Enseigner en même temps la Recherche Opérationnelle avec de bonnes pratiques de programmation, c'est possible !

Nicolas Dupin[1], Dominique Quadri[1]

Université Paris-Saclay, Laboratoire Interdisciplinaire des Sciences du Numérique (LISN)
{nicolas.dupin,dominique.quadri}@universite-paris-saclay.fr

Mots-clés : *recherche opérationnelle, enseignement, programmation parallèle.*

Dans la lignée des discussions sur le lien entre algorithmes de RO et questions d'implémentation, nous souhaitons partager des retours d'expérience sur ce sujet en enseignement. Nous avons expérimenté par la RO un apprentissage décloisonné d'algorithmes et de bonnes pratiques de programmation. Nous nous étions rendus compte que les étudiants de L3 ou de M1 n'avaient plus beaucoup codé depuis leurs L1 et L2, la L3 étant une année où les étudiants découvraient beaucoup d'unités d'enseignements (UE) (autres que la programmation) leurs permettant ensuite de choisir leur master. Ainsi, nous avons souhaité que la programmation soit plus présente dans nos UE de RO, à travers l'implémentation des algorithmes présentés. Une telle approche permet une présentation attractive de la RO comme un domaine faisant des ponts avec d'autres techniques de l'informatique. Trois illustrations sur différentes UE d'informatique de l'Université Paris-Saclay sont présentées dans la suite. Cette présentation vise à ouvrir des discussions, et susciter des partages de bonnes pratiques en enseignement de la RO.

1 Implémentation de solveurs de sac à dos

Avant de mettre un solveur de PLNE dans les mains d'étudiants de niveau L3, l'algorithme de Branch&Bound (B&B) est implémenté en TP sur le problème de sac à dos. Les mécanismes de bornes et de recherche arborescente B&B peuvent être appréhendés sans avoir à implémenter l'algorithme du simplexe, grâce au calcul glouton de relaxation continue. L'implémentation d'un tel TP était auparavant dans un langage au choix, et beaucoup d'étudiants implémentaient une version récursive pour un parcours DFS de l'arbre, similairement à leurs cours d'OCaml. Un code à remplir en C++ est à présent fourni pour leur permettre d'expérimenter d'autres types de parcours (BFS, aléatoire, meilleure relaxation), mais aussi différentes heuristiques primales. Le code fourni comprend un parseur de données d'entrée et des fonctions élémentaires pour concentrer le travail sur l'implémentation des opérateurs clés de l'optimisation. L'architecture du code fourni comprend des solveurs gloutons (entier et continu), un solveur de programmation dynamique, un solveur heuristique Kernel Search ([1]) basé sur la programmation dynamique, et des classes nécessaires pour le solveur B&B. Un tel travail mobilise des connaissances en POO, en illustrant des bonnes pratiques pour l'efficacité et la généricité du code avec plusieurs variantes d'opérateurs. Il a également été proposé un travail d'approfondissement sur l'algorithmie et la qualité de la programmation, en implémentant un démarrage à chaud de la relaxation continue à chaque noeud de l'arbre B&B, mais aussi en réfléchissant aux conteneurs et opérateurs adaptés pour plus d'efficacité du code, quitte à être moins générique sur les parcours d'arbre. Au final, le problème du sac à dos présente une grande richesse pour illustrer et mobiliser des techniques de programmation pour des étudiants de niveau L3.

2 Programmation parallèle et méta-heuristiques

La programmation parallèle et distribuée avec MPI (Message Passing Interface) est enseignée en M1 à l'université Paris-Saclay. Il est possible d'enseigner MPI conjointement avec une introduction au Machine Learning [5]. Dans la version 2020-2021 du cours, l'UE est rattachée à la filière ANO (Advanced Network and Optimisation), un cadre adapté pour des applications est d'illustrer la parallélisation d'algorithmes d'optimisation combinatoire. La programmation dynamique est un cadre d'étude intéressant pour une implémentation MPI, pour l'extensivité de l'espace mémoire. Des problèmes de recherche arborescente permettent d'illustrer la parallélisation master-worker avec MPI et des communications asynchrones pour optimiser l'équilibre de charge avec des tâches de durées variables et non connues a priori. Un tel TP peut se faire sur la recherche de sous-isomorphisme dans un graphe par algorithme de backtracking distribué (comme dans [5]), ou dans une version optimisation combinatoire avec critère d'élagage comme dans le problème Maximum Independent Set. Ce cours est aussi l'occasion de présenter une vision générale des méta-heuristiques dans le parcours ANO, dans l'esprit de [6]. Les méta-heuristiques de population ou l'hybridation d'heuristiques sont des cadres privilégiés à une implémentation distribuée avec MPI : la parallélisation est à gros grain avec peu de synchronisation et de communications, cela correspond aux conditions d'utilisation pratique de MPI, comme expérimenté par [4].

3 Travaux d'études en Recherche (TER)

Les TER constituent une initiation à la recherche pour étudiants de M1, sur des créneaux de 4h par semaine le second semestre. Une première phase comporte des TP traditionnels, pour acquérir les compétences nécessaires à la réalisation d'un sujet spécifique par binôme, avec une fort part expérimentale. Un sujet a été proposé autours d'algorithmes de clustering appliqué au cas spécifique d'un Front de Pareto en 2D [3, 2]. Ce cas spécifique permet une résolution optimale par programmation dynamique, dont l'implémentation efficace (en séquentiel ou en parallèle), illustre l'intérêt de techniques de programmation. La partie expérimentale, avec plusieurs variantes de solveurs à comparer, illustre l'intérêt d'utiliser proprement la POO et l'héritage pour de la bonne factorisation de code. Un tel cadre permet des approfondissements sur des questions d'algorithmie ou de parallélisation pour accélérer la résolution exacte de solveurs. D'autres approfondissements s'intéressent à expérimenter des techniques heuristiques pour une résolution efficace en temps contraint, ou réaliser des expériences statistiques sur des extremas locaux. Sur de tels projets, la qualité de la programmation est incontournable, au profit de l'efficacité des implémentations, et de la génération automatique de tableaux de résultats pour répondre à des expériences numériques. En outre, un tel projet permet d'apporter des éclairages sur les algorithmes de clustering tels que présentés dans le cours de Machine Learning (ML), illustrant la complémentarité entre les domaines de la RO et du ML.

Références

[1] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28(5) :1130–1154, 1980.

[2] N. Dupin, F. Nielsen, and E. Talbi. k-medoids clustering is solvable in polynomial time for a 2d Pareto front. In *World Congress on Global Optimization*, pages 790–799. Springer, 2019.

[3] N. Dupin, F. Nielsen, and E. Talbi. Unified Polynomial Dynamic Programming Algorithms for P-Center Variants in a 2D Pareto Front. *Mathematics*, 9(4) :453, 2021.

[4] N. Dupin and E. Talbi. Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. *Int. Trans. Oper. Res.*, 27(1) :219–244, 2020.

[5] F. Nielsen. *Introduction to HPC with MPI for Data Science*. Springer, 2016.

[6] E. Talbi. *Metaheuristics : from design to implementation*, volume 74. John Wiley & Sons, 2009.