v_j $\forall(i,j)$. Ensuite on calcule $\bar{C}_{ij}$ pour les variables hors bases à partir de : $\bar{C}_{ij} = c_{ij} - (v_j - u_i)$.

2. Changement de base

Supposons que $X_{i_1 j_1}$ hors base et $\bar{C}_{i_1 j_1} \leq 0$. Alors on peut faire entrer la variable $X_{i_1 j_1}$ dans la base. Dans le graphe associé à cette solution, l'ajout de l'arc $(i_1 j_1)$ crée un cycle $(i_1, j_1, i_2, j_2, i_3, \ldots, i_p, j_p, i_1)$.

Soit θ la quantité transportée sur $(i_1 j_1)$, on doit rééquilibrer les quantités transportées.

On envoie θ en plus sur $(i_1 j_1)$, θ en moins sur $(i_1 j_p)$, ..., θ en plus sur $(i_p j_p)$, et θ en moins sur $(i_1 j_p)$. $x_{0i_2 j_1} = x_{i_2 j_1} - \theta x_{0i_3 j_2} = x_{i_3 j_2} - \theta \ldots x_{0i_1 j_p} = x_{i_1 j_p} - \theta$.

Comme toutes ces quantités doivent rester positives, on prend :

$\theta = min\,(x_{i_2 j_1}, x_{i_3 j_2}, \ldots, x_{i_1 j_p})$, ce qui annule la (les) variable(s) pour la(les)quelle(s) le minimum est atteint. La variable $x_{i_1 j_1}$ entre en base en échange de l'une des variables qui s'annule (on a intérêt à garder hors base, c'est à dire à maintenir à 0, celle qui a un coût unitaire de transport maximum).

On poursuit ensuite l'algorithme jusqu'`a l'obtention d'une solution optimale, c'est à dire que tous les $\bar{c}_{ij} \leq 0$. Si tous les $\bar{c}_{ij}$ des variables hors base sont strictement négatifs, cette solution optimale est unique.

3. Exemple :

Appliquons cette méthode à une solution fournie par Balas-Hammer.
Soit A la solution dans l'exemple de Balas-Hammer:

35	*	*	25
*	30	*	*
15	45	30	*

TABLEAU 6 : SOLUTION INITIALE

Cette solution est de base car elle possède $m + n - 1$ soit $4 + 3 - 1 = 6$ valeur.
Le coût est de $2945 = \sum x_{ij} * c_{ij}$.

1. Première itération :

Traçons le graphe représentent la solution et les coûts associés aux arcs retenus :

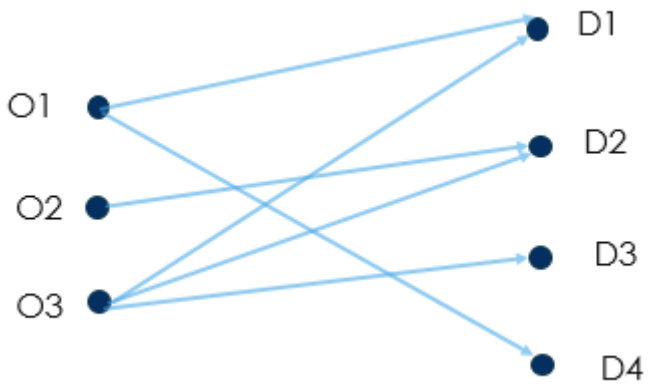


FIGURE 4 : GRAPHE DE LA SOLUTION INITIALE

Créons un ensemble de potentiels sur cet arbre en affectant, au sommet O3 la valeur « 0 » puis on calcule les autres potentiels grâce à la formule $c_{ij} = v_j - u_i$.

	D1	D2	D3	D4	ui
O1	11	*	*	10	8
O2	*	16	*	*	5
O3	19	21	20	*	0
vj	19	21	20	18	

TABLEAU 7 : CALCUL DES POTENTIELS

Cherchons s'il existe une nouvelle liaison permettant, en créant un circuit de cout marginal négatif d'améliorer la solution précédente. Le cout marginal du circuit obtenu par ajout de l'arc (i,j) est : $\bar{C}_{ij} = c_{ij} - (v_j - u_i)$. On obtient :

	-1	-2	
3		0	5
			4

TABLEAU 8 : COUT MARGINAUX

La solution peut donc être améliorée grâce aux arcs $(O1, B2)\ ou\ (O1, B3)$, en modifiant les quantités transportées sur les liaisons des cycles $(O1\ B2\ O3\ O1)\ ou\ (O1\ D3\ O3\ D1\ O1)$.

