Projet d'Optimisation Stochastique

Problème du voyageur de commerce

-

Document de synthèse des résultats

Enseignant : Abdel LISSER

Groupe : 3

Membres du groupe : Adrien LAVILLONNIERE
Corentin MANSCOUR
Hien Minh NGUYEN

Rédigé le : 04/11/2018

Nombre de pages : 14

Table des matières

Introduction

Ce document vise à montrer les résultats de notre programme ainsi que nos réflexions pour venir à bout de certains problèmes rencontrés durant son élaboration.

I. Avancée du projet

I. 1) Recuit déterministe

La résolution de problèmes déterministes par recuit simulé est fonctionnel, et propose des résultats exploitables (qui seront discutés plus tard dans ce rapport). Une bonne parties des fonctionnalités discutées en cours et en TD ont été implémentées :

- La recherche de solution voisine peut s'effectuer via les algorithmes 2-opt, 3-opt ou 4-opt.
- Le choix de la température initiale se fait selon la méthode KirkPatrick.

Un aspect auquel nous n'avons pas pu nous intéresser est l'évolution de la température au fur et à mesure de l'algorithme. En effet, celle-ci descend de manière constante et naïve, sans prendre en compte le taux d'acceptation de nouvelles solutions en fin de palier. Si nous devions travailler sur d'éventuelles améliorations de notre algorithme, ce serait la première chose sur laquelle nous nous pencherions.

I. 2) Recuit stochastique

Notre recuit stochastique semble fonctionner, et proposer des résultats cohérents, malgré la confusion qui a entouré son développement. Il intègre les différents éléments du recuits déterministe, en plus de contenir divers fonction permettant de tester si telle ou telle solution satisfait la contrainte en probabilité définie dans notre document technique.

Faute d'avoir complètement saisi ce qui était attendu pour faire marcher correctement ce recuit pour problème stochastique, voici la démarche que nous avons effectuée :

Tout d'abord nous avons fait l'hypothèse que les variables aléatoires étaient indépendantes les unes des autres, ce qui nous permettait de générer une matrice de covariance avec des élément nulles en dehors de la diagonale (représentable, donc, par un simple vecteur). Chaque arc se voyait alors accorder une variance correspondant à 20% de sa moyenne.

Ensuite nous avons fixé Z comme étant la solution optimale de l'équivalent déterministe du problème à résoudre, majoré de 25%. Cependant, cette définition de Z nous a mené à un premier problème : lors du lancement de notre recuits, les solutions trouvées possédaient un coût tellement élevé comparées à la solution optimale déterministe qu'aucune de nos solutions ne satisfaisaient la contrainte de probabilité. Impossible donc de démarrer notre recuit.

Nous avons donc fixé Z comme étant la solution ACTUELLE gérée par le programme, majorée toujours de 25%. Un autre problème à alors surgit : soit toutes nos solutions étaient acceptées, soit elles étaient toutes rejetées si nous baissions trop bas le coefficient de majoration. Alors nous avons décidé de générer aléatoirement chaque variance, plutôt que d'avoir une variance fixe de 20% pour tout arc. Et nous avons fini par trouver un équilibre entre ordre de grandeur de la variance et coefficient de majoration, équilibre où certaines de nos solutions sont rejetées mais d'autres acceptées.

A l'équilibre que nous avons trouvé, le coefficient de majoration de Z est de l'ordre de 1 à 2 %, ce qui est bien loin des 20-30% préconisés. Ainsi, nous nous doutons que ceci n'est pas la méthode que vous attendiez, mais c'est la seule que nous avons développé qui a eu le mérite de produire des résultats analysables.

Pour finir, les fonctionnalités permettant de créer une centaine d'instance de notre problème pour tester empiriquement si notre solution satisfait la contrainte de probabilité n'ont pas été implémentées.

