Rapport de Soutenance 1

Janvier/Mai 2023

Alicia Houset, Ange Mercoyrol-dol, Nathan Fontaine, Dimitri Brancourt

Avant-propos

Bonjour, voici le premier rapport de soutenance après deux semaines de travail suite à l'envoie du cahier des charges

Table des matières

1 Introduction

1.1 Rappel du sujet

OptiPlan est une application de gestion du temps. Pour être plus précis, c'est une application qui nous permet de gérer des tâches ou des projets sur un emploi du temps virtuel. Le but de cette application est l'optimisation de notre temps. Nous voulons donc pouvoir mettre à jour un emploi du temps dans lequel nous pouvons ajouter ou supprimer des tâches ou des projets de manière efficace tout en mettant à jour l'avancement jour après jour. Il est également nécessaire d'implémenter différents moyens de communication entre cette application et l'utilisateur, tels qu'une interface graphique ou des notifications, afin d'améliorer son effcacité. Bien sûr, nous devrons sauvegarder l'emploi du temps afin qu'il soit effectif.

2 Fonctionnalités du projet

2.1 Algorithme de structure d'emploi du temps

2.1.1 Intro

Ce que nous entendons par algorithme de structure d'emploi du temps, c'est simplement l'ensemble du code permettant de calculer les jours suivant une date donnée et de gérer nos structures de données.

2.1.2 Gestion des structures

Tout d'abord commencons par le commencement, afin de pouvoir gérer efficacement les données à utiliser dans un emploi du temps.
Pour ce faire on utilise les struct, on a donc 3 structures dans notre projet, la structure week qui représente une semaine, la structure day qui représente une journée et enfin la structure action qui représente les actions à faire et à ajouter dans l'emploi du temps.

La structure week est composée d'une liste de 7 jours, ainsi que d'un label, qui est une simple chaîne de caractères décrivant briévement la semaine (Vacances, Exams ...).
La strucutre day est composée de la liste d'actions, ainsi que du

nombre d'actions, le nom de la journée, et enfin la date, jour mois et année.
Enfin, la structure action est composée de l'heure de début, de fin ainsi que du nom de l'action et sa courte description.

2.1.3 Calcul des jours

Pour la partie du calcul des jours, il n'y a pas de secrets, il fallait simplement gérer tous les cas un par un.
C'est à dire, il faut gérer les cas du changement de mois, qui varie en fonction du nombre de jour dans chaque mois, le cas du changement d'année en décembre et efin le cas du mois de février.

2.1.4 Difficultés rencontrées

Pour cette partie la seule difficulté était de trouver une façon efficace de gérer les données de la semaine.
La première idée était de créer des listes de données, mais il s'est rapidement avéré que les structures de données était bien plus pratiques et efficace.

2.2 Interface

Pour l'interface graphique de notre application, nous avons utilisé Glade. Glade est un outil interactif de conception graphique GTK+. Il prend en charge la partie gestion/génération de l'interface pour permettre au développeur de se concentrer sur la partie développent et sur le code dit ńutileż. GtkBuilder permet quant à lui de lire ces fichiers dynamiquement.

Dans un premier temps, il a fallut se familiariser avec Glade, connaître et comprendre ses différentes fonctionnalité pour ensuite les initialiser avec GTK dans le code. Nous avons alors commencer l'interface en ayant pour le moment la base pour notre application. Cest à dire la page principal où sera affiché lemploi du temps, avec différentes boutons. Nous avons donc mis plusieurs bouton à laide de GtkButton, soit un bouton Add pour lequel lorsquon appuie dessus nous pouvons ajouter un événement au calendrier, des boutons Next et Return afin de voir le planning des semaines précédentes ou des semaines suivantes.

Un bouton Paramètres pour changer les différents paramètres de linterface, un bouton Research qui permettra à l'utilisateur de rechercher la date dun événement particulier et pour finir un bouton Exit pour fermer l'interface et donc quitter l'application.

