REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

Faculté des Mathématiques et de l'Informatique

Département d'Informatique

N° : ……

DOMAINE : Mathématiques et Informatique

FILIERE : Informatique

OPTION : Intelligence Artificielle

Mémoire présenté pour l'obtention
Du diplôme de Master Professionnel

Par : Djaidja Badr

Intitulé

Planification automatique de l'emploi du temps universitaire

Soutenu le 12/06/2024 devant le jury composé de :

Pr. Bourahla Mustapha	Université de M'sila	Président
M. Khettaf Abdelouahab	Université de M'sila	Rapporteur
Dr. Barkat Abdelbasset	Université de M'sila	Examinateur

Année universitaire : 2023 / 2024

Remerciement

Nous remercions tout d'abord le bon DIEU pour nous avoir donnée le courage et la santé pour accomplir ce travail.

« (Celui qui ne remercie pas les gens, ne remercie pas Allah.) »

[Authentique Hadith]

Ce travail n'aurait pu aboutir à des résultats sans l'aide et l'encouragement de nombreuses personnes que nous remercions.

Je voudrais remercier mon cher enseignant, Mr. KHETTAF ABDELOUAHAB, pour son soutien et ses conseils pertinents et le temps qu'il m'a consacré tout au long de cette période, et savoir répondre à toutes mes questions, et grâce à lui j'ai déjà eu ce sujet, et parce que j'ai toujours su que je pouvais compter dessus, j'ai appris beaucoup de choses de vous mon enseignant, je dois vous dire : merci mon cher enseignant et mille mercis.

Que Dieu vous bénisse.

J'aimerais également à remercier les membres du jury, le Professeur Mr. BOURAHLA MUSTAPHA et le docteur Mr. BARKAT ABDELBASSET, d'avoir lu attentivement ce résumé pour en juger le contenu. Je les remercie aussi pour leurs leçons.

Que Dieu vous bénisse.

Je remercie toute ma famille, mes amis et tous ceux qui m'ont aidé et encouragé à réaliser ce travail, retrouvez ici ma gratitude et ma sincérité et ma gratitude du cœur.

Dédicace

Tout d'abord, je remercie DIEU le tout puissant et miséricordieux, qui m'a donné la force et la patience d'accomplir ce Modeste travail.

Remerciez DIEU pour l'amour, les remerciements et la gratitude pour commencer et conclure.

Et leur dernière revendication est que grâce à DIEU le Seigneur des deux mondes.

Au premier maître et enseignant qui nous a fait sortir des ténèbres de l'ignorance à la lumière de la science et nous a guidés sur le chemin de la science Notre maître, notre bien-aimé, notre modèle et notre patronage le jour du jugement Mohammad que DIEU le bénisse et lui accorde la paix.

S'il vous plaît, parents, je n'oublierai pas toute ma vie et tout ce que nous avons eu, maladie et vous l'avez porté sur nous.

« Cher papa »

Le symbole du don et du sacrifice et l'idéal de moi Aoun était mon bon dans l'adversité

« Cher mama »

C'est toi qui possèdes le paradis sous le pied. Vous me saluez avec un sourire.

Merci et mille fois merci

« Mes frères SAIF, ZAKkARIA, NADJEM et mon frère qui n'est pas de ma mère YOUNES »

« Mon oncle, et ma tante »

Ce grand entrepôt de pouvoir et d'amour.

Je remercie tous mes amis Qui j'ai connu dans ma vie.

Table des matières

Liste des tableaux

Liste des figures

CHAPITRE 01 Problème d'emploi du temps

CHAPITRE 02 Modélisation CSP

CHAPITRE 03 Implémentation

Liste des abréviations

CSP	**C**onstraint **S**atisfaction **P**roblem
PET	**P**roblème **d'E**mploi du **T**emps
ETU	**E**mploi du **T**emps **U**niversitaire
PETU	**P**roblème **d'E**mploi du **T**emps **U**niversitaire
AG	**A**lgorithme **G**énétique
RT	**R**echerche **T**abou
AC-1	**A**rc **C**onsistance **1**
AC-3	**A**rc **C**onsistance **3**
GET	**G**énère **E**t **T**este
SRA	**S**imple **R**etour **A**rrière
BT	**B**ack**T**racking
ISO	**I**nternational **S**tandardization for **O**rganization

INTRODUCTION GENERALE

La planification des emplois du temps universitaires est un défi complexe qui a un impact direct sur l'efficacité du processus éducatif et la satisfaction des étudiants et des enseignants. La génération manuelle de ces emplois du temps est souvent laborieuse et sujette à des erreurs, d'où la nécessité d'une solution automatique.

Ce travail s'inscrit dans le cadre de l'optimisation de la planification universitaire, en utilisant des méthodes avancées de programmation et de modélisation pour répondre aux besoins spécifiques de l'Université.

Dans le contexte universitaire, la gestion efficace de l'emploi du temps est cruciale pour assurer le bon déroulement des activités académiques, optimiser les ressources disponibles et répondre aux besoins variés des étudiants et des enseignants.

La complexité de cette tâche est souvent sous-estimée, car elle implique la coordination de multiples contraintes et préférences, telles que les disponibilités des salles, les préférences des enseignants, les exigences des programmes d'études et les contraintes logistiques.

En utilisant des techniques de résolution de contraintes (CSP), cette approche vise à modéliser et à résoudre les problèmes d'emploi du temps en tenant compte de toutes les contraintes pertinentes, tout en optimisant des objectifs spécifiques tels que la minimisation des conflits d'horaire et l'optimisation de l'utilisation des ressources.

Ce document est structuré en trois chapitres principaux. Le premier chapitre introduit le problème de la planification des emplois du temps, en détaillant les défis et les objectifs de cette tâche, ainsi que les méthodes existantes pour y parvenir. Le deuxième chapitre se concentre sur la modélisation du problème en termes de Problèmes de Satisfaction de Contraintes (CSP), en expliquant les concepts fondamentaux et en présentant des modèles appliqués au domaine universitaire. Le troisième chapitre décrit l'implémentation pratique de notre solution, en utilisant le langage Prolog et le logiciel Scryer Prolog, et présente les résultats obtenus.

CHAPITRE 01
Problème d’emploi du temps

1. Introduction

La gestion du temps dans les établissements d'enseignement est un défi majeur, en particulier dans les établissements d'enseignement supérieur. La complexité de la gestion des horaires exige une planification minutieuse pour répondre aux divers besoins des étudiants, des enseignants et du personnel administratif.

La création d'horaires équilibrés et efficaces est essentielle pour assurer le bon déroulement des activités scolaires et administratives et pour offrir une expérience d'apprentissage idéale à toutes les parties prenantes.

Donc, La planification automatique des horaires académiques est une approche visant à résoudre ce défi en identifiant les problèmes organisationnels des horaires des universités et en soulignant l'importance d'une approche automatique pour les résoudre.

2. La planification automatique :

2.1. Définition

La planification automatique est un processus informatique qui consiste à générer automatiquement des plans ou des schémas d'action pour atteindre un objectif donné, en prenant en compte un ensemble de contraintes, de préférences et de conditions spécifiques.

Cette approche vise à automatiser la tâche complexe de la planification, qui peut être sujette à des erreurs ou à des inefficacités lorsqu'elle est réalisée manuellement.

2.2. Objectif

L'objectif de la planification automatique dans le cadre de ce travail est de développer un outil qui permette de générer automatiquement l'emploi du temps universitaire en tenant compte des contraintes spécifiques liées aux cours, aux salles, aux enseignants et aux étudiants.

L'automatisation de ce processus vise à optimiser la répartition des ressources afin de faciliter l'organisation et de maximiser l'efficacité de l'emploi du temps.

2.3. Avantages

La planification automatique offre une multitude d'avantages qui touchent à différents aspects de la gestion des activités.

- Elle permet une meilleure utilisation des ressources disponibles.
- Une réduction des coûts opérationnels.

- Une amélioration de la qualité des produits ou services.
- Une plus grande flexibilité pour s'adapter aux changements.
- Une diminution des erreurs humaines.
- Ainsi qu'une meilleure adaptabilité aux besoins spécifiques de chaque organisation.

3. Emploi du temps

3.1. Définition

L'emploi du temps est un plan représentatif définissant le temps nécessaire pour élaborer des tâches et des objectifs fixés préalablement sous forme de créneaux horaires. Il consiste à gérer les charges de travail dans le temps tout en prenant compte des ressources humaines et matérielles disponibles **[2].**

Donc on peut définir le problème de l'emploi du temps comme chercher à gérer les ressources d'une manière satisfaisante en respectant au maximum que possible les contraintes de temps **[3]**.

3.2. Les domaines d'application

Le problème de l'emploi du temps se pose en général dans tous les secteurs :

3.2.1. Les secteurs économiques

Dans le domaine économique, la gestion du temps est cruciale pour optimiser la productivité et la rentabilité des entreprises.

Cela inclut la planification des tâches, des réunions et des projets pour atteindre les objectifs commerciaux et respecter les délais.

3.2.2. Les secteurs industriels

Dans les secteurs industriels, la gestion du temps est essentielle pour coordonner les opérations de fabrication, garantir l'efficacité des processus de production et minimiser les temps d'arrêt coûteux.

3.2.3. Les secteurs administratifs

Dans les structures administratives, la gestion du temps est nécessaire pour organiser les réunions, gérer les documents et les flux de travail, et respecter les échéances pour les rapports et les projets.

3.2.4. Les établissements scolaires

Dans les écoles, la gestion du temps est importante pour les élèves et les enseignants afin de planifier les cours, les devoirs, les examens et les activités parascolaires de manière efficace, en maximisant le temps d'apprentissage.

3.2.5. Les instituts

Dans les instituts de formation professionnelle ou académique, la gestion du temps est cruciale pour organiser les programmes de formation, planifier les sessions d'apprentissage et évaluer les progrès des étudiants.

3.2.6. Les universités

Dans les universités, la gestion du temps est essentielle pour les étudiants et les enseignants afin de gérer les cours, les recherches, les publications et les engagements universitaires tout en respectant les délais académiques.

3.2.7. Les établissements scolaires

Les entreprises de services telles que les agences de consulting, les cabinets d'avocats et les agences de marketing dépendent fortement de la gestion du temps pour fournir des services de haute qualité à leurs clients, en planifiant efficacement les projets et les engagements.

3.2.8. Les professions libérales

Les professionnels indépendants tels que les consultants, les avocats, les médecins et les architectes doivent bien gérer leur temps pour gérer leurs clients, leurs dossiers et leurs engagements professionnels de manière efficace.

Les résultats issus d'un secteur peuvent être utilisés dans un autre secteur mais nous nous intéressons particulièrement aux établissements universités.

3.3. Objectif

L'objectif de l'emploi du temps est d'organiser efficacement les activités, les ressources et le temps disponible afin d'atteindre certains résultats spécifiques. Voici quelques-uns des objectifs principaux de la gestion de l'emploi du temps :

3.3.1. Optimisation des ressources

Assurer que les ressources telles que le temps, le personnel, les équipements et les matériaux sont utilisées de manière efficace et économique.

3.3.2. Maximisation de la productivité

Organiser les activités de manière à ce qu'elles soient accomplies dans un délai raisonnable et avec le moins de gaspillage possible.

3.3.3. Respect des contraintes

Tenir compte des contraintes telles que les délais, les disponibilités de ressources et les préférences des parties prenantes pour élaborer un emploi du temps réalisable.

3.3.4. Amélioration de la qualité

Structurer les activités de manière à ce qu'elles puissent être réalisées avec soin et attention, ce qui peut contribuer à la qualité des résultats obtenus.

3.3.5. Réduction des conflits

Minimiser les conflits potentiels entre les activités concurrentes ou les demandes concurrentes sur les ressources.

3.3.6. Adaptabilité

Créer des emplois du temps flexibles qui peuvent être ajustés en fonction des changements imprévus ou des nouvelles priorités qui surviennent.

En somme, l'objectif global de la gestion de l'emploi du temps est de créer un plan organisé et réalisable qui permet d'optimiser les performances et d'atteindre les objectifs fixés dans un cadre donné.

3.4. Définition de l'emploi du temps universitaire

L'emploi du temps universitaire (ETU**)** (planification des horaires universitaires) est un calendrier détaillé qui répertorie les cours, les séminaires, les conférences et autres activités académiques prévues pour une période donnée dans une université ou un établissement d'enseignement supérieur.

L'emploi du temps universitaire est conçu pour optimiser l'utilisation des ressources disponibles (les salles, les laboratoires, les équipements spécialisés et les ressources humaines) tout en répondant aux besoins académiques des étudiants et des enseignants.

3.5. Différentes variantes pour la formalisation d'emploi du temps

Il existe un grand nombre de variantes du problème d'emploi du temps qui diffère les uns des autres selon l'université impliqué et le type de contraintes.

3.5.1. Emploi du temps des étudiants

Créer un programme hebdomadaire pour tous les étudiants universitaires avec des matières et des salles et des enseignants et tenir compte de toutes les contraintes possibles.

3.5.2. Emploi du temps des enseignants

Créer un programme hebdomadaire pour tous les étudiants universitaires avec des matières et des salles et des enseignants et tenir compte de toutes les contraintes possibles.

3.5.3. Emploi du temps des cours

Créer un programme hebdomadaire pour tous les cours de l'université en réduisant au minimum que possible les chevauchements des cours ayant les étudiants et les enseignants.

3.5.4. Emploi du temps des examens

Créer un programme pour les examens d'un ensemble d'étudiants. L'emploi du temps d'un examen est un peu différent de l'emploi du temps des cours car il impose autres contraintes.

3.6. Spécification des besoins

La gestion du temps relie quatre ressources différentes : les enseignants, les salles, les étudiants et les matières.

Cette partie a pour but de planifier l'emploi du temps pour l'université ou le collège.

Les étudiants sont rassemblés pour chaque leçon dans des groupes, dans des salles cour, des salles TD ou des salles TP, où chaque matière est supervisée par un ou plusieurs enseignants. Le conflit se caractérise par le partage des ressources entre plusieurs leçons en même temps. L'idée de base de notre travail est de créer un calendrier qui permet de vérifier de manière interactive l'état des sessions tout en résolvant ces litiges pour éviter tout conflit.

3.7. L'objectif de l'emploi du temps universitaire

L'objectif de l'emploi du temps universitaire est de créer un calendrier d'activités académiques qui permette une expérience d'apprentissage efficace et équilibrée pour les étudiants tout en soutenant les objectifs institutionnels et pédagogiques de l'université.

Voici quelques-uns des objectifs clés :

3.7.1. Optimisation de l'utilisation du temps

L'emploi du temps universitaire vise à maximiser l'utilisation efficace du temps disponible pour l'apprentissage en planifiant judicieusement les cours, les examens, les séminaires et autres activités académiques.

3.7.2. Équilibrage des charges de travail

Il cherche à répartir équitablement les charges de travail entre les différents jours de la semaine et les différentes périodes de la journée afin d'éviter des périodes de surcharge ou de sous-utilisation.

3.7.3. Satisfaction des exigences du programme

L'emploi du temps doit répondre aux exigences spécifiques de chaque programme d'études en garantissant que tous les cours obligatoires et recommandés sont proposés à des moments où les étudiants peuvent les suivre.

3.7.4. Prise en compte des préférences des étudiants

Il tient compte, dans la mesure du possible, des préférences des étudiants en ce qui concerne les horaires des cours, les préférences des enseignants et les obligations personnelles des étudiants.

3.7.5. Facilitation de la réussite académique

Un bon emploi du temps universitaire prend en considération les rythmes d'apprentissage des étudiants et vise à créer un environnement qui favorise la concentration, la participation et l'engagement dans les cours.

3.7.6. Minimisation des conflits d'horaires

Il cherche à éviter les conflits d'horaires entre les différents cours et activités académiques afin de permettre aux étudiants de suivre les cours de leur choix sans rencontrer de difficultés logistiques.

3.7.7. Utilisation efficace des ressources

L'emploi du temps doit optimiser l'utilisation des ressources institutionnelles telles que les salles de classe, les laboratoires, les bibliothèques et le personnel enseignant.

En bref, l'objectif ultime de l'utilisation de l'emploi du temps de l'Université est de mieux répartir les ressources disponibles dans les périodes disponibles.

4. Problème d'emploi du temps universitaire PETU

4.1. Définition

Le PETU fait référence à la difficulté de planifier et d'organiser de manière efficace les horaires des cours, des examens, des activités académiques et des autres engagements au sein d'une institution d'enseignement supérieur. Il s'agit de trouver un équilibre entre les besoins des étudiants, les disponibilités des enseignants et des salles de classe, ainsi que les exigences du programme d'études.

Ce problème peut être complexe en raison de plusieurs facteurs, tels que :

4.1.1. Nombre d'étudiants et de cours

Les institutions universitaires proposent généralement une grande variété de cours et de programmes, ce qui rend difficile la création d'un emploi du temps qui répond aux besoins de tous les étudiants et qui évite les conflits d'horaires entre les cours.

4.1.2. Disponibilité des enseignants

Les enseignants ont souvent des contraintes de disponibilité en raison de leurs autres responsabilités académiques, de leurs recherches ou de leurs engagements extérieurs à l'université, ce qui peut compliquer la planification des cours.

4.1.3. Utilisation efficace des ressources

Il est important d'optimiser l'utilisation des salles de classe, des laboratoires et d'autres installations afin de maximiser l'efficacité et de répondre aux besoins de tous les cours et activités.

4.1.4. Préférences des étudiants

Les étudiants peuvent avoir des préférences quant aux horaires des cours, aux périodes d'examen ou aux plages horaires pour les activités parascolaires, ce qui peut rendre la planification plus complexe.

4.1.5. Contraintes logistiques

Des contraintes telles que la distance entre les bâtiments, les temps de déplacement des étudiants et des enseignants, ainsi que les disponibilités des transports en commun peuvent également influencer la planification de l'emploi du temps.

En somme, le PETU consiste à élaborer des horaires qui permettent aux étudiants de suivre les cours nécessaires pour leur programme d'études, tout en tenant compte des

contraintes et des préférences des enseignants et des étudiants, et en maximisant l'utilisation efficace des ressources disponibles.

