

Rapport Final Projet Planning



BELHAJ Wael
DINH Khang Pierre
FEBRISSY Lilian
HANG Victor
MIRZICA-VIGE Lucas
ZERROUK Samuel

CY Tech - Cycle ingénieur
Filière Mathématiques appliquées option Maths-Finance
Professeur Référent : Zaouche DJAOUIDA
Année universitaire : 2023-2024

Table des matières

1	Introduction générale	3
2	État de l'art	4
2.1	Introduction	4
2.2	Coloration de Graphes	4
2.3	Méthode gloutonne	5
2.3.1	Algorithme de recherche A*	5
2.4	Méthode de recherche locale	5
2.4.1	Méthode du recuit simulé	5
2.4.2	Méthode de recherche tabou	6
2.5	Méthode évolutive	7
2.5.1	Algorithme génétique	7
2.5.2	Algorithme de colonies de fourmis	7
2.5.3	Méthode de décomposition	8
3	Modélisation mathématique	9
3.1	Problème de coloration des graphes	9
3.2	Problème de minimisation	10
3.2.1	Contrainte d'adjacence (Contrainte dure)	10
3.2.2	Contrainte d'équilibre (Contrainte souple)	10
3.2.3	Contrainte de session (Contrainte dure)	10
3.2.4	Contrainte de Durée des Examens (Contrainte dure)	10
3.3	Algorithme Génétique pour la Création du Planning	10
4	Construction des bases de données	12
4.1	Administrateur.csv	12
4.1.1	Structure	12
4.1.2	Construction détaillée	12
4.2	emploi_du_temps.csv	13
4.2.1	Structure	13
4.2.2	Construction détaillée	13
4.3	Etudiant.csv	13
4.3.1	Structure	13
4.3.2	Construction détaillée	14
4.4	listeSalle.csv	14
4.4.1	Structure	14
4.4.2	Construction détaillée	14
4.5	Matieres.csv	15

4.5.1	Structure	15
4.5.2	Construction détaillée	15
4.6	Salle_Place.csv	15
4.6.1	Structure	15
4.6.2	Construction détaillée	16
5	Fonctionnement de l'application	17
5.1	Introduction	17
5.2	Connexion Sécurisée	17
5.3	Visualisation des Emplois du Temps	18
5.4	Recherche par Nom	19
6	Déroulement du Projet	21
6.1	ZERROUK Samuel et MIRZICA-VIGE Lucas	21
6.2	HANG Victor	21
6.3	BELHAJ Wael et DINH Khang Pierre	22
6.4	FEBRISY Lilian	22
7	Conclusion générale	24
8	Annexes	26
8.1	Diagramme UML	26
8.2	Bibliographie	27

Chapitre 1

Introduction générale

La planification des examens universitaires est un défi de taille qui croît en complexité avec l'augmentation du nombre d'étudiants, de cours et de contraintes logistiques. La tâche consiste à attribuer des horaires et des salles d'examen de manière à éviter les conflits d'horaires pour les étudiants, à maximiser l'utilisation des ressources disponibles et à respecter diverses contraintes académiques et administratives.

L'objectif principal de ce projet est de développer une application de gestion des emplois du temps qui automatisera ce processus. Cette application vise à faciliter la consultation et la gestion des horaires pour les étudiants et les administrateurs de l'université. Elle doit être capable de gérer des données complexes provenant de diverses sources, de les intégrer de manière cohérente et de produire des plannings optimisés.

Pour atteindre cet objectif, nous avons choisi d'utiliser l'algorithme génétique, une méthode heuristique inspirée du processus de sélection naturelle. Cet algorithme est particulièrement adapté aux problèmes d'optimisation complexe comme la planification des emplois du temps, car il peut explorer un large espace de solutions et s'adapter à des contraintes variées.

Nous voulons modéliser un planning d'examen d'une université avec plusieurs filières. Chaque filière a un nombre fixe d'étudiants et des matières spécifiques à valider. Deux matières d'une même filière ne peuvent pas être effectuées en même temps. Modélisons notre problème par un graphe non orienté $G = (V, E)$:

- Chaque nœud $v \in V$ correspond à un examen.
- Chaque arête $(u, v) \in E$ indique que les examens u et v ne peuvent pas être planifiés en même temps.

Chapitre 2

État de l’art

2.1 Introduction

Dans le cadre de notre projet, nous avons été amenés à nous documenter sur les algorithmes existants qui pourraient nous être utiles. À travers cette partie, nous réalisons la synthèse de tous les algorithmes que nous avons étudiés.

2.2 Coloration de Graphes

La coloration de graphes est un problème issu de la théorie des graphes consistant à attribuer des couleurs à chaque sommet d’un graphe de telle sorte que deux sommets adjacents n’aient pas la même couleur. L’objectif est de minimiser le nombre total de couleurs utilisées. La coloration de graphes est une théorie performante pour la planification efficace et optimisée des ressources et des activités. La coloration de graphes offre une méthodologie structurée pour organiser efficacement les tâches, les ressources et les contraintes temporelles, facilitant ainsi la conception de plannings optimisés et fonctionnels. Malheureusement, il n’existe pas d’algorithme général capable de trouver une coloration optimale pour tous les types de graphes en un temps raisonnable. En fait, le problème de la coloration de graphes est NP-complet, ce qui signifie qu’il est très difficile voire impossible de trouver une solution optimale dans un temps polynomial pour tous les cas possibles. Il existe néanmoins plusieurs méthodes heuristiques, évolutives et exactes qui peuvent être utilisées pour trouver des solutions de bonne qualité, bien que celles-ci ne soient pas nécessairement optimales dans la pratique.