L'interface est pour le moment au niveau de commencement et de visualisation, malgré le fait que les bouton soit initialisés, elle nest pas entièrement fonctionnelle, ce qui est lobjectif pour les prochaines soutenance.

2.3 Implémentation du calendrier

Pour les prochines soutenances, le principal objectif sera l'implémentation d'un système d'interaction entre l'interface utilisateur et le code dans notre application afin de permettre aux utilisateurs de saisir leurs besoins d'emplois du temps, tels que les horaires de travail et les préférences de temps libre, via l'interface utilisateur. Ces informations seront ensuite transmises au code pour optimiser les emplois du temps de manière efficace.

Nous voudrion également mettre en place des champs de saisie pour les horaires de travail et les préférences de temps libre, ainsi qu'un bouton de soumission pour envoyer ces informations au code. Il faudra également intégrer des mécanismes de validation pour garantir que les données saisies sont correctes.

Pour améliorer cette fonctionnalité, nous prévoyons de mettre en place un système de sauvegarde de données pour permettre aux utilisateurs de conserver leurs emplois du temps précédents et de les modifier facilement. Nous prévoyons également de mettre en place des mécanismes de suggestion d'horaires, en fonction des préférences de l'utilisateur et des contraintes du code, pour aider les utilisateurs à optimiser leur emploi du temps de manière plus efficace.

Enfin, nous souhaitons également améliorer l'interface utilisateur en la rendant plus intuitive en ajoutant des fonctionnalités telles que la possibilité de visualiser l'emploi du temps généré et de le modifier directement.

2.4 Algorithme de gestion de temps

La programmation par contrainte est une réponse au problème posé par la gestion d'un emploi du temps de façon optimisée. En effet un emploi du temps représente en lui-même de nombreuses contraintes (impossibilité de poser un créneau sur les mêmes horaires, difficultés de changer d'horaires certaines activités, transports, etc).

Mais qu'est ce que la programmation par contrainte ?

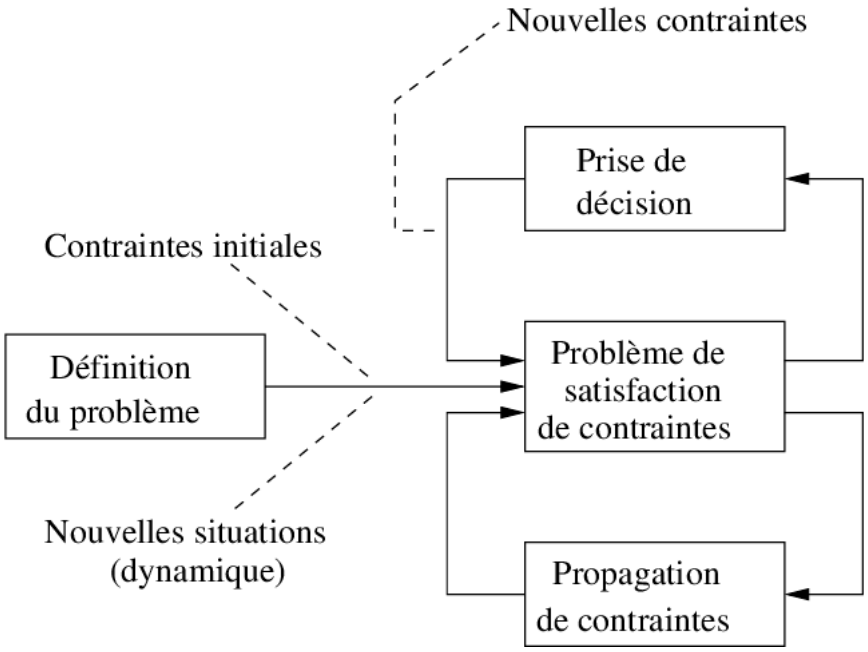
La programmation par contraintes est une technique de résolution des problèmes combinatoires complexes issue de la programmation logique et de l'intelligence artificielle et apparue à la fin des années 1980. Elle consiste à modéliser un problème par un ensemble de relations logiques, des contraintes, imposant des conditions sur l'instanciation possible d'un ensemble de variables définissant une solution du problème. Un solveur de contraintes calcule une solution en instanciant chacune des variables à une valeur satisfaisant simultanément toutes les contraintes.