4.2. Description du problème à résoudre

À l'université, un groupe d'étudiants et d'enseignants est censé avoir un horaire afin que ce groupe d'étudiants soit enseigné par un enseignant et que ce processus prenne la forme de leçons afin que chaque leçon prenne un sujet, une salle et un type pour cette leçon.

Pour résoudre le PETU, il est nécessaire d'allouer un certain nombre de périodes consécutives, de sorte que chaque période est un nombre naturel.

4.3. Ressources utilisées dans un PETU

Pour créer ETU, nous devons savoir quelles ressources sont organisées dans ce l'emploi du temps :

4.3.1. L'enseignants

Les enseignants jouent un rôle essentiel en tant que ressource dans la planification automatique de ETU.

Ils fournissent des informations sur leur disponibilité et leurs préférences de temps, ainsi que sur leurs aptitudes et compétences. Ils aident également à clarifier les contraintes personnelles et à communiquer sur les besoins spéciaux, et favorisent la flexibilité dans la gestion des défis d'horaire pour répondre efficacement aux besoins des étudiants et des établissements.

4.3.2. Groupes d'étudiants

Les étudiants sont la principale catégorie de l'université qui est directement affectée par le problème de l'emploi du temps.

Les étudiants peuvent être considérés comme des personnes à la recherche d'une éducation et d'une formation universitaires, et leurs études et leurs besoins personnels nécessitent une coordination efficace de l'emploi du temps.

Pour résoudre le problème de ETU, il faut également éviter les horaires conflictuels et s'assurer que les leçons nécessaires sont disponibles au bon moment pour tous les étudiants.

4.3.3. Les salles

Les salles dans le contexte du problème de ETU se réfèrent aux espaces réels utilisés pour tenir des leçons, des conférences et d'autres événements académiques.

Les salles peuvent être définies comme des zones qui doivent être allouées efficacement et structurées dans l'horaire pour répondre aux besoins des élèves et des enseignants. Le problème de ETU consiste à relever le défi de fournir des salles appropriées au bon moment et d'allouer des salles en fonction de la capacité d'hébergement et de l'équipement nécessaire pour chaque activité académique.

Il faut tenir compte de facteurs tels que la disponibilité des installations, les exigences en matière d'études lors de l'organisation de l'emploi du temps de l'université et le choix des salles adaptées aux activités universitaires.

4.3.4. Les matières

Le terme "matières" désigne les cours enseignés et complétés au cours du semestre.

Les matières peuvent être définis comme du contenu académique offert par les enseignants dans les salles de classe, les laboratoires et d'autres espaces. Le problème de ETU comprend l'organisation et la planification du matériel d'une manière qui assure leur compatibilité avec le calendrier des étudiants et offre des possibilités appropriées pour l'apprentissage et le développement académique.

Cela exige une distribution équilibrée des matières sur des jours et des heures différents, en évitant les incohérences dans l'emploi du temps, ainsi que la fourniture d'infrastructures et de ressources pour fournir des documents de haute qualité et dans un environnement d'apprentissage approprié.

4.3.5. Le temps

"Le temps" représente les périodes spécifiques au cours desquelles des activités académiques telles que des leçons, des conférences et des examens sont organisées.

Le temps est un facteur essentiel pour organiser l'emploi du temps d'étudiants et des enseignants et assurer la mise en œuvre efficace des activités académiques.

La bonne compatibilité des horaires nécessite une coordination efficace entre les tables d'étude des matières, la disponibilité des salles, des enseignants et des besoins d'étudiants.

4.4. Les contraintes

- **Une contrainte :** c'est une règle obligatoire qui réduit la liberté d'action, un événement ou une obligation **[4]**.

Ces contraintes sont subdivisées en deux catégories :

4.4.1. Les contraintes dures

Ce sont des contraintes qui doivent être satisfaite car la violation d'un de ces contraintes peut générer un emploi du temps inacceptable.

- Un enseignant peut enseigner un module dans une salle pendant une séance.
- Un groupe occupe une salle pendant une séance.
- Chaque enseignant ne peut enseigner que le module de ces compétences.
- Le module doit avoir un nombre limite de séance (TD, cours, TP).

4.4.2. Les contraintes préférence

La violation de ces contraintes n'a aucun effet sur la génération d'une solution satisfiable.

- Il faut que l'affectation des salles et des périodes de temps permette de satisfaire au mieux les préférences des enseignants.
- Il faut que l'affectation des salles aux différentes séances permette de satisfaire aux mieux certaines préférences.

4.5. Toutes les contraintes possibles dans un PETU

On a ajouté toutes les contraintes possibles à ce problème :

- Chaque créneau se compose d'enseignant, module, salle, type de leçon et groupe d'étudiants une fois par semaine.
- Ne pas chevaucher les leçons de l'enseignant dans le sens où l'enseignant ne peut pas enseigner deux leçons en même temps.
- Ne pas chevaucher les salles à l'heure de l'étude en ce sens que la salle ne peut pas occuper deux leçons en même temps.
- Chaque groupe d'étudiants a un ou plusieurs temps libre, plusieurs jours libre, L'étudiant a au moins une pause de 45 minutes aux heures de midi.
- Chaque enseignant a un ou plusieurs temps libres, plusieurs jours libre, L'enseignant a au moins une pause de 45 minutes aux heures de midi.
- Deux groupes d'étudiants qui s'appartiennent n'étudient pas en même temps.

- Disponibilité d'enseignants : L'emploi du temps essaie de respecter la journée de disponibilité des enseignants.
- Ne pas chevaucher dans les leçons d'étudiants dans le sens où groupe d'étudiants ne peut pas enseigner deux leçons en même temps.
- Chaque matière doit avoir un nombre limite de séance (TD, cours, TP).
- Il faut que l'affectation des salles et des périodes de temps permette de satisfaire au mieux les préférences des enseignants.
- Il faut que l'affectation des salles aux différentes séances permette de satisfaire aux mieux certaines préférences.
- Contraintes sur les horaires : On ne peut commencer une unité pédagogique si on ne peut la finir dans les horaires prédéfinis. Par exemple, on ne peut commencer un cours de deux heures à 17h.
- Toutes les unités pédagogiques doivent être positionnées dans l'emploi du temps.
- Contraintes de choix de salles : les affectations des salles doivent respecter (TD, cours, TP).
- Contraintes de capacité de salle : Une unité pédagogique ne peut être planifiée dans une salle ne permettant pas de recevoir l'ensemble des étudiants lié à cette unité.

En tenant compte de toutes ces contraintes, les planificateurs de l'emploi du temps universitaire doivent trouver un équilibre entre les besoins des enseignants, des étudiants et de l'institution pour créer un emploi du temps fonctionnel et équitable.

4.6. Contraintes qui nous intéressent pour résoudre le PETU

Nous nous intéressons à ces contraintes car elles sont fondamentales à ce problème et nous donneront une solution efficace :

- Chaque créneau se compose d'enseignant, module, salle, type de leçon et groupe d'étudiants une fois par semaine.
- Ne pas chevaucher les leçons de l'enseignant dans le sens où l'enseignant ne peut pas enseigner deux leçons en même temps.
- Ne pas chevaucher les salles à l'heure de l'étude en ce sens que la salle ne peut pas occuper deux leçons en même temps
- Chaque groupe d'étudiants à un ou plusieurs temps libres et plusieurs jours libres.
- Chaque enseignant à un ou plusieurs temps libres et plusieurs jours libres.
- Chaque matière doit avoir un nombre limite de séance (TD, cours, TP).

Ce sont les contraintes importantes que nous devons atteindre.

5. Quelques méthodes et techniques de résolution du PET

Le tableau ci-dessous résume les techniques qui ont automatisé l'emploi du temps dès dernières années :

Année	Méthodes											Auteurs
	CSP[a]	PL[b]	RT[c]	AG[d]	Co[e]	PPC[f]	HC[g]	RS[h]	RN[i]	CF[j]	LA[k]	
1960											●	Appleby,Blake&Newman
1961											●	Lewis
1969		●										Lawrie
1970-1980			●	●				●				
1994				●								Corne
1998					●							David
1999	●											Potts et al.
2002			●	●								Paquete&Stutzle/Sheibani
2003						●	●	●				Merlot et al.
2004			●	●				●		●		Naji Azimi
2005									●			Corr et al

[a]Probleme de satisfaction de contraintes
[b]Programmation lineaire
[c]Recherche tabou
[d]Algorithme genetique
[e]Techniques de satisfaction de contraintes
[f]Programmation par contraintes
[g]Hill-climbing
[h]Recuit simulé
[i]Reseau de neurones
[j]Colonies de fourmis
[k]Look-ahead

Tableau 1. 1**:** Récapitulation des méthodes d'automatisation d'un emploi du temps **[14]**.

5.1. Programmation linéaire

Les techniques de programmation linéaire (PL) -les premières appliquées à la planification du temps- ont été développées à partir du domaine plus large de la programmation mathématique. La programmation mathématique s'applique à la classe de problèmes caractérisée par un grand nombre de variables qui se croisent dans des limites imposées par un ensemble de conditions restrictives **[11].**

Le mot "programmation" signifie planification dans ce contexte et est lié au type d'application **[12].**

Ce schéma de programmation a été développé pendant la seconde guerre mondiale en vue de trouver des stratégies optimales pour mener l'effort de guerre et utilisé ensuite dans les domaines de l'industrie, du commerce et des services gouvernementaux **[13]**.

La programmation linéaire est ce sous-ensemble de la programmation mathématique qui concerne l'affectation efficace de ressources limitées à des activités connues dans le but d'atteindre un objectif souhaité, comme maximiser les profits ou minimiser les coûts **[12]**.

La construction d'un modèle de programmation linéaire implique trois étapes successives de résolution de problèmes. La première étape identifie les variables de décision

inconnues ou indépendantes. La deuxième étape nécessite l'identification des contraintes et la formulation de ces contraintes sous forme d'équations linéaires. Enfin, dans la troisième étape, la fonction objective est identifiée et écrite comme une fonction linéaire des variables de décision **[5]**.

5.2. Algorithme génétique (AG)

Les algorithmes génétiques constituent une approche puissante pour résoudre divers problèmes d'optimisation, y compris le problème complexe de l'emploi du temps universitaire. Imaginez le défi de planifier les cours, les salles de classe et les enseignants de manière à maximiser l'utilisation des ressources tout en respectant les contraintes comme les disponibilités des enseignants, les préférences des étudiants et les capacités des salles. Les algorithmes génétiques offrent une méthode itérative inspirée de la sélection naturelle et de la génétique pour trouver des solutions efficaces à ces problèmes.

L'idée fondamentale derrière les algorithmes génétiques est de créer une population initiale de solutions potentielles, représentées sous forme de chromosomes ou de génotypes. Dans le contexte de l'emploi du temps universitaire, chaque solution pourrait être une disposition spécifique des cours, des enseignants et des salles de classe. Ensuite, des opérations telles que la sélection, le croisement et la mutation sont appliquées pour générer de nouvelles solutions, en favorisant celles qui répondent le mieux aux objectifs de l'optimisation.

Les objets de l'algorithme génétique proposé sont les suivants :

- Enseignant - avec carte d'identité et le nom.
- Groupe d'étudiants - avec pièce d'identité et le nombre d'étudiants (taille du groupe)
- Salles - avec pièce d'identité et le nom de la salle, ainsi que le nombre de sièges et des informations sur l'équipement (ordinateurs).
- Cours - avec ID et le nom du cours.
- Matières - avec pièce d'identité et le nom de la matière.
- Classe - avec une référence à la matière et au cours auxquels la classe appartient, une référence au professeur qui enseigne et une liste des groupes d'étudiants qui assistent à la classe. Il enregistre également le nombre de places (somme des tailles des groupes d'étudiants) nécessaires dans la salle de classe, si la classe a besoin d'ordinateurs dans la salle de classe, et la durée de la classe (en heures).

Nous ne considérons que les contraintes dures suivantes :

- Une classe A peut-être assignée à une chambre libre.
- Aucun enseignant ou groupe d'étudiants ne peut avoir plus d'une classe à la fois.

- Une salle doit avoir suffisamment de places pour tous les étudiants.
- La salle doit avoir de l'équipement (ordinateurs) si la classe l'exige.

Si le nombre total de classes est L, alors un chromosome pour une liste de classes est représenté par une table de hachage avec des éléments L, où pour chaque élément la clé est l'ID de classe (entier de 1 à L) et la valeur est le nombre de l'espace-temps dans le vecteur, dont appartient la première heure de cette classe (entier de 1 à working_days * number_of_rooms *classes_per_day).

L'aptitude du chromosome est calculée comme suit :

- Chaque classe peut avoir 0 à 8 points.
- Si une classe utilise une salle gratuite, nous ajoutons 1 à son score.
- Si une classe nécessite de l'équipement et que la salle assignée est équipée, nous augmentons le score de la classe.
- Si une classe est située dans une salle avec un nombre suffisant de sièges, nous augmentons son score.
- Si un professeur n'a pas d'autres classes à ce moment, nous incrémentons le score de la classe.
- Si l'un des groupes d'étudiants qui assistent à la classe n'a pas d'autre classe à ce moment, nous incrémentons le score de la classe.
- Si la durée de la classe est d'une heure ou si tous les créneaux horaires de la classe dont la durée est supérieure à une heure sont attribués en une journée de travail, nous augmentons le score de la classe.
- Si la salle préférée de la classe n'est pas pointée ou si elle coïncide avec la salle préférée, nous incrémentons le score de la classe.
- Si le bâtiment universitaire préféré de la classe n'est pas pointé ou si cela coïncide avec le bâtiment universitaire préféré, nous incrémentons le score de la classe.
- Le score total d'un horaire de classe est la somme des points de toutes les classes.
- La valeur d'aptitude est calculée comme schedule_score / maximum_score, où maximum_score est total_number_of_classes * 8.

La valeur de fitness est comprise entre 0 et 1.

Une opération de croisement combine des données dans les tables de hachage de deux parents, puis elle crée une nouvelle table de hachage. Un croisement « divise » les tables de hachage des deux parents en parties de taille aléatoire. Le nombre de parties est défini par

le nombre de points de croisement (plus un) et dans notre cas, il est de 2. Ensuite, il copie alternativement des parties des parents au nouveau chromosome.

Une opération de mutation prend une classe au hasard et la déplace vers une autre fente choisie au hasard. Le nombre de classes qui vont être déplacées dans une seule opération est défini par la taille de la mutation et dans notre cas c'est 2.

5.3. L'heuristique

L'heuristique est une règle générale d'action applicable à plusieurs situations permettant d'aboutir plus rapidement à la solution.

Pour le cas de l'emploi du temps l'heuristique remplit directement l'horaire complet d'un cours ou d'un groupe de cours à la fois dans la mesure où aucun conflit ne surgit. Un système Schola est un exemple typique de cette méthode **[2]**.

Ce système est basé sur les trois stratégies suivantes :

A) Assigner le cours le plus urgent à la période la plus favorable.

B) Quand une période peut être affectée seulement pour un cours, assigner une période à un cours.

C) Déplacer un cours déjà programmé à une période libre afin de laisser la période au cours cherchant à être affecté.

Le système **SCHOLA** programme les cours en alternant les stratégies **A** et **B** autant que possible. Quand plusieurs cours ne peuvent être programmés de cette façon, elle commence à employer la stratégie **C [2]**.

La stratégie **A** est le noyau du système et elle est utilisée dans plusieurs systèmes en définissant l'urgence et la période favorable différemment.

Un cours est urgent quand le professeur et la classe ont peu de disponibilité pendant qu'il y a beaucoup de cours à dispenser. Une période est favorable quand un petit nombre de cours peut être programmé à la période selon la disponibilité des professeurs et salles.

L'utilisation de la stratégie **B** est effectuée dans le cas où il y a un défaut au niveau de la stratégie **A**. La stratégie **C** fournit une forme limitée de retour arrière à récupérer les défauts de la stratégie **A**.

5.4. Méthode manuelle

Les planificateurs peuvent utiliser des feuilles de calcul ou des logiciels de traitement de texte pour organiser les horaires des cours en fonction des contraintes et des préférences des enseignants, des étudiants et de l'institution.

En utilisant cette méthode manuelle avec soin et précision, un calendrier universitaire fonctionnel peut être élaboré qui répond aux besoins et aux limites de l'organisation tout en minimisant les conflits et les problèmes potentiels.

Cette méthode peut être laborieuse et sujette aux erreurs humaines, mais elle offre une certaine flexibilité pour prendre en compte diverses considérations.

5.5. Recherche Tabou

Recherche Tabou (RT) est une technique méta-heuristique qui guide une procédure de recherche heuristique locale pour explorer l'espace de solution au-delà de l'optimalité locale **[7]**. Formellement, une métaheuristique est définie comme une stratégie maîtresse qui guide et modifie d'autres heuristiques pour produire des solutions au-delà de celles normalement générées dans une quête d'optimalité locale dans les problèmes d'optimisation **[7]**.

Les procédures méta-heuristiques se sont énormément améliorées grâce aux progrès réalisés ces dernières années, en particulier grâce au développement de nouvelles implémentations de RT qui sont plus efficaces pour résoudre des problèmes difficiles que ce qui était considéré auparavant possible **[8]**.

5.5.1. Méthode de résoudre

Pour résoudre ce problème de ETU, nous devons introduire la notation où nous allons essayer de faire une meilleure routine en utilisant un certain score. Nous verrons comment la notation nous aidera à résoudre ce problème en utilisant la notation **[10]**.

Notation

À la **Figure 1.1**, nous pouvons voir que le pointage nous aide à obtenir une meilleure routine de pointage. Nous verrons maintenant comment le pointage fonctionne **[10]**.

Par exemple :

Supposons qu'une université nommée « Exemple d'université » ne contienne qu'un seul département. Nommons-le Département d'informatique à l'Université Example. Devinez qu'il y a 4 (quatre) enseignants et 50 (cinquante) étudiants avec 10 (dix) cours.

