



ÉCOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE ET
D'ANALYSE DES SYSTÈMES

MÉMOIRE DE PROJET DE FIN DE 2ÈME ANNÉE

Génération d'emploi du temps par satisfaction des contraintes



Soutenu par :
Ougunir Safaa
Taddist Afaf

Sous la direction de :
M. Elhamlaoui Mahmoud

Remerciement

Avant d'entamer le rapport de notre projet, nous tenons à exprimer nos sincères remerciements à tous les professeurs et le corps administratif de l'École Nationale Supérieure de l'Informatique et l'Analyse des Systèmes pour les efforts qu'ils fournissent tout au long de l'année.

En particulier, nous voudrions remercier Monsieur ELHAMLAOUI Mahmoud pour son encadrement et sa disposition à nous orienter tout au long de la réalisation de notre projet malgré ses diverses occupations.

Nous remercions enfin tous les internautes éventuels qui nous ont aidé avec leurs diverses ressources informatiques que l'on détaillera dans la partie Bibliographie. Ces ressources nous ont été indispensables durant nos tout premiers pas avec les méthodes de résolution.

Nous présentons également nos remerciements à tous les membres du jury qui nous ont fait l'honneur d'accepter de juger notre travail.

Nous souhaitons que notre travail soit à la hauteur du niveau estimé.

Résumé

Le présent document constitue une synthèse de notre projet de fin de deuxième année. Ce projet a pour objectif de concevoir et réaliser une application web pour génération des emplois du temps par satisfaction des contraintes au sein l'École Nationale Supérieure de l'Informatique et l'Analyse des Systèmes.

Afin d'atteindre notre but, on a divisé le projet en trois grandes parties :

Chapitre I : Présentation générales du problème Dans cette partie on a essayé tout d'abord de présenter le problème et d'où vient sa complexité. On a également abordé les différentes approches de résolution adoptées pour ce problème.

Chapitre II : Modélisation et Algorithme Dans cette partie on a essayé formuler le problème tout en modélisant nos données et nos contraintes à l'aide des différentes classes et algorithmes.

Chapitre III : Réalisation Dans cette partie concrète non avons présenter l'environnement du travail matériel et logiciel .Par la suite , on a cité quelques bibliothèques CSP Python .Et enfin , on a présenté l'état de l'application.

A travers ce document, nous allons décrire plus en détail chaque partie de la réalisation de ce projet.

Sommaire

1	Configuration de l'environnement	2
1.1	Installation du framework Django en utilisant pip	2

Table des figures

1.1	Capture version python	2
1.4	Commande pour installer pip pour python3	3
1.5	Commande pour installer django 1.10	3
1.6	Commande pour verifier la version de django	3

Introduction

Les problèmes d'ordonnancement des personnels sont devenus de plus en plus diffusés dans notre vie réelle. Le problème d'emploi du temps est une instance des problèmes d'ordonnancement des tâches et en particulier des problèmes d'ordonnancement des personnels. Ce problème est omniprésent dans tous les aspects pratiques de la société moderne. Il joue un rôle très important dans plusieurs types d'organisation tels que les hôpitaux, les sociétés de transport et les universités. Nous nous intéressons dans ce projet, à la résolution du problème d'emploi du temps des cours universitaires de l'ENSIAS.

Le problème d'emploi du temps est bien connu comme un problème NP-complet. Autrement dit, aucun des algorithmes existants n'est capable de résoudre à lui seul toutes les instances du problème dans un temps polynômial.

D'autre part, le problème d'emploi du temps peut être assimilé à un problème d'ordonnancement cyclique. Un problème d'ordonnancement cyclique est un problème d'ordonnancement où un nombre fini de tâches génériques liées par des contraintes de précédence ou de ressources doivent être exécutées une infinité de fois.

Dans notre problématique, nous choisissons de traiter le problème d'emploi du temps consacré au cours, qui se définit comme un ensemble de cours universitaires qui se déroulent tout au long des périodes spécifiques pendant cinq ou six jours par semaine, dirigés par un nombre limité d'enseignants et de salles de classes nécessitant une meilleure gestion pour bien contenir le nombre important d'étudiants inscrits.

Ce problème se modélise généralement par un réseau de contraintes, encore appelé problème de satisfaction de contraintes (CSP pour Constraints Satisfaction Problem). La résolution d'un CSP consiste à trouver l'assignation consistante de valeurs à des variables prenant leurs valeurs dans des domaines discrets et finis. Le problème peut être résolu de multiples façons : approche multicritère, algorithmes génétiques, heuristique spécifique, etc.

Première partie

Présentation générales du problème

Chapitre 1

Problème d'emploi du temps des cours universitaires

Dans ce chapitre, nous commençons par la présentation du problème d'emploi du temps en générale. Puis, nous citons les principaux travaux de recherche qui ont été élaborés pour la résolution de ce problème.

1.1 Présentation du problème d'emploi du temps

Le problème d'emploi du temps consiste à l'affectation d'un ensemble de tâches et /ou activités à un ensemble de ressources en se basant sur des périodes de temps bien précises, soumettant à un ensemble de contraintes à respecter.

Une tâche est un travail élémentaire dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de chaque ressource. Une ressource est un moyen technique ou humain, dont la disponibilité limitée ou non est connue a priori.

Le problème d'emploi du temps consiste à définir un certain nombre d'affectations qui permettent d'assigner plusieurs ressources sur une période fixe de temps. Il s'adresse habituellement aux organisations où un ensemble de tâches doivent être accomplies par un ensemble d'employés ayant leurs propres qualifications, contraintes et préférences.

Les contraintes considérées peuvent différer d'un problème à un autre suivant la spécificité ainsi que les caractéristiques attendues de l'emploi du temps recherché. Ces contraintes sont souvent classées en deux catégories, la première regroupe les contraintes dures (un emploi du temps qui ne satisfait pas ce genre de contraintes est infaisable ou inacceptable), la seconde catégorie regroupe des contraintes appelées souvent contraintes molles, souples ou de préférence dont la satisfaction a différents degrés d'importance mais dont le non respect n'empêche pas une application plus au moins acceptable de l'emploi du temps trouvé. Typiquement ces contraintes de préférence sont utilisées pour exprimer ce que doit être un « bon » emploi du temps. Ces contraintes sont plus difficiles à formaliser que les contraintes dures et leur traitement est plus délicat.

En ce qui nous concerne, nous nous sommes concentrées plus précisément sur le problème d'emploi du temps universitaire.

1.2 Problème d'emploi du temps universitaire

Le problème d'emploi du temps universitaire est une instance des problèmes d'ordonnement cyclique les plus connus dans la littérature, il s'agit d'ordonner les tâches (qui possède un caractère cyclique) d'un ensemble d'enseignants en leur allouant un ensemble de salles et en leur fixant leurs dates de début et de fin.