I. 3) CPLEX déterministe

La résolution déterministe par CPLEX a été implémentée dans le programme. Pour des raisons de complexité, la contrainte des sous-tours du problème du VDC est ajoutée itérativement.

Une amélioration possible, outre une optimisation avec les différentes classes implémentant Callback de CPLEX (comme la classe LazyConstraintCallback), serait de mettre en place un temps maximum d'exécution. Une fois le temps imparti écoulé, on vérifie la qualité de la solution et, à l'aide d'un algorithme glouton/naïf, on rends cette solution réalisable (par exemple, en supprimant les sous-tours pour le problème du VDC).

I. 4) CPLEX stochastique

Pour la résolution stochastique par CPLEX, et comme nous n'avons pas de jeux de données correspondant à un problème stochastique, nous avons fait le choix d'associer une variance aléatoire à chaque arc. On ajoute ensuite la contrainte probabiliste en fonction des variances, en calculant au préalable une solution déterministe au problème.

II. Discussions sur les résultats

II. 1) Comparaison générale entre recuit simulé et CPLEX

Selon l'algorithme de résolution choisi, les résultats et le temps d'exécution varient grandement. Par exemple, pour le fichier kroC100 (voir tableau 5), le temps de calcul du recuit simulé (environ 15s) est bien supérieur au temps pris par CPLEX (2,5s). Cela peut être expliqué par les paramètres contraignants que nous avons appliqué au recuit pour la résolution de ce problème (coefficient de température à 0,99, taux d'acceptation minimum à 0,1%, nombre d'itération maximal par palier de 65).

CPLEX est capable de donner un résultat avec une différence de seulement 0,01% avec la valeur optimale, tandis que la différence pour le recuit est de 0,79%. Néanmoins, ces résultats restent tous les deux proches de la valeur optimale : nous les considérons donc comme acceptables (tableau 5).

	Fichier	**Coût (calculé)**	**Coût (optimal)**	**% diff**	**Tps calcul (s)**
Recuit simulé	kroC100.xml	20913	20749	0,79%	14,995
CPLEX	kroC100.xml	20751	20749	0,01%	2,507
Recuit simulé	att48.xml	10658	10628	0,28%	0,663
CPLEX	att48.xml	10628	10628	0,00%	0,932

Tableau 1 : comparaison entre recuit et CPLEX pour les fichiers kroC100 et att48.

On remarque toutefois que pour certains réglages, comme celui du *taux d'acceptation minimum*, le temps de calcul du recuit devient inférieur à celui de CPLEX. Dans le cas où ce taux d'acceptation est élevé, le recuit ne permet pas de donner la valeur optimale exacte mais reste plus rapide que CPLEX. Par exemple, dans le tableau 2, nous avons réglé un taux d'acceptation minimum élevé, de l'ordre de 5%, pour la résolution du fichier kroB200. Nous remarquons donc que le temps de calcul requis pour le recuit est considérablement inférieur à celui de CPLEX, accompagné d'un coût final de nettement moins bonne qualité (10% de différence par rapport à la solution optimale).

	Fichier	Coût (calculé)	Coût (optimal)	% diff	Tps calcul (s)
Recuit simulé	kroB200.xml	32137	29437	9,17%	5,916
CPLEX	kroB200.xml	29440	29437	0,01%	32,02

Tableau 2 : comparaison entre recuit et CPLEX pour le fichier kroB200

Les différences peuvent s'expliquer par le fait que notre recuit simulé n'est pas codé de manière optimale comparé à CPLEX, un logiciel du commerce conçu pour ce genre de problème. De plus, les coefficients et paramètres du recuit simulé ont été choisis de manière *empirique*, à travers nos expériences. Des paramètres plus optimaux et un algorithme de génération de solution initiale plus adapté diminueraient sans doute le temps de calcul.