Donc,

Département : 1

Enseignants : 4

Étudiants : 50

Cours : 10

Préférence des enseignants : élevée

Établissement des préférences de l'enseignant (p. ex., ordre fondé sur l'ancienneté) :

Enseignant, A = 4000

Enseignant, B = 300

Enseignant, C = 200

Enseignant, D = 100

Si nous rompons les promesses à l'enseignant A, nous obtiendrons 4000 points pour lui et si les autres enseignants sont bien, nous obtiendrons +600 (=300 + 200 + 100) pour tous. Nous traitons les préférences de l'enseignant comme ceci.

Alors qu'un étudiant suit plusieurs cours, si l'un de ses cours prévus en même temps à la routine, alors la routine nous donnera une grande valeur négative (10000). En attribuant ce type de valeurs négatives, nous générons une routine avec un score de pénalité total, puis nous le jetons à Tabu Window (**Figure 1.1**) en utilisant Tabu Search.

Chaque fois qu'il se compare à la routine stockée dans Tabu Window (TW) et si la routine actuelle est meilleure que toute autre routine dans Tabu Window (**Figure 1.1**), il stockera cela dans le dow Tabu Win et jettera la dernière routine de la fenêtre.

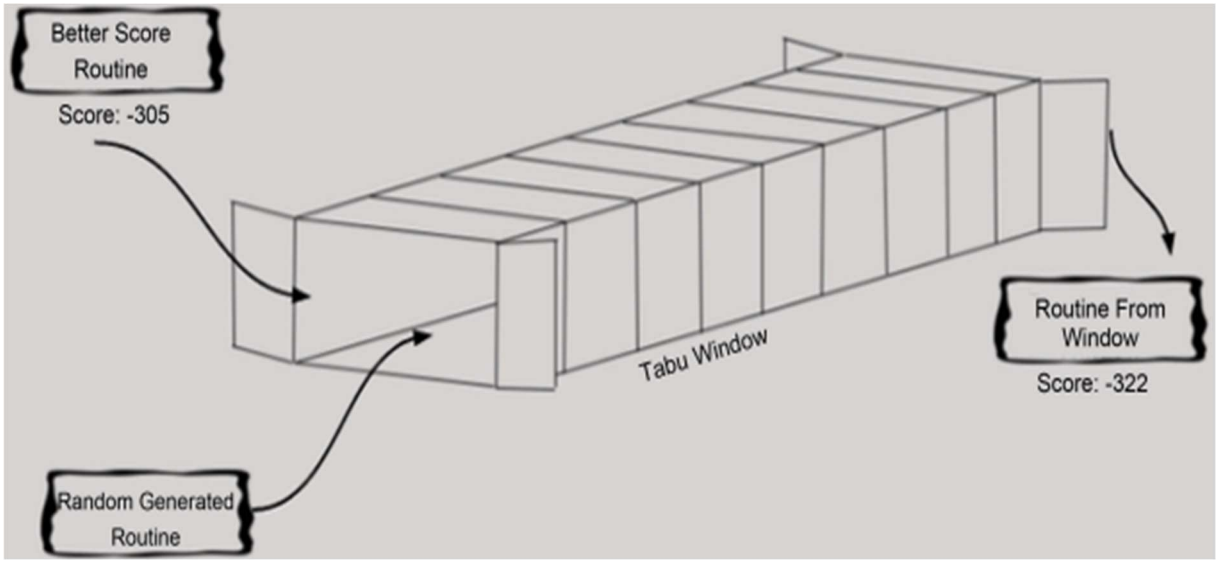


Figure 1. 1 **:** Tabu Window **[10]**.

5.5.2. Algorithme (RT)

```
Input : TabuList_size
Output : S_best
1.  S_best ← ConstructInitialSolution();
2.  TabuList ← φ;
3.  while ¬ StopCondition() do
4.      CandidateList ← φ;
5.      for S_candidate ∈ Sbest_neighbourhood do
6.          if ¬ ContainsAnyFeatures (S_candidate, TabuList) then;
7.              CandidateList ← S_candidate;
8.          end
9.      end
10.     S_candidate ← LocateBestCandidate(CandidateList);
11.     if Cost(S_candidate) ≤ Cost(S_best) then
12.         S_best ← S_candidate;
13.         TabuList ← FeatureDifferences(S_candidate, S_best);
14.         while TabuList > TabuList_size do
15.             DeleteFeature(TabuList);
16.         end
17.     end
18. end
19. S_best;
```

Figure 1. 2 **:** Pseudo-code pour Tabu Search **[9]**.

5.5.3. Positifs et négatifs

La création manuelle d'une routine semestrielle prend beaucoup de temps. Un coordonnateur ministériel investit toujours beaucoup de temps à cette fin.

Parfois, d'autres enseignants et agents de section travaillent avec un coordonnateur ministériel pour générer une routine semestrielle.

Un générateur d'horaires universitaires réduira le temps et les ressources. Les étudiants et les enseignants auront la meilleure routine possible pour un semestre particulier grâce à ce générateur.

Bien qu'il génère automatiquement une routine, il ne sera pas possible de conserver toutes les préférences en permanence.

Parfois, il peut arriver que certains enseignants ne reçoivent pas leur horaire selon leurs préférences. Si nous utilisons plusieurs campus pour un département particulier, alors les étudiants doivent voyager beaucoup et les enseignants aussi.

5.5.4. Analyse coûts-avantages

Le générateur d'horaires de l'Université sera un travail rentable. Cela permettra à nos coordonnateurs ministériels, agents de section, agents de programme et enseignants de gagner un temps précieux. Il réduira les conflits de classe et les problèmes de conflits d'examen.

6. Conclusion

Dans ce premier chapitre, nous avons parlé de notre problème de tous les côtés, puis nous avons mentionné nos principaux objectifs et tout sur le problème pour le résoudre.

Ensuite, nous avons détaillé les recherches les plus connues dans le domaine de l'ordonnancement des tâches dans les établissements universitaires, ainsi que quelques moyens et techniques pour résoudre le problème.

Dans le chapitre suivant, nous modéliserons ce problème par problème de satisfaction des contraintes (CSP).

CHAPITRE 02
Modélisation CSP

1. Introduction

L'élaboration d'un emploi du temps universitaire est un défi complexe qui nécessite la prise en compte de multiples contraintes et exigences. La modélisation de ce problème, souvent abordée à travers le prisme de la satisfaction des contraintes, vise à trouver une solution optimale qui concilie les besoins des étudiants, des enseignants et de l'institution dans son ensemble.

Pour modéliser le problème d'emploi de temps universitaire on a recouru aux modélisations de Problème de Satisfaction des Contraintes (CSP).

2. Modélisation

2.1. Définition

La modélisation est le processus de création d'une représentation simplifiée d'un système ou d'un phénomène du monde réel. Cette représentation peut prendre différentes formes, telles que des équations mathématiques, des diagrammes, des graphiques ou des simulations informatiques.

Dans le cas de la modélisation d'un emploi du temps universitaire, les variables représentent les différents cours et activités, tandis que les contraintes incluent des exigences telles que l'indisponibilité de certaines salles à certains moments, les préférences des enseignants pour les horaires, etc. L'objectif est de trouver une solution satisfaisante qui permette de maximiser l'utilisation des ressources disponibles tout en répondant aux besoins et aux préférences des différentes parties prenantes, comme les étudiants, les enseignants et l'administration.

L'objectif principal de la modélisation est de comprendre, analyser et prédire le comportement ou les résultats du système étudié, ce qui peut être utile dans de nombreux domaines, notamment la science, l'ingénierie, l'économie et la biologie.

2.2. Définition d'un modèle

Un modèle est une représentation abstraite et moins compliqué, d'une chose (system, processus, phénomène etc.) du monde réel en vue de lui donner une idée ou de l'expliquer **[15]**.

Un modèle donne le moyen de réduire la complexité d'un phénomène en écartant les détails qui n'influencent pas son comportement de manière significative. Il reflète ce que le

concepteur croit important pour la compréhension et la prédiction du phénomène modélisé. Les limites de ce dernier dépendent des objectifs du modèle **[15]**.

2.3. Tâche de modélisation

Modéliser un système avant son accomplissement permet d'une façon plus avantageuse de comprendre son fonctionnement **[16]**.

C'est aussi un bon moyen de réprimer sa complexité et d'assurer sa capacité, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer **[16]**.

Cette communication est essentielle pour aboutir à une compréhension commune aux différentes parties précise d'un problème donné **[16]**.

3. Le problème de satisfaction de contraintes CSP

3.1. Définition

Le problème de satisfaction des contraintes (CSP) est un concept en informatique et en intelligence artificielle qui consiste à trouver une solution satisfaisante à un ensemble de contraintes données.

Dans un CSP, un ensemble de variables doivent être assignées à des valeurs de manière à respecter un ensemble de contraintes qui spécifient les relations entre ces variables.

3.2. L'objectif de CSP

L'objectif général de la résolution de problèmes CSP est de trouver une solution satisfaisante qui respecte toutes les contraintes spécifiées.

Le but est donc de trouver une combinaison de valeurs pour les variables qui satisfasse toutes les contraintes, si une telle combinaison existe.

Dans le cas d'emploi du temps universitaire, l'objectif est de trouver une solution satisfaisante permettant de tirer le meilleur parti des ressources disponibles tout en respectant les besoins et les préférences des différents acteurs, tels que les élèves, les enseignants et la direction.

En résumé, l'objectif de la résolution de problèmes CSP est d'atteindre une solution qui respecte toutes les contraintes du problème donné.

3.3. La forme de CSP

Un CSP est un problème modélisé sous forme de contraintes posées sur des variables prenant valeur dans un domaine.

CSP est un triplet (X, D, C) où :

- X un ensemble $\{X_1, \ldots\ldots, X_k\}$ de variables.
- D un ensemble $\{D_1, \ldots\ldots, D_k\}$ de domaines : $X_i \in D_i$, $i = 1, \ldots, n$.
- C un ensemble $\{C_1, \ldots\ldots, C_t\}$ de contraintes où chaque C_i définit un sous-ensemble du produit cartésien des domaines des variables sur lesquelles elle porte :
  $C_i\ (X_{i1}, \ldots\ldots, X_{ik}) \subseteq D_{i1} \times \ldots \times D_{ik}$.

Exemple : modélisation de problème des 4 reines

Placer 4 reines sur échiquier 4 × 4 de telle sorte qu'aucune ne soit en prise (pas sur la même ligne, pas sur la même colonne, pas sur la même diagonale).

- $X = \{X_{11}, X_{12}, X_{13}, X_{14}, X_{21}, X_{22}, X_{23}, X_{24}, X_{31}, X_{32}, X_{33}, X_{34}, X_{41}, X_{42}, X_{43}, X_{44}\}$ ➔ Variables
- $D(X_{ij}) = \{0,1\}$ pour i Є {1,2,3,4} et j Є {1,2,3,4} ➔ Domaines
- $C = C_{lig}$ U C_{col} U C_{dm} U C_{dd} avec

- Il y a une reine par ligne
  $C_{lig} = \{X_{i1} + X_{i2} + X_{i3} + X_{i4} = 1 \ / \ i \in \{1,2,3,4\}\}$
- Il y a une reine par colonne
  $C_{col} = \{X_{1j} + X_{2j} + X_{3j} + X_{4j} = 1 \ / \ j \in \{1,2,3,4\}\}$
- Les reines doivent être sur des diagonales montantes différentes
  C_{dm} = pour tout couple de variables différentes X_{ij} et X_{kl}, i+j=k+l => $X_{ij} + X_{kl} \leq 1$.
  Exp : (X_{21} et X_{12}), (X_{31} et X_{22}), …
- Les reines doivent être sur des diagonales descendantes différentes
  C_{dd} = pour tout couple de variables différentes X_{ij} et X_{kl}, i-j=k-l => $X_{ij} + X_{kl} \leq 1$.
  Exp : (X_{11} et X_{22}) , (X_{11} et X_{33}) , (X_{21} et X_{32}) , …

- Une solution du problème des 4 reines, pour cette modélisation, est l'ensemble des affectations A.
  A = {(X_{11},0), (X_{21},1), (X_{31},0), (X_{41},0), (X_{12},0), (X_{22},0), (X_{32},0), (X_{42},1), (X_{13},1), (X_{23},0), (X_{33},0), (X_{43},0), (X_{14},0), (X_{24},0), (X_{34},1), (X_{44},0)}

3.4. Espace de recherche d'un CSP

Soit un CSP P = (X, D, C), L'espace de recherche est l'ensemble des affectations possibles que l'on notera S [17].

$$S = D_{x1} \times D_{x2} \times ... \times D_{xn} \quad (2.1)$$

L'espace de recherche est égal au produit cartésien de l'ensemble des domaines des variables et une affectation s sera assimilée à un élément de cet ensemble S (i.e. s = (d_1, ..., d_n)).

Une affectation s ∈ S satisfait une contrainte c_k ∈ C si toutes les variables de var (c_k) sont instanciées par s et si la relation définie par c_k est vérifiée pour les valeurs des variables de var(c_k) données par s. Pour simplifier, on notera $s \in c_k$.

3.5. Solution d'un CSP

Une solution d'un CSP P = (X, D, C) est une affectation s ∈ S qui satisfait toutes les contraintes. On note Sol(P) l'ensemble des solutions correspond à tout élément de l'espace de recherche (affectations) qui appartenant aux valeurs permises pour chaque contrainte [17].

$$Sol(P) = \{s \in S \mid \forall c \in C, s \in c\} \quad (2.2)$$

Exemple

Soit P = (X, D, C) un CSP tel que :

- X = {x, y, z}.
- D(x) = D(y) = {2, 3, 4, 5, 6}, D(z) = {8, 9, 10, 11, 12}.
- C = {(c_1 : x * 3 = z ∧ y * 2 = z)}.

- A_1 = {< x, 2 >, < y, 2 >, < z, 8 >} est une affectation totale et inconsistante.
  L'affectation A_1 viole la contrainte c1 : (x * 3 = z ∧ y * 2 = z).
- A_2 = {< x, 3 >, < y, 4 >, < z, 9 >} est une affectation totale et inconsistante.
  L'affectation A_2 viole la contrainte c1 : (x * 3 = z ∧ y * 2 = z).
- A_3 = {< x, 3 >, < y, 4 >, < z, 9 >} est une affectation totale et inconsistante.
  L'affectation A_3 viole la contrainte c1 : (x * 3 = z ∧ y * 2 = z).
- A_4 = {< x, 4 >, < y, 6 >, < z, 12 >} est une affectation totale et consistante.
  Donc, A_4 est une solution.

3.6. Résolution d'un problème de satisfaction de contraintes

La résolution d'un problème de satisfaction de contraintes (CSP) consiste à trouver une assignation de valeurs à un ensemble de variables qui respecte toutes les contraintes imposées.

La résolution est assurée par des méthodes de base qui peuvent être envisagées selon deux étapes, à savoir : des techniques de propagation de contraintes (ou de filtrage), les algorithmes de recherche.

3.6.1. Filtrage de la résolution

Les méthodes de filtrage permettent de réduire la taille du problème courant en supprimant les valeurs devenues inconsistantes dans les domaines des variables correspondants et ainsi réduire l'espace de recherche pour améliorer la performance des algorithmes de résolution [18].

L'objectif principal des techniques de filtrage est la détection précoce d'instanciations partielles localement ou totalement incohérentes. Une des techniques les plus utilisées est la technique de renforcement de la consistance locale ou consistance d'arc [19].

Prenant l'exemple ci-avant.

Le graphe de ce CSP avant le filtrage est illustré dans la Figure 2.1 :

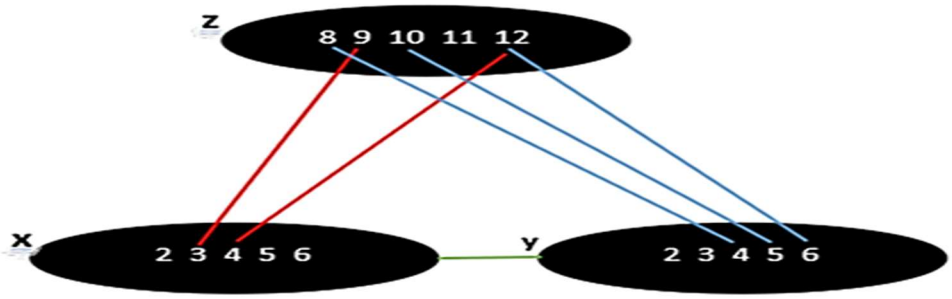


Figure 2. 1 **:** Graphe de CSP avant le filtrage

Le résultat de filtrage par consistance d'arc de CSP doit représenter dans la Figure 2.2.

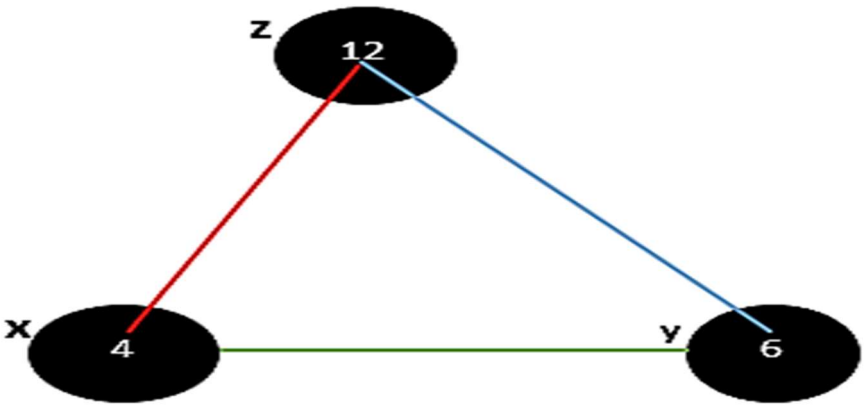


Figure 2. 2 : Graphe de CSP après le filtrage

3.7. Algorithmes de résolution des CSP

Les algorithmes de résolution des CSP visent à explorer l'espace des solutions de manière systématique et efficace afin de trouver une combinaison d'affectations de valeurs qui satisfait toutes les contraintes imposées.

