Compte rendu : projet CSDC 17

Table des matières

Parser Java

La première partie du projet consistait à convertir les données du xlsx en données utilisables par la partie EASEA. Pour ce faire, nous sommes partis sur un Parser en Java réalisé à l'aide de la librairie Apache Poi qui extrait les informations importante dans le fichier excel. Elle les transforme et les stocke dans un format spécifique facile à utiliser qui sera expliqué plus en détail dans la partie structure des données.

Nous avons donc commencé par étudier le fichier Excel afin de regarder les informations que nous souhaitions conserver. Nous avons cependant corrigé une des données de l'Excel qui était erroné : dans l'onglet du track "8", il y avait un intervalle où il manquait un crochet ouvrant. Nous l'avons donc rajouté pour éviter de faire un traitement spécifique à ce cas.

Pour créer le fichier qui sera utilisé plus tard, le parser va parcourir les différents onglets (les tracks) qui nous intéressent. Chaque onglet sera parcouru ligne par ligne, en ignorant la première ligne qui contient le type des informations des colonnes.

En même temps qu'il récupère les informations, le parser va les formater pour qu'elles soient uniformes et puissent être lues plus facilement par la suite. Ainsi, il a fallu convertir les différents formats de fuseau horaire à l'aide d'une regex (expression régulière).

Il a également fallu uniformiser les heures des durées et des intervalles de disponibilités qui étaient écrites de deux manières différentes.

La première était un nombre décimal alors que la deuxième était sous forme heures:minutes. Une fois le traitement de l'Excel fini, il ne restait plus qu'à stocker les résultats dans un fichier texte où les informations sont séparées par des |, pour permettre au programme de bien interpréter les données.

Structure des données

Pour simplifier le problème d'affectation des papiers dans l'emploi du temps, nous avons décidé de stocker directement une date de démarrage pour chaque paper plutôt qu'une date de démarrage de session suivi d'un ordre de passage dans la session.

Pour commencer, on a défini deux structures : Paper qui représente les données propres à un Papier et une structure Interval permettant de représenter les intervalles de disponibilité des chercheurs.

La structure Paper comprend trois identifiants (ID pour le paper, ID_Session et ID_Track), la durée du track (duration), le fuseau horaire du chercheur (fuseauHoraire) et 3 listes d'intervalles pour stocker les disponibilités du chercheur sur chaque jour de la conférence (dispoDay1, dispoDay2 et dispoDay3).

La structure Interval quant à elle contient une heure de début et une heure de fin.

Import des données

Après la finalisation du parser Java, on pouvait désormais procéder à l'import des données présentes dans le fichier texte généré. Ce fichier texte a été formaté afin d'être facilement importé par le programme.

Au moment de l'import, on récupère les différentes données dans le fichier txt qui sont séparées par des '|' et on les stocke dans une liste de Paper.

Quelques particularités au moment de l'import sont :

- Lorsqu'une donnée est absente dans le xls (marqué par une * dans le txt), on affecte une valeur par défaut au champ concerné.
- Pour l'import des intervalles, nous avons défini une fonction strToInterval capable de convertir les différents intervalles présents dans le fichier en un tableau d'Interval.

Classes du Génome

Dans le génome, nous manipulons des dates, pour ce faire nous avons dû créer une classe CustomDate car on ne pouvait pas utiliser la structure tm dans la section de déclaration des classes du génome.

Notre classe du génome contient une liste de CustomDate correspondant à la liste des dates où la présentation de chaque papier va démarrer.

Initialisation

La fonction d'initialisation affecte à chacun des papiers une date de démarrage (startDate) au hasard en respectant les contraintes de la conférence (date comprise entre le 30/09 à partir de 6h et le 1/10 jusqu'à 23h59). On obtient alors une première liste de date de démarrage des différents papiers qui sera ensuite améliorée au fur et à mesure des itérations.

Croisement

Pour simuler le croisement de deux emplois du temps, en l'occurrence parent1 et parent2, nous avons utilisé la fonction tossCoin(). On itère sur la liste des papiers pour sélectionner aléatoirement le papier de l'un des deux parents comme source pour l'emploi du temps fils généré. On affecte donc l'un ou l'autre horaire de démarrage du papier actuellement parcouru à la variable children en fonction du résultat du lancer de pièce.

Mutation

La fonction mutation va permettre de faire muter un certain pourcentage de la population, on parcourt l'ensemble des dates du génome avec une certaine chance de muter.

Pour définir le taux de mutation, nous avons repris la formule présente dans le programme de 2015. On utilise ensuite la fonction tossCoin(), qui prend en paramètre le taux de mutation (en réalité le biais du lancer de pièce). Si mutation il y a, alors on affecte une nouvelle date aléatoire (de façon similaire à l'initialisation) au papier concerné, sinon on conserve la date actuelle.

Evaluation

Pour la partie évaluation, nous sommes partis sur un score qu'on cherche à minimiser : à chaque règle non respectée, nous avons augmenté le score du planning d'un montant choisi au préalable.

Afin de faciliter la vérification au niveau des dates, nous avons créé des fonctions utilitaires pour convertir les dates et ajouter des durées. Nous avons également une fonction qui vérifie que deux intervalles de temps ne se chevauchent pas, en traitant les différents cas possibles.

Dans un premier temps, nous vérifions que l'horaire de présentation du paper respecte bien les contraintes horaires du chercheur. Nous vérifions ensuite qu'il n'y a pas de chevauchement de papiers lors d'une même session ou de papiers empiétant sur une session plénière. Nous avons également réfléchi à la vérification de l'écart pour éviter les fractures entre les sessions, mais nous n'avons pas finalisé l'algorithme, seule une ébauche est définie en commentaire.

Résultat

Une fois que l'algorithme a terminé de s'exécuter, on stocke le meilleur résultat dans le fichier output.txt. Ce fichier contient la liste des informations sur l'emploi du temps généré.

Germain CLAUSS et Kévin BIER