Projet d'Optimisation Stochastique

Implémentation de l'algorithme du Recuit Simulé

Problème du Voyageur de Commerce
Document de synthèse de résultats

Auteur :
Jeremie Bosom
Erwan Chaussy

Professeur :
M.Abdel Lisser

20 janvier 2014

1 Problématique

Nous avons eu plusieurs problèmes pour améliorer nos résultats lors de ce projet. C'est pour cela que nous avons implémentés des options afin de pouvoir tester plus facilement les différentes possibilités que nous avions.

La première question que nous nous sommes posée fut de savoir s'il valait mieux utiliser l'algorithme du Plus Proche Voisin ou de la Plus Proche Insertion, afin d'obtenir une solution acceptable de base. Après avoir répondu à cette question, nous nous sommes demandés quelle était l'influence de notre algorithme pour trouver une solution voisine sur les résultats du recuit simulé. Enfin nous nous sommes interrogés sur l'influence des paramètres du recuit sur nos solutions.

Ce document a pour but de retracer notre réflexion sur comment améliorer notre algorithme et les résultats que nous avons obtenus.

2 PPV ou PPI

Nous avons implémentés les algorithmes du Plus Proche Voisin et de la Plus Proche Insertion que nous abrégerons respectivement PPV et PPI. Nous pensions, de prime abord, que la PPI nous permettrait d'obtenir un cycle de base plus optimisé que le PPV. Néanmoins, après quelques tests, il s'est avéré que ce n'est pas le cas. Non seulement la PPI est plus gourmande au niveau du temps que le PPV, mais en plus la PPI obtient des résultats qui sont sensiblement moins bon que le PPV. C'est pourquoi nous avons décidé d'utiliser le PPV.

Afin d'améliorer le PPV, nous avons décidé d'essayer d'utiliser du multi-thread. Ainsi nous avons implémenté le Plus Proche Voisin Threadé, abrégé PPVT, qui fait du multi-thread pour trouver le point le plus proche du point actuellement étudié. Nous avons ainsi réussi à accélérer l'acquisition d'un cycle de base.

3 Influence de l'algorithme pour trouver une solution voisine

Nous avons choisi d'utiliser un algorithme proche de celui du 2-opt pour trouver une solution voisine pour la simple raison qu'avec les autres algorithmes que nous avons implémentés nous n'avions aucune amélioration. Vu le peu d'intérêt de ces résultats nous ne les montrerons pas ici. Néanmoins, nous avons laissé les algorithmes que nous avons implémentés dans notre code source. Pour voir les résultats qu'ils permettent, il suffit de lancer le programme en mode debuggage avec l'option "-verbose" en ayant changé l'appel à ces fonctions.

4 Influence des paramètres du recuit simulé

Lors de nos premiers essais les paramètres n'étaient pas du tout optimaux. Nous obtenions des résultats très proches de ceux donnés par le PPVT ou le PPV. Avoir la possibilité de modifier ces réglages grâce aux options que nous avions écrites nous fut très pratique.

5 Résultats finaux

Voici les résultats que nous avons obtenus et les pourcentages d'erreur par rapport aux valeurs optimales.

Instance	Coûts optimaux	Résultats	Différence (en %)
a280	2 579	2 888	12.0
att48	10 628	11 196	5.34
att532	27 686	31 778	14.8
br17	39	77	97.4
brazil58	25 395	26 573	4.64
fl1577	22 204	27 736	24.9
fl3795	28 723	32 339	12.6
fnl4464	182 566	204 748	12.2
kroB100	22 141	24 687	11.5
korB150	26 130	31 117	19.1
kroB200	29 437	31 322	6.40
kroC100	20 749	23 184	11.7
kroD100	21 294	23 381	9.80
pla7397	23 260 728	26 840 880	15.4
pr2392	378 032	442 813	17.1
rl5915	565 040	665 876	17.8
rl5934	554 070	648 874	17.1
u1060	224 094	290 935	29.8
vm1084	239 297	270 554	13.1
vm1748	336 556	396 296	17.8