- De 35 unités pour le premier (gain de 35*1=35)
- De 30 unités pour le premier (gain de 30*2=60)

L'ajout de l'arc $(O1, D3)$ crée le cycle $(O1\ D3\ O3\ D1\ O1)$, on ne peut pas diminuer de plus de 30 les quantités transportées sur $O3 - D3$ et $O1 - D1$ car on est limité à 30 en $O3 - D3$.

En retenant ce cycle donnant le plus grand gain on obtient :

5	*	30	25
*	30	*	*
45	45	0	*

TABLEAU 8 : SOLUTION DE LA PREMIÈRE ITÉRATION

Le coût de cette solution est de 2885.

2. Deuxième itération :

L'arbre associé à la nouvelle solution est le suivant :

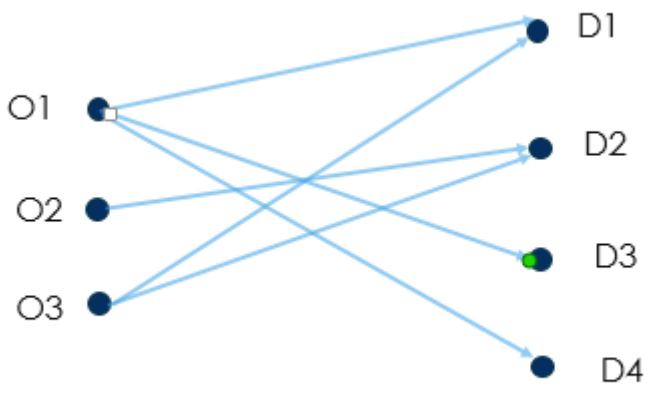


FIGURE 5 : GRAPHE DE LA SOLUTION DE LA PREMIÈRE ITÉRATION

Les potentiels :

	D1	D2	D3	D4	ui
O1	11	*	10	10	8
O2	*	16	*	*	5
O3	19	21	*	*	0
vj	19	21	18	18	

TABLEAU 9 : POTENTIELS

Les coûts marginaux :

	-1		
3		2	5
		2	4

TABLEAU 10 : LES COUTS MARGINAUX

Modifions de 5 unités le long du cycle $(O1\ D2\ O3\ B1\ O1)$. Le gain est de $5 * 1 = 5$. La nouvelle solution obtenue est :

0	5	30	25
*	30	*	*
50	40	0	*

TABLEAU 11 : SOLUTION DE LA DEUXIÈME ITÉRATION

Le cout de cette solution est de : 2880.

3. Troisième itération :

L'arbre associé à la solution précédente :

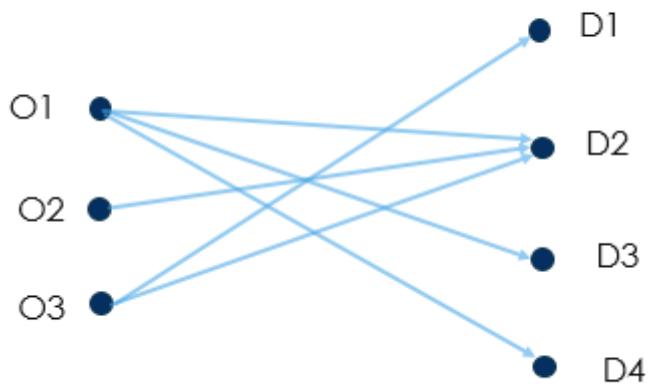


FIGURE 6: GRAPHE DE LA SOLUTION DE LA DEUXIEME ITERATION

Les potentiels :

	D1	D2	D3	D4	ui
O1	*	12	10	10	9
O2	*	16	*	*	5
O3	19	21	*	*	0
vj	19	21	19	19	

TABLEAU 12 : POTENTIELS

Les couts marginaux :

1			
3		1	4
		1	3

TABLEAU 13 : COUTS MARGINAUX

Tous les couts marginaux sont positifs : la solution précédente est donc optimale.
FIN.