Les types d'application traités par la programmation par contraintes sont très variés. Parmi les domaines dapplication les plus classiques, on pourra citer la gestion du temps ou d'affectation de ressources, la planification, etc.

La propagation des contraintes seule est généralement insuffisante pour exhiber une solution, il est alors nécessaire d'appliquer un algorithme de recherche pour trouver une ou plusieurs solutions.

Un exemple de programmation par contrainte que nous avons utilisé au S3 est l'algorithme pour la résolution du sudoku. Ici les contraintes étaients les règles du sudoku et la méthode de recherche la plus commune est la recherche arborescente avec retour arrière.

La principale originalité de la programmation par contrainte c'est qu'il sagit d'une approche qui favorise la séparation entre la définition formelle du problème, les mécanismes d'analyse et de filtrage du système de contraintes induit par cette définition (propagation) et les méthodes de résolution(voir image ci-dessous).



Cette caractéristique nous paraît particulièrement importante pour la conception de systèmes flexibles permettant la coopération entre l'humain et le système. Ici cela permettrait à l'utilisateur de choisir des contraintes différentes car la notion d'un agenda parfait est subjectif à chacun (certaines personnes préfèrent se coucher et de se lever plus tôt et inversement).

Ensuite, en programmation par contraintes, le traitement des contraintes nest pas globalisé, mais au contraire particularisé en fonction du type de la contrainte traitée, suivant l'assertion "1 contrainte = 1 algorithme".

Nous avons ici bien défini ce qu'est la programmation par contrainte. Nous allons voir ici une proposition de comment cet algorithme va être réalisé. Ce qui a déjà été fait, ce qui est à faire ainsi que ce qui peut-être modifié. Ils existent plsuieurs étapes essentielles

2.4.1 Collecte de données

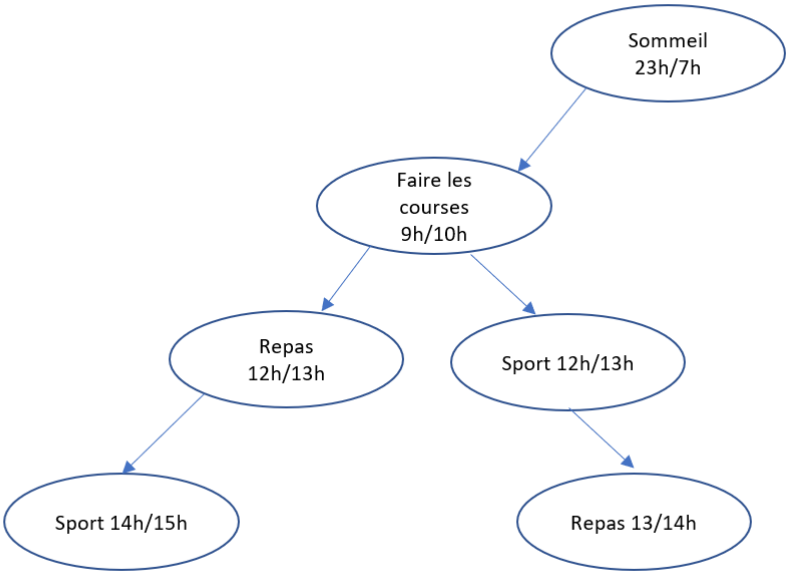
La première étape pour créer un algorithme de satisfaction des contraintes pour un emploi du temps optimisé consiste à collecter toutes les données nécessaires. Ces données peuvent inclure les informations de l'utilisateur, telles que les heures de travail, les heures d'étude, les heures de loisirs, les préférences personnelles, etc. Les données peuvent également inclure des informations sur les contraintes qui s'appliquent à chaque variable, telles que l'heure de début et de fin, la durée, le temps de trajet,le besoin de repos après, etc. Toutes ces

informations sont inclues à l'intérieurs de la Structure Action que l'on peut récupérer en récupérant une à une toutes les actions de la semaine que l'on souhaite gérer.