Nous expliquerons certains des algorithmes suivants :

3.7.1. L'algorithme « AC-1 »

Arc consistance-1 est une méthode de propagation de contraintes utilisée dans la résolution des problèmes de satisfaction de contraintes.

Son objectif principal est d'établir l'arc-consistance dans un réseau de contraintes en éliminant de manière itérative les valeurs incompatibles des domaines des variables.

Concrètement, l'algorithme AC-1 examine chaque paire de variables et leurs contraintes associées, et il élimine les valeurs des domaines des variables qui ne satisfont pas les contraintes.

Cette procédure est répétée de manière itérative jusqu'à ce qu'aucune modification supplémentaire ne puisse être effectuée.

L'algorithme AC-1 est relativement simple et efficace pour éliminer les valeurs incompatibles des domaines des variables, ce qui réduit l'espace de recherche et facilite la résolution des CSP.

Cependant, il peut ne pas toujours garantir l'arc-consistance globale dans des cas complexes, et des variantes plus sophistiquées comme AC-3 ou AC-4 peuvent être nécessaires dans de tels cas.

```
Début
    Q ← (i, j)      / {(i, j) est un arc et i <> j}
Répéter
        CHANGE ← Faux
Pour tout (i, j) de Q Faire
        CHANGE ← Révise (i, j) OU CHANGE
Fin Pour
Jusqu'à NON CHANGE
Fin.
```

```
Fonction Révise (i, j)
Début
    SUPPRESSION ← Faux
Pour chaque valeur v (i) de dom (i) Faire
S'il n'existe pas de valeur v (j) dans dom (j) support de v (i) Alors
        Supprimer v (i) de dom (i)
        SUPPRESSION ← Vrai
Fin Si
Fin Pour
 Retourner SUPPRESSION
Fin
```

Figure 2. 3 **:** L'algorithme AC-1 [20].

3.7.2. L'algorithme « AC-3 »

Arc consistance 3 est une méthode de propagation de contraintes utilisée dans la résolution des problèmes de satisfaction de contraintes (CSP).

Son objectif est d'établir l'arc-consistance dans un réseau de contraintes en éliminant de manière itérative les valeurs incompatibles des domaines des variables.

Concrètement, l'algorithme AC-3 examine les arcs (paires de variables) du réseau de contraintes et utilise une technique appelée « recherche de conflits » pour détecter et éliminer les valeurs incompatibles.

Lorsqu'une valeur est éliminée du domaine d'une variable, l'algorithme examine toutes les contraintes qui impliquent cette variable et utilise la recherche de conflits pour vérifier si d'autres valeurs doivent également être éliminées.

L'algorithme AC-3 est plus puissant que AC-1 car il peut détecter et éliminer plus de valeurs incompatibles dans un réseau de contraintes donné.

Cependant, il peut être plus coûteux en termes de temps de calcul, surtout dans des réseaux de contraintes complexes.

Malgré cela, AC-3 est largement utilisé dans la résolution de CSP en raison de son efficacité et de sa capacité à améliorer la propagation de contraintes.

```
Début
   Q ← (i, j)                    / {(i, j) est un arc et i <> j}
Tant que Q est non vide Faire
     Extraire un arc (k, m) de Q
Si Révise (k, m) Alors
        Q ← Q U (i, k)          / {(i, k) est un arc et i <> k et i <> m}
Fin Si
Fin Tant que
 Fin.
```

Figure 2. 4 : L'algorithme AC-3 [20].

3.7.3. L'algorithme « Génère Et Teste »

L'algorithme "Génère et Teste" est une méthode de résolution simple utilisée dans la recherche de solutions pour certains types de problèmes, y compris les problèmes de satisfaction de contraintes (CSP).

L'approche de cet algorithme consiste en deux étapes principales :

- **Génération :** Dans cette étape, l'algorithme génère toutes les combinaisons possibles d'affectations de valeurs aux variables du problème.
- **Test :** Après avoir généré une affectation de valeurs, l'algorithme teste si cette affectation satisfait toutes les contraintes du problème. Si c'est le cas, l'affectation est considérée comme une solution valide. Sinon, elle est rejetée.

L'algorithme GET est simple à comprendre et à mettre en œuvre, mais il peut être inefficace pour les problèmes de grande taille en raison du nombre exponentiel de combinaisons à explorer.

De plus, il peut ne pas être adapté aux problèmes où il existe des contraintes locales qui permettent d'éliminer certaines combinaisons dès le stade de génération, avant même de les tester.

```
Fonction GenereEtTeste (A, (X, D, C)) retourne un booléen
(X, D, C) : un CSP sur les domaines finis
A : une affectation partielle pour (X, D, C)
si l'affectation partielle A peut être étendue en une solution pour (X, D, C)
        Retourner vraie sinon faux
Début
Si toutes les variables de X sont affectées à une valeur dans A alors
          /* A est une affectation totale */
          Si A est consistante Alors
          /* A est une solution */
               Retourner vrai
          Sinon
                Retourner faux
          Fin Si
     Sinon  /* A est une affectation partielle */
          Choisir une variable Xi de X qui n'est pas encore affectée à une valeur dans A
Pour toute valeur Vi appartenant à D (Xi) Faire
       Si GenereEtTeste (A U {(Xi, Vi)}, (X, D, C)) = vrai Alors
Retourner vrai
     Fin Pour
    Retourner faux
 Fin Si
   Fin.
```

Figure 2. 5 **:** L'algorithme GET [21].

3.7.4. L’algorithme Simple Retour Arrière

L’algorithme de recherche de base de résolution des problèmes CSPs est l’algorithme de simple retour arrière (SRA) ou « backtracking » chronologique [22].

L'algorithme "SRA" est une technique de recherche exhaustive utilisée pour résoudre les problèmes de satisfaction de contraintes (CSP) et d'autres problèmes de recherche de solutions.

L'idée principale de cet algorithme est d'explorer toutes les possibilités en avançant progressivement dans une branche de l'arbre de recherche, et en revenant en arrière (d'où le terme "backtracking") lorsque les conditions ne sont pas satisfaites pour explorer d'autres branches.

L'algorithme « SRA » est souvent utilisé pour résoudre des problèmes où les contraintes sont difficiles à modéliser de manière efficace pour d'autres méthodes, ou lorsque des contraintes globales sont en jeu.

Bien qu'il puisse être coûteux en termes de temps de calcul pour certains problèmes de grande taille, il est souvent efficace pour les problèmes de petite à moyenne taille.

Fonction BT (A, V) **retourne** un booléen
Début
Si $V = \emptyset$ **alors** **Retourner** vrai
Sinon
Choisir $x \in V$
$d \leftarrow d_x$
Consistance ← **faux**
Tant que $d \neq \emptyset$ **et non** consistance **Faire**
Choisir v dans d
$d \leftarrow d \backslash \{v\}$
Si $\exists c \in C$ telle que c viole $A \cup \{x \leftarrow v\}$ **Alors**
Consistance ← BT ($A \cup \{x \leftarrow v\}, V \backslash \{x\}$)
Fin Si
Fin Tant que
Retourner Consistance
Fin Si
Fin.

Figure 2. 6 **:** L'algorithme BT [23].

4. Exemples de modèles CSP d'un PETU

On va présenter ici deux modèle CSP pour le problème PETU :

4.1. Modèle 1

Selon **[24],** La formulation du problème PETU est la suivante :

- Problème= {Enseignants, Salles, Cours, Horaires, Contraintes}.
- Enseignants= {E_1, ……E_i, …..E_e} est l'ensemble des e enseignants.
- Salles= {S_1, ..…S_j, …..S_s} est l'ensemble des s salles.
- Cours= {C_1, ..…C_l, ……C_c} est l'ensemble des c cours.
- Horaires= {H_1, ..….H_k, …..H_h} est l'ensemble des h horaires.

Les contraintes :

- Un enseignant E_i peut donner au plus un cours C_l pendant l'horaire H_k dans une salle S_j:

$$\sum_{S_j} \sum_{C_l} \times \left[E_i , S_j, C_l, H_k \right] \leq 1 \; ; \; \forall E_i , \forall H_k \qquad (2.3)$$

- Une salle S_j peut contenir au plus un cours C_l donné par un seul enseignant pendant l'horaire H_k :

$$\sum_{S_j} \sum_{C_l} \sum_{H_k} \times \left[E_i , S_j, C_l, H_k \right] \leq 1 \; ; \; \forall S_j \qquad (2.4)$$

- Un cours C_l peut être donné au plus par un unique enseignant E_i dans une salle S_j à un certain horaire H_k :

$$\times \left[E_i , S_j, C_l, H_k \right] = 1 \; ; \; \forall C_l \qquad (2.5)$$

4.2. Modèle 2

Le problème consiste à programmer un ensemble de cours (conférences et tutoriels) dans différents créneaux horaires sous réserve de satisfaire à la contrainte suivante : aucun étudiant ne participe à plus d'un cours en même temps, la salle doit être assez grande pour tous les étudiants présents, aucun sujet de base n'est programmé en même temps, et une seule classe est programmée dans une pièce à un créneau horaire donné **[1]**.

- $S = \{s_1, s_2, \ldots, s_n\}$ soit l'ensemble des étudiants.
- $L = \{l_1, l_2, \ldots, l_o\}$ soit l'ensemble des matières enseignées.
- $T = \{t_1, t_2, \ldots, t_m\}$ soit les périodes d'enseignement disponibles.
- $R = \{r_1, r_2, \ldots, r_p\}$ soit l'ensemble des salles disponibles.

Où

- S_l est l'ensemble des étudiants qui suivent le sujet *l*.
- T_l est l'ensemble des plages horaires attribués au sujet *l*.
- R_l est l'ensemble des salles attribuées au sujet *l*.

Alors

- T_{li} est le nombre de créneaux d'enseignement pour les matières l_i.
- S_{li} est l'ensemble des l'étudiant pouvant d'assister au sujet l_i
- R_{li} est l'ensemble des salles qui peuvent être attribuées au sujet l_i.

L'emploi du temps réalisable est un calendrier dans lequel tous les événements ont été attribués un créneau horaire et une salle afin que les contraintes suivantes soient satisfaites :

- Aucun étudiant ne fréquente plus d'une matière à la même heure :

$$\forall(S_{li} = S_{lj}) \ \exists \ T_{li} \neq T_{lj} \qquad (2.6)$$

Où $S_{li} = S_{lj}$ représente l'étudiant qui prend deux matières l_i et l_j, ces deux matières ne peuvent pas être tenues en même temps.

- Dans chaque salle, il n'y a qu'une matière disponible à tout moment.

$$\forall(l_i = l_j) \ \exists \ T_{li} \neq T_{lj} \qquad (2.7)$$

Où $T_{li} \neq T_{lj}$ représente le créneau alloué aux matières l_i et l_j.

Lorsque les deux matières l_i et l_j sont affectés à la même salle, ils doivent être affectés à différents créneaux horaires.

- La taille de la salle R_j « R_{Size} (R_j) » est satisfaite pour toutes les fonctionnalités requises par les sujets R_{li} et est assez grand pour tous les tailles des étudiants présents S_{Size} (S_i).

$$\forall(R_{li} = R_j) \ \exists \ R_{Size}(R_j) > S_{Size}(S_i) \quad (2.8)$$

Où S_{Size} (S_{li}) représente la taille des étudiants qui fréquentent la matière li.

- Certains créneaux horaires ont été réservés pour des événements spéciaux E. Par conséquent, ces créneaux horaires ne doivent pas être attribués aux matières.

  $$\forall(T_j = E) \ \exists \ T_l \neq T_j \quad (2.9)$$

  Où $T_l \neq T_j$ signifie que l'ensemble des créneaux horaires attribué au sujet l ne peut pas être égal au créneau horaire spécial T_j lorsque T_j est égal à l'événement spécial E.

5. Modélisation CSP du problème étudié

5.1. Variables

- Variables des matières : Ce sont toutes des matières étudiées dans le département.
  M = {m_1, ……., m_n}
- Variables types des leçons : Tl = {C, TD, TP}
  C: Cours, TD: Travaux Diriges, TP: Travaux Pratique
- Variables des salles : Ce sont toutes les salles de département.
  S = {s_1, ……, s_o}
- Variables d'enseignants : tous les enseignants de département
  E = {e_1, ……, e_p}
- Variables des groupes d'étudiants : Tous les groupes d'étudiants dans le département et chaque section ou sous-groupes sont considérés comme un groupe d'étudiants parce que nous ferons le timing pour chacun de ces éléments seuls.
  G = {g_1, ……, g_q}
- Variables des créneaux horaires : le numéro de créneaux par semaine
  H = {h_1, ……, h_r}
- Variables des jours : les jours où les étudiants peuvent étudier. J

5.2. Domaines

- Supposons que : Le nombre de créneaux par jour est : 6 et le nombre de jours par semaine est : 5 alors le nombre de créneaux par semaine est : 6 * 5 = 30.
- Toutes les variables ont le même domaine parce que chaque variable prend un certain temps ou plus, sauf les variables jours est le domaine en nombres naturels 0 à 4.
- Domaine de la variable jours est un ensemble de jours de la semaine et les a représentés en nombres naturels 0 à 4.

- L'ensemble de périodes (créneaux) dans la semaine, y compris les périodes de nourriture, est représenté par un en nombres naturels 0 à 29, voir tableau 2.1 .
- D (matières) = D (salles) = D (enseignants) = D (groupes d'étudiants) = {0, 1, 2,…, 29}.
- D (jours) = {0, 1, 2, 3, 4}.

	Dimanche « 0 »	**Lundi « 1 »**	**Mardi « 2 »**	**Mercredi « 3 »**	**Jeudi « 4 »**
Créneau 1	**0**	**6**	**.**	**.**	**.**
Créneau 2	**1**	**.**	**.**	**.**	**.**
Créneau 3	**2**	**.**	**.**	**.**	**.**
Créneau 4	**3**	**.**	**.**	**.**	**.**
Créneau 5	**4**	**.**	**.**	**.**	**.**
Créneau 6	**5**	**.**	**.**	**.**	**29**

Tableau 2. 1 **:** Tableau montrant les domaines des variables.

5.3. Contraintes

- **C1:** Un enseignant e_i peut donner au plus une matière m_j au groupe g_t pendant l'horaire h_k dans une salle s_n:

$$\forall e \in E, \exists g \in G, \exists m \in M, \exists s \in S: \ \times [e, g, m, s] = h^4 \mid h = 0..29 \qquad (2.10)$$

Exemple :

- Un enseignant e_1 peut donner au plus une matière m_2 au groupe g_3 pendant l'horaire h_7 dans une salle s_2: (e_1 / m_2 / g_3 / s_2)

$$(e \times g \times m \times s) = 7^4 \mid h = 7$$

	Dimanche « 0 »	**Lundi « 1 »**	**Mardi « 2 »**	**Mercredi « 3 »**	**Jeudi « 4 »**
Créneau 1	**X**	**X**	**X**	**X**	**X**
Créneau 2	**X**	**e_1 / m_2 / g_3 / s_2**	**X**	**X**	**X**
Créneau 3	**X**	**X**	**X**	**X**	**X**
Créneau 4	**X**	**X**	**X**	**X**	**X**
Créneau 5	**X**	**X**	**X**	**X**	**X**
Créneau 6	**X**	**X**	**X**	**X**	**X**

Tableau 2. 2 **:** Tableau montrant la contrainte C1.

- **C2 :** un enseignant peut enseigner une seule matière à la même heure :

$$\forall\left(e_i = e_j\ , e_i = m_t = h_1, e_j = m_k = h_2\ \right)\ \exists\ h_1 \neq h_2 \qquad (2.11)$$

Où $e_i = e_j\ , e_i = m_t = h_1, e_j = m_k = h_2$ représente que l'enseignant qui prend deux matières m_t et m_k, ces deux matières ne peuvent pas être tenues en même temps.

- **C2 :** Aucun étudiant ne fréquente plus d'une matière à la même heure :

$$\forall\left(g_i = g_j\ , g_i = m_t = h_1, g_j = m_k = h_2\ \right)\ \exists\ h_1 \neq h_2 \qquad (2.12)$$

Où $g_i = g_j\ , g_i = m_t = h_1, g_j = m_k = h_2$ représente que l'étudiant qui prend deux matières m_t et m_k, ces deux matières ne peuvent pas être tenues en même temps.

- **C3 :**Dans chaque salle, il n'y a qu'une matière disponible à tout moment.

$$\forall\left(m_i = m_j, m_i = s_k = h_1\ , m_j = s_t = h_2\ \right)\ \exists\ h_1 \neq h_2 \qquad (2.13)$$

Où $h_1 \neq h_2$ représente le créneau alloué aux matières m_i et m_j.

- **C4 :**Certains créneaux horaires ont été réservés pour des événements spéciaux E. Par conséquent, ces créneaux horaires ne doivent pas être attribués aux matières.

$$\forall\left(h_j = E\right)\ \exists\ (m_l = h_i \neq h_j) \qquad (2.14)$$

Où $(m_l = h_i \neq h_j$ signifie que l'ensemble des créneaux horaires attribué au matière m_l ne peut pas être égal au créneau horaire spécial h_j lorsque h_j est égal à l'événement spécial E (Example : Nourriture (12 :30 – 14 :00).

6. Conclusion

Dans ce deuxième chapitre, nous avons parlé de modélisation et du problème de la satisfaction des contraintes à tous égards, puis nous avons mentionné nos principaux objectifs et nous avons parlé d'algorithmes pour résoudre ce problème.

Ensuite, nous avons modélisé le problème de PETU par CSP et mentionné certains modèles aux chercheurs avant.

Dans le chapitre suivant, nous allons simuler cette modélisation en programme PROLOG qui fait la génération automatique de l'emploi du temps universitaire.

CHAPITRE 03
Implémentation

1. Introduction

Pour implémenter notre modèle CSP on aura recouru de langage Prolog. Vu sa syntaxe déclarative et son moteur d'inférence basé sur la logique, il est particulièrement bien adapté pour modéliser et résoudre des problèmes où il s'agit de trouver des solutions satisfaisant un ensemble de contraintes complexes.