2.3 Méthode gloutonne

2.3.1 Algorithme de recherche A*

L'algorithme A* est une méthode de recherche utilisée pour trouver le chemin le plus court dans un graphe. Il utilise une fonction de coût pour évaluer les nœuds et choisit le chemin avec le coût estimé le plus bas.

Avantages

- Optimalité : L'algorithme A* garantit de trouver le chemin optimal du nœud de départ au nœud d'arrivée, ce qui est essentiel dans la planification des examens pour minimiser les conflits d'horaires et maximiser l'efficacité.
- Flexibilité : La fonction heuristique peut être adaptée pour prendre en compte différentes contraintes et objectifs spécifiques à la planification des examens, telles que les préférences des étudiants, les capacités des salles, etc.

Inconvénients

- Complexité : La complexité de l'algorithme A* peut augmenter rapidement avec la taille de l'instance du problème, ce qui peut rendre son utilisation inefficace pour de grandes écoles d'ingénieurs avec de nombreux étudiants et examens.
- Dépendance de la fonction heuristique : La qualité de la solution trouvée par l'algorithme A* dépend fortement de la qualité de la fonction heuristique utilisée. Si la fonction heuristique n'est pas bien conçue, l'algorithme peut prendre énormément de temps pour converger ou fournir des solutions non-optimales.

2.4 Méthode de recherche locale

2.4.1 Méthode du recuit simulé

La méthode du recuit simulé est une méthode de recherche heuristique qui permet d'explorer l'espace des solutions en acceptant parfois des solutions de qualité inférieure dans le but d'éviter de rester coincé dans des optima locaux.

Avantages

- Exploration efficace de l'espace des solutions : Le recuit simulé peut explorer efficacement l'espace des solutions, ce qui est essentiel dans la planification d'examens où il peut y avoir un grand nombre de combinaisons possibles.
- Évitement des optima locaux : En acceptant parfois des solutions de qualité inférieure, le recuit simulé peut éviter de rester coincé dans des optima locaux et trouver des solutions de meilleure qualité.

- Facilité d’ajustement des paramètres : La méthode du recuit simulé offre plusieurs paramètres (comme le nombre de matière et le nombre d’élèves) qui peuvent être ajustés pour contrôler la vitesse de convergence et l’exploration de l’espace des solutions.

Inconvénients

- Qualité de la solution : Bien que le recuit simulé puisse trouver des solutions de haute qualité, il ne garantit pas de trouver la meilleure solution possible. Il peut être difficile de déterminer quand arrêter l’algorithme pour obtenir une solution satisfaisante.
- Sensibilité aux paramètres : Le comportement du recuit simulé peut être sensible aux paramètres choisis, tels que le nombre de matière et le nombre d’élèves. Trouver les bons paramètres peut être un défi et nécessiter des essais et des ajustements.

2.4.2 Méthode de recherche tabou

La méthode de recherche tabou est une méthode de recherche locale qui explore l’espace des solutions en se déplaçant de solution en solution, en évitant de rester coincée dans des optima locaux grâce à l’utilisation d’une mémoire à court terme appelée liste tabou.

Avantages

- Efficacité dans la gestion des optima locaux : La recherche tabou est bien adaptée pour gérer des problèmes d’optimisation combinatoire comme la planification d’examens, où il peut être facile de rester coincé dans des optima locaux.
- Exploration efficace de l’espace des solutions : La méthode de recherche tabou explore efficacement l’espace des solutions en se déplaçant de solution en solution, ce qui peut être bénéfique dans des problèmes complexes comme la planification d’examens.
- Facilité d’implémentation : La recherche tabou est relativement simple à implémenter et peut être adaptée pour prendre en compte différentes contraintes et objectifs spécifiques à la planification d’examens.

Inconvénients

- Qualité de la solution : Bien que la recherche tabou puisse trouver des solutions de haute qualité, elle ne garantit pas de trouver la meilleure solution possible. La qualité de la solution dépend des paramètres choisis et de la manière dont la méthode est mise en œuvre.
- Sensibilité aux paramètres : Le comportement de la recherche tabou peut être sensible aux paramètres choisis, tels que la taille de la liste tabou et le nombre

d'itérations. En effet, la gestion de la mémoire liée à la liste tabou peut être compliquée et la complexité spatiale et temporelle de l'algorithme constitue un frein à l'utilisation de la recherche tabou. Ainsi, trouver les bons paramètres peut nécessiter des essais et des ajustements.

2.5 Méthode évolutive

2.5.1 Algorithme génétique

Les algorithmes génétiques sont des méthodes d'optimisation inspirées du processus biologique de la sélection naturelle. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable.

Avantages

- Exploration de l'espace des solutions : Les algorithmes génétiques peuvent explorer efficacement l'espace des solutions, ce qui peut être bénéfique dans des problèmes complexes comme la planification d'examens.
- Adaptabilité : Les opérateurs génétiques peuvent être adaptés pour prendre en compte différentes contraintes et objectifs spécifiques à la planification d'examens, tels que les préférences des étudiants ou des enseignants.

Inconvénients

- Complexité : La mise en œuvre et le réglage des paramètres d'un algorithme génétique peuvent être complexes, en particulier pour des problèmes de grande taille comme la planification d'examens.
- Dépendance des paramètres : Les performances d'un algorithme génétique peuvent dépendre sensiblement des paramètres choisis, tels que la taille de la population, les taux de croisement et de mutation.

2.5.2 Algorithme de colonies de fourmis

L'algorithme de colonie de fourmis imite le comportement des fourmis cherchant de la nourriture, en utilisant des phéromones pour trouver des chemins optimaux vers une solution dans un espace de recherche.