II. 2) Influence du nombre de villes sur le temps de calcul

Après l'implémentation du recuit simulé et de CPLEX dans leur version déterministe et stochastique, nous avons pu effectuer des tests sur certains fichiers du jeu de données fourni par M. Lisser. Les fichiers de **plus de 1000 villes étant trop longs** pour être traités de manière optimale, nous les avons laissé de côté pour nous concentrer plus spécifiquement sur des fichiers ayant une taille maximale de 532 (att532).

Voici le comparatif entre CPLEX et le recuit pour le même fichier (tableau 4). Nous pouvons remarquer que les temps de calculs sont assez longs, surtout pour le recuit simulé qui a eu besoin de 6h (avec les paramètres de la première ligne).

	Tx temp	Tx accept	nbIteMax	nb palier	Coût final	Coût opti.	% diff	Tps (s)	Tps (h)
Recuit	0,99	1%	20	$4n^2$	30671	27686	10,78%	23578	6,55
Recuit	0,90	1%	20	n	50324	27686	81,77%	7,5	0
CPLEX	-	-	-	-	27686	27686	0,00%	4630	1,28

Tableau 3 : comparatif des résultats de la résolution du fichier att532 par le recuit déterministe et CPLEX déterministe.

L'utilisateur ne pouvant pas modifier les valeurs de CPLEX, nous avons donc pu en déduire que les temps de résolution de CPLEX étaient relativement constants et du même ordre de grandeur. Nous pouvons donc tracer un **graphe des temps de résolution** en fonction du nombre de villes à partir des résolutions que nous avons faites (figure 1).

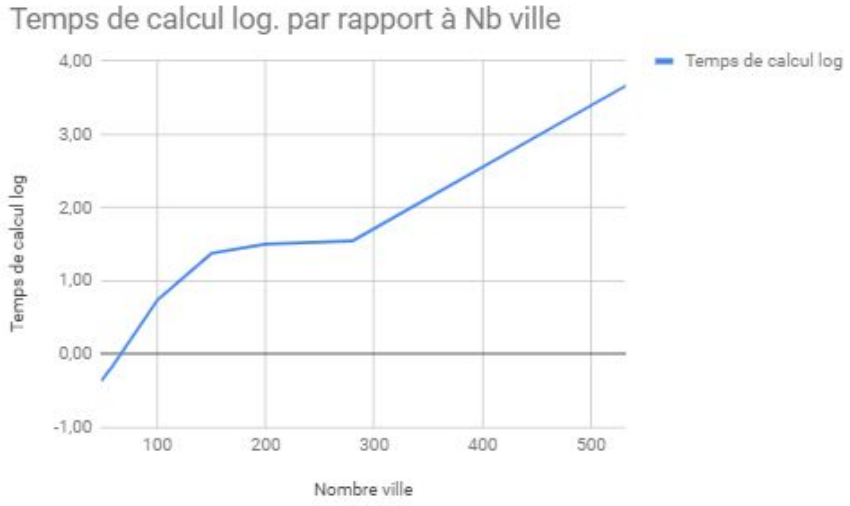

Figure 1 : graphe des temps de résolution (en logarithmique) en fonction du nombre de villes d'un fichier

Au contraire, le tableau 3 nous montre que le temps de résolution du recuit peut varier entre 23578 et 7 secondes pour un même fichier : ce temps dépend donc énormément des paramètres donnés au recuit, et plus particulièrement le nombre d'itération par palier de $4n^2$. Nous ne pouvons donc pas prévoir exactement à l'avance le temps de calcul requis pour la résolution via le recuit simulé. Malgré tout, il existe une corrélation forte entre le temps de résolution et le nombre d'itérations et de paliers du recuit, eux-même influencé par les paramètres (voir partie III.).

II. 3) Choix des paramètres du recuit simulé

Choix des algorithmes de la solution initiale

Pour déterminer la solution initiale à partir de laquelle nous allons appliquer l'algorithme du recuit simulé, nous avons eu le choix entre plusieurs algorithmes : le Plus Proche Voisin et un algorithme naïf et glouton qui lie la ville 1 à la ville 2, puis la ville 2 à la ville 3 et ainsi de suite.