2. Environnement de travail

2.1. Langage PROLOG

Prolog est un langage de programmation logique créé par Alain Colmerauer et Robert Kowalski en 1972. L'idée derrière Prolog est d'essayer d'exprimer une tâche dans un langage similaire à la logique du premier ordre **[25]**.

Les systèmes Prolog incluent l'unification et la non-détermination comme concepts clés sur lesquels nous construisons des programmes.

Un programme Prolog est composé de prédicats qui définissent une relation entre ses arguments. Un prédicat est fait à partir de clauses. Un article peut être un fait ou une règle.

C'est encore aujourd'hui l'un des meilleurs exemples et l'un des langages les plus populaires dans le domaine de la programmation logique. C'est parce que Prolog nous permet de résoudre avec élégance de nombreuses tâches avec des programmes courts et généraux.

2.1. Logiciel (Scryer Prolog)

Scryer prolog est une implémentation moderne de Prolog qui vise à être conforme à la norme ISO Prolog tout en intégrant également plusieurs fonctionnalités contemporaines pour prendre en charge les besoins de programmation avancés.

Scryer prolog est un logiciel libre destiné à être un environnement de production de force industrielle et un banc d'essai pour la recherche de pointe dans la logique et la programmation par contrainte [25].

3. Etude de cas : "Département d'Informatique- Université de M'sila"

Dans cette étude, nous avons utilisé les masters 1 et 2 de notre département comme exemple pour présenter les résultats :

- **Master 1 :** Nous avons quatre spécialités

- M1 Informatique Décisionnelle et Optimisation (IDO).

- M1 Réseaux et Technologies de l'Information et de la Communication (RTIC).
- M1 Système d'Information et Génie Logiciel (SIGL).
- M1 Intelligence artificielle (IA).

- **Master 2 :** Nous avons quatre spécialités

- M2 Informatique Décisionnelle et Optimisation (IDO).
- M2 Réseaux et Technologies de l'Information et de la Communication (RTIC).
- M2 Système d'Information et Génie Logiciel (SIGL).
- M2 Intelligence artificielle (IA).

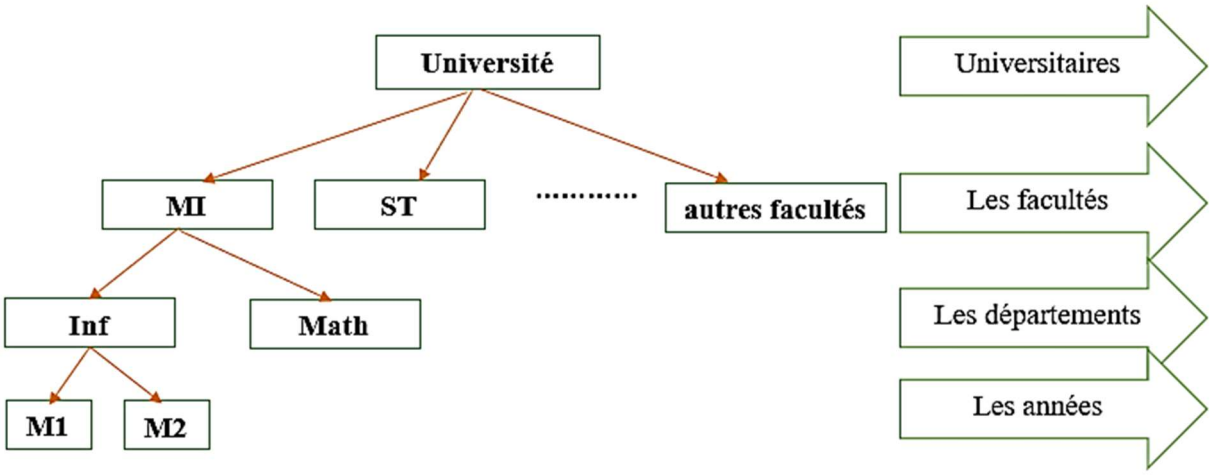


Figure 3. 1 : Hiérarchie de spécialités

3.1. Ressources dans exemple

En raison de l'énorme quantité de ressources qu'on étudie dans cet exemple (matières, enseignants, groupe d'étudiants et salles), et avec toutes ces complexités, on prend le défi pour déterminer le calendrier de chaque ressource sans conflit.

Après une étude approfondie du problème de l'ordonnancement, nous avons constaté que la première étape consiste à identifier toutes les ressources pour le traitement automatique de ces ressources, et pour ce faire, on a défini et expliqué ces ressources dans l'exemple réfléchi :

3.1.1. Les enseignants

Dans cet exemple, on a recouru un groupe d'enseignants qui enseigne dans ce département, en particulier les années étudiées dans notre cas, afin que chaque enseignant qui enseigne sa propre matière et un groupe d'étudiants.

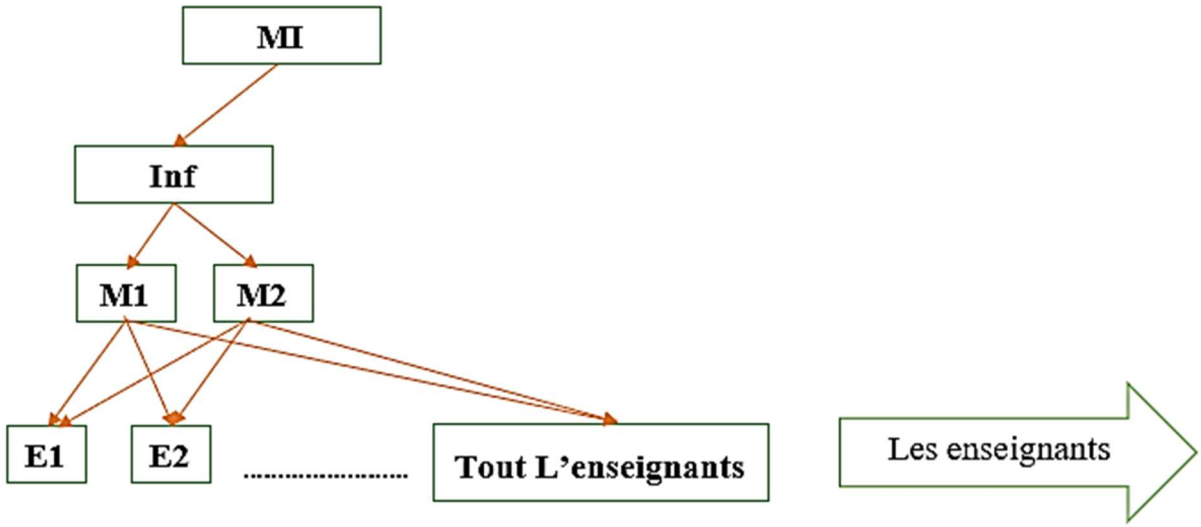


Figure 3. 2 : Hiérarchie des enseignants

3.1.2. Groupes d'étudiants

Dans cette étude, on ne se soucie pas de la spécialisation, de la section ou du sous-groupe, ce qui nous intéresse, c'est qu'il y a un groupe d'étudiants qui étudient dans une salle et étudient une matière et qui sont enseignés par un enseignant particulier.

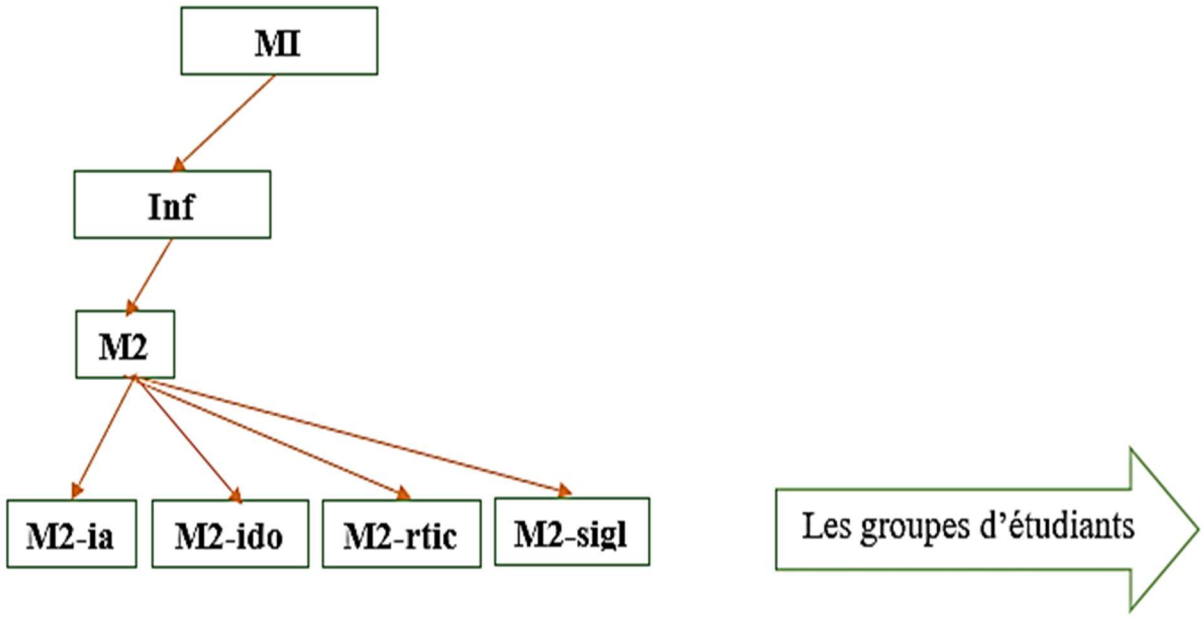


Figure 3. 3 : Hiérarchie des groupes d'étudiants master 2

3.1.3. Les salles

Dans cette étude, on a trois types de salles où les étudiants y étudient selon le type de leçon (C, TD, TP).

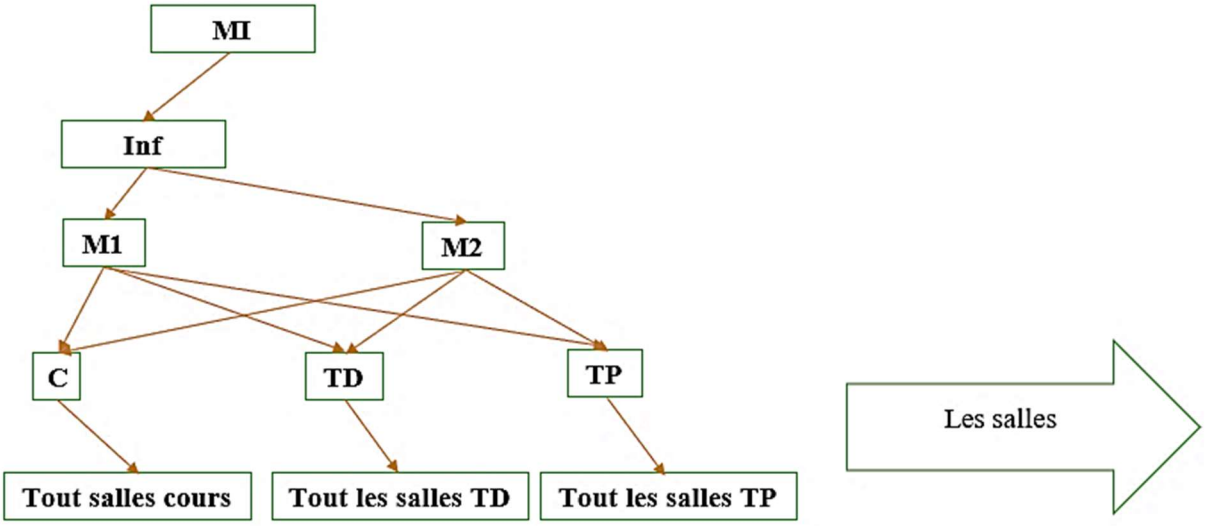


Figure 3. 4 : Hiérarchie des salles

3.1.4. Les matières

Chaque groupe d'étudiants étudie ses propres matières.

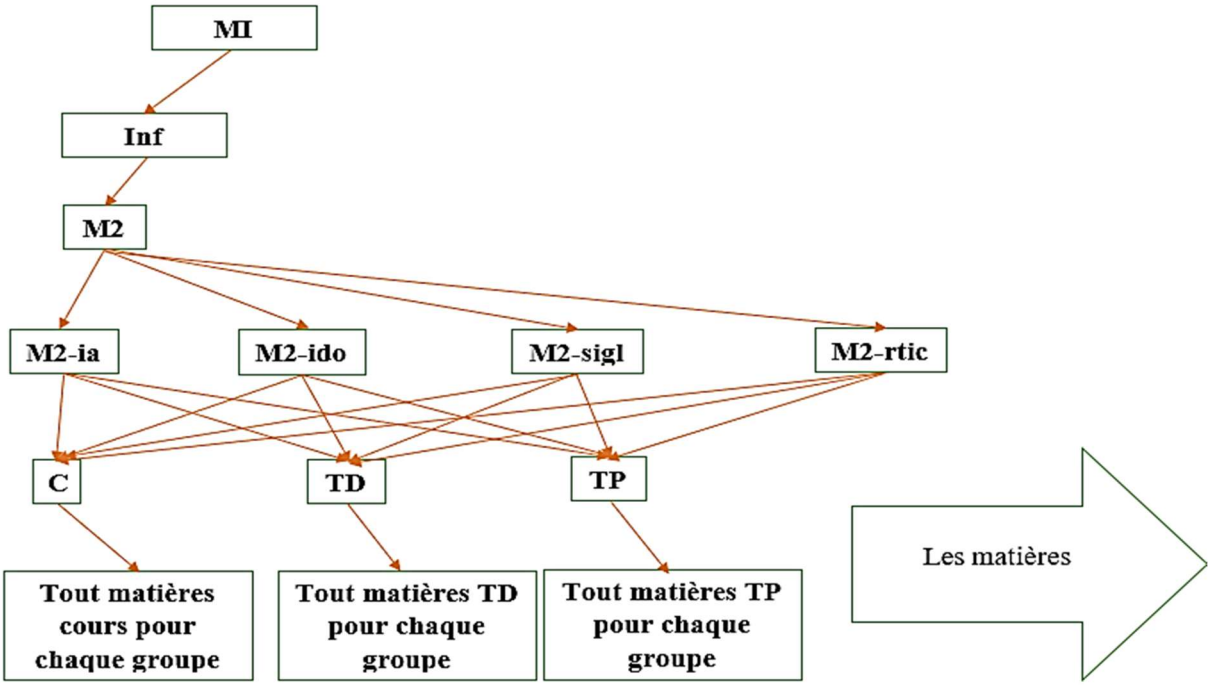


Figure 3. 5 : Hiérarchie des matières master 2

Enfin, on a mentionné et expliqué toutes les ressources et créé la hiérarchie de toutes les ressources qu'on a utilisé dans notre application pour générer un emploi du temps du département.

4. Base de connaissances

On explique les prédicats dans notre base de connaissances qui nous aident à construire ou à créer un emploi du temps efficace et satisfaisant pour toutes les restrictions étudiées.

En logique, un prédicat est une expression qui affirme quelque chose à propos d'un sujet. Un prédicat peut être vrai ou faux en fonction des valeurs des variables auxquelles il s'applique. Par exemple, dans l'expression "x est un nombre premier", " premier" est le prédicat, et "x" est la variable. Premier(X)

Ce qui suit tous les prédicats utilisés:

4.1. g_Etud/1.

On présente par ce prédicat tous les groupes d'étudiants qu'on a, comme le montre la Figure 3.6.

```
% ------------ Groupe_Etudiants -----------------
g_Etud('M1-Ia&Sigl').
g_Etud('M1-Ia&Rtic').
g_Etud('M1-ia').
g_Etud('M1-ia-sg1').
g_Etud('M1-ia-sg2').
g_Etud('M1-ido').
g_Etud('M1-ido-sg1').
g_Etud('M1-ido-sg2').
g_Etud('M1-sigl').
g_Etud('M1-sigl-sg1').
g_Etud('M1-sigl-sg2').
g_Etud('M1-rtic').
g_Etud('M1-rtic-sg1').
g_Etud('M1-rtic-sg2').
g_Etud('M2-Ido&Sigl&Rtic').
g_Etud('M2-Sigl&Rtic').
g_Etud('M2-ia').
g_Etud('M2-ido').
g_Etud('M2-sigl').
g_Etud('M2-rtic').
```

Figure 3. 6 : Extrait de Base de connaissance – prédicat « **g_Etud** » -

4.2. g_Appartient/2.

On présente par ce prédicat tous les groupes d'étudiants qui appartiennent à l'autre afin que deux groupes d'étudiants du même ensemble n'étudient pas en même temps comme le montre la Figure 3.7

```
% ------------ Groupe_Etudiants_Appartient -----------------
g_Appartient('M1-Ia&Sigl', 'M1-ia').
g_Appartient('M1-Ia&Sigl', 'M1-ia-sg1').
g_Appartient('M1-Ia&Sigl', 'M1-ia-sg2').
g_Appartient('M1-Ia&Sigl', 'M1-sigl').
g_Appartient('M1-Ia&Sigl', 'M1-sigl-sg1').
g_Appartient('M1-Ia&Sigl', 'M1-sigl-sg2').
g_Appartient('M1-Ia&Rtic', 'M1-ia').
g_Appartient('M1-Ia&Rtic', 'M1-ia-sg1').
g_Appartient('M1-Ia&Rtic', 'M1-ia-sg2').
g_Appartient('M1-Ia&Rtic', 'M1-rtic').
g_Appartient('M1-Ia&Rtic', 'M1-rtic-sg1').
g_Appartient('M1-Ia&Rtic', 'M1-rtic-sg2').
g_Appartient('M1-ia', 'M1-ia-sg1').
g_Appartient('M1-ia', 'M1-ia-sg2').
g_Appartient('M1-ido', 'M1-ido-sg1').
g_Appartient('M1-ido', 'M1-ido-sg2').
g_Appartient('M1-sigl', 'M1-sigl-sg1').
g_Appartient('M1-sigl', 'M1-sigl-sg2').
g_Appartient('M1-rtic', 'M1-rtic-sg1').
g_Appartient('M1-rtic', 'M1-rtic-sg2').
g_Appartient('M2-Ido&Sigl&Rtic', 'M2-ido').
g_Appartient('M2-Ido&Sigl&Rtic', 'M2-sigl').
g_Appartient('M2-Ido&Sigl&Rtic', 'M2-rtic').
g_Appartient('M2-Sigl&Rtic', 'M2-sigl').
g_Appartient('M2-Sigl&Rtic', 'M2-rtic').
```

Figure 3. 7 : Extrait de Base de connaissance – prédicat « g_Appartient » -

4.3. enseignant/1.