De nos jours, la construction d'un calendrier qui satisfait tous les besoins d'un établissement universitaire est une tâche très importante, mais elle est extrêmement difficile. Avec le progrès réalisé dans les technologies matérielles et logicielles, la communauté scientifique continue à travailler sur ce problème, afin d'élaborer des procédures formelles et automatisées pour élaborer des calendriers efficaces et souhaitables.

1.3 Problème d'emploi du temps des cours universitaires

Dans notre problématique, nous choisissons de traiter le problème d'emploi du temps consacrés au cours.

Et afin d'avoir une meilleure solution à ce problème, il faut prendre en compte l'ensemble des contraintes du problème qui doivent être satisfaites.

Les contraintes Hard :

- Deux lectures ne peuvent pas être programmées dans la même salle et dans la même période de temps.
- Les lectures à donner par un même enseignant ne peuvent pas être programmées dans la même période de temps.
- Une salle ne peut être affectée qu'à une seule lecture dans une même période de temps.
- Deux séances de lectures en groupe ne peuvent pas se dérouler dans la même période pour le même groupe.
- Le nombre d'étudiants doit être inférieur ou égale à la capacité de la salle à affecter.

Les contraintes Soft :

- Il faut que l'affectation des périodes de temps permette de satisfaire au mieux certaines préférences (Pas de lectures Mercredi apres midi).

1.4 La méthode de résolution

De nombreuses approches ont été proposées dans la littérature pour résoudre ce problème. Ces méthodes sont regroupées en deux classes : une première classe contenant les méthodes centralisées et une deuxième classe contenant les méthodes décentralisées. Nous nous intéressons aux approches centralisées et plus précisément l'approche basée sur les problèmes de satisfaction de contraintes .

1.4.1 Approches centralisées

Plusieurs approches centralisées ont été proposées pour la résolution de ce problème. Les premières tentatives de résolution étaient les méthodes basées sur la théorie des graphes, la programmation linéaire et les techniques de satisfaction des contraintes. Mais, ces méthodes n'ont pas donné de solutions traitant toutes les instances et les contraintes de ce problème.

C'est pour cela, ils ont cédé le pas à d'autres types de méthodes adaptées à ce type de problème, à savoir les méta-heuristiques. Cette famille de recherche approchée est dotée de mécanismes généraux lui permettant une bonne investigation de l'espace de recherche, mais généralement, elle est non-déterministe et ne donne aucune garantie d'optimalité. Ce qui a permis l'apparition d'autres types de méthodes, à savoir les techniques d'hybridations des premières méthodes avec les méta-heuristiques.

Approches basées sur la programmation linéaire et la théorie des graphes

La programmation linéaire est un outil très puissant de la recherche opérationnelle. C'est un outil générique qui peut résoudre un grand nombre de problèmes. Une fois le problème est modélisé sous la forme d'équations linéaires, des méthodes assurent sa résolution de manière exacte.

On distingue dans la programmation linéaire, la Programmation Linéaire en Nombres Réels (P.L.N.R), pour laquelle les variables des équations sont dans \mathbb{R}^+ et la Programmation Linéaire en Nombres Entiers (P.L.N.E), pour laquelle les variables sont dans \mathbb{N} . Bien entendu, il est possible d'avoir les deux en même temps

La théorie des graphes est un domaine très riche contenant des modèles et des applications permettant de résoudre plusieurs types de problèmes difficiles. En effet, le Graph Coloring Problem est considéré comme un de ces fameuses applications les plus connues dans la littérature.

Approches basées sur les problèmes de satisfaction de contraintes

Les problèmes de satisfaction de contraintes ou C.S.P (Constraint Satisfaction Problem) sont des problèmes mathématiques où l'on cherche des états ou des objets satisfaisant un certain nombre de contraintes ou de critères.

Définition : Un problème de satisfaction de contraintes est un quadruplet (X, D, C, R) défini par :

- Un ensemble de n variables : $X = (X_1, X_2, \dots, X_n)$
- Un domaine discret et fini de valeurs associé à chaque variable X_i , noté D_i : $D = (D_1, D_2, \dots, D_n)$
- Un ensemble de m contraintes : $C = (C_1, C_2, \dots, C_m)$ où une contrainte C_i est définie par un sous-ensemble de variables : $C_i = (C_{i1}, C_{i2}, \dots, C_{ik})$
- Une relation R_i est associée à chaque contrainte C_i et définissant l'ensemble des combinaisons de valeurs permises par C_i , R_i est définie par un sous-ensemble du produit cartésien $D_{i1}, D_{i2}, \dots, D_{ik}$.

L'ensemble des relations est noté R : $R = (R_1, R_2, \dots, R_m)$.

Une relation peut être exprimée de deux façons :

- En extension : par l'utilisation d'un ensemble de tuples permis.
- En compréhension : par l'utilisation de prédicats ou de fonctions caractéristiques.

\implies Une instanciation d'un sous-ensemble $Y \subseteq X$ est dite consistante (ou cohérente) si et seulement si elle satisfait toutes les contraintes qui impliquent toutes les variables de Y .

\implies Une solution de $P = (X, D, C, R)$ est une instanciation consistante de l'ensemble de toutes les variables de X sur D .

\implies Un C.S.P est dit cohérent si et seulement s'il possède au moins une solution.

La formulation du modèle C.S.P pour la résolution du problème d'emploi du temps universitaire a été basée sur trois formules :

Timetabling = $(\exists f \in \text{Courses} \rightarrow \text{Times}) \text{ Suitable}(f)$:

\implies Timetabling c'est la relation d'affectation d'un cours C à une période T .

$\text{Suitable}(f) = (\forall c, c' \in \text{Courses}) (c \neq c') \text{ Conflict}(c, c', f)$:

\implies Suitable permet de vérifier la relation de conflit entre deux types de cours c et c' .

$\text{Conflict}(c, c', f) = (\exists X \in \text{Students}) c \in s(X) \wedge c' \in s(X) \wedge \prec c, b \succ \in f \wedge \prec c', b' \succ \in f \wedge b \neq b' = \emptyset$

\implies Conflit est une fonction qui permet de tester l'existence d'un conflit entre les deux cours c et c' pour un seul étudiant X .

Approches basées sur les méta-heuristiques

Les méta-heuristiques forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficiles souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle pour lesquels on ne connaît pas de méthodes exactes plus efficaces.

Une méta-heuristique est formellement défini comme un processus de génération itératif, qui permet de guider une heuristique en combinant intelligemment différents types de concepts afin d'avoir une meilleure exploration et exploitation de l'espace de recherche

Les méta-heuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique (cas du recuit simulé), en biologie de l'évolution (cas des algorithmes génétiques) ou encore en éthologie (cas des algorithmes de colonies de fourmis ou de l'optimisation par essaims particuliers).

En effet, cette famille d'algorithmes a été utilisée dans plusieurs travaux de recherche permettant de trouver des méthodes génériques pouvant optimiser une large gamme de problèmes difficiles, tel que celui de l'emploi du temps universitaire.