Nous avons tout d'abord implémenté l'**algorithme glouton**. La solution qui en découle est loin de la solution optimale. L'algorithme a cependant le mérite de prendre beaucoup moins de temps comparé à celui du Plus Proche Voisin.

Nous avons ensuite implémenté l'algorithme du **Plus Proche Voisin** et avons remarqué que la plupart du temps, les solutions avaient du mal à s'éloigner de la solution optimale. Le coût restait très élevé, malgré le changement d'autres paramètres et l'augmentation du nombre d'itérations. L'algorithme glouton semble donc donner de **meilleurs résultats** que le Plus Proche Voisin.

Choix des paramètres du recuit simulé

Afin d'obtenir des résultats optimaux pour l'algorithme du recuit, nous avons décidé des paramètres suivants :

Température initiale

La température initiale est déterminée par la méthode de Kirkpatrick, détaillée dans le document technique de ce programme. Elle est donc calculée automatiquement par le programme, sans intervention de l'utilisateur.

Coefficient de changement de la température

Celui-ci est fixé par défaut à 0.99, une valeur semblant retourner les meilleurs résultats. Cependant, l'utilisateur est libre de définir n'importe quelle valeur entre 0 et 1, lui donnant plus de flexibilité pour la résolution de certains problèmes.

Nombre d'itérations maximum par palier

Quand le compteur d'itération atteint le maximum défini, si l'évolution est supérieur au taux, on recommence le palier. Sinon, on baisse la température[1].

Taux d'acceptation minimal

Celui-ci a été déterminé en fonction des tests que nous avons effectués au cours de l'élaboration du programme. Nous avons fixé ce taux à 5%, mais celui-ci pourra être modifié dans le programme par l'utilisateur.

[1] http://idboard.net:10000/maths/2018/03/11/recuit-simule/

Multiplicateur de température initiale

Nous avons ajouté un coefficient multiplicateur de la température à régler par l'utilisateur. Celui-ci permet de multiplier la température initiale, afin d'allonger le temps de calcul, permettre plus d'itérations et ainsi obtenir un meilleur coût final.

II. 4) Choix des paramètres stochastiques

Coefficient alpha

Ce coefficient représente la contrainte de risque à satisfaire. Il est fixé à 95% à l'origine et peut être modifié par l'utilisateur.

II. 5) Spécificités des jeux de données

Lors des tests que nous avons pu effectuer sur de gros fichiers, nous avons remarqué qu'outre le temps requis pour parcourir les algorithmes, le temps requis pour calculer les coordonnées à partir d'un fichier XML était considérable (environ 37 minutes pour le fichier fl3795.xml). Nous avons donc décidé d'implémenter un deuxième parseur, qui prenait des fichiers TSP pour le même problème, et qui permettait ainsi de considérablement réduire le temps de traitement de chaque problème.

Voici quelques temps de résolution de CPLEX déterministe, avec et sans fichier TSP pour l'affichage des villes (tableau 3). On remarque que sur de petites valeurs, comme kroB100, le temps que prend le parsing du fichier TSP est un désavantage face au temps de calcul des coordonnées. Au contraire, pour de plus grands fichiers, le gain de temps est réel et augmente en fonction du nombre de villes.

Fichier	**Avec fichier TSP**	**Sans fichier TSP**
a280	41, 463 s	44,075 s
att532	1h18	1h37
ch150	19,06 s	17,94 s

Tableau 3 : comparatif des temps d'affichage avec parsing de fichier TSP et sans

C'est pour cette raison que le choix a été donné à l'utilisateur de choisir ou non d'utiliser un fichier TSP, grâce à la checkbox "Get cities position from a TSP file". Il faut également que le fichier TSP soit présent en parallèle du fichier XML.

