Un Algorithme de Colonie de Fourmis pour la Planification de Formations en Santé sous Contrainte de Ressources *

Simon Caillard, Corinne Lucet, Laure Brisoux-Devendeville

Laboratoire MIS (UR 4290), Université de Picardie Jules Verne
33 rue Saint-Leu, 80039 Amiens Cedex 1, France
{simon.caillard, corinne.lucet, laure.devendeville}@u-picardie.fr

Mots-clés : *recherche opérationnelle, optimisation, santé, emploi du temps, planification*

1 Introduction

SimUSanté, située à Amiens, France, est l'un des plus grands centres de formation et de simulation en santé, avec plus de 400 formations proposées dans les différents domaines de la santé pour de nombreux publics différents. Le personnel dispose de multiples compétences et les types salles où se déroulent les formations sont variées et modulables. Planifier les sessions de formations en vue d'obtenir un emploi du temps qui respecte l'ensemble des contraintes de temps et de ressources est difficile et de première importance. Ce problème est dérivé de celui du Curriculum-Based Course Timetabling problem (CB-CTT)[4] qui est NP-Difficile[3]. Pour résoudre ce problème, nous proposons dans cet article un algorithme MinMax de colonie de fourmis hybridé avec une recherche locale adaptée[2].

2 Définition du problème

Soit H un ensemble de créneaux sur lequel un ensemble de ressources R, composé d'employés et de salles, est disponible ou non. Soit Λ l'ensemble des types de ressources. Un type de ressource peut correspondre à une compétence associée à un ou plusieurs employés ou à une caractéristique de salle. Pour chaque type de ressources $\lambda \in \Lambda$, on connait $qtavail_\lambda^t$ la quantité de ressources disponible au créneau $t \in H$. De par le grand nombre de formations proposées, les ressources du centre sont à la fois variées et flexibles et ainsi chaque ressource possède un ensemble de types de ressources. De plus, il existe un ensemble de sessions S où toute session $s \in S$ est composée d'un ensemble d'activités A_s. Chaque activité $a \in A_s$ a une durée, un ensemble de prédécesseurs $pred_a \in A_s$ ainsi que pour tout $\lambda \in \Lambda$, une quantité $qtreq_\lambda^a$ de ressources de type λ nécessaire à sa réalisation. On note $A = \bigcup_{s \in S} A_s$ l'ensemble des activités.

Une solution Sol est un ensemble de triplets (a, t, c), avec $a \in A$ une activité, $t \in H$ son créneau de début et $c \subseteq R$ l'ensemble des ressources qui répond à ses besoins et qui lui sont affectées. On note UA l'ensemble des activités qui n'ont pu être planifiées. Pour toute session s, Sol_s est l'ensemble des triplets relatifs à s, t_{start_s}, son créneau de début et t_{end_s} celui de sa fin.

Notre objectif est de planifier des activités, tout en minimisant la fonction objective $Eval()$ définie par l'équation 1. $Eval(Sol)$ calcule pour une solution Sol la somme des makespan de l'ensemble des sessions plus la somme des pénalités associées aux activités non planifiées. Minimiser $Eval()$ revient donc à planifier un maximum d'activités, tout en compactant l'emploi du temps.

*Ce projet est supporté par la région Hauts de France

$$Eval(Sol) = \sum_{s \in S}(t_{end_s} - t_{start_s}) + |UA| \times |H| \quad (1)$$

3 Méthode de résolution

Pour résoudre le problème de SimUSanté, $SimUACO$, un algorithme MinMax de colonie de fourmis[5] est utilisé conjointement avec la recherche locale $SimULS$[2]. Le principe de ce type d'algorithme s'inspire du comportement des fourmis qui déposent des phéromones sur leur chemin afin de guider d'autres fourmis et permettre ainsi de trouver le chemin le plus rapide entre leur nid et une source de nourriture.

Soit K un ensemble de fourmis. À chaque itération, chaque fourmi $k \in K$ va construire une solution $Sol^k = \{(a_1, t_1, c_1), \ldots, (a_i, t_i, c_i)\}$ qui représente le chemin emprunté par k. Puis, pour toute fourmi $k \in K$, $SimULS$ est appliqué à Sol^k avec une probabilité de $\beta\%$. Enfin, l'ensemble des phéromones sont évaporées de $\gamma\%$ et des phéromones additionnelles sont déposées sur le chemin correspondant à la meilleure solution trouvée depuis le début de l'algorithme.

Pour construire Sol^k, une fourmi k sélectionne de manière itérative chaque activité et lui affecte un créneau et des ressources. Ce processus se fait en deux temps. Pour sélectionner une activité a, la fourmi k utilise une règle de proportionnalité aléatoire se basant sur une heuristique qui estime le nombre de possibilités de planification restantes de a. Pour affecter un créneau et des ressources à a, la règle de proportionnalité aléatoire utilisée se base sur les phéromones déposées par les fourmis précédentes ainsi que sur une heuristique composée de 3 critères : la variation de $Eval(Sol^k)$, le nombre de créneaux inutilisables, et le nombre de ressources multitypes utilisées.

4 Résultats

Pour tester notre algorithme, nous avons généré des instances[1], reprenant toutes les caractéristiques du problème de SimUSanté, à partir de celles du CB-CTT. Nous comparons les résultats obtenus par $SimUACO$, $SimULS$ et un modèle mathématique[2] implémenté sous CPLEX. CPLEX parvient à obtenir les résultats optimaux uniquement sur de petites et moyennes instances. $SimULS$ obtient des résultats ayant un écart moyen de 11,50% avec les résultats optimaux de CPLEX, tandis que $SimUACO$ améliore en moyenne de 6% les résultats obtenus par $SimULS$. L'écart avec l'optimalité tombe alors à 4.95%. Les résultats seront détaillés lors de la conférence.

Références

[1] Simon Caillard, Laure Brisoux-Devendeville, and Corinne Lucet. Health Simulation Center Simusanté®'s Problem Benchmarks. https ://mis.u-picardie.fr/en/Benchmarks-GOC/.

[2] Simon Caillard, Laure Brisoux-Devendeville, and Corinne Lucet. Local Search for a Planning Problem with Resource Constraints in Health Simulation Center. preprint hal-03162224 : https ://hal.archives-ouvertes.fr/hal-03162224, March 2021.

[3] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In *Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, UK, August 29 - September 1, 1995, Selected Papers*, volume 1153 of *Lecture Notes in Computer Science*, pages 283–295. Springer, 1995.

[4] Luca Di Gaspero, Barry McCollum, and Andrea Schaerf. The Second International Timetabling Competition (itc-2007) : Curriculum-Based Course Timetabling (track 3). Technical report, 2007.

[5] Thomas Stützle and Holger H. Hoos. Max–min ant system. *Future Generation Computer Systems*, 16(8) :889–914, 2000.