4. Conclusion :

Grace à la solution de Balas-Hammer, Charnes et Cooper n'a besoin que de quelques itérations pour atteindre l'optimum.
Le programme de Charne et Cooper donne des résultats pour des solutions initiales admissible, alors que pour des solutions qui n'ont pas $m + n - 1$ éléments il bug. Donc il faut rendre ces solutions admissibles aussi.

4. Graphe max :

Une solution admissible pour Charnes et Cooper est une matrice qui satisfait la condition d'un arbre maximal ; cette matrice doit avoir m+n-1 éléments ≥ 0 et son graphe ne forme pas de cycle.

Quelques solutions fournies par Balas-Hammer ne contiennent pas m+n-1 éléments, d'où le bug dans Charnes et Cooper.

La solution était de créer une fonction qu'on a appelé Graphemax, cette fonction utilise la solution de Balas-Hammer pour la transformer en solution admissible si elle ne l'est pas.

La fonction Graphemax trouve tous les sous arbres puis créer une liaison entre eux de façon à ne pas avoir de cycle.

Exemple : le graphe d'une matrice donne :

-Deux sous arbre :

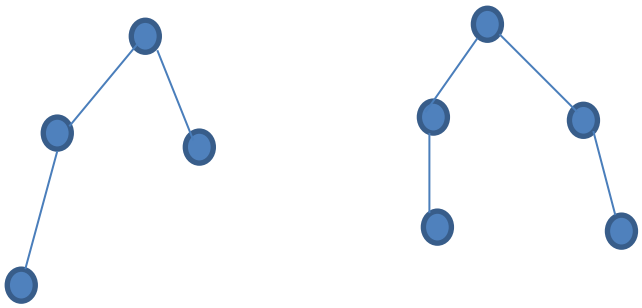


FIGURE 7 : DEUX ARBRES

-création d'un arbre maximal par l'ajout d'un arc:

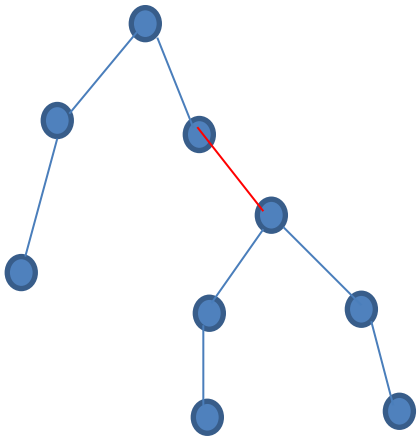


FIGURE 7 : DEUX ARBRES

5. Algorithme :

-si la matrice contient m colonnes et n lignes : les lignes seront numéroté de 1 à n et les colonnes de $n+1$ à $m+n$.

-un tableau « M » de $m+n$ cases contiendra les numéros des lignes et colonnes de 1 à $m+n$.

-une matrice « X » de taille $(n, m+n)$ contiendra les combinaisons des sous arbres.

-on parcourt la première ligne, on pose le numéro de la ligne dans X et on le coche dans M, si cette ligne est liée à une colonne on note son numéro dans X et on le coche de M, si la colonne est liée à une ligne on refait la même chose. Quand il n'y a plus de liaison, on commence à remplir la deuxième ligne dans X en parcourant M sachant qu'elle a été modifiée.

-chaque ligne dans X contiendra les numéros des cases composant un sous arbre.

-on prend un numéro de ligne de la première ligne de X et un numéro de colonne de la ligne qui la suit dans X, puis on remplit la case (numéro de ligne, numéro de colonne) par zéro.

6. Exemple :

Matrice A :

N°	5	6	7	8	9	10
1	1	1	1	*	*	*
2	*	*	1	*	*	*
3	*	*	*	*	*	1
4	*	*	*	1	1	*

Cette matrice contient sept éléments, pour qu'elle soit une solution admissible il lui faut deux autres éléments ≥0.