Approches basées sur l'hybridation des méthodes

1.4.2 Approches décentralisées

Outre la première génération de chercheurs qui ont beaucoup travaillé sur les approches centralisées, nous avons eu l'apparition d'une autre comité de chercheurs, qui cherchent à diminuer tout type d'exécution séquentielle, afin d'atteindre de nouvelles approches parallèles et fortement distribuées.

En effet, nous avons distingué l'existence de quelques approches distribuées pour la résolution de notre problème d'emploi du temps universitaire.

Approches basées sur la satisfaction distribuée des contraintes appliquées dans les systèmes multi-agents

Approches basées sur les systèmes multi-agents

conclusion

Au niveau de ce premier chapitre, nous avons défini notre problématique, puis nous avons cité nos principaux objectifs à atteindre. Ensuite, nous avons détaillé les travaux de recherche les plus connus dans le domaine d’ordonnancement des tâches dans les établissements universitaires, ainsi que les méthodes les plus utilisées pour le problème d’emploi du temps universitaire.

Deuxième partie

Modélisation et Algorithme

Chapitre 2

Modélisation du problème

Après avoir donner une présentation générale du problème de génération d'emploi du temps dans le premier chapitre , nous allons essayer de modéliser ce problème afin pouvoir mieux le comprendre.

2.1 Formulation du problème

Le problème de l'emploi du temps consiste à affecter une séance pendant une période de temps et à une salle sous un ensemble de contraintes. La première chose est de savoir « c'est quoi une séance » ?

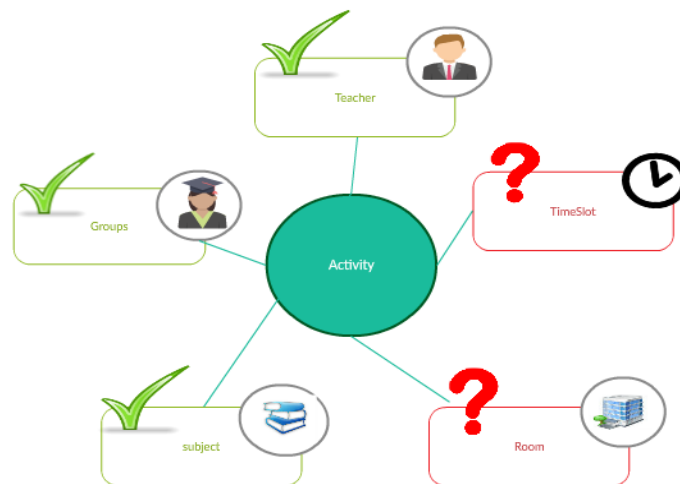


FIGURE 2.1 – Definition of Activity

Selon la figure??, une séance « Activity » est une combinaison d'un professeur « teacher », d'une classe « Group », d'une matière « subject », d'une salle « Room » et d'une période de temps « Time Slot ».

Tous les cinq composants forment ensemble une séance et seront construits par le programme d'horaire. Bien sûr, les programmes ne construisent pas tout ; certaines combinaisons ont déjà été données dans l'entrée. L'administration décide quel professeur enseigne quelle matière et soumis à quelle classe. La seule chose que les programmes ont à faire est d'attribuer un créneau et une salle à chaque leçon.

En résumé un programme d'horaire doit retourner la combinaison suivante :
Activity (teacher, subject, class, room ?, timeslot ?)

2.2 Modélisation des données

Chaque type de données est modélisé par une classe. La figure ?? présente les divers classes obtenues :

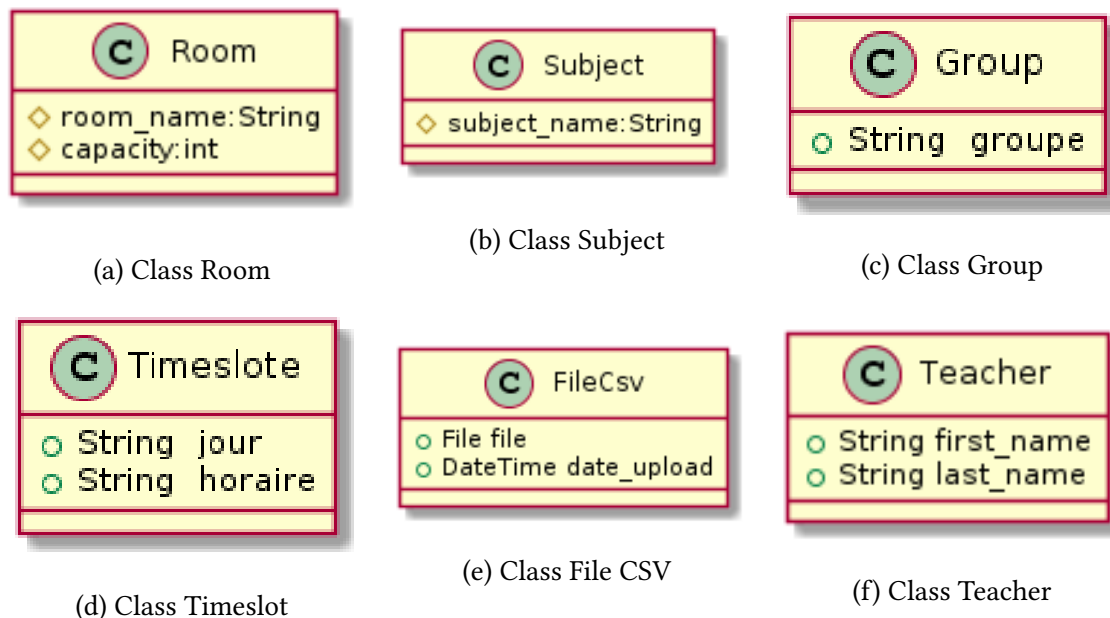


FIGURE 2.2 – Class of data

2.3 Modélisation des contraintes

On a deux catégories de contraintes :

- Les contraintes Hard seront modélisées par des méthodes ,qui seront intégrées dans la classe Activity,figure ??.
- La contrainte soft (pas de seances mercredi après midi) : notre ensemble de timeslot va devenir :
 $TS=\{\text{Lundi , Mardi , Mercredi(matin) , jeudi ,vendredi }\}$