2.4.2 Création d'une structure de données

Une fois que toutes les données nécessaires ont été collectées, la prochaine étape consiste à créer une structure de données qui représente l'emploi du temps. Cette structure de données doit inclure des variables pour chaque action : l'heure de début et de fin, la durée, si cette action peut-être déplacée ou non etc. Les variables doivent être liées les unes aux autres pour refléter les contraintes qui s'appliquent à chaque action.

Pour ce problème nous pensons utiliser des graphes orientés, avec un graphe par jour. Cela permet d'avoir une image assez rapide des solutions que les algorithmes propose à la fin afin que l'utilisateur puisse choisir à la fin ce qui lui convient le mieux.



2.4.3 Définition des contraintes

Une fois que la structure de données est créée, la prochaine étape consiste à définir les contraintes qui s'appliquent à chaque variable. Ces contraintes peuvent inclure des contraintes de temps, telles que l'heure de début et de fin, la durée, le jour de la semaine, etc. Les contraintes peuvent également inclure des contraintes de ressources, telles que si

l'on doit prévoir de rentrer chez nous afin de prendre tel ou tel dossier, etc. Les contraintes peuvent également être liées les unes aux autres pour refléter les relations entre les différentes variables. Par exemple, la variable représentant la flexibilité sera liée au type de l'action. Un repas est plus facilement déplaçable qu'un rendez-vous médical.

2.4.4 Définition des objectifs

Une fois que les contraintes sont définies, la prochaine étape consiste à définir les objectifs de l'optimisation. Ces objectifs peuvent être spécifiques à l'utilisateur et peuvent être pondérés pour refléter leur importance relative. Par exemple, l'utilisateur peut vouloir minimiser le temps de trajet entre les activités, maximiser le temps de travail ou d'étude, minimiser les conflits de calendrier, ou maximiser le temps de loisirs.

2.4.5 Satisfaction des contraintes

Une fois que les données, les contraintes et les objectifs sont définis, l'algorithme de satisfaction des contraintes peut être appliqué.Il existe plusieurs approches pour appliquer cet algorithme, mais la méthode la plus courante est l'utilisation d'un solveur de contraintes. Un solveur de contraintes est un outil qui permet de trouver une solution qui respecte toutes les contraintes et maximise les objectifs définis. Le solveur de contraintes utilise un algorithme qui itère sur toutes les variables de l'emploi du temps et qui cherche une valeur pour chacune qui respecte les contraintes. Cet algorithme utilise des techniques de propagation de contraintes pour éliminer les valeurs qui ne respectent pas les contraintes et pour restreindre l'espace de recherche.

La méthode de propagation de contraintes est une technique qui consiste à utiliser des règles logiques pour déduire des informations sur les variables à partir des contraintes qui s'appliquent à elles. Cette technique permet d'éliminer les valeurs qui ne respectent pas les contraintes et de réduire l'espace de recherche. Les règles logiques utilisées par la propagation de contraintes sont basées sur des propriétés mathématiques telles que la transitivité, la symétrie, la réflexivité, etc.

Une fois que le solveur de contraintes a trouvé une solution qui respecte toutes les contraintes, il peut appliquer une méthode

d'optimisation pour maximiser les objectifs définis. Cette méthode d'optimisation peut être basée sur une recherche locale ou globale, selon la complexité de l'emploi du temps et le nombre d'objectifs définis. La méthode de recherche locale consiste à partir d'une solution initiale et à chercher une solution voisine qui maximise les objectifs. La méthode de recherche globale consiste à explorer tout l'espace de recherche pour trouver la meilleure solution possible.

Tout ceci pour dire que comme dit auparavant dans l'introduction de la programmation par contrainte. Il existera un "algorithme" par objectif qui privilégiera ce planning en fonction de la pondération de l'importance des objectifs donnés en variable.