Le graphe associé est :

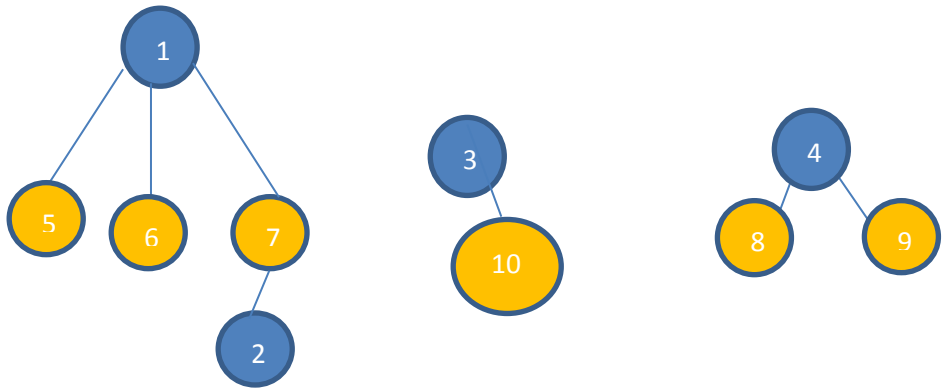


FIGURE 9 : GRAPHE DE L'EXEMPLE

1. Remplir le tableau M :

1	2	3	4	5	6	7	8	9	10

2. Remplir la matrice X :

Premier itération :

On commence par le premier élément non barré de M.

De la matrice A on remarque que 1 est lié à 5, 6 et 7, de ça part 7 est lié à 2.

1	2			5	6	7			

M devient :

-2	-2	3	4	-2	-2	-2	8	9	10

Deuxième itération :

Le premier élément non barré dans M est 3, dans A 3 est lié à 10 et 10 n'est lié à rien.

1	2			5	6	7			
		3							10

M devient :

-2	-2	-2	4	-2	-2	-2	8	9	-2

Troisième itération :

Le 4 est le premier élément non barré, il est lié à 8 et 9, et les deux ne sont liés à rien.

1	2			5	6	7			
		3							10
			4				8	9	

M devient :

-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

3. Liaison :

Chaque ligne dans X définit un arbre, pour construire un arabe maximal il faut les lier en évitant les cycles.

On lie le premier élément dans la première ligne et l'élément le plus proche de 4 dans la ligne suivante, puis le premier élément dans cette ligne et l'élément le plus proche de 4 dans la ligne suivante, ainsi de suite.

Dans cette exemple, on lie 1 avec 10 et 3 avec 8 en mettant des zéros dans les cases (1,10) et (3,8) dans la matrice A.

N°	5	6	7	8	9	10
1	1	1	1	*	*	0
2	*	*	1	*	*	*
3	*	*	*	0	*	1
4	*	*	*	1	1	*

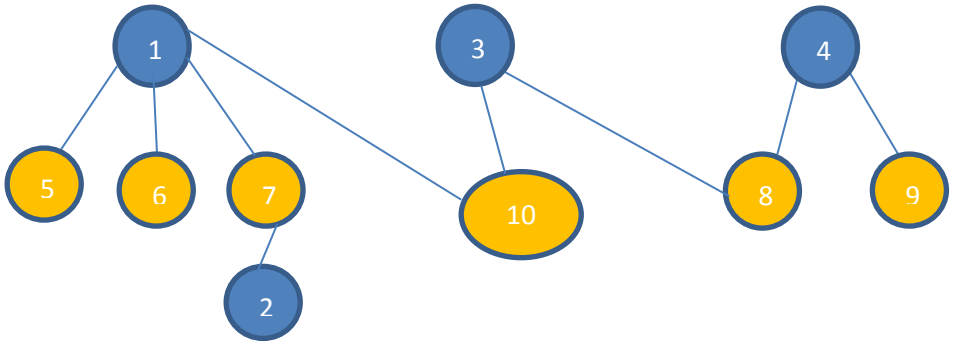


FIGURE 10 : ARBRE MAXIMAL (SOLUTION ADMISSIBLE)