Nous notons une différence particulière pour le fichier **br17.xml.** Celui-ci semble contenir des données asymétriques (les coûts entre deux ville dépend du sens de parcours). Notre algorithme est incapable de déterminer la position des villes d'un tel fichier et ne peut donc pas afficher le problème pour br17.xml. En revanche, il est possible de résoudre le problème et afficher le coût final, sans représentation graphique.

Enfin, sans doute à cause d'un problème dans le parsing, il nous a été impossible de résoudre le fichier **a280**. En effet, lorsque nous exécutons notre recuit sur a280, il nous redonne notre solution initiale à l'identique. Impossible de changer quoi que ce soit. C'est le seul fichier qui nous a posé ce genre de problème.

III. Etude de l'influence de certains paramètres

Influence du coefficient multiplicateur de la température et du taux d'acceptation minimum de la solution

Afin de mieux optimiser notre recuit simulé déterministe, nous avons décidé de récolter des données concernant deux paramètres de l'algorithme : le coefficient multiplicateur de la température et le taux d'acceptation minimum de la solution.

Lorsque l'on diminue le taux d'acceptation minimum ou que l'on augmente le coefficient multiplicateur de la température, on remarque que la qualité de la solution augmente. Cela est causé par le fait que le nombre de paliers augmente lorsque les paramètres sont plus strictes. En effet, un coefficient multiplicateur de la température qui augmente implique une diminution de la température plus faible et donc un nombre de paliers plus élevés. Il en est de même pour la diminution du taux d'acceptation minimum.

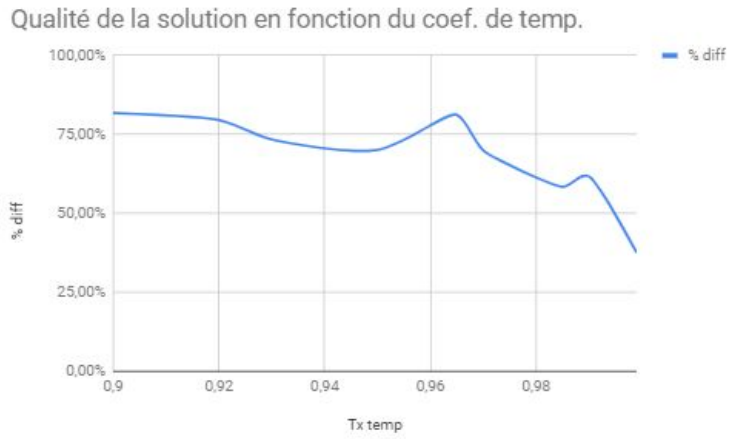

Figure 2 : graphe montrant l'évolution du coût de la solution (%diff), en fonction du coefficient multiplicateur de la température (Tx temp), appliqué au fichier att532. Un bon coût correspond à un % faible.

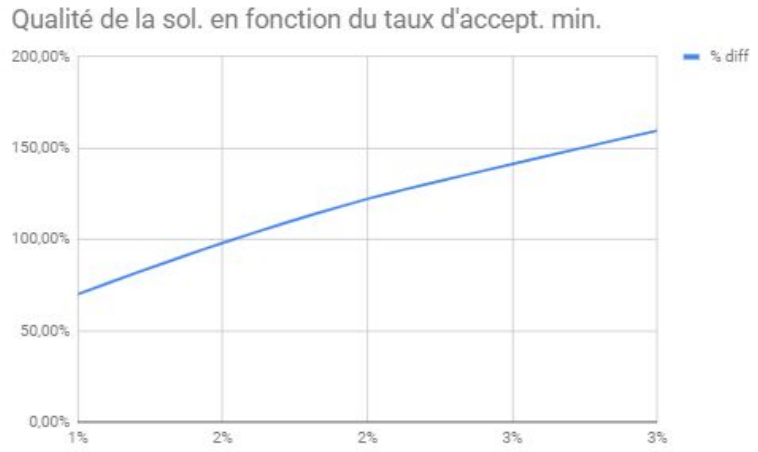

Figure 3 : graphe montrant l'évolution du coût de la solution (%diff), en fonction du coefficient multiplicateur de la température (Tx accept), appliqué au fichier att532. Un bon coût correspond à un % faible.