Avantages

- Adaptabilité : Les colonies de fourmis peuvent être adaptées pour prendre en compte différentes contraintes et objectifs spécifiques à la planification d'examens, telles que les préférences des étudiants ou des enseignants.

- Exploration de l'espace des solutions : Les colonies de fourmis peuvent explorer efficacement l'espace des solutions, en particulier dans des environnements complexes où d'autres méthodes peuvent avoir du mal à converger.

Inconvénients

- Complexité : La mise en œuvre et le réglage des paramètres d'un algorithme génétique peuvent être complexes, en particulier pour des problèmes de grande taille comme la planification d'examens.
- Dépendance des paramètres : Les performances d'un algorithme génétique peuvent dépendre sensiblement des paramètres choisis, tels que la taille de la population, les taux de croisement et de mutation.

2.5.3 Méthode de décomposition

La méthode de décomposition consiste à diviser le problème global de la planification des examens en sous-problèmes plus petits et plus gérables. Ces sous-problèmes peuvent être résolus indépendamment les uns des autres, puis leurs solutions peuvent être combinées pour obtenir une solution globale.

Avantages

- Gestion de la complexité : La méthode de décomposition permet de gérer efficacement la complexité du problème global en le divisant en sous-problèmes plus simples et plus gérables.
- Parallélisme : Les sous-problèmes peuvent être résolus en parallèle, ce qui peut permettre une réduction significative du temps de calcul nécessaire pour trouver une solution globale.

Inconvénients

- Coordination des sous-problèmes : La résolution des sous-problèmes de manière indépendante peut rendre difficile la coordination et la garantie d'une solution globale cohérente et de haute qualité.
- Solutions sous-optimales : Les solutions obtenues pour chaque sous-problème peuvent être sous-optimales par rapport à la solution globale, ce qui peut conduire à des compromis dans la qualité de la solution finale.

Chapitre 3

Modélisation mathématique

3.1 Problème de coloration des graphes

Nous avons un planning $P = \{S_i \mid i \in \mathbb{N}\}$ avec S_i des sessions d'examens. Le problème de coloration des graphes consiste à trouver une fonction $c : V \rightarrow P$ avec $\text{card}(P) \leq k$, telle que pour chaque arête $(u, v) \in E$, u et v ne soient pas de la même couleur. Nous utiliserons l'algorithme de Welsh-Powell pour cela.

Algorithm 1 Algorithme de coloration de Welsh-Powell

```
1: Entrée : Un graphe  $G = (V, E)$ 
2: Sortie : Une coloration des sommets du graphe
3: Trier les sommets  $V$  en ordre décroissant de degrés
4: Initialiser la couleur  $c$  à 0
5: Initialiser un tableau  $\text{couleurs}[v] = -1$  pour chaque sommet  $v$ 
6: for chaque sommet  $v_i$  do
7:   if  $\text{couleurs}[v_i] = -1$  then
8:     Incréments la couleur  $c$ 
9:      $\text{couleurs}[v_i] \leftarrow c$ 
10:   for chaque sommet  $v_j$  après  $v_i$  do
11:     if  $\text{couleurs}[v_j] = -1$  et  $v_j$  n'est pas adjacent à un sommet de couleur  $c$ 
12:       then
13:          $\text{couleurs}[v_j] \leftarrow c$ 
14:       end if
15:     end for
16:   end if
17: end for
Retourner le tableau  $\text{couleurs}$ 
```

3.2 Problème de minimisation

Nous devons minimiser le nombre de sessions $i = C(v)$ avec $i \in \{0, 1, \dots, k\}$, en respectant les contraintes suivantes :

$$\min(i) = C(V)$$

3.2.1 Contrainte d'adjacence (Contrainte dure)

Deux matières d'une même filière ne peuvent pas avoir la même horaire :

$$\forall (u, v) \in E, \quad C(u) \neq C(v)$$

3.2.2 Contrainte d'équilibre (Contrainte souple)

Équilibrer le nombre de matières par session pour minimiser la variance :

$$\mu = \frac{1}{i} \sum_{s \in S} n_s, \quad \sigma^2 = \frac{1}{i} \sum_{s \in S} (n_s - \mu)^2$$

3.2.3 Contrainte de session (Contrainte dure)

Chaque session s a une capacité maximale P_{\max} en termes de places :

$$\forall s \in S, \quad \sum_{u \in V} \mathbb{I}_{\{C(u)=s\}} p_u \leq P_{\max}$$

3.2.4 Contrainte de Durée des Examens (Contrainte dure)

La durée de la session D_s doit être au moins égale à la somme des durées de tous les examens qui s'y déroulent :

$$D_s \geq \sum_{u \in V} \mathbb{I}_{\{C(u)=s\}} \cdot d_u$$

3.3 Algorithme Génétique pour la Création du Planning

L'algorithme génétique utilise une population de solutions pour représenter les différents plannings d'examens. Il évolue cette population au fil des générations en sélectionnant les parents les plus adaptés, en croisant ces parents pour générer des enfants, et en appliquant une mutation à la population.