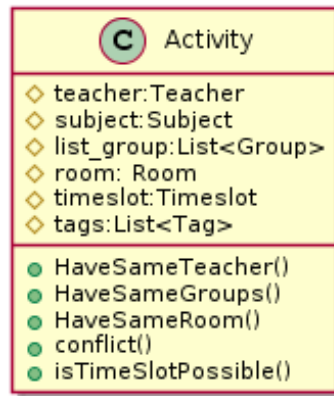


FIGURE 2.3 – Class Activity

2.4 Diagrammes de cas d'utilisation

Use Case Diagram

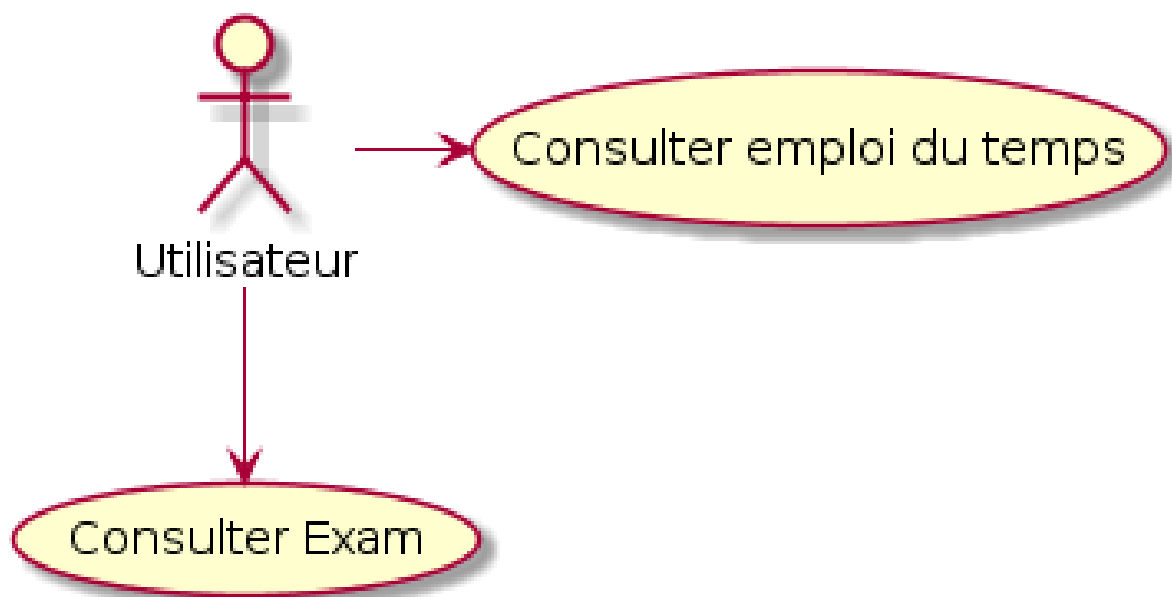


FIGURE 2.4 – Diagramme de cas d'utilisation des utilisateurs

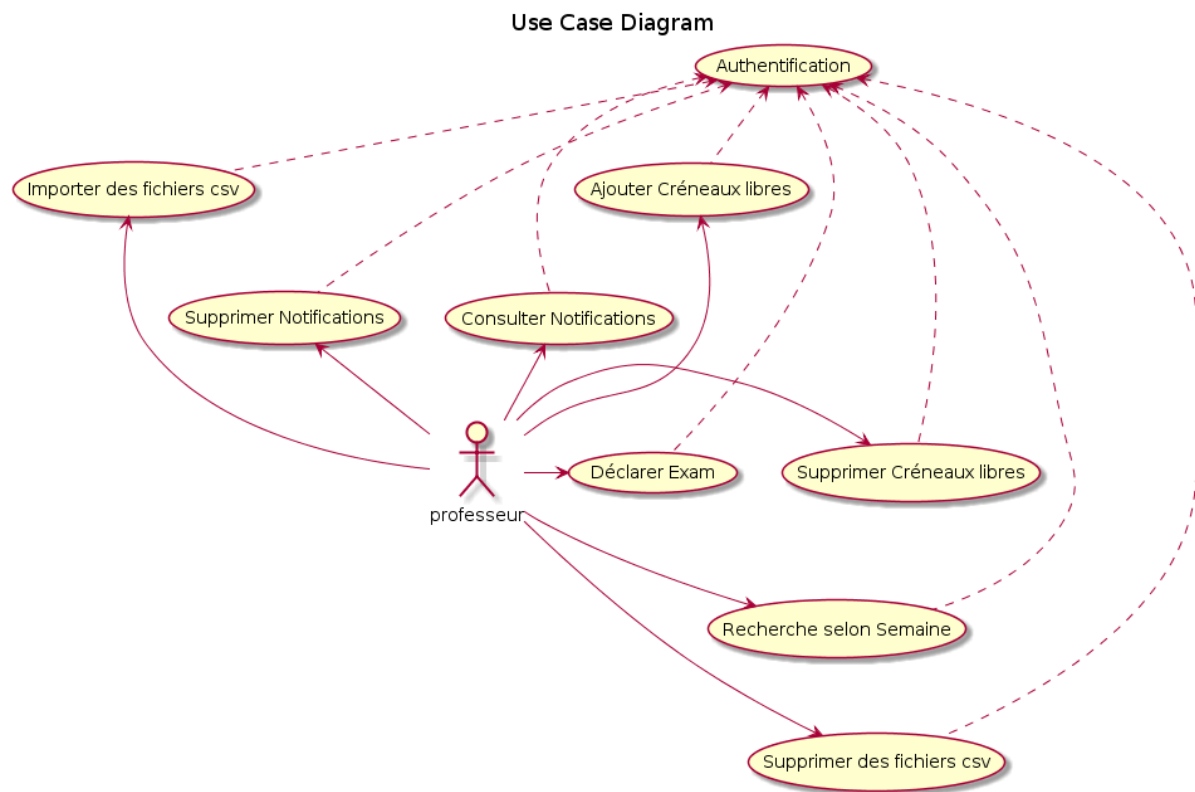


FIGURE 2.5 – Diagramme de cas d'utilisation des professeurs

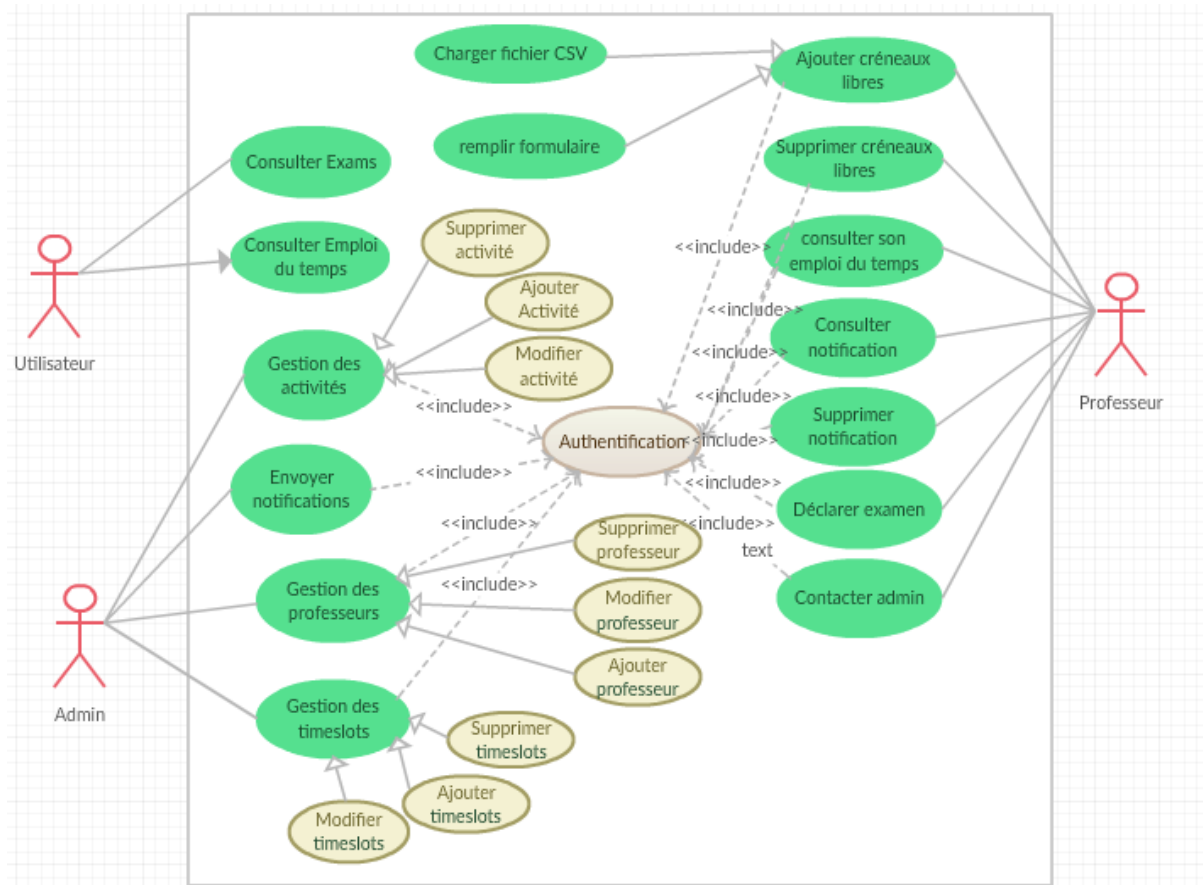


FIGURE 2.6 – Diagramme de cas d'utilisation

Conclusion

Dans ce deuxi me chapitre, nous avons essay  de mod liser notre probl me   travers la sp cification des diff rentes Classes pour mod liser les donn es ainsi que les divers m thodes pour repr senter les contraintes soft et hard.

Chapitre 3

Algorithme

Ce chapitre est consacré aux algorithmes d'affectation des timeslots et des salles pour les divers activités("Cours","TD","TP").

3.1 Présentation générale de l'algorithme

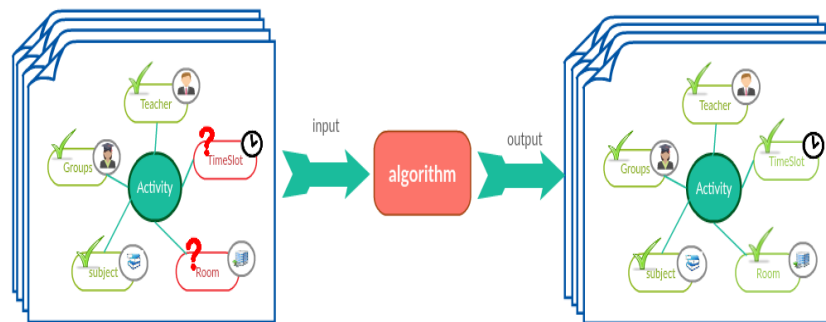


FIGURE 3.1 – Functioning of Algorithm

Comme la figure ?? le montre nous allons avoir comme entrées une listes des activités $A=[A1,A2,\dots\dots\dots]$ dont les professeurs ,les groupes et les matière sont connus mais les timeslots et les salles sont inconnus,ainsi qu'une liste des salles disponibles et une liste de longueur 18 des timeslots $T=[T1,T2,\dots]$. Les timeslots représentent les cases de l'emploi du temps de la façon suivante :

	8h → 10h	8h → 10h	8h → 10h	8h → 10h
Lundi	T[1]	T[2]	T[3]	T[4]
Mardi	T[5]	T[6]	T[7]	T[8]
Mercredi	T[9]	T[10]		
Jeudi	T[11]	T[12]	T[13]	T[14]
Vendredi	T[15]	T[16]	T[17]	T[18]

FIGURE 3.2 – Timeslot Representation

D'après la figure ??,T[1] est une liste des activités qui a pour date lundi de 8 à 10h.

3.2 Les contraintes

- Deux activités ne peuvent pas se chevaucher pour 3 raisons :si ils ont le même professeur ou bien un groupe en commun ou la même salle. Notre algorithme va se baser sur

ces trois contraintes.

- L'activité contient également une liste de tags qui représente une sorte de modélisation de contraintes. en effet ,ils permettent de donné des conditions sur les salles à affecter ou bien les horaires prenant l'exemple d'un tag dont le nom est « G1 » il indiquera la liste des salle possible pour une séance par exemple l1,12,A1,A2 .. on bien un tag « TP » qui oblige de choisir une salle de TP comme l1,l2.. dans ce cas on le choix d'un amphi ou salle simple sera éliminé.