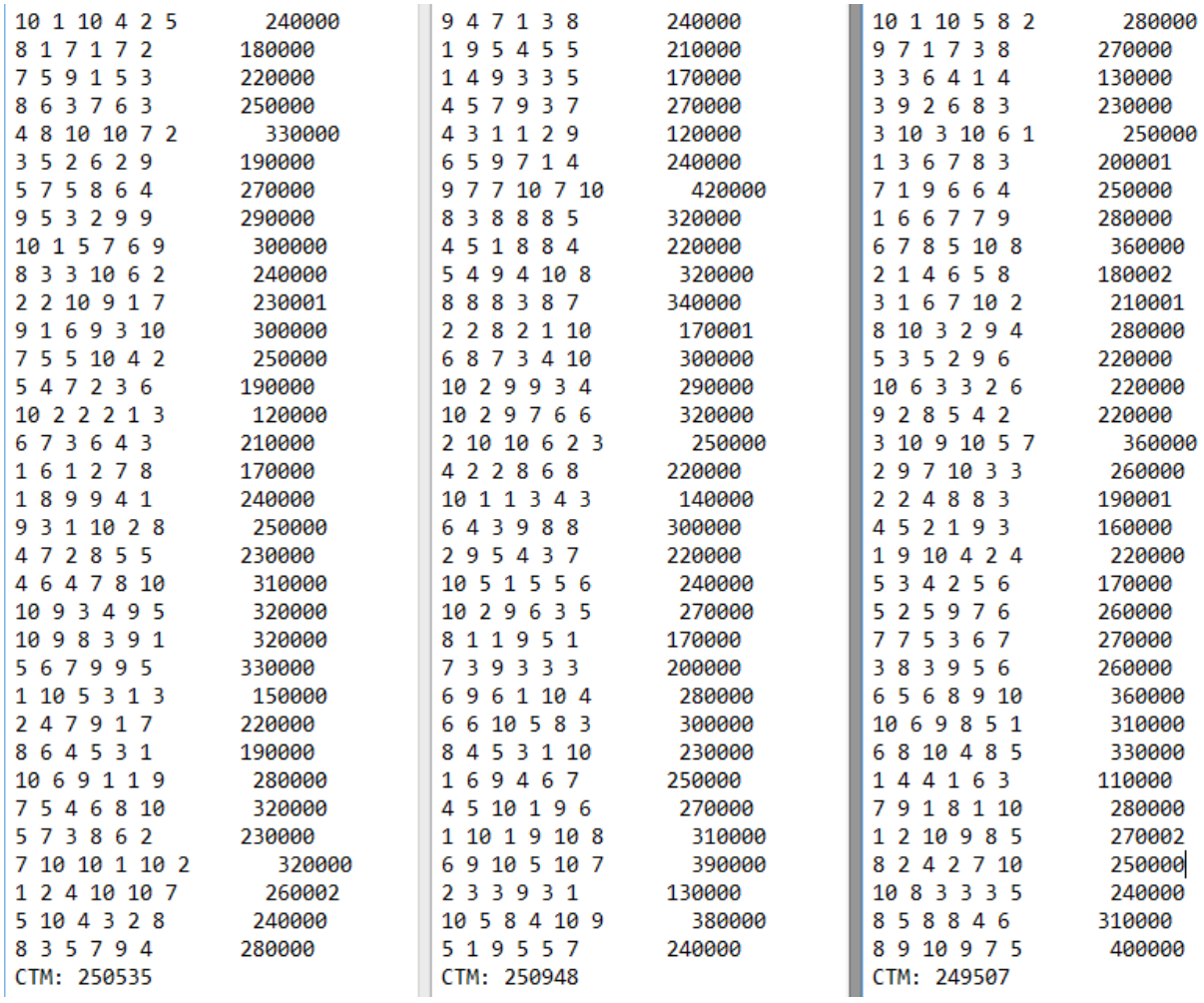
7. Conclusion :

Cette fonction rend la solution de Balas-Hammer admissible, donc toutes les solutions seront traitables par Charnes et Cooper.