Algorithm 2 Algorithme Génétique pour la Création du Planning

```
1: procedure ALGORITHMEGÉNÉTIQUE( $G, T_{\max}, N, p_m, \text{variance\_cible}$ )
2:    $population \leftarrow \text{InitialiserPopulation}(G, N)$ 
3:    $meilleure\_solution \leftarrow population[0]$ 
4:    $meilleure\_variance \leftarrow \text{calculerVariance}(meilleure\_solution)$ 
5:    $iterations\_sans\_amelioration \leftarrow 0$ 
6:   while  $iterations\_sans\_amelioration < T_{\max}$  et  $meilleure\_variance >$ 
    $variance\_cible$  do
7:      $population \leftarrow \text{selectionnerParents}(population, G)$ 
8:      $enfants \leftarrow \text{croiserParents}(population, G)$ 
9:      $population \leftarrow \text{muterPopulation}(population, p_m)$ 
10:    for each  $solution$  in  $population$  do
11:       $variance\_solution \leftarrow \text{calculerVariance}(solution)$ 
12:      if  $variance\_solution < meilleure\_variance$  and
         $\text{estSolutionValide}(solution, G)$  then
13:         $meilleure\_solution \leftarrow solution$ 
14:         $meilleure\_variance \leftarrow variance\_solution$ 
15:         $iterations\_sans\_amelioration \leftarrow 0$ 
16:      else
17:         $iterations\_sans\_amelioration \leftarrow iterations\_sans\_amelioration + 1$ 
18:      end if
19:    end for
20:  end while
21:  return  $meilleure\_solution, meilleure\_variance$ 
22: end procedure
```

Chapitre 4

Construction des bases de données

Les jeux de données fournis proviennent de différentes sources au sein d’une université et sont essentiels pour la gestion administrative et académique. Ils couvrent des informations sur les administrateurs, les étudiants, les salles de classe, les matières enseignées et l’emploi du temps des cours. Ces fichiers sont construits à partir de systèmes de gestion intégrés qui permettent de coordonner et de synchroniser les activités et les ressources de l’université. Voici une analyse détaillée de la construction de chaque fichier.

4.1 Administrateur.csv

4.1.1 Structure

Lucas Martin	Nom de l’administrateur
111	Identifiant unique de l’administrateur
SunnyDay21	Mot de passe de l’administrateur

4.1.2 Construction détaillée

1. **Collecte des données :**
 - Les informations sur les administrateurs (noms, identifiants, mots de passe) sont probablement collectées à partir du système de gestion des ressources humaines (HRMS) ou du système d’information des utilisateurs de l’université.
2. **Traitement des données :**
 - Extraction des données pertinentes depuis la base de données RH ou d’un annuaire LDAP (Lightweight Directory Access Protocol).
 - Vérification et nettoyage des données pour s’assurer de leur exactitude (par exemple, suppression des doublons, formatage uniforme des noms).
3. **Organisation des données :**

- Structure des données en colonnes avec le nom de l'administrateur, l'identifiant et le mot de passe.
- Exportation des données nettoyées dans un fichier CSV pour une manipulation et une intégration faciles dans d'autres systèmes de gestion.

4.2 emploi_du_temps.csv

4.2.1 Structure

Session	Numéro de la session de cours
Date	Date de la session
Début	Heure de début du cours
Fin	Heure de fin du cours
Matière	Nom de la matière enseignée

4.2.2 Construction détaillée

1. Collecte des données :

- Les horaires des cours, les dates, les matières et les plages horaires sont recueillis à partir du logiciel de planification académique utilisé par l'université.

2. Traitement des données :

- Extraction des données de planification depuis le système de gestion des emplois du temps.
- Synchronisation des informations pour assurer la cohérence entre les différentes entrées (par exemple, vérifier que les horaires ne se chevauchent pas pour une même salle ou un même professeur).

3. Organisation des données :

- Structuration des données en colonnes correspondant aux sessions, dates, heures de début et de fin, et matière.
- Formatage des dates et heures dans un format standardisé.
- Exportation des données dans un fichier CSV.

4.3 Etudiant.csv

4.3.1 Structure

ABDELMALEK Mohammed Mehdi	Nom de l'étudiant
---------------------------	-------------------

219036	Numéro d'identification unique de l'étudiant
kJ7pEa9F	Mot de passe de l'étudiant
GI	Groupe de l'étudiant

4.3.2 Construction détaillée

1. **Collecte des données :**
 - Les informations sur les étudiants (noms, numéros d'identification, mots de passe, groupes) sont collectées depuis le système de gestion des étudiants (SIS) de l'université.
2. **Traitement des données :**
 - Extraction des données depuis la base de données étudiante.
 - Nettoyage des données pour s'assurer qu'il n'y a pas d'erreurs (par exemple, élimination des enregistrements en double, vérification des groupes assignés).
3. **Organisation des données :**
 - Mise en forme des données en colonnes pour chaque attribut (nom, identifiant, mot de passe, groupe).
 - Formatage uniforme des noms d'étudiants et des identifiants.
 - Exportation des données dans un fichier CSV.

4.4 listeSalle.csv

4.4.1 Structure

Salles	Nom de la salle
capacité	Capacité d'accueil de la salle

4.4.2 Construction détaillée

1. **Collecte des données :**
 - Les informations sur les salles (noms et capacités) sont collectées à partir du système de gestion des infrastructures ou du logiciel de réservation des salles de l'université.
2. **Traitement des données :**
 - Extraction des données depuis la base de données des infrastructures.
 - Vérification des capacités des salles pour s'assurer de leur exactitude.
3. **Organisation des données :**

- Structuration des données en colonnes avec le nom de la salle et sa capacité.
- Exportation des données dans un fichier CSV.

4.5 Matieres.csv

4.5.1 Structure

GI	Matières du programme de Génie Informatique
Unnamed : 1	Nombre d'heures pour chaque matière du programme de Génie Informatique
GMF	Matières du programme de Génie Mathématique pour la Finance
Unnamed : 3	Nombre d'heures pour chaque matière du programme de Génie Mathématique pour
GMI	Matières du programme de Génie Mathématique et Informatique
Unnamed : 5	Nombre d'heures pour chaque matière du programme de Génie Mathématique et In

4.5.2 Construction détaillée

1. Collecte des données :

- Les informations sur les matières et les heures de cours sont recueillies à partir des programmes académiques et des descriptifs des cours fournis par les départements de l'université.