2.4.6 Validation et évaluation

Une fois que le solveur de contraintes a trouvé une solution qui respecte toutes les contraintes et maximise les objectifs, la dernière étape consiste à valider et évaluer la solution. Cette étape peut inclure des tests manuels pour s'assurer que la solution est réalisable dans la pratique. Elle peut également inclure des tests automatisés pour évaluer les performances de la solution par rapport aux objectifs définis.

L'évaluation des performances peut être basée sur des métriques telles que le temps de travail ou d'étude, le temps de trajet, le nombre de conflits de calendrier, le temps de loisirs, etc. Ces métriques peuvent être pondérées pour refléter l'importance relative des objectifs définis. L'évaluation peut également inclure une analyse de sensibilité pour déterminer l'impact des différentes contraintes sur les performances de la solution.

L'utilisateur fera partie de ce processus de validation et choisira le planning qui correspond pour lui le plus à ses objectifs donnés auparavant.

Pour le moment la définition des contraintes ainsi que des objectifs ont été fait. La collecte des données est en cours avec la mise en relation du calendrier, qui a été implémenté ,et de l'algorithme principal. Une fois les données récupérées celle-ci vont être mise en forme dans un graphe orienté où les arêtes seront pondérées en fonction du temps libre entre les deux actions. Ainsi si plusieurs objectifs pourront être différents : maximiser le temps-libre et minimiser le nombre d'interruption,

prioriser ces temps libre lors des jours de repos, augmenter le nombre de pause afin que la productivité soit plus importante. Plusieurs options seront fournies en fonction de l'objectif mis en priorité.

La pose des bases était importante afin de ne pas devoir revenir en arrière si le modèle n'était pas correctement compatible avec l'algorithme. Néanmoins le code de l'algorithme sera la priorité de la deuxième soutenance afin de pouvoir ensuite mettre l'algorithme avec l'interface pour la troisième soutenance. Ainsi nous sommes loin d'être en avance sur cette partie voire même un léger retard sur les objectifs peut-être observé. Celui ci peut toutefois être largement rattrapé.

3 Bilan

Pour résumer, nous expliquons ce qu'est un algorithme de structure d'emploi du temps et les structures de données utilisées pour gérer efficacement les données à utiliser dans un emploi du temps. Les structures comprennent une structure semaine, une structure jour et une structure action. Le texte aborde également la gestion de la logique pour le calcul des jours et comment il faut gérer tous les cas un par un.

Nous parlons ensuite de l'interface graphique pour l'application et comment l'outil Glade a été utilisé pour la conception graphique de l'interface, et GtkBuilder pour la lecture des fichiers dynamiquement. Il explique comment l'interface a été mise en place avec différents boutons pour ajouter des événements au calendrier, voir le planning des semaines précédentes ou suivantes, rechercher la date d'un événement particulier, changer les paramètres de l'interface, quitter l'application, etc.

Enfin, nous abordons les prochains objectifs pour l'application, qui sont l'implémentation d'un système d'interaction entre l'interface utilisateur et le code pour permettre aux utilisateurs de saisir leurs besoins d'emplois du temps, la mise en place d'un système de sauvegarde de données pour permettre aux utilisateurs de conserver leurs emplois du temps précédents et de les modifier facilement, et la mise en place de mécanismes de suggestion d'horaires pour aider les utilisateurs à optimiser leur emploi du temps de manière plus efficace.

En conclusion, l'algorithme de structure d'emploi du temps est un outil important pour la gestion efficace des données d'un emploi du temps. L'utilisation de structures de données appropriées pour la semaine, le

jour et l'action est essentielle pour une gestion efficace des données. L'interface utilisateur est également importante pour permettre aux utilisateurs de saisir leurs besoins d'emplois du temps et de modifier facilement les données. Les prochains objectifs de l'application visent à améliorer l'interaction utilisateur/code et à proposer des fonctionnalités supplémentaires pour une gestion plus efficace des emplois du temps.