Cette augmentation du temps de résolution permet donc un meilleur résultat, mais nous ne savons pas encore quel est le paramètre qui contribue le plus à cette augmentation. Afin d'améliorer le recuit, il faudrait effectuer encore plus de tests et de simulations pour pouvoir trouver une forte corrélation entre un paramètre et une amélioration du coût final.

Conclusion

Sur les fichiers que nous avons pu tester, l résultats que nous avons obtenus par CPLEX et par le recuit semblent se rapprocher des coûts optimaux indiqués sur le site. Le temps de calcul élevé pour certains fichiers pourraient venir de méthodes non-optimisées de notre programme. Concernant les temps de calcul, nous pourrions implémenter une méthode permettant d'arrêter le programme sur un input de l'utilisateur, enlever les sous-tours de la dernière solution connue et l'afficher.

L'impact de certaines modifications du programme (utilisation d'un tableau de N villes pour stocker les solutions) a eu un impact conséquent sur la vitesse d'exécution de celui-ci.

Le programme que nous avons conçu pourra être utilisé pour divers problèmes linéaires. Il peut être également couplé à plusieurs autres types de solveurs. Grâce à l'aspect générique de notre programme, tous ces ajouts pourront être fait sans avoir à coder de nouveau certaines parties communes, telles que la classe du problème linéaire ou toutes les fonctions d'affichage.

Ce projet nous a permis de mieux comprendre les principes de la programmation stochastique et de l'optimisation d'un problème linéaire. En codant nous-même l'algorithme du recuit et en découvrant le fonctionnement du solveur CPLEX, nous avons pu voir en détail comment s'effectuait la résolution d'un tel problème. De plus, les problèmes d'optimisation étant présent dans la vie de tous les jours et plus spécialement dans la carrière d'un ingénieur informatique, ce projet a été un bon premier pas pour nous initier à ce domaine.

Annexe

Annexe 1 : tableau montrant l'évolution du coût de la solution (colonne coût final), en fonction du coefficient multiplicateur de la température (colonne Tx temp), appliqué au fichier att532.

Fichier	Tx temp	Tx accept	nbIteMax	nb palier	Mult. temp	Coût final	Coût opti.	% diff	Temps (s)
att532	0,9	1%	20	n	1	50324	27686	81,77%	7,5
	0,92	1%	20	n	1	49716	27686	79,57%	7,2
	0,93	1%	20	n	1	48014	27686	73,42%	7,6
	0,95	1%	20	n	1	47093	27686	70,10%	9
	0,965	1%	20	n	1	50189	27686	81,28%	10
	0,97	1%	20	n	1	47055	27686	69,96%	11,19
	0,985	1%	20	n	1	43853	27686	58,39%	15,9
	0,99	1%	20	n	1	44769	27686	61,70%	21,7
	0,999	1%	20	n	1	38095	27686	37,60%	186,2

Annexe 2 : tableau montrant l'évolution du coût de la solution (colonne coût final), en fonction du taux d'acceptation (colonne Tx accept), appliqué au fichier att532.

Fichier	Tx temp	Tx accept	nbIteMax	nb palier	Mult. temp	Coût final	Coût opti.	% diff	Temps (s)
att532	0,95	1%	20	n	1	47093	27686	70,10%	9
	0,95	1,50%	20	n	1	54845	27686	98,10%	7,3
	0,95	2%	20	n	1	61515	27686	122,19%	6,859
	0,95	3%	20	n	1	71870	27686	159,59%	5,4