Pour notre premier algorithme on va pas utiliser les tags, vu qu'on a choisi que les contraintes basiques.

3.3 L'algorithme 1 : affectation des activités aux timeslots

POUR chaque activité Ai FAIRE

 POUR chaque time Slot Tt Faire

 SI Tt vérifie les contraintes de Ai ALORS

 POUR chaque activité Aj déjà affectée au timeslot Tt FAIRE

 SI At et Aj peuvent se chevaucher alors

 Ajouter Tt dans la liste des activités affectées au timeslot Tt

 Attribuer le timeslot à l'activité

 SINON

 Sortir pour passer vérifier le prochain timeslot

 FIN SI

 FIN POUR

 FIN SI

 FIN POUR

FIN POUR

3.4 L'algorithme 2 : affectation des salles

- On note ici que l'entrée de cette algorithme sera la liste des salles disponible ainsi que la sortie du premier algorithme c'est-à-dire une liste des timeslots ou chaque timeslots a une liste des activités affectées .
- Chaque timeslot va également contenir une liste des salles déjà occupé qu'on va remplir au fur et à mesure.

L'algorithme est le suivant :

POUR chaque timeSlot Tt Faire

 POUR chaque Ai activité affectée à ce timeslot

 POUR chaque Sj dans la liste des salles

 Si la salle Sj n'est pas dans la liste des salles occupées de Tt

 Affecter la salle Sj à l'activité

 Ajouter la salle sj à la liste des salles occupees de Tt

 sinon

 Sortir pour passer à l'autre salle

 FIN SI

 Fin POUR

FIN POUR

FIN POUR

Conclusion

Transformer ces algorithmes en Python va augmenter la complexité du problème, on a besoin d'une bibliothèque CSP qui va nous aider à simplifier le maximum la génération d'une solution

Troisième partie

Réalisation

Chapitre 4

Environnement du travail

Pour pouvoir mener à bien un projet informatique, il est nécessaire de choisir des technologies permettant de simplifier sa réalisation. Pour cela, après avoir compléter la modélisation du problème et la proposition d'algorithme dans la partie précédente, nous allons aborder la partie implémentation dans ce qui suit. Je commence par présenter l'environnement matériel et logiciel et ensuite.