5. Conclusion :

1. Analyse :

On a réussi à résoudre les bugs remontés par le programme trouvé sur internet sur le site http://www.polytech-lille.fr/cours-algos-calcul-scientifique/Charnes.swf, j'ai reprogrammé Balas-Hammer, découvert le problème dans Chanes et Cooper et programmer une fonction Graphemax qui le résout. Sur ce programme nous avons ajouté des fonctions qui calculent le cout total et le cout total moyen pour différents vecteurs générés aléatoirement. Nous avons testé le programme 3 fois avec 10 000 simulations et nous avons obtenu le résultat suivant:



```
10 1 10 4 2 5       240000
8 1 7 1 7 2        180000
7 5 9 1 5 3        220000
8 6 3 7 6 3        250000
4 8 10 10 7 2       330000
3 5 2 6 2 9        190000
5 7 5 8 6 4        270000
9 5 3 2 9 9        290000
10 1 5 7 6 9        300000
8 3 3 10 6 2        240000
2 2 10 9 1 7        230001
9 1 6 9 3 10        300000
7 5 5 10 4 2        250000
5 4 7 2 3 6        190000
10 2 2 2 1 3        120000
6 7 3 6 4 3        210000
1 6 1 2 7 8        170000
1 8 9 9 4 1        240000
9 3 1 10 2 8        250000
4 7 2 8 5 5        230000
4 6 4 7 8 10        310000
10 9 3 4 9 5        320000
10 9 8 3 9 1        320000
5 6 7 9 9 5        330000
1 10 5 3 1 3        150000
2 4 7 9 1 7        220000
8 6 4 5 3 1        190000
10 6 9 1 1 9        280000
7 5 4 6 8 10        320000
5 7 3 8 6 2        230000
7 10 10 1 10 2       320000
1 2 4 10 10 7       260002
5 10 4 3 2 8        240000
8 3 5 7 9 4        280000
CTM: 250535
```

```
9 4 7 1 3 8        240000
1 9 5 4 5 5        210000
1 4 9 3 3 5        170000
4 5 7 9 3 7        270000
4 3 1 1 2 9        120000
6 5 9 7 1 4        240000
9 7 7 10 7 10       420000
8 3 8 8 8 5        320000
4 5 1 8 8 4        220000
5 4 9 4 10 8        320000
8 8 8 3 8 7        340000
2 2 8 2 1 10        170001
6 8 7 3 4 10        300000
10 2 9 9 3 4        290000
10 2 9 7 6 6        320000
2 10 10 6 2 3       250000
4 2 2 8 6 8        220000
10 1 1 3 4 3        140000
6 4 3 9 8 8        300000
2 9 5 4 3 7        220000
10 5 1 5 5 6        240000
10 2 9 6 3 5        270000
8 1 1 9 5 1        170000
7 3 9 3 3 3        200000
6 9 6 1 10 4        280000
6 6 10 5 8 3        300000
8 4 5 3 1 10        230000
1 6 9 4 6 7        250000
4 5 10 1 9 6        270000
1 10 1 9 10 8       310000
6 9 10 5 10 7       390000
2 3 3 9 3 1        130000
10 5 8 4 10 9       380000
5 1 9 5 5 7        240000
CTM: 250948
```

```
10 1 10 5 8 2       280000
9 7 1 7 3 8        270000
3 3 6 4 1 4        130000
3 9 2 6 8 3        230000
3 10 3 10 6 1       250000
1 3 6 7 8 3        200001
7 1 9 6 6 4        250000
1 6 6 7 7 9        280000
6 7 8 5 10 8        360000
2 1 4 6 5 8        180002
3 1 6 7 10 2        210001
8 10 3 2 9 4        280000
5 3 5 2 9 6        220000
10 6 3 3 2 6        220000
9 2 8 5 4 2        220000
3 10 9 10 5 7       360000
2 9 7 10 3 3        260000
2 2 4 8 8 3        190001
4 5 2 1 9 3        160000
1 9 10 4 2 4        220000
5 3 4 2 5 6        170000
5 2 5 9 7 6        260000
7 7 5 3 6 7        270000
3 8 3 9 5 6        260000
6 5 6 8 9 10        360000
10 6 9 8 5 1        310000
6 8 10 4 8 5        330000
1 4 4 1 6 3        110000
7 9 1 8 1 10        280000
1 2 10 9 8 5        270002
8 2 4 2 7 10        250000
10 8 3 3 3 5        240000
8 5 8 8 4 6        310000
8 9 10 9 7 5        400000
CTM: 249507
```

Nous remarquons que le cout total moyen est plus ou moins le même à chaque testable quand le nombre de simulation est grand.

