SCIENCES SORBONNE UNIVERSITÉ

Master 1 ANDROIDE

2019-2020

P-ANDROIDE
Automatisation de construction de planning

Cahier des charges

Kevin Nartz,
Ariane Ravier,
Anthéa Richaume,
Bassem Yagoub

Supervisé par Pierre Fouilhoux et Cédric Herpson

Contexte et définition du problème

Ce P-ANDROIDE porte sur l'automatisation de construction de planning. Il s'agit de proposer une répartition optimale des postes respectant un ensemble de contraintes données.

Le projet trouvera une application au sein du service des sages-femmes du Centre Hospitalier Intercommunal de Créteil (CHIC), qui nous a fourni une liste de contraintes à respecter au mieux et des exemples de calendriers qui les respectent afin d'avoir une idée du résultat attendu.

Objectif du projet

Proposer une méthode de construction de planning à l'aide de divers outils de recherche opérationnelle. Cette méthode doit proposer la meilleure solution possible (approchée ou non) à partir des contraintes données, et ce en un temps raisonnable.

Jusqu'ici, les plannings du CHIC sont construits à la main, ce qui représente un total de 4 à 5 jours de travail pour deux sages-femmes. Notre objectif est de réussir à réduire ce temps de construction à moins de deux jours.

Si ce projet a pour application concrète une satisfaction des contraintes propres au service des sages-femmes du CHIC, le logiciel proposé devra s'appliquer à des cas plus généraux.

Contraintes organisationnelles

Voici la liste des contraintes identifiées :

- Fournir une solution respectant les contraintes légales de travail
- Les calculs d'une solution doivent se réaliser en temps raisonnable (5 à 10 mn maximum)

- Le logiciel produit doit être libre (Licence GPL ou LGPL)
- Le logiciel doit être gratuit à la production et à la distribution
- Doit être rendu au plus tard le 22 mai 2020

Spécifications techniques du logiciel

1. Saisie des données

La saisie des données dans le logiciel à réaliser pourra se faire manuellement. Étant donné que la responsable construit ses plannings à partir de fichiers Excel recensant les contraintes fixes (congés, postes fixes, etc.), il serait idéal que le logiciel puisse également prendre en entrée un fichier de format CSV.

On pourra pré-définir certains types de contraintes, que l'utilisateur pourra choisir ou non d'appliquer.

2. Génération de planning

Le logiciel retournera une suggestion de planning. Si la construction a nécessité des modifications (e.g. relâchement de contrainte), il retournera également un à plusieurs plannings approchés réalisés à partir d'étapes à définir. Ces étapes indiqueront alors de manière incrémentale de plus en plus de contraintes à respecter afin de converger vers une solution qui les respecte toutes, pouvant servir de base à la construction manuelle.

Dans l'idéal, et comme précisé plus haut, la génération du planning ne devra pas prendre plus de 5 à 10 minutes.

3. Visualisation des résultats

L'objectif sera d'offrir un moyen de lecture des solutions directement via l'interface, ou sous un format CSV. Idéalement, il s'agira de pouvoir diviser la visualisation du planning par service ou par agent.

Stratégie d'approche

1) <u>Modélisation</u>

Dans un premier temps, il s'agira de modéliser les agents selon leurs caractéristiques et les contraintes qui peuvent les affecter (e.g. travail à mi-temps, agent débutant, etc.), les services et les postes.

Il nous faudra aussi formaliser les contraintes et les organiser par classes, pour pouvoir notamment distinguer les contraintes liées aux questions légales, de celles préférentielles.

Parallèlement, nous pourrons chercher à produire des instances de tailles et difficultés diverses.

Nous prévoyons de consacrer environ deux semaines à cette étape de modélisation.

2) Résolution

On explorera ensuite deux méthodes de résolution pour essayer de trouver des solutions acceptables.

a) Approche par Programmation linéaire en nombres entiers (PLNE)[1]

La première méthode consistera à formaliser le problème sous forme de programme linéaire en nombres entiers (PLNE). Nous utiliserons des variables binaires pour exprimer la présence d'un agent à un poste précis tel jour. La taille du problème (et des solutions à vérifier) pouvant grandir très rapidement, nous chercherons un moyen de réduire sa taille sans perdre de données par des "pré-calculs" de solutions sur les jours par exemple. Pour résoudre le PLNE nous utiliserons MIPCL, un solveur de programmes linéaires libre (licence LGPL) en C++.

Cette approche a l'avantage de proposer des solutions optimales exactes, mais son inconvénient principal est sa difficulté pour passer à l'échelle pour des tailles importantes et son manque de flexibilité et de lisibilité. En effet, si le programme linéaire ne trouve pas de solution exacte, il nous est difficile de déterminer de quelles contraintes proviendrait l'incompatibilité.

b) Approche heuristique[2]

Pour la deuxième méthode, on s'attachera à la conception de méthodes heuristiques permettant notamment d'obtenir des solutions approchées. On pourra notamment s'intéresser à différents types d'heuristiques telles que les heuristiques dédiées, les méta-heuristiques ou encore les algorithmes génétiques.

On définira dans ce cadre un *checker* et un évaluateur permettant respectivement de valider et de donner un retour sur la qualité de la solution obtenue.

Cette approche a l'avantage d'être extrêmement flexible, nous assurant donc de trouver une solution. Cependant, elle ne nous assure pas de trouver la solution optimale, et il nous faudra donc évaluer la qualité des différentes solutions. Cela nécessitera postérieurement un travail de *checking* et d'évaluation, effectué grâce à

---

[1] Genova, Krasimira, et Vassil Guliashki. « Linear Integer Programming Methods and Approaches–A Survey ». *Cybernetics and Information Technologies* 11 (1 janvier 2011).

[2] Silver, Edward. « An overview of heuristic solution methods ». *Journal of The Operational Research Society - J OPER RES SOC* 55 (26 mai 2004): 936‑56. https://doi.org/10.1057/palgrave.jors.2601758.

un *checker*, mais également, pour des instances de taille réduite, par comparaison avec les solutions obtenues par PLNE.

Ces deux tâches étant indépendantes, et la structure de notre groupe le permettant, nous envisageons de les réaliser en parallèle sur une période d'au moins un mois.

Nous procéderons à une comparaison des résultats obtenus selon chacune des approches pour déterminer laquelle utiliser pour notre méthode de construction de de planning. Nous pourrons éventuellement utiliser une combinaison des deux approches.

Le code sera développé en parallèle de nos avancées.

3) <u>Réalisation de l'interface</u>

Enfin, nous consacrerons le temps restant à la réalisation d'une interface utilisateur à l'aide de QT sous licence GPL en C++.

Notre proposition de solution théorique sera validée par nos encadrants, et la qualité des solutions concrètes sera évaluée par la responsable du planning au CHIC.

D'autre part, celle-ci nous fournira les plannings des mois à venir afin de nous permettre de comparer les résultats obtenus.

Planning prévisionnel