4.1 Choix des outils

4.1.1 Environnement matériel

Pour la réalisation de l'application, on a utilisé un 1 pc portable pour le développement ayant les caractéristiques suivantes

- Intel core I7 2.5 GHz 8 Go de mémoire vive.
- Ubuntu 16.04

4.1.2 Environnement logiciel

Django



FIGURE 4.1 – Logo Django

Django est un framework Web de haut niveau Python qui favorise un développement rapide et un design propre et pragmatique. Construit par des développeurs expérimentés, il gère une grande partie du problème de développement web, afin que vous puissiez vous concentrer sur l'écriture de votre application sans avoir besoin de réinventer la roue. C'est gratuit et open source.

Principales fonctionnalités :

- Mapping relationnel-objet
- Interface d'administration automatisée
- Design élégant d'URL
- Système de gabarit
- Système de cache
- Internationalisation

Il existe de nombreux framework web, dans différents langages de programmation. Pourquoi utiliser spécifiquement Django et pas un autre ?

Nous avons choisi Django pour plusieurs raisons.

- La simplicité d'apprentissage.
- La qualité des applications réalisées.
- La rapidité de développement.
- La sécurité du site Internet final.
- La facilité de maintenance des applications sur la durée.

Bootstrap

Bootstrap est une collection d'outils utile à la création de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

AlertifyJS



FIGURE 4.2 – Logo AlertifyJS

AlertifyJS est un framework javascript pour développer de belles boîtes de dialogue et notifications de navigateurs.

Chapitre 5

Implémentation pour les CSP python

5.1 Introduction

La programmation de contraintes est souvent réalisée dans une programmation impérative via une bibliothèque distincte. Par la suite, on va étudier certaines bibliothèques populaires[8] pour la programmation des contraintes.

5.2 python-constraint

Le module de contraintes Python[5] offre des solutions de résolution des problèmes de résolution de contraintes (CSP) sur des domaines finis en Python simple et pur. CSP est une classe de problèmes qui peuvent être représentés en termes de variables (a, b, \dots), de domaines (a dans $[1, 2, 3], \dots$) et de contraintes ($a < b, \dots$)

python-constraint peut être installé en utilisant PIP : `$ sudo pip install python-constraint`

5.3 Numberjack

Numberjack[2] est un paquetage de modélisation écrit en Python pour la programmation de contraintes. Python bénéficie d'une grande communauté de programmation active, Num-

berjack est donc un outil parfait pour intégrer la technologie CP dans de plus grandes applications. Il est conçu pour supporter un certain nombre de solveurs C / C ++ sous-jacents de manière transparente et efficace. Il existe un certain nombre de back-ends disponibles : trois solveurs de programmation entiers mixtes (Gurobi, CPLEX et SCIP), des solveurs de satisfaction (MiniSat, Walksat et bien d'autres), un solveur de programmation de contraintes (Mistral) et un solveur pondéré de satisfaction de contraintes (Toulbar2).

- Numberjack offre une plate-forme de modélisation de haut niveau.
- Numberjack bénéficie directement des fonctionnalités et des modules de Python.
- Numberjack utilise des solveurs C / C ++ sous-jacents efficaces.

5.4 Cassowary

Une mise en œuvre pure de Python de l'algorithme de résolution des contraintes de Cassowary[9]. Cassowary est l'algorithme qui constitue le noyau du mécanisme de présentation visuelle OS X et iOS.

Pour installer Cassowary, exécutez : `$ sudo pip install cassowary`

5.5 Artelys Kalis

Artelys Kalis[10] is an innovative constraints programming component able to solve quickly and efficiently a wide range of combinatorial problems such as scheduling, timetabling, resource allocation, equipment or network configuration, etc.

5.6 Google CP Solver[1]

Chapitre 6

Présentation de l'application

Le Responsive Web design est une approche de conception Web qui vise à l'élaboration de sites offrant une expérience de lecture et de navigation optimales pour l'utilisateur quelle que soit sa gamme d'appareil (téléphones mobiles, tablettes, liseuses, moniteurs d'ordinateur de bureau). Par la suite, nous allons présenter notre état de réalisation.