Nous ferons d'autre tests sur des instances plus grande; un grand nombre de casernes (10, 20, 30), puis le coupler à l'algorithme génétique et la méthode tabou. Les approches testées seront mises en perspectives par rapport à l'état de l'art du domaine.

2. Perceptions Personnelle :

C'était très intéressant de travailler dans le même bureau que mais professeur, j'ai pu découvrir leur rôle de chercheur et bénéficier de leur manière de travail et d'organisation. J'étais motivé pour m'impliquer à ce projet dès qu'on me l'a proposé, il vient enrichir les enseignements des cours d'optimisation avec tous les algorithmes que j'ai pu utiliser.

J'ai pu durant ce stage avec l'aide de mes tuteurs, surmonter toutes les difficultés auxquelles j'ai été confronté, comme le manque de documentation et la succession de bugs.

Tout cela entre le cadre d'une expérience qui marquera mon projet professionnel, car elle fût très positives et me donna une idée sur le monde de la recherche.

6. ANNEXES :

1. Algorithme de Graphemax :

```
debut
    entier i=1, j=1, e=1, b=1, a=1, u=1, z=1, n=4, m=6
    entier n,m
    entier M[PMAX], s[NMAX][MMAX], X[NMAX][MMAX], r[NMAX][MMAX]
        pour i allant de 1 a n
            debut pour
                pour j allant de 1 a m
                debut pour
                r[i][j]=s[i][j]
                fin pour
```

```
        fin pour
pour i allant de 1 a m+n
        debut pour
                pour j allant de 1 a n+m
                debut pour
                X[i][j]=0
                fin pour
pour i allant de 1 a n+m
        debut pour
        M[i]=i;
        fin pour
pour i allant de 1 a m+n
        /debut pour
                si s[i][j]!=-1 && s[i][j]!=-2
                alors X[i][n+j]=n+j;M[n+j]=-2;s[i][j]=-2
                fin si
pour a allant de 1 a n
        //debut pour
                si s[a][j]!=-1 && s[a][j]!=-2
                alors X[i][a]=a;M[a]=-2;s[a][j]=-2
                fin si

pour b allant de 1 a m
        ///debut pour
                si s[a][b]!=-1 && s[a][b]!=-2
                alors X[i][n+b]=n+b;M[n+b]=-2;s[a][b]=-2
                fin si
pour z allant de 1 de n
        ////debut pour
                si s[z][b]!=-1 && s[z][b]!=-2
                alors X[i][z]=z;M[z]=-2;s[z][b]=-2
                fin si
        /fin pour
        //fin pour
```

```
                ///fin pour
                ////fin pour
        pour i allant de 1 a m+n
                debut pour
                        pour j allant de 1 a n+m
                        afficher X[i][j]
                        fin pour
                fin pour
        pour i allant de 1 a n
                debut pour
                        pour j allant de 1 a m
                        afficher s[i][j]
                        fin pour

                fin pour
entier i=1
        tant que (i<(n+1) faire
                debut tant que
                        pour j allant de 1 a n+1
                                debut pour
                                        si X[i][j]!=0
                                        alors a=X[i][j]
                                        i++
                                        d=i
                                        fin si
                                        tant que (i<(n+1) faire
                                                pour k allant de n+a de m+n+1
                                                        debut pour
                                                        si X[i][k]!=0
                                                        alors r[a][b-n]=0
                                                        afficher a,b
                                                        afficher r[a][b-n]
                                                        k=m+n+1
                                                        i=n+1
```

```
                                                    fin si
                                                fin pour
                                            fin tant que
                                    i++
                        fin tant que
                        i=d-1
                        j=n+1
                pour i allant de 1 a n
                        debut pour
                                pour j allant de 1 a m
                                        debut pour
                                        afficher r[i][j]
                                        fin pour
                        fin pour
fin
```