2. Traitement des données :

- Compilation des données des différents programmes (GI, GMF, GMI).
- Vérification des heures de cours pour chaque matière pour assurer qu'elles correspondent aux exigences du programme.
- Nettoyage des données pour s'assurer qu'elles sont complètes et correctement formatées.

3. Organisation des données :

- Structuration des données en colonnes pour chaque programme, avec les matières et les heures de cours correspondantes.
- Correction des noms de colonnes manquantes ou incorrectes si nécessaire.
- Exportation des données dans un fichier CSV.

4.6 Salle_Place.csv

4.6.1 Structure

Salle	Nom de la salle
--------------	-----------------

Place	Nombre de places disponibles dans la salle
--------------	--

4.6.2 Construction détaillée

1. Collecte des données :

- Les informations sur les salles et leur capacité en termes de places sont collectées à partir du système de gestion des infrastructures ou du logiciel de réservation des salles.

2. Traitement des données :

- Extraction des données depuis la base de données des infrastructures.
- Vérification des capacités des salles pour s'assurer de leur exactitude.

3. Organisation des données :

- Structuration des données en colonnes avec le nom de la salle et le nombre de places.
- Exportation des données dans un fichier CSV.

Ces étapes de construction impliquent une coordination étroite entre les différents départements de l'université (administratif, académique, infrastructure) et l'utilisation de systèmes informatiques intégrés pour assurer une gestion efficace et cohérente des données.

Chapitre 5

Fonctionnement de l'application

5.1 Introduction

L'application de gestion des emplois du temps a été conçue pour faciliter la consultation et la gestion des emplois du temps des étudiants et des administrateurs au sein de l'université. Développée à l'aide de QT Designer, cette application utilise une interface utilisateur intuitive pour permettre une interaction aisée avec les données académiques. Elle intègre plusieurs fonctionnalités clés, notamment la connexion sécurisée, la visualisation des emplois du temps et la recherche par nom.

5.2 Connexion Sécurisée

Lors du lancement de l'application, l'utilisateur est accueilli par une page de connexion sécurisée. Cette page permet de s'assurer que seuls les utilisateurs autorisés peuvent accéder aux informations sensibles. L'utilisateur doit entrer son numéro étudiant et son mot de passe pour se connecter.

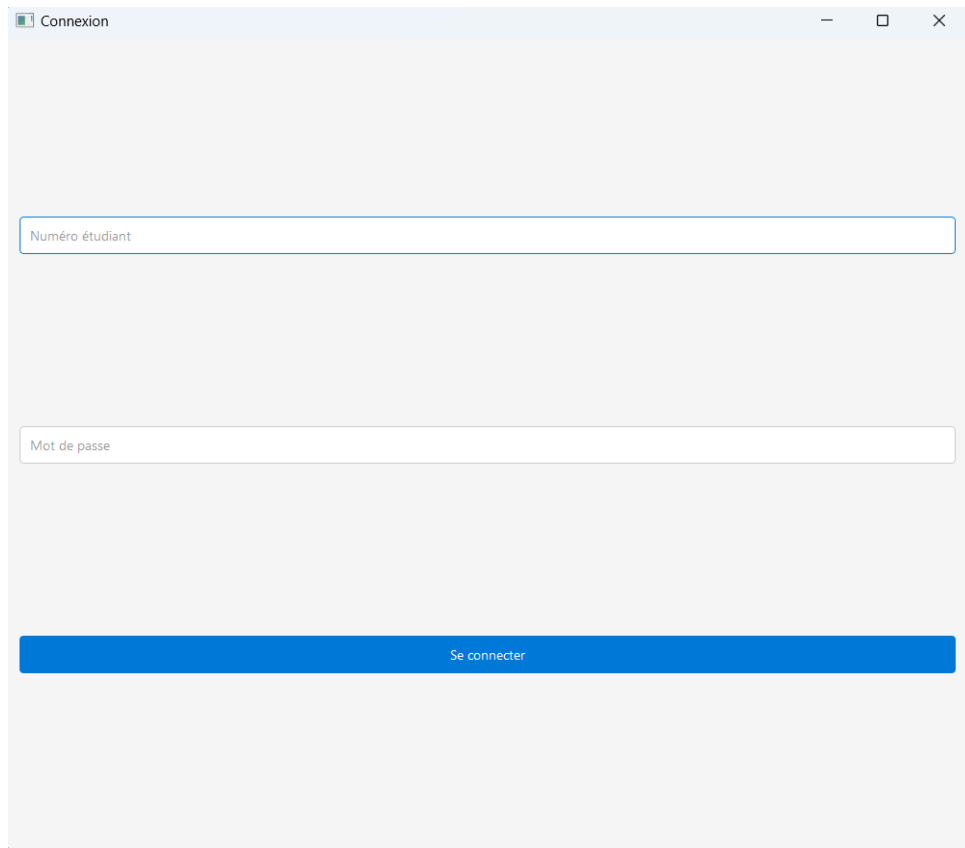
The image shows a web browser window with a title bar that says 'Connexion'. The page has a light gray background. There are two input fields: the first is labeled 'Numéro étudiant' and the second is labeled 'Mot de passe'. Below these fields is a blue button with the text 'Se connecter'.

FIGURE 5.1 – Page de connexion sécurisée

5.3 Visualisation des Emplois du Temps

Une fois connecté, l'utilisateur peut accéder à la page de visualisation des emplois du temps. Cette page affiche un emploi du temps hebdomadaire avec des créneaux horaires colorés pour chaque cours. Les cours sont affichés avec leur nom, la date, et l'heure de début et de fin. Cela permet aux étudiants de visualiser rapidement et facilement leurs cours planifiés.