6.1 Présentation de différentes interfaces format pc

6.1.1 Interface accueil

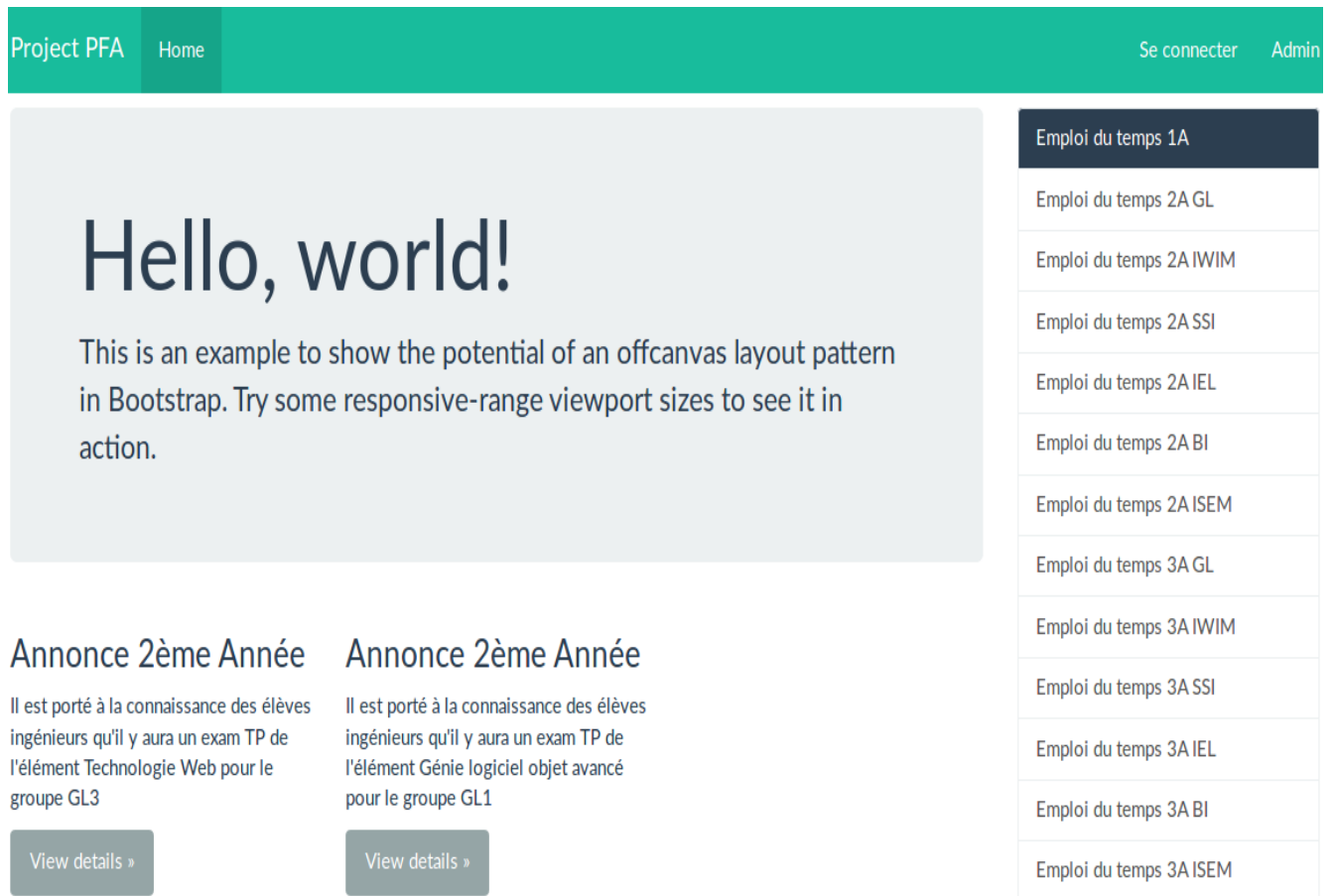


FIGURE 6.1 – Capture d'accueil

La figure ?? illustre la première interface de l'application. Elle permet de visualiser les exams prévus et les emplois du temps selon la filière.

6.1.2 Interface de connexion

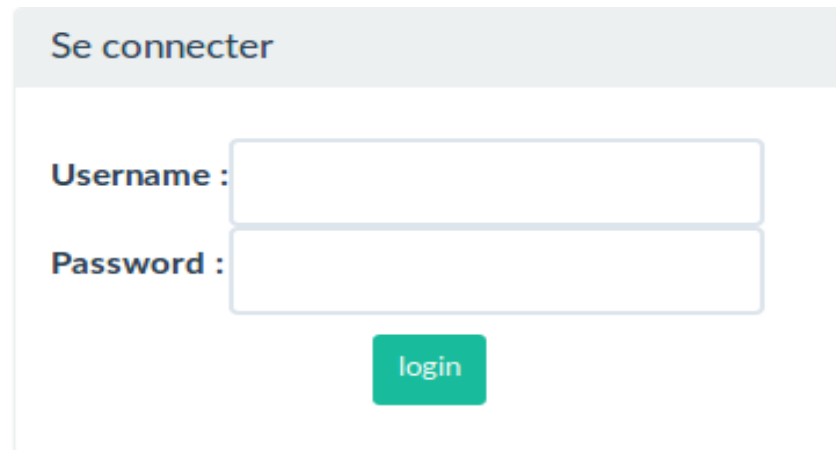
A screenshot of a login interface. At the top, there is a light gray header bar with the text "Se connecter". Below this, there are two input fields: "Username :" and "Password :". The "Password :" field has a small eye icon on the right side. Below the input fields is a green button with the text "login".

FIGURE 6.2 – Capture du login

Un professeur peut se connecter via la interface definit dans la figure ?? .En effet,Administrateur se charger de la création des comptes.

6.1.3 Interfaces d'ajout de créneaux libres

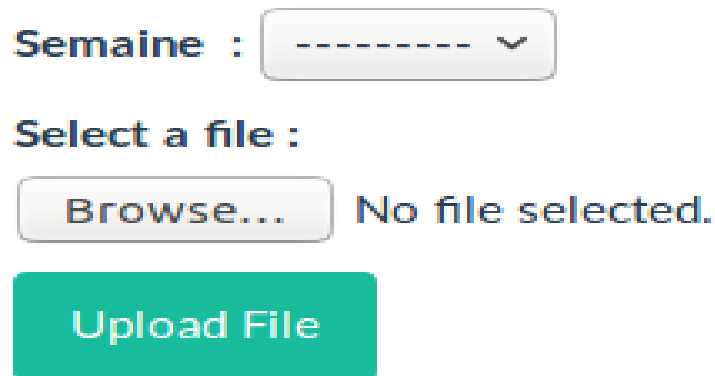
A screenshot of a file upload interface. It features a "Semaine :" label followed by a dropdown menu showing "-----" and a downward arrow. Below this is a "Select a file :" label. Underneath, there is a "Browse..." button and the text "No file selected.". At the bottom, there is a large green button with the text "Upload File".

FIGURE 6.3 – Capture d'importation du fichier csv

Créneau libre :

Semaine :

Valider

FIGURE 6.4 – Capture d'ajout manuel

Ajouter votre créneaux libres en important un fichier CSV ou
bien manuellement

Télécharger exemplaire de CSV [Semaine_0.csv](#)

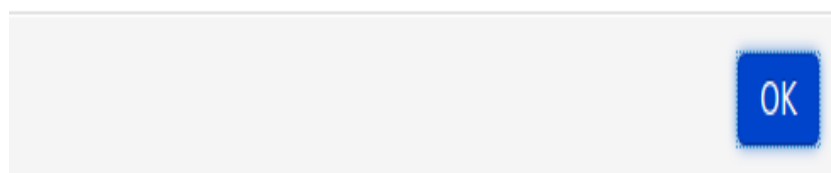


FIGURE 6.5 – Capture d'exemplaire du fichier csv à télécharger

Mes fichiers

Semaine	Date	Télécharger	Modifier	Supprimer
S15	23 juin 2017 01:11	Semaine_15.csv		
S1	10 mai 2017 15:11	Semaine_1.csv		

FIGURE 6.6 – Capture des fichier csv

Un utilisateur a deux possibilités pour ajouter ses créneaux libres soit en important un fichier CSV, figure ?? ou bien manuellement en utilisant le formulaire de la figure ?? . Un fois accéder à l'interface d'ajout de créneaux libres , une fenêtre s'ouvre permettant de télécharger un exemplaire de fichier csv , voir figure ?? . Le professeur peut également visualiser tous les fichiers csv avec possibilité de suppression, d'après la figure ?? .

6.1.4 Interface de déclaration d'examen

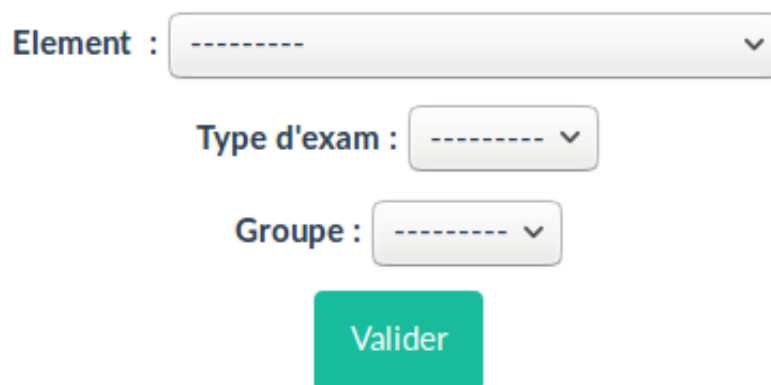
The image shows a web form for declaring an exam. It consists of three stacked dropdown menus. The first is labeled 'Element :', the second 'Type d'exam :', and the third 'Groupe :'. Each dropdown menu has a dashed line and a downward arrow icon. Below these fields is a green rectangular button with the white text 'Valider'.