2. Algorithme de Balas-Hammer :

```
pl_balas_hammer(c,s:matrice d'entier,n,m:entier)

   /*declaration des variables*/
   p,NMAX,MMAX,i,j,q,l,co,x,k,a,b,d,z,y,S,f,N:entier;
   NMAX=n+1;
   MMAX=m+1;
   v:matrice d'entier;

   /*copier la matrice c dans v pour la conserver*/
   v <- c

   tant que (S!=0) /*condition d arret , S est la somme de l'offre*/
     S=0;
```

```
/*calcul des deltas colonne*/
pour j allant de 1 à m

  /*recherche du min et sa position dans une colonne*/
  a <- c[1][j];
  b=1;
  pour i allant de 1 à n
    si ((a>c[i+1][j]) et (c[i+1][j]!=-1) ou (a=(-1)))
      a <- c[i+1][j];
      b <- i+1;

   /*recherche du 2ème min et sa position dans une colonne*/
   p <- c[1][j];
   d=1;
   pour i allant de 1 à n
   si (((p>c[i+1][j]) et (i+1!=b) et (c[i+1][j]!=-1)) ou (d==b) ou (p==(-1)))
      p <- c[i+1][j];
      d <- i+1;

    /*calcule de la difference entre les deux min*/
    si (((p=-1) et (a!=-1)) ou ((p!=-1) et (a=-1)))
    c[n+1][j] <- max(p,a);

    sinon si((p=(-1)) et (a=(-1)))
    c[n+1][j] <- -1;

    sinon
    c[n+1][j] <- p-a;

    /*calcul des deltas ligne*/
    /*meme procedure, au lieux de parcourire par ligne on parcoure les colonnes*/

    /*recherche du delta max */
    q <- c[1][m+1];
```

```
d <- 1;
pour i allant de 1 à n
  si((q<c[i+1][m+1]) ou (q=-1))
    q <- c[i+1][m+1];
    d <- i+1;

p=c[n+1][1];b=1;
pour j allant de 1 à m
  si(p<c[n+1][j+1] ou (p=-1))
    p <- c[n+1][j+1];
    b <- j+1;
/*recuperer la position de delta max*/
si ( p >= q )
   l <- n+1;
   co <- b;
sinon
   l <- d;
   co <- m+1;

si ( l=n+1 )
  a <- c[1][co];
  z <- 1;
  /*cherche le min dans la colonne corespendante*/
  pour i allant de 1 à n
    si ((a>c[i+1][co] et (c[i+1][j]!=-1) ou a=-1))
      a <- c[i+1][co];
        z <- i+1;
      s[z][co] <- min(c[z][0],c[0][co]);

  si (c[0][co]=c[z][0])
      pour i allant de 1 à n
        c[z][i] <- -1;
      pour i allant de 1 à n
        c[i][co] <- -1;
```

```
            c[0][co]=c[0][co]-s[z][co];
            c[z][0]=c[z][0]-s[z][co];
        sinon si (s[z][co]=c[z][0])
            pour i allant de 1 à n
              c[z][i] <- -1;
              c[0][co]=c[0][co]-s[z][co];
              c[z][0]=c[z][0]-s[z][co];
        sinon
             pour i allant de 1 à n
               c[i][co] <- -1;
               c[z][0]=c[z][0]-s[z][co];
               c[0][co]=c[0][co]-s[z][co];
    sinon
      a=c[l][1];
      y=1;
      pour i allant de 1 à n
        si ((a>c[l][i+1] et (c[l][i+1]!=-1)) ou a==-1)
          a=c[l][i+1];
            y=i+1;
          s[l][y]=min(c[l][0],c[0][y]);

      si (s[l][y]=c[0][y])
         pour i allant de 1 à n
           c[i][y]=-1;
           c[0][y]=c[0][y]-s[l][y];
           c[l][0]=c[l][0]-s[l][y];
      sinon
           pour i allant de 1 à n
            c[l][i]=-1;
            c[l][0]=c[l][0]-s[l][y];
            c[0][y]=c[0][y]-s[l][y];


pour i allant de 1 à n
```

```
    pour j allant de 1 à m
        c[i][j]=v[i][j];
FIN
```