On présente par ce prédicat tous les enseignants qu'on a, comme le montre la Figure 3.8.

```
% ------------ Enseignant -----------------
enseignant('E1').
enseignant('E2').
enseignant('E3').
enseignant('E4').
enseignant('E5').
enseignant('E6').
enseignant('E7').
enseignant('E8').
enseignant('E9').
enseignant('E10').
enseignant('E11').
enseignant('E12').
enseignant('E13').
enseignant('E14').
enseignant('E15').
enseignant('E16').
enseignant('E17').
enseignant('E18').
enseignant('E19').
enseignant('E20').
enseignant('E21').
enseignant('E22').
enseignant('E23').
enseignant('E24').
enseignant('E24').
enseignant('E26').
enseignant('E27').
enseignant('E28').
```

Figure 3. 8 : Extrait de Base de connaissance – prédicat « enseignant » -

4.4. salle/1.

On présente par ce prédicat tous les salles qu'on a, comme le montre la Figure 3.9.

```
% ------------ Salles -----------------
salle(mi04).
salle(mi03).
salle(mm01).
salle(mm02).
salle(mm03).
salle(mm05).
salle(s01).
salle(s02).
salle(s03).
salle(s04).
salle(s05).
salle(s06).
salle(s07).
salle(lab01).
salle(lab02).
salle(lab03).
salle(lab04).
salle(lab13).
salle(lab14).
salle(lab15).
salle(lab16).
salle(lab17).
salle(lab18).
salle(lab19).
salle(lab20).
```

Figure 3. 9 : Extrait de Base de connaissance – prédicat « salle » -

4.5. type_salle/2.

On présente par ce prédicat toutes types de salles qu'on a, selon le type de leçons (C, TD, TP) enseigné dans ces salles, comme le montre la Figure 3.10.

```
% ------------ TypeSalles -----------------
% ------------ SallesCour -------------
type_salle(mi04, c).
type_salle(mi03, c).
type_salle(mm01, c).
type_salle(mm02, c).
type_salle(mm03, c).
type_salle(mm05, c).
% ------------ SallesTd -------------
type_salle(s01, td).
type_salle(s02, td).
type_salle(s03, td).
type_salle(s04, td).
type_salle(s05, td).
type_salle(s06, td).
type_salle(s07, td).
% ------------ SallesTp -------------
type_salle(lab01, tp).
type_salle(lab02, tp).
type_salle(lab03, tp).
type_salle(lab04, tp).
type_salle(lab13, tp).
type_salle(lab14, tp).
type_salle(lab15, tp).
type_salle(lab16, tp).
type_salle(lab17, tp).
type_salle(lab18, tp).
type_salle(lab19, tp).
type_salle(lab20, tp).
```

Figure 3. 10 : Extrait de Base de connaissance – prédicat « type_salle» -

4.6. g_etud_matière/2.

Par ce prédicat, on affecte les matières à leur propre groupe d'étudiants, comme le montre la Figure 3.11.

```
% ------------ M1-ia -------------
g_etud_matière('M1-ia', algoAv).
g_etud_matière('M1-ia', reprConai).
g_etud_matière('M1-ia', bdda).
g_etud_matière('M1-ia', bigData).
g_etud_matière('M1-ia', techAgent).
g_etud_matière('M1-ia', priETapp).
g_etud_matière('M1-ia', ang1).
```

Figure 3. 11 : Extrait de Base de connaissance – prédicat « g_etud_matière » -

4.7. g_etud_enseignant/2.

Par ce prédicat, on détermine les enseignants de chaque groupe d'étudiants enseigné, comme le montre la Figure 3.12.

```
g_etud_enseignant('M1-ia', 'E1').
g_etud_enseignant('M1-ia', 'E3').
g_etud_enseignant('M1-ia', 'E2').
g_etud_enseignant('M1-ia', 'E4').
g_etud_enseignant('M1-ia', 'E5').
g_etud_enseignant('M1-ia', 'E6').
g_etud_enseignant('M1-ia', 'E7').
```

Figure 3. 12 : Extrait de Base de connaissance – prédicat « g_etud_enseignant » -

4.8. matière_enseignant/2.

Par ce prédicat, on détermine les matières de spécialité de chaque enseignant, comme le montre la Figure 3.13.

```
matière_enseignant(algoAv, 'E1').
matière_enseignant(reprConai, 'E3').
matière_enseignant(bdda, 'E2').
matière_enseignant(bigData, 'E4').
matière_enseignant(techAgent, 'E5').
matière_enseignant(priETapp, 'E6').
matière_enseignant(ang1, 'E7').
```

Figure 3. 13 : Extrait de Base de connaissance – prédicat « matière_enseignant » -

4.9. matière_salle/2.

Dans ce prédicat, on propose que chaque matière soit enseignée dans une salle privée en tenant compte du type de leçons (C, TD, TP) pour chaque matière, comme le montre la Figure 3.14.

```
matière_salle(algoAv, mi03).
matière_salle(bdda, mi03).

matière_salle(algoAv, s03).
matière_salle(algoAv, lab13).
matière_salle(algoAv, lab13).
matière_salle(reprConai, mm03).
matière_salle(reprConai, s02).
matière_salle(reprConai, lab14).
matière_salle(reprConai, lab14).
matière_salle(bdda, lab15).
matière_salle(bdda, lab15).
matière_salle(bigData, mm03).
matière_salle(bigData, s02).
matière_salle(techAgent, mm03).
matière_salle(techAgent, lab01).
matière_salle(techAgent, lab01).
matière_salle(priETapp, mm03).
matière_salle(priETapp, s06).
matière_salle(ang1, mm01).
```

Figure 3. 14 : Extrait de Base de connaissance – prédicat « matière_enseignant » -

4.10. g_etud_matière_type/4.

Par ce prédicat, on détermine le nombre et le type de leçons de toutes les matières pour chaque groupe d'étudiants. Dans notre exemple, chaque matière a une seule leçon, mais le type est déterminé selon la matière, comme le montre la Figure 3.15.

```
% -------------------------- M2-IDO-COURandTDandTP ------------------------
g_etud_matière_type('M2-ido', outIA, c, 1).
g_etud_matière_type('M2-ido', outIA, tp, 1).
g_etud_matière_type('M2-ido', fouille, c, 1).
g_etud_matière_type('M2-ido', fouille, td, 1).
g_etud_matière_type('M2-ido', admBDD, c, 1).
g_etud_matière_type('M2-ido', admBDD, tp, 1).
g_etud_matière_type('M2-ido', techNOdec, c, 1).
g_etud_matière_type('M2-ido', techNOdec, tp, 1).
g_etud_matière_type('M2-ido', ordon, c, 1).
g_etud_matière_type('M2-ido', ordon, td, 1).
g_etud_matière_type('M2-ido', semin, c, 1).
g_etud_matière_type('M2-ido', modSim, c, 1).
g_etud_matière_type('M2-ido', modSim, td, 1).
g_etud_matière_type('M2-ido', methOptim, c, 1).
g_etud_matière_type('M2-ido', methOptim, td, 1).
g_etud_matière_type('M2-ido', methOptim, tp, 1).
```

Figure 3. 15 : Extrait de Base de connaissance – prédicat « g_etud_matière_type» -

4.11. leçon/6.

Ce prédicat est une tête de la règle la plus importante parce qu'elle combine toutes les prédictions précédentes et après réflexion prédit et nous donne toutes les leçons par exemple, pour chaque type de leçon et nombre de fois répétées et un groupe d'étudiants qui étudient une matière particulière dans une salle particulière et sont enseignés par l'enseignant comme le montre la Figure 3.16.

Ce Prédicat est ceux qui sont appliqués dans Code Source.

```
leçon(Groupes, M, Ens, Salles, Type, N) :-
    enseignant(Ens),
    g_Etud(Groupes),
    salle(Salles),
    g_Appartient(Groupes1, Groupes2),
    Groupes1 \= Groupes2,
    type_salle(Salles, Type),
    matière_salle(M, Salles),
    g_etud_matière(Groupes, M),
    g_etud_enseignant(Groupes, Ens),
    matière_enseignant(M, Ens),
    g_etud_matière_type(Groupes, M, Type, N).
```

Figure 3. 16 : Extrait de Base de connaissance – règle « leçon » -

4.12. Nombre de créneaux par (jour / semaine)

- **nombre_créneaux_par_jour**/1: désigne le nombre de créneaux par jour
- **nombre_créneaux_par_semaine/1** : désigne le nombre de créneaux par semaine.

Ce sont des nombres naturels comme le montre la Figure 3.17.

```
nombre_créneaux_par_semaine(30).
nombre_créneaux_par_jour(6).
```

Figure 3. 17 : Extrait de Base de connaissance – nombre des créneaux -

4.13. Créneaux et jours libres pour un groupe d'étudiant

- **g_etud_tempslibre/2 :** désigne les créneaux libres pour un groupe d'étudiants ou un enseignant
- **g_etud_jourlibre/2** : désigne les jours libres pour un groupe d'étudiants ou un enseignant.

ce sont des nombres naturels comme le montre la Figure 3.18.

```
g_etud_jourlibre('M1-ia', 4).
g_etud_tempslibre('M1-ia', 0).
g_etud_tempslibre('M1-ia', 1).
g_etud_tempslibre('M1-ia', 3).
g_etud_tempslibre('M1-ia', 9).
g_etud_tempslibre('M1-ia', 15).
g_etud_tempslibre('M1-ia', 21).
g_etud_tempslibre('M1-ia', 27).
```

Figure 3. 18 : Extrait de Base de connaissance – (créneaux / jours) libre pour un groupe-

4.14. Créneaux et jours libres pour un enseignant

- **enseignant _tempslibre/2 :** désigne les créneaux libres pour un enseignant
- **enseignant _jourlibre/2** : désigne les jours libres pour un enseignant.

ce sont des nombres naturels comme le montre la Figure 3.19.

```
enseignant_jourlibre('E1', 0).
enseignant_jourlibre('E1', 4).
enseignant_tempslibre('E1', 6).
enseignant_tempslibre('E1', 7).
enseignant_tempslibre('E1', 21).
```

Figure 3. 19 : Extrait de Base de connaissance – (créneaux / jours) libre pour un enseignant-

En fin, on a expliqué et montré tous les prédicats qu'on a utilisé pour résoudre le problème d'emploi du temps de notre département.

5. Résultats

On est heureux de vous fournir un compte rendu détaillé des résultats de nos solutions pour résoudre le problème de planification automatique de l'emploi du temps universitaire.

Il se trouve dans cette partie les résultats qu'on a obtenus après avoir étudié et résolu le problème, où on soulignera les améliorations qu'on a réalisées dans la précision de l'emploi du temps et l'efficacité de l'utilisation des ressources académiques.

on a affiché trois types de l'emploi du temps :

- Emploi du temps des groupes d'étudiants.
- Emploi du temps des enseignants.
- Emploi du temps des salles.

5.1. Emploi du temps des groupes d'étudiants

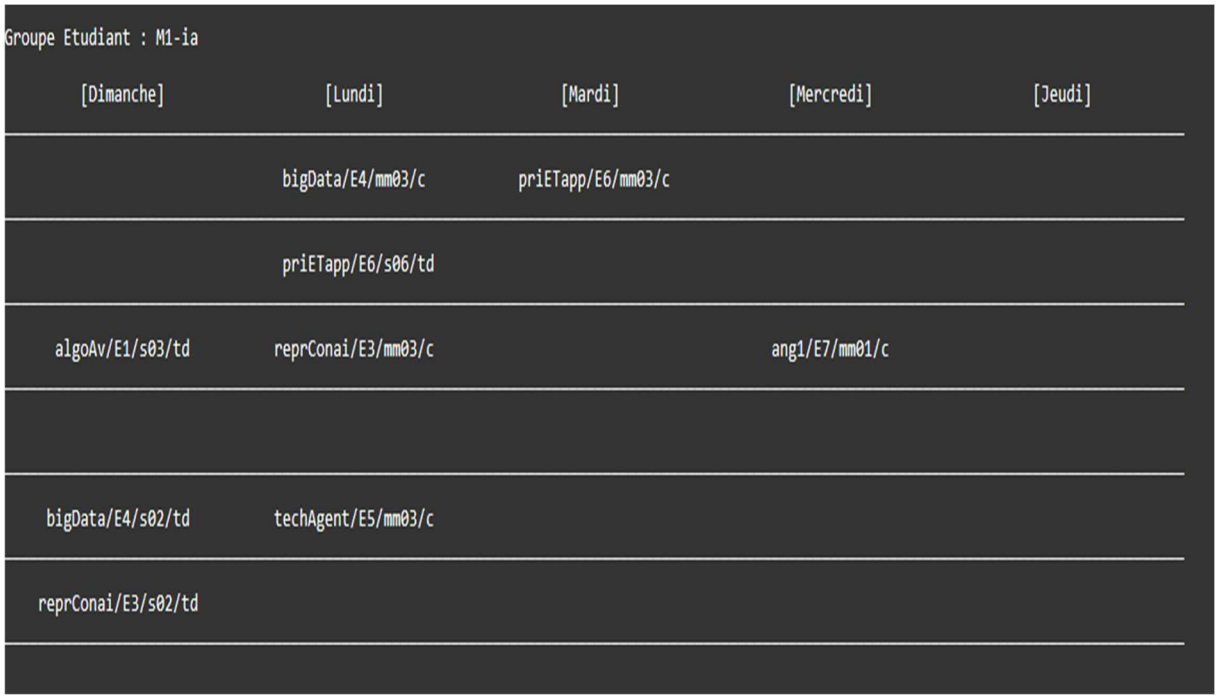
Notre solution a permis d'optimiser des emplois du temps, en tenant compte des préférences des étudiants et des exigences pédagogiques, ce qui a conduit à une amélioration significative de la satisfaction des étudiants et à une meilleure organisation des cours.

Exemple :

On a affiché l'emploi du temps de certains groupes d'étudiants dans notre département (Département Informatique M'sila), mais il n'y a pas de chevauchement entre eux.

On a affiché l'emploi de temps des groupes **('M1-ia'**, '**M1-ia-sg1'**, **'M1-ia-sg2'**, **'M1-rtic'**, **'M1-rtic-sg1'**, **'M1-rtic-sg2'**, **'M2-ido'**, **'M2-**sigl') comme le montre les Figures ci-après :

On note dans les Figures (Figure 3.20, Figure 3.21, Figure 3.22) que ces groupes d'étudiants n'étudient pas en même temps parce qu'ils s'appartiennent (**'M1-ia', 'M1-ia-sg1'**), (**'M1-ia', 'M1-ia-sg2'**).



Groupe Etudiant : M1-ia

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
	bigData/E4/mm03/c	priETapp/E6/mm03/c		
	priETapp/E6/s06/td			
algoAv/E1/s03/td	reprConai/E3/mm03/c		ang1/E7/mm01/c	
bigData/E4/s02/td	techAgent/E5/mm03/c			
reprConai/E3/s02/td				

Figure 3. 20 : Affichage d'emploi du temps de groupe 'M1-ia'

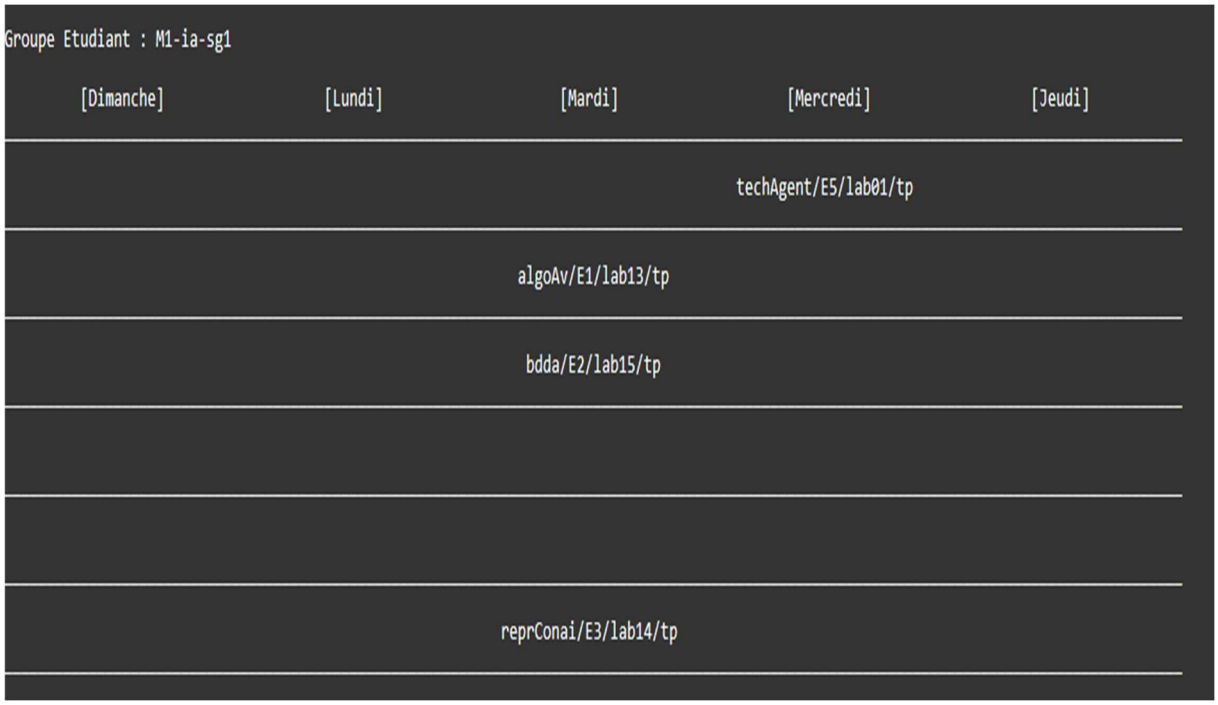


Figure 3. 21 : Affichage d'emploi du temps de groupe 'M1-ia-sg1'

