Sujet 59 : Algorithmes pour la gestion des contraintes en optimisation combinatoire

Matthieu Caron

March 7, 2016

Chapter 1

Introduction

1.1 Présentation du sujet

Un problème d'optimisation combinatoire a pour but de trouver une solution optimisant un ou plusieurs objectifs et répondant à un ensemble de contraintes. Par exemple, le problème du sac-à-dos classique vise à sélectionner un sous-ensemble d'éléments dans une collection de n éléments en maximisant une ou plusieurs fonctions profit et satisfaisant une ou plusieurs contraintes de ressource (la capacité du sac), basée sur des fonctions poids. Une solution est alors spécifiée par une chaîne binaire de taille n, de sorte que chaque variable indique si l'élément correspondant est inclus dans le sous-ensemble des éléments sélectionnés (le sac) ou non.

Contrairement aux approches classiques de la programmation mathématique, les métaheuristiques sont des méthodes de haut niveau à usage général qui sont relativement simples à développer tout étant en mesure de fournir des solutions efficaces en pratique à des problèmes d'optimisation combinatoire difficiles et de grande taille. Lors de la conception de métaheuristiques pour ces problèmes optimisation combinatoire, il existe essentiellement trois catégories générales pour la gestion des contraintes : (1) pénaliser les solutions irréalisables, (2) réparer les solutions irréalisables, ou (3) concevoir une représentation et des opérateurs spécifiques pour le problème à résoudre. Il est bien entendu que la performance d'une technique de gestion de contraintes est fortement liée aux caractéristiques du problème à résoudre. Cependant, savoir quelle approche donnera de meilleures résultats reste une question ouverte.

1.2 Pourquoi avoir pris ce sujet

Tout d'abord car j'ai beaucoup aimé le cours d'Algo de Sophie Tison, ça m'a donné envie de trouver un sujet en rapport avec la résolution de problèmes difficiles. En suite parce que ce sujet allait m'apprendre quelquechose, car jusqu'à présent les algorithmes vus en cours nous apprenez à résoudre des problèmes mono-objectifs. Enfin parce que c'était un projet orienté recherche et qu'il me permettait une bonne autonomie dessus (travail seul, liberté du langage et de l'implémentation)

Chapter 2

le Projet

Pour le moment le travail demandé à été de travailler sur le problème de Flow-Shop de permutation. Il m'a donc fallut étudier le problème de Flow-Shop qui m'était inconnu pour ensuite pouvoir le modéliser. Enfin l'algorithme d'optimisation multi-objectifs demandé est celui du Pareto local search (PLS) à tester et comparer avec des variantes.

2.1 Flow-Shop de permutation

C'est un problème d'ordonnancement, on possède n taches à effectuer sur m machines et on connait p_{ij} soit le temps que met la tache i sur la machine j et enfin on demande a ce que chaque tache soit fini a un certain donné, soit d_i date de fin de la tache i.
Les contraintes :

- toutes les taches doivent s executer une fois sur chaque machine de la machine 1 à la machine m
- l'ordre des taches a effectuer reste le même pour chaque machine
- une machine ne peut traiter qu'une tache a la fois

On appelle ce problème Flow-Shop de permutation car les solutions quandidates peuvent être représentée sous forme d'une permutation soit l'ordre dans lequel on va effectuer les taches.

2.2 Pareto local search

Recherche locale, parce qu'a chaque tour on va observer ce qu'on va appeler un voisin et voir si ce voisin est une meilleure solution que la solution actuelle et sinon observer le voisin suivant et ainsi avancer vers une solution optimisée. Et optimum de Pareto du nom de son iventeur Vilfredo Pareto on ne peut pas améliorer le résultat d'un objectif sans déteriorer un autre.

2.3 Implémentation et Résultats

Le langage choisit est le python, pas pour une optimisation machine mais plutôt parce que je le connais bien et ça me permet de coder rapidemment dessus. Une classe pour l'instance du problème une classe pour une solution candidate, et les classes pour les différents voisinages.