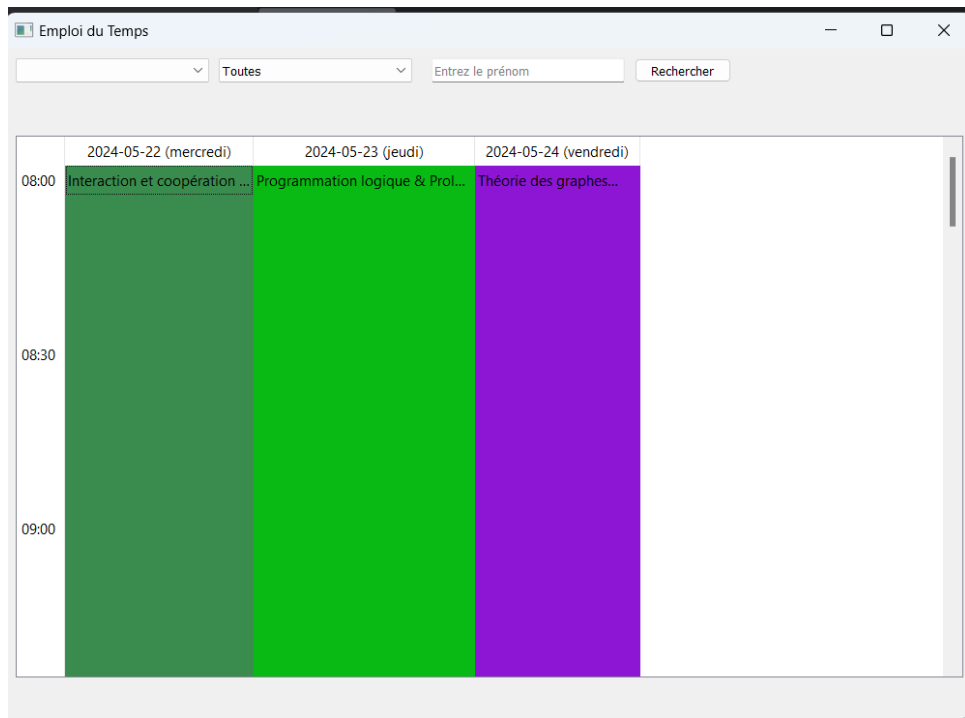


FIGURE 5.2 – Visualisation des emplois du temps

5.4 Recherche par Nom

L'application inclut également une fonctionnalité de recherche qui permet de filtrer les emplois du temps par nom d'étudiant. Cette fonctionnalité est particulièrement utile pour les administrateurs qui doivent gérer plusieurs emplois du temps. En entrant le nom de l'étudiant dans la barre de recherche, l'administrateur peut afficher l'emploi du temps spécifique de cet étudiant.

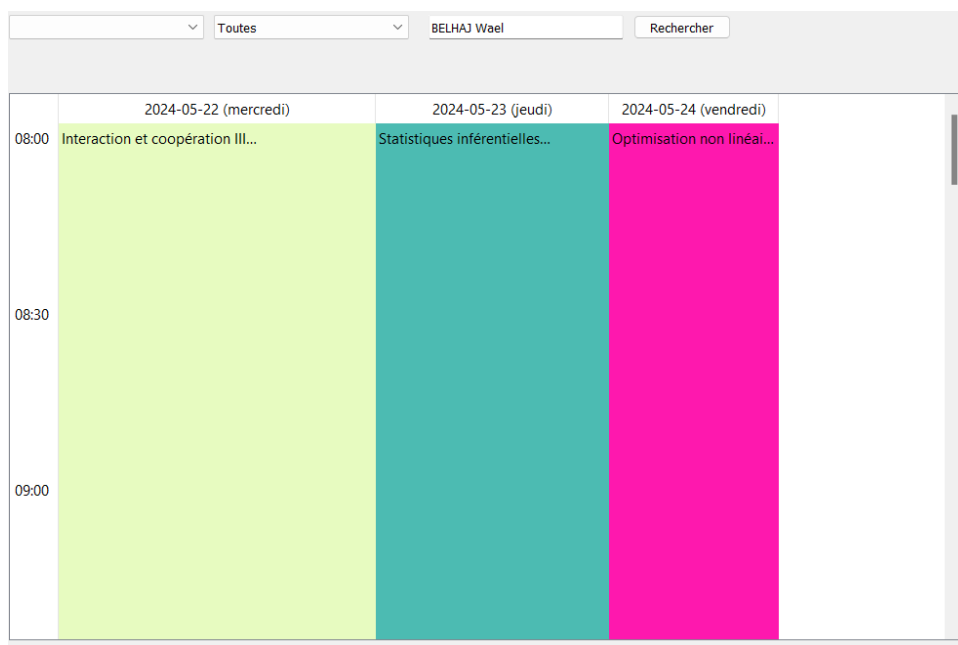


FIGURE 5.3 – Recherche par nom d'étudiant

Chapitre 6

Déroulement du Projet

Le projet d'emploi du temps s'est déroulé sur une période de trois semaines, entre le 4 mai et le 26 mai 2024. Voici un résumé détaillé des contributions de chaque membre de l'équipe :

6.1 ZERROUK Samuel et MIRZICA-VIGE Lucas

- **Responsabilités :**
 - **Modélisation UML :** Samuel et Lucas ont collaboré pour créer un modèle UML détaillé, représentant les différentes entités et leurs relations au sein du système de gestion des emplois du temps.
 - **Base de données CSV :** Ils ont développé la base de données en format CSV en utilisant les données fournies par CY Tech, garantissant la structure et la cohérence des fichiers.
 - **Gestion des fichiers CSV :** En utilisant Python, ils ont géré et manipulé les fichiers CSV pour les rendre compatibles avec les autres parties du projet, notamment l'interface utilisateur.
 - **Développement des classes :** Samuel et Lucas ont créé les classes nécessaires pour le projet, encapsulant les fonctionnalités et les données pour une meilleure modularité et réutilisabilité.
- **Objectif :**
 - Rendre toutes les données exploitables pour l'interface utilisateur afin de permettre une intégration fluide et une gestion efficace des emplois du temps.