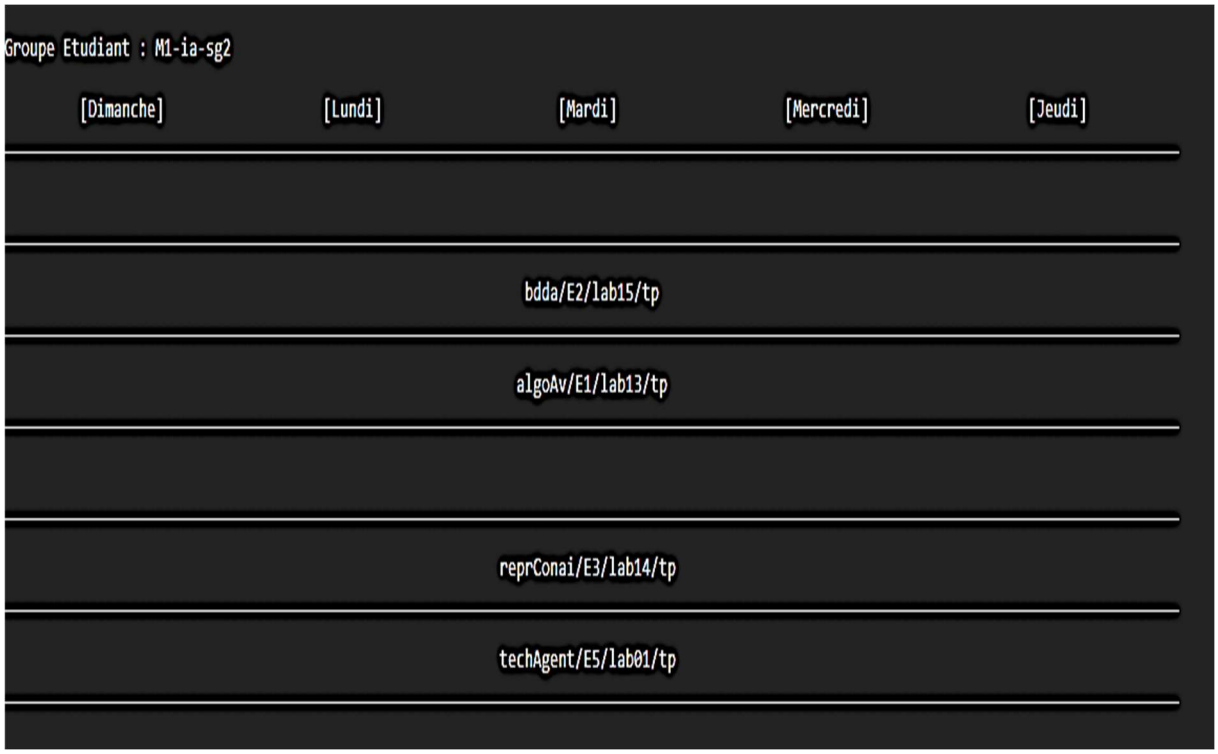


Figure 3. 22 : Affichage d'emploi du temps de groupe 'M1-ia-sg2'

Nous notons dans les Figures (Figure 3.23, Figure 3.24, Figure 3.25) que ces groupes d'étudiants n'étudient pas en même temps parce qu'ils s'appartiennent (**'M1-rtic', 'M1-rtic-sg1'**), (**'M1-rtic', 'M1-rtic-sg2'**).

Groupe Etudiant : M1-rtic

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
	techWeb/E20/mm01/c			
	qOs/E4/s05/td	algoAv_r/E19/mm01/c	ethiq/E13/mm02/c	
bdda_r/E2/s01/td		ang1/E7/mm01/c		
sysDist/E21/mm01/c	algoAv_r/E19/s02/td			
qOs/E4/mm01/c	resProt/E23/mm01/c			

Figure 3. 23 : Affichage d'emploi du temps de groupe 'M1-rtic'

Groupe Etudiant : M1-rtic-sg1

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
			qOs/E4/lab15/tp	
			sysDist/E21/lab01/tp	
		bdda_r/E2/lab13/tp	techWeb/E20/lab14/tp	
		resProt/E23/lab02/tp		

Figure 3. 24 : Affichage d'emploi du temps de groupe 'M1-rtic-sg2'

Groupe Etudiant : M1-rtic-sg2

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
			techWeb/E20/lab14/tp	
			qOs/E4/lab15/tp	
		resProt/E23/lab02/tp	sysDist/E21/lab01/tp	
		bdda_r/E2/lab13/tp		

Figure 3. 25 : Affichage d'emploi du temps de groupe 'M1-rtic-sg2'

Les Figures précédents montrent l'emploi du temps pour certains groupes d'étudiants Master 1.

Les Figures (Figure 3.26, Figure 3.27) représentent l'emploi de temps des groupes d'étudiants (**'M2-ido', 'M2-sigl'**).

Groupe Etudiant : M2-ido

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
	semin/E18/mm02/c	admBDD/E22/mm02/c	methOptim/E11/s01/td	
	techNOdec/E9/mm02/c	fouille/E22/mm02/c	outIA/E6/lab20/tp	
modSim/E5/mm05/c	ordon/E18/s02/td	fouille/E22/s05/td	outIA/E6/mm03/c	
modSim/E5/s06/td	techNOdec/E9/lab17/tp	methOptim/E11/lab18/tp		
ordon/E18/mm02/c	admBDD/E22/lab16/tp	methOptim/E11/mm01/c		

Figure 3. 26 : Affichage d'emploi du temps de groupe 'M2-ido'

Groupe Etudiant : M2-sigl

[Dimanche]	[Lundi]	[Mardi]	[Mercredi]	[Jeudi]
	ows/E15/s07/td	dmri/E4/mm01/c		
	ows/E15/lab19/tp			
dmri/E4/lab20/tp	dad/E30/lab16/tp		poc/E22/mm02/c	
poc/E22/lab17/tp	testLog/E30/s05/td	redacSci/E24/mm01/c		
testLog/E30/mm05/c	dad/E30/mm03/c			

Figure 3. 27 : Affichage d'emploi du temps de groupe 'M2-sigl'

Les Figures précédents montrent l'emploi du temps pour certains groupes d'étudiants Master 2.

En fin, on a montré quelques groupes d'étudiants et nous constatons qu'il n'y a pas de chevauchement ni en termes de groupes appartenant les uns aux autres, ni de chevauchement des enseignants ou de salles.

5.2. Emploi du temps des enseignants

Les emplois du temps des enseignants doivent prendre en compte leurs disponibilités, leurs préférences personnelles et leurs charges de travail.

On a affiché l'emploi du temps de certains enseignants dans notre département. Exemple **(E1**, **E3**, **E11**, **E30)** comme le montre les Figures ci-après :

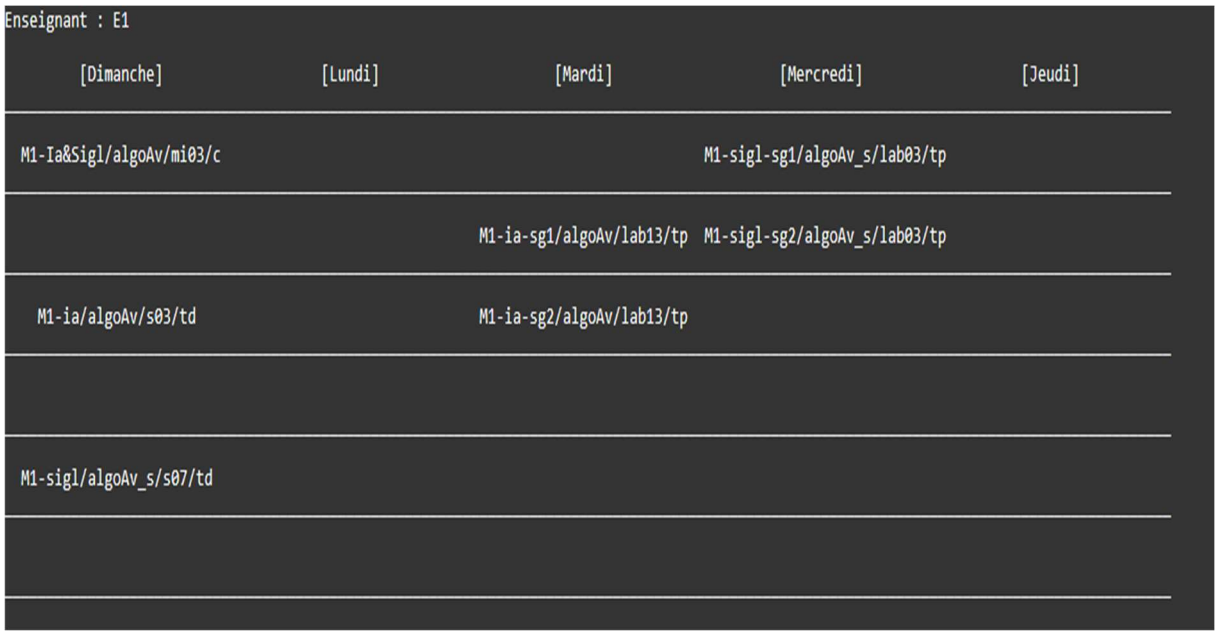


Figure 3. 28 : Affichage d'emploi du temps d'enseignant E1.

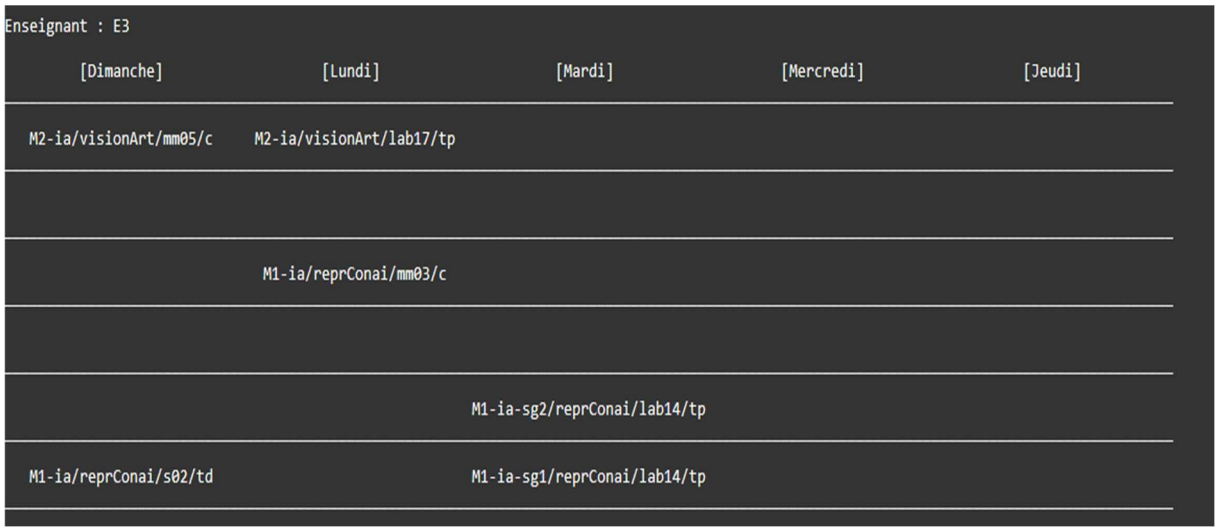


Figure 3. 29 : Affichage d'emploi du temps d'enseignant E3

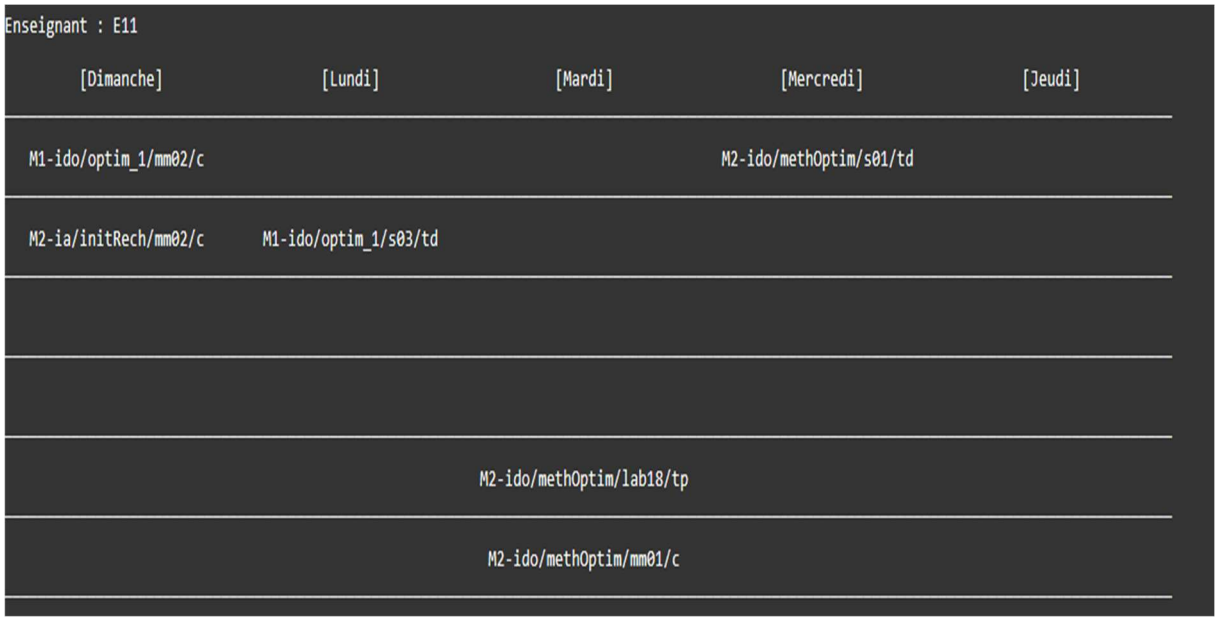


Figure 3. 30 : Affichage d'emploi du temps d'enseignant E11

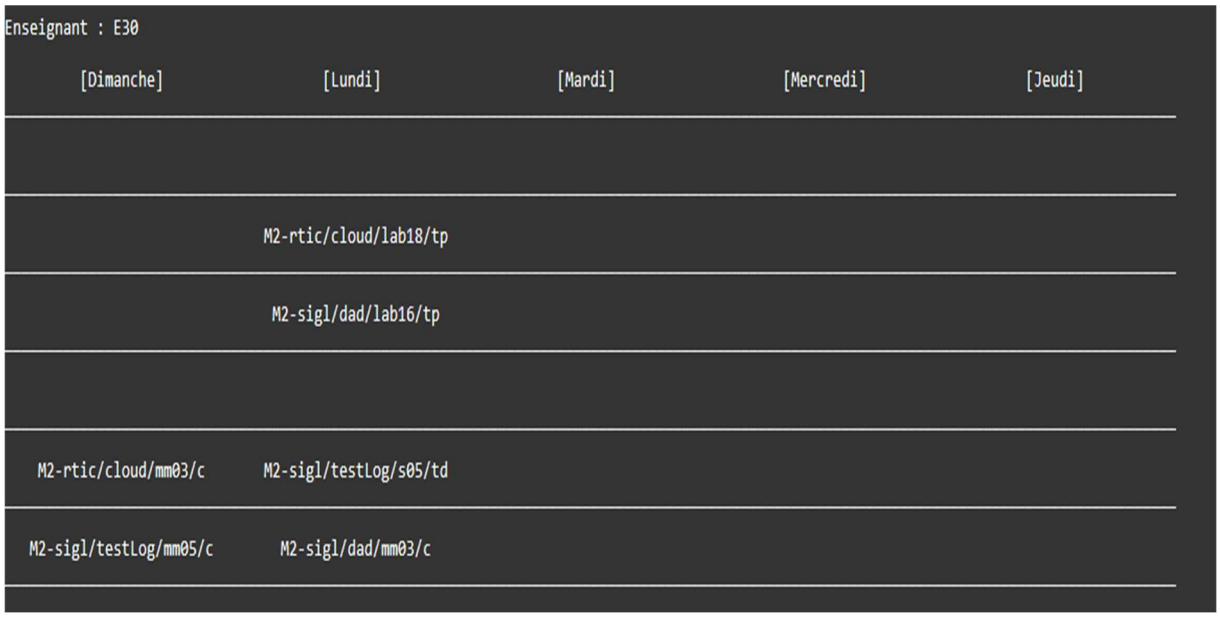


Figure 3. 31 : Affichage d'emploi du temps d'enseignant E30

Enfin, on a montré quelques enseignants et constaté qu'il n'y avait pas de chevauchement pour chaque enseignant.

5.3. Emploi du temps des salles

La gestion des emplois du temps des salles est essentielle pour une utilisation efficace des ressources académiques.

Grâce à notre application, on a pu réduire les périodes de non-utilisation des salles, augmenter la disponibilité des espaces pour les activités académiques et améliorer la gestion des ressources matérielles de l'université.