FIGURE 6.7 – Capture de déclaration d'examen

Ce formulaire , figure ?? , permet à un professeur de déclarer un exam pour un groupe donnée tout en précisant le type d'exam .

6.1.5 Interface de notification

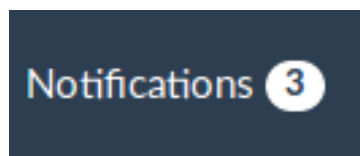


FIGURE 6.8 – Capture de barre de notification

Veuillez Ajouter plus de créneaux libres avant le vendredi 24/06/2017

L'emploi du temps est généré . Bon journée

Veuillez remplir au minimum 20 heures !

Voir Tout

FIGURE 6.9 – Capture des notifications

23 juin 2017 02:04	
Veuillez Ajouter plus de créneaux libres avant le vendredi 24/06/2017	
23 juin 2017 02:07	
L'emploi du temps est généré . Bon journée	
23 juin 2017 02:08	
Veuillez remplir au minimum 20 heures !	

FIGURE 6.10 – Capture de la liste des notifications

Un professeur peut recevoir des notifications de la part de l'administrateur. La figure ?? et ?? donne plus de détails sur les notifications, le nombre total des notifications et leurs contenus. Or la figure ?? montre la possibilité de suppression des notifications .

6.1.6 Interface Contacter Admin



The screenshot shows a web form for contacting an administrator. It features a label 'Subject :' followed by a text input field. Below this is a large, empty rectangular text area. At the bottom center of the form is a green button with the text 'Envoyer'.

FIGURE 6.11 – Capture du formulaire

Le professeur peut contacter l'administrateur selon la figure ??.

6.1.7 Interface de recherche



The screenshot shows a search bar with a light blue border. Inside the bar, the text 'Recherche... Ex:S1' is displayed in a light blue, monospaced font.

FIGURE 6.12 – Capture de la barre de recherche

Semaine S15

Jour	Créneau	Modifier	Supprimer	Jour/Horaire	8h-10h	10h-12h	14h-16h	16h-18h
Lundi	8h-10h			Lundi				
Lundi	10h-12h			Mardi				
Lundi	14h-16h			Mercredi				
Lundi	16h-18h			Jeudi				
Mardi	8h-10h			Vendredi				
Mardi	10h-12h							
Jeudi	8h-10h							
Jeudi	10h-12h							
Jeudi	14h-16h							
Jeudi	16h-18h							
Vendredi	8h-10h							
Vendredi	10h-12h							
Vendredi	14h-16h							
Vendredi	16h-18h							

FIGURE 6.13 – Capture du résultat de la recherche

Un professeur peut effectuer une recherche selon le nom de la semaine par exemple S1,S2.... à l'aide de la barre de recherche ,figure ?? .Ce dernier reçoit comme réponse ces créneaux libres sous forme d'une liste et un tableau,voir figure ??.

6.1.8 Espace Administrateur



Administration de Django

Nom d'utilisateur :
admin

Mot de passe :
.....

Connexion

FIGURE 6.14 – Capture du login Admin

Administration du site

AUTHENTIFICATION ET AUTORISATION		
Groupes	+ Ajouter	Modifier
Utilisateurs	+ Ajouter	Modifier
LOG		
Contacts	+ Ajouter	Modifier
Exams	+ Ajouter	Modifier
File csvs	+ Ajouter	Modifier
Filieres	+ Ajouter	Modifier
Free times	+ Ajouter	Modifier
Groups	+ Ajouter	Modifier
Notifications	+ Ajouter	Modifier
Rooms	+ Ajouter	Modifier
Semaines	+ Ajouter	Modifier
Subjects	+ Ajouter	Modifier
Tags	+ Ajouter	Modifier
Timeslotes	+ Ajouter	Modifier
Type elements	+ Ajouter	Modifier
Type exams	+ Ajouter	Modifier
Years	+ Ajouter	Modifier

FIGURE 6.15 – Capture de la page d’accueil Admin



FIGURE 6.16 – Capture de la barre Admin

Conclusion

La perfection ne fait probablement pas partie de ce monde et même si elle y était, nos ordinateurs (ainsi que nous mêmes) seraient encore très loin de l'atteindre. Il n'existe donc pas de technique exacte ou d'algorithme universel : à titre d'exemple, seul le filtrage par consistances n'est pas suffisant pour aboutir à une solution quasi optimale pendant un temps limité, ou encore, la simple application de quelque heuristique (comme le best first ou le first fail) n'est pas la réponse définitive. Une "règle d'informaticien" qui est toujours valide est "chercher si une solution existe déjà" avant de se pencher sur le problème. L'application du meilleur algorithme peut être parfois très coûteuse si l'on a "gaspillé" énormément de temps pour le trouver,

Bibliographie

- [1] <https://github.com/google/or-tools>, 2004.
- [2] <http://numberjack.ucc.ie/>, 2007.
- [3] <https://github.com/futurecore/python-csp>, 2009.
- [4] <https://github.com/futurecore/python-csp>, 2009.
- [5] <https://labix.org/python-constraint/>, 2011.
- [6] <https://web.stanford.edu/class/cs227/Lectures/lec14.pdf>, 2011.
- [7] Constraint satisfaction problems. <https://courses.csail.mit.edu/6.034s/handouts/spring12/csp.pdf>, 2012.
- [8] <http://3s-cms.enstb.org/uvf2b206/?p=260>, 2012-2016.
- [9] <https://cassowary.readthedocs.io/en/latest/>, 2013.
- [10] <https://www.artelys.com/en/optimization-tools/kalis>, 2016.
- [11] <https://www.cs.cmu.edu/afs/cs/academic/class/15381-s07/www/slides/020107CSP.pdf>, 2017.
- [12] Fahiem Bacchus. Constraint satisfaction problems. <http://www.cs.toronto.edu/~fbacchus/Presentations/CSP-BasicIntro.pdf>.
- [13] SALMAN Nader FERRO Luca, KHIN Samith. *Résolution pratique de problèmes NP-complets*. juin 2005.
- [14] Olfa Belkahla Driss Houssem Eddine Nouri. *Résolution multi-agents du problème d'emploi du temps universitaire*. Éditions Universitaires Européennes, Janvier 2015.
- [15] Michael Rovatsos. *Smart Searching Using Constraints*. février 2016.

- [16] Barbara M. Smith, Sally C. Brailsford, Chris N. Potts. *Constraint satisfaction problems : Algorithms and applications*. European journal of operational research, Octobre 1998.
- [17] OLIVER SCHULTE. Constraint satisfaction problems. <http://www.cs.sfu.ca/CourseCentral/310/oschulte/mychapter6.pdf>, 2011.
- [18] Michael Sioutis. Constraint satisfaction problems in python. <http://www.cril.univ-artois.fr/~sioutis/files/CSPsPy.pdf>, 2011.