6.2 HANG Victor

- **Responsabilités :**
 - **Conception graphique :** Victor a conçu l'interface utilisateur en utilisant

Qt Designer, en veillant à ce qu'elle soit intuitive et facile à utiliser pour les administrateurs et les étudiants.

- **Développement de l'interface** : Il a développé les fonctionnalités de l'interface utilisateur, intégrant les données des fichiers CSV pour afficher les emplois du temps de manière claire et organisée.
- **Gestion du dépôt GitHub** : Victor a maintenu le dépôt GitHub, assurant la coordination des contributions de l'équipe, le suivi des versions et la résolution des conflits de fusion.
- **Objectif** :
 - Réaliser une application fonctionnelle et attrayante qui permet aux utilisateurs de consulter et de gérer les emplois du temps de manière efficace.

6.3 BELHAJ Wael et DINH Khang Pierre

- **Responsabilités** :
 - **Modélisation de l'algorithme génétique** : Wael et Pierre ont travaillé sur la modélisation de l'algorithme génétique en tenant compte des contraintes spécifiques liées à la planification des cours.
 - **Implémentation de l'algorithme** : Ils ont implémenté l'algorithme en Python, optimisant les paramètres pour obtenir les meilleurs résultats possibles en termes de planification.
 - **Rédaction du rapport scientifique** : Ils ont rédigé un rapport scientifique détaillé, documentant les étapes de conception et de mise en œuvre de l'algorithme, ainsi que les résultats obtenus.
 - **Tests unitaires** : Wael et Pierre ont développé et exécuté des tests unitaires pour s'assurer que toutes les parties de l'application fonctionnaient correctement et répondaient aux spécifications.
- **Objectif** :
 - Fournir une solution conceptuelle performante et intégrée à l'interface utilisateur, permettant une planification efficace et sans erreurs des emplois du temps.

6.4 FEBRISSY Lilian

- **Responsabilités** :
 - **Préparation de la présentation** : Lilian a préparé une présentation complète, utilisant PowerPoint et LaTeX, pour résumer toutes les étapes du projet de A à Z.

- **Documentation des étapes du projet** : Il a documenté toutes les étapes importantes, des réunions de planification initiale à la phase finale de test et d'intégration.
- **Synthèse pour l'oral final** : Lilian a synthétisé les informations clés et préparé les membres de l'équipe pour l'oral final, en veillant à ce que tous les aspects du projet soient couverts de manière cohérente et claire.
- **Objectif** :
 - Cadrer, synthétiser et présenter le projet de manière professionnelle et compréhensible pour l'oral final, en mettant en avant les réalisations et les contributions de chaque membre de l'équipe.

Chapitre 7

Conclusion générale

Le projet de planification des emplois du temps universitaire a permis de mettre en œuvre une solution complète et efficace, répondant aux besoins spécifiques des étudiants et des administrateurs. Le choix de l’algorithme génétique s’est avéré judicieux en raison de sa robustesse, de sa rapidité et de sa capacité à gérer efficacement la complexité inhérente à ce type de problématique.

L’algorithme génétique, inspiré par les principes de la sélection naturelle, permet d’explorer un vaste espace de solutions possibles grâce à des opérateurs de sélection, de croisement et de mutation. Cette approche a non seulement permis de trouver des solutions optimales, mais a également offert la flexibilité nécessaire pour intégrer diverses contraintes spécifiques, telles que les préférences des étudiants, la capacité des salles et la non-concurrence des examens pour une même filière.

En comparaison avec d’autres méthodes explorées, telles que les algorithmes de recherche locale ou les méthodes heuristiques, l’algorithme génétique s’est distingué par sa capacité à éviter les optima locaux et à converger vers des solutions de haute qualité. Les algorithmes de recherche tabou et de recuit simulé, bien qu’efficaces dans certains contextes, n’ont pas offert le même niveau de performance globale et d’adaptabilité que l’algorithme génétique.

Le projet s’est déroulé sur trois semaines, durant lesquelles chaque membre de l’équipe a apporté des contributions significatives. Samuel Mirzica-Vige et Lucas Zerrouk ont assuré la création et la gestion des bases de données en format CSV, garantissant l’exploitabilité des données pour l’interface utilisateur. Victor Hang a conçu et développé une interface utilisateur intuitive et fonctionnelle, facilitant l’accès et la gestion des emplois du temps. BELHAJ Wael et DINH Khang Pierre ont travaillé sur la modélisation et l’implémentation de l’algorithme génétique, optimisant ses paramètres pour obtenir les meilleurs résultats possibles. Lilian Febrissy a synthétisé et documenté l’ensemble du projet, préparant une présentation claire et exhaustive pour l’oral final.

En résumé, ce projet a démontré l’efficacité de l’algorithme génétique pour la planification des emplois du temps universitaires, tout en mettant en lumière l’importance d’une

approche collaborative et multidisciplinaire. Les résultats obtenus montrent une nette amélioration de la gestion des emplois du temps, répondant aux attentes des utilisateurs finaux et posant les bases pour de futures améliorations et extensions du système.