On a affiché l'emploi du temps de certains salles de notre département. Comme exemple les salles **(mi03**, **mm01**, **s01**, **lab13**) comme le montre les Figures ci-après :

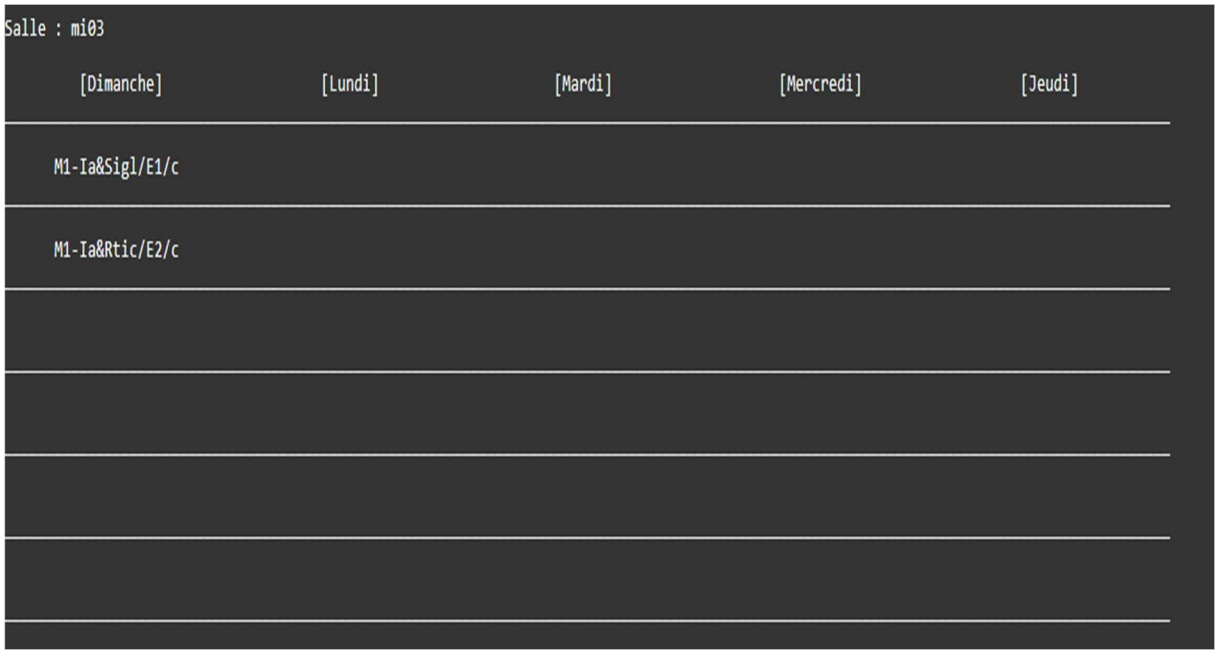


Figure 3. 32 : Affichage d'emploi du temps de salle mi03

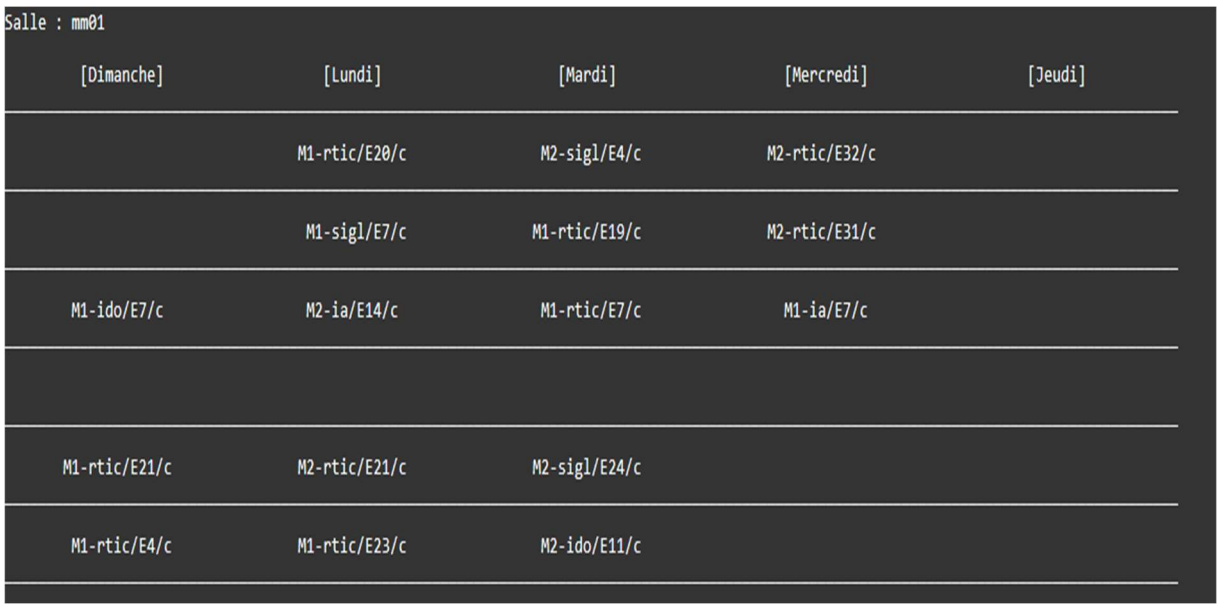


Figure 3. 33 : Affichage d'emploi du temps de salle mm01

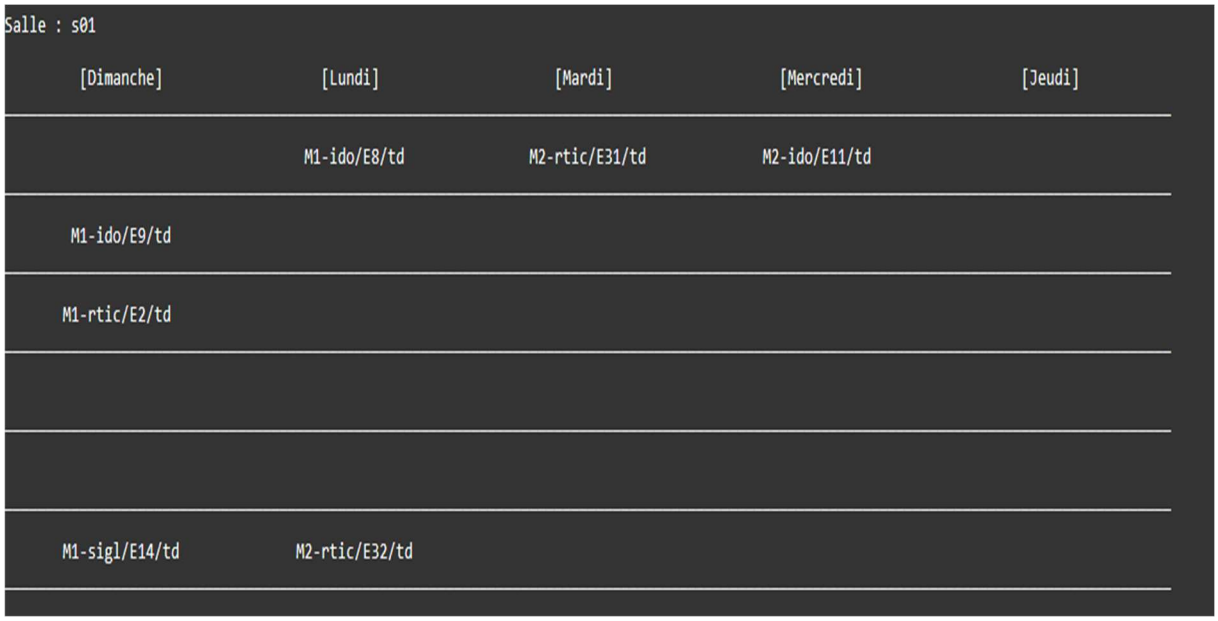


Figure 3. 34 : Affichage d'emploi du temps de salle s01

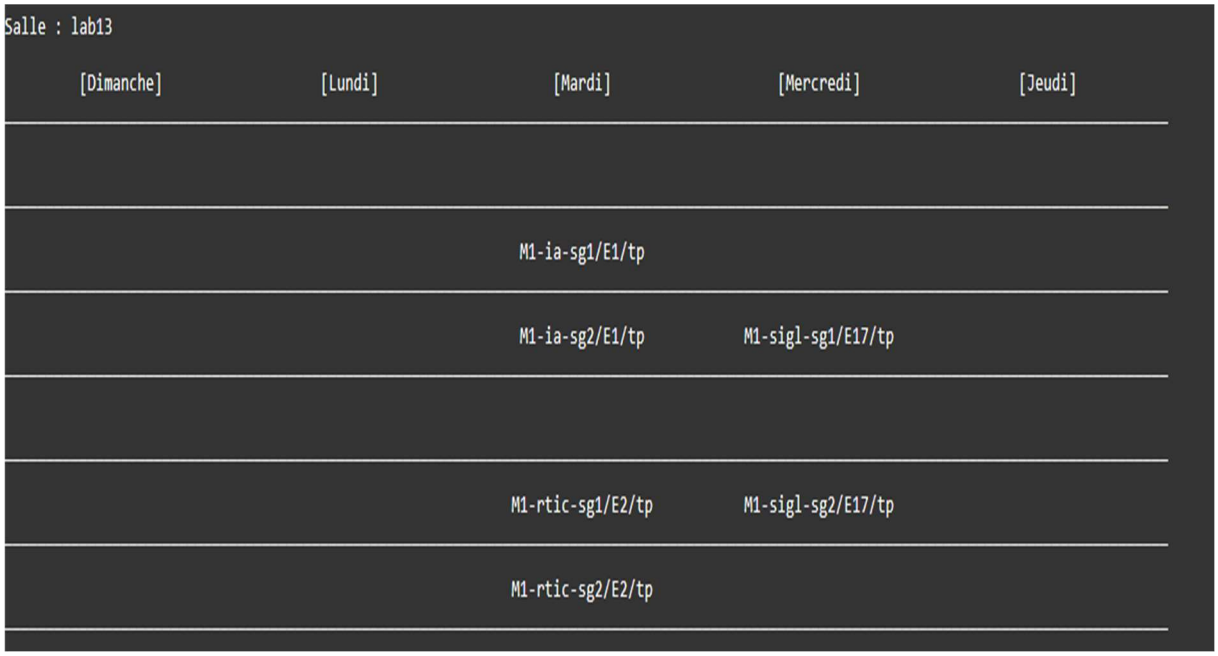


Figure 3. 35 : Affichage d'emploi du temps de salle lab13

5.4. Temps d'exécution

On note que le temps d'exécution n'a pas été long, il a été très court comme le montre la **Figure 3.46**.

```
% CPU time: 2.484s, 7_702_981 inferences
true
```

Figure 3. 36 : Affichage de temps d'exécution

6. Conclusion

Notre solution de planification automatique de l'emploi du temps universitaire a montré des résultats prometteurs dans l'optimisation des ressources et la satisfaction des utilisateurs.

L'utilisation de Prolog et de Scryer Prolog a permis de traiter efficacement les contraintes complexes et de générer des emplois du temps équilibrés pour les groupes d'étudiants, les enseignants et les salles.

Les temps d'exécution rapides de notre solution la rendent adaptée à une utilisation en milieu universitaire, où les ajustements fréquents des emplois du temps sont souvent nécessaires.

CONCLUSION GENERALE

La planification automatique des emplois du temps universitaires est une tâche complexe qui nécessite la prise en compte de multiples contraintes et ressources.

À travers ce mémoire, on a développé une solution en utilisant Prolog et Scryer Prolog pour automatiser cette planification, et ce en offrant des résultats significatifs en termes d'efficacité et de satisfaction des utilisateurs.

Notre travail a démontré que l'utilisation de Prolog, couplée à une modélisation basée sur les Problèmes de Satisfaction de Contraintes (CSP), permet de traiter efficacement les contraintes complexes et variées des emplois du temps universitaires. Les résultats obtenus montrent une amélioration notable de la gestion des ressources académiques, avec des emplois du temps optimisés pour les étudiants, les enseignants et les salles.

Les temps d'exécution rapides et les solutions équilibrées générées par notre application soulignent son potentiel pour une application pratique dans les universités. Cependant, des améliorations peuvent encore être apportées, notamment en intégrant davantage de variables et en explorant des techniques d'optimisation plus avancées.

En conclusion, notre approche représente une avancée significative dans la planification automatique des emplois du temps universitaires, offrant une solution robuste et adaptable aux défis actuels du secteur éducatif. on espère que ce travail servira de base à de futures recherches et développements dans ce domaine, contribuant ainsi à l'amélioration continue de la gestion des ressources académiques.

BIBLIOGRAPHIE

[1] Lixi Zhang and SimKim Lau, Constructing university timetable using constraint satisfaction programming approach. School of Economics and Information System University of Wollongong, NSW 2500, Australia.

[2] Hamidani Hicham et Chikha Belgacem Abderrazzak, Conception et développement d'une application d'optimisation basée sur les systèmes multi-agents cas d'étude « le problème de l'emploi du temps ».Mémoire Master, Universite Kasdi Merbah Ouargla , 30 juin 2013.

[3] Abbes Faicel, Omri Abd El Ouahebe et Coulibaly Souleyman, La mise au point d'un système de génération automatique de l'emploi du temps basé sur les Systèmes Multi-agents, Université 08 mai 1945 Guelma, 2006.

[4] Benhalouche Amel et Bachiri Khadidja, Aide à la gestion d'emploi du temps par visualisation, Universite Abdelhamid Ibn Badis Mostaganem , 2019.

[5] Nathan Campus, Automating Class Schedule Generation in the Context of a University Timetabling Information System, Griffith University.

[6] B. Midodashvili , L. Midodashvili , P. Midodashvili, Genetic algorithm and university timetable problem. 2011.

[7] Glover F. and Laguna M., Tabu Search. Kluwer Academic Publishers, Boston, MA, 1997.

[8] Glover, F., A Template for Scatter Search and Path Relinking. Lecture Notes in Computer Science, 1997.

[9] Brownlee, J. Clever Algorithms Nature-Inspired Programming Recipes. lulu.com, 2012.

[10] Tanzila Islam, Zunayed Shahriar, Mohammad Anower, Perves and Monirul Hasan theme University Timetable Generator Using Tabu Search, Department of Computer Science and Engineering, Southeast University, Dhaka, Bangladesh, 2016.

[11] Thompson, E. P., Time, Work-discipline and Industrial Capitalism. Past and Present, 1967.

[12] Feiring B. R., Linear Programming 60: An Introduction Publication, 1986.

[13] Bunday, B. D., Basic Optimization Methods Edward Arnold, 1984.

[14] BOUNAB Yasmin, Programmation par contraintes des emplois du temps. Université Abdelhamid Ibn Badis de Mostganem

[15] RAMDANE Latifa, Automatisation de la génération de l'emploi du temps. Université Abdelhamid Ibn Badis Mostaganem, 2016.

[16] Franck Barbier, UML 2 et MDE, Ingénierie des modèles avec études de cas, ISBN 9782100495269.

[17] HAMIOUD Amina et HANTAT Selma, Approche hybride pour la résolution d'un DisCSP : génération automatique d'emploi du temps. Université Mohammed Seddik Benyahia Jijel 2022.

[18] L. Lobjois and M. Lemaitre. Coopération entre méthodes complètes et incomplètes pour la résolution de (v) csp : une tentative d'inventaire. Journées Nationales de la Résolution Pratique de Problèmes NP-complets, pages 67–73, 1997.

[19] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. Information sciences, 7 :95–132, 1974.

[20] V. Kumar. Algorithms for constraint-satisfaction problems : A survey. AI magazine, 13(1) :32–32, 1992.

[21] C. Solnon. Un cours" électronique" sur la programmation par contraintes. In JFPLC, pages 77–80, 2003.

[22] J.R. Bitner and E.M. Reingold. Backtrack programming techniques. Communications of the ACM, 18(11) :651–656, 1975.

[23] C. Terrioux. Approches structurelles et coopératives pour la résolution des problemes de satisfaction de contraintes. PhD thesis, Aix-Marseille 1, 2002.

[24] Mme Mallem Zouina et Pr Kholladi Mohamed-Khireddin et Melle Sahraoui Sabrina, Conception d'un système pour l'emploi du temps universitaire, Université Mentouri Constantine,.

[25] https ://www.scryer.pl/ Consulté le 14 mars 2024

Résumé

La planification automatique des emplois du temps universitaires par le biais du problème de satisfaction de contraintes (CSP) optimise la création des emplois du temps en prenant en compte diverses contraintes. Cette méthode identifie les besoins spécifiques des emplois du temps universitaires, y compris la gestion des ressources et la satisfaction des contraintes. Le problème est modélisé en termes de CSP, avec une définition précise des variables, des domaines et des contraintes spécifiques. L'implémentation pratique utilise le langage Prolog et le logiciel Scryer Prolog, et les résultats sont appliqués à un département universitaire. Cette approche permet de générer des emplois du temps pour les groupes d'étudiants, les enseignants et les salles, tout en optimisant le temps d'exécution.

***Mots-clés** : planification automatique, emploi du temps universitaire, problème de satisfaction de contraintes (CSP).*

الملخص

التخطيط التلقائيً للجداول الزمنية الجامعية من خلال مشكلة إرضاء القيود (CSP) يعمل على تحسين إنشاء الجداول الزمنية مع مراعاة القيود المختلفة. تحدد هذه الطريقة احتياجات الجدول الزمني الجامعي بدقة، بما في ذلك إدارة الموارد وتلبية القيود. يتم نمذجة المشكلة من حيث CSP ، مع تعريف دقيق للمتغيرات والمجالات والقيود المحددة. يتم التنفيذ العملي باستخدام لغة البرمجة Prolog وبرنامج Scryer Prolog ، مما يظهر نتائج تطبيقية على قسم جامعي. تمكن هذه الطريقة من توليد جداول زمنية لمجموعات الطلاب وأعضاء هيئة التدريس والقاعات الدراسية، مع تحسين وقت التنفيذ بشكل ملحوظ..

***الكلمات المفتاحية:** التخطيط التلقائي، الجدول الزمني للجامعة، مشكلة إرضاء القيود.*

Abstract

Automatic university timetable planning through the constraint satisfaction problem (CSP) optimizes the creation of schedules by considering various constraints. This approach identifies the specific needs of a university timetable, including resource management and constraint satisfaction. The problem is modeled in terms of CSP, with a precise definition of variables, domains, and specific constraints. Practical implementation is carried out using the Prolog language and Scryer Prolog software, demonstrating results applied to a university department. This method generates timetables for student groups, teachers, and classrooms while optimizing execution time.

Keywords: *automatic planning, university timetable, constraint satisfaction problem (CSP).*