Chapitre 8

Annexes

8.1 Diagramme UML

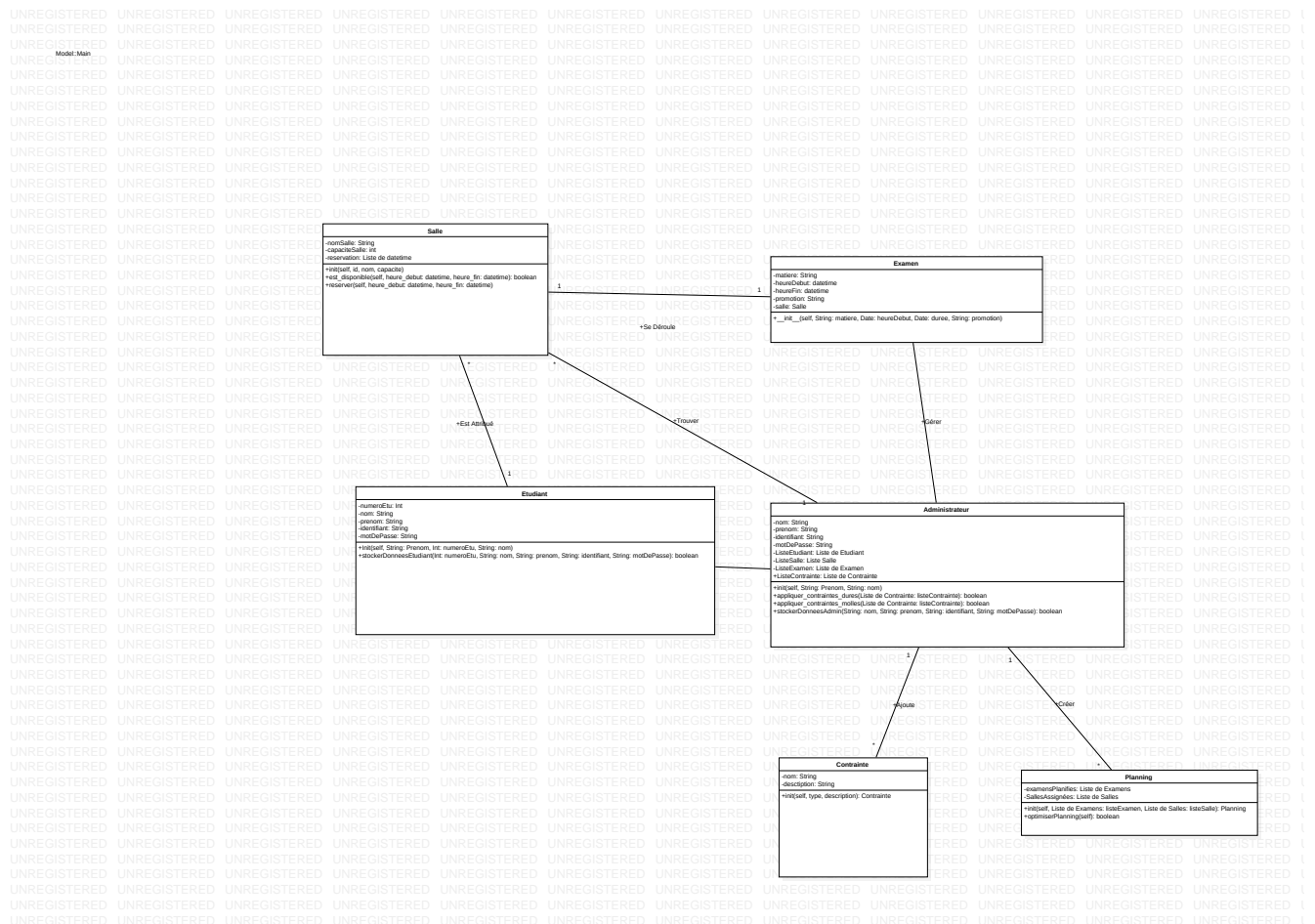


FIGURE 8.1 – Diagramme UML du projet de planification

8.2 Bibliographie

- Mohammad Dib, “Tabu-NG : hybridation de programmation par contraintes et recherche locale pour la résolution de CSP”, Novembre 2011. Disponible à : <https://theses.hal.science/tel-00644178/document>
- Alain Hertz, Nicolas Zufferey, “La coloration des sommets d’un graphe par colonies de fourmis”, Avril 2008. Disponible à : <https://www.gerad.ca/~alainh/G-2008-29.pdf>
- RAMDANE Latifa, “Automatisation De La Génération De l’Emploi Du Temps”, 2015-2016. Disponible à : <http://e-biblio.univ-mosta.dz/bitstream/handle/123456789/9691/MINF142.pdf?sequence=1&isAllowed=y>
- Dominique de Werra, Daniel Kobler, “Coloration de graphes : fondements et applications”, Décembre 2002. Disponible à : <http://www.numdam.org/item/10.1051/ro:2003013.pdf>
- Douali Ikram, “Modélisation numérique des emplois du temps des classes d’un établissement scolaire”, 2015-2016. Disponible à : <https://memoirepfe.fst-usmba.ac.ma/download/3297/pdf/3297.pdf>
- Daniel Cosmin Porumbel, “Algorithmes Heuristiques et Techniques d’Apprentissage : Applications au Problème de Coloration de Graphe”, Avril 2010. Disponible à : <https://theses.hal.science/tel-00644178>
- Hamioud Amina, Hantat Selma, “Approche hybride pour la résolution d’un DisCSP : génération automatique d’emploi du temps”, 2021-2022. Disponible à : <http://dspace.univ-jijel.dz:8080/xmlui/bitstream/handle/123456789/12819/Inf.IA.02-22.pdf?sequence=1&isAllowed=y>