



الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جامعة وهران 2 محمد بن أحمد
Université d'Oran 2 Mohamed Ben Ahmed
معهد الصيانة والأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle

Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel

Spécialité : Maintenance de l'Automatisme et de l'Instrumentation

Thème

IMS-SCHEDULER

Un système de gestion d'emplois du temps

Présenté et soutenu publiquement par :

MANSOUR Marwa
SAADNA BAH AMAR Kawtar

Devant le jury composé de :

Nom et Prénom	Grade	Établissement	Qualité
A. BOULEKHRAS	MCA	IMSI - Univ. Oran 2	Président
A. MOULAI-KHATIR	MCA	IMSI - Univ. Oran 2	Encadrant
A. SENOUCI	MCB	IMSI - Univ. Oran 2	Examinatrice

Année universitaire 2025/2026

Dédicace

Nous dédions ce travail...

*À nos très chers parents,
qui nous ont offert leur amour inconditionnel,
leurs sacrifices silencieux et leur soutien indéfectible.
Vous êtes notre force, notre fierté et notre raison d'avancer.
Ce travail est avant tout le vôtre.*

*À nos frères et sœurs,
pour leur tendresse, leur encouragement
et les moments précieux partagés ensemble.*

*À toutes nos familles,
pour leurs prières et leur soutien moral
qui nous ont accompagnées tout au long de ce parcours.*

*À tous nos amis et collègues de promotion,
pour les souvenirs inoubliables,
la solidarité et les rires partagés durant ces années.*

*À tous ceux qui ont cru en nous
et ont contribué, de près ou de loin,
à l'aboutissement de ce travail.*

MARWA & KAWTAR

Remerciements

Avant tout, nous remercions Allah, le Tout-Puissant, de nous avoir accordé la santé, la volonté et la patience nécessaires pour mener à bien ce travail.

Nous tenons à exprimer notre profonde gratitude à notre encadrant M. Moulai Khatir pour ses précieux conseils, sa disponibilité et son soutien constant tout au long de l'élaboration de ce mémoire. Sa rigueur scientifique et ses orientations éclairées nous ont permis de structurer notre réflexion et d'approfondir nos connaissances dans le domaine de la planification automatique.

Nous adressons également nos sincères remerciements à Mme Aissani, responsable chargée de la planification des emplois du temps au sein de l'université, pour sa disponibilité et les précieuses informations qu'elle nous a communiquées sur le fonctionnement réel de la planification à l'IMSI ont été d'une aide indispensable pour la réalisation de ce travail.

Nos sincères remerciements vont également aux membres du jury qui ont accepté d'évaluer ce travail. Leurs remarques et leurs observations contribueront sans aucun doute à l'amélioration de ce mémoire.

Nous remercions chaleureusement l'ensemble du corps enseignant du département Génie Industriel de l'IMSI Université Oran 2 Mohamed Ben Ahmed pour la qualité de la formation dispensée tout au long de notre cursus, ainsi que le personnel administratif pour sa coopération et les informations mises à notre disposition.

Enfin, nous adressons notre plus sincère gratitude à nos familles pour leur soutien indéfectible et leurs encouragements permanents, ainsi qu'à nos collègues de promotion pour leur solidarité et les moments d'entraide partagés durant ces années d'études.

Résumé

Ce mémoire présente la conception et le développement du système IMS-Scheduler, une application web destinée à améliorer la gestion des emplois du temps universitaires. Face aux limitations des méthodes traditionnelles, souvent manuelles et sources d'erreurs, ce projet propose une solution numérique moderne répondant aux besoins réels des établissements d'enseignement supérieur. Il vise également à réduire la charge administrative et à optimiser l'utilisation des ressources disponibles, telles que les salles et les créneaux horaires.

Mots-clés : Gestion des emplois du temps - Application web - Gestion des ressources - Optimisation

Abstract

This thesis presents the design and development of the IMS-Scheduler system, a web application intended to improve the management of university timetables. Addressing the limitations of traditional methods, which are often manual and prone to errors, this project proposes a modern digital solution that meets the real needs of higher education institutions. It also aims to reduce the administrative burden and optimize the use of available resources, such as classrooms and time slots.

Keywords : Timetable Management - Web Application - Resource Management - Optimization

المخلص

تتقدم هذه الأطروحة تصميم وتطوير نظام IMS-Scheduler، وهو تطبيق ويب يهدف إلى تحسين إدارة الجداول الدراسية الجامعية. وبمعالجة أوجه القصور في الأساليب التقليدية، التي غالباً ما تكون يدوية وعرضة للأخطاء، يقترح هذا المشروع حلاً رقمياً حديثاً يلبي الاحتياجات الفعلية لمؤسسات التعليم العالي. كما يهدف إلى تخفيف العبء الإداري وتحسين استخدام الموارد المتاحة، مثل قاعات الدراسة والفترات الزمنية.

الكلمات المفتاحية: إدارة الجداول الزمنية - تطبيق ويب - إدارة الموارد - تحسين وتطوير

TABLE DES MATIÈRES

Dédicace	I
Remerciements	II
Résumé	III
Introduction Générale	1
1 Étude de l'Existant et État de l'Art	3
1.1 Présentation de l'IMSI	3
1.1.1 Présentation de l'établissement	3
1.1.2 Organisation académique	4
1.1.3 Ressources disponibles	4
1.2 Analyse de la méthode manuelle actuelle	5
1.2.1 Procédure actuelle de génération des emplois du temps	5
1.2.2 Méthode actuelle de planification	5
1.2.3 Outils utilisés	6
1.3 Difficultés et limites du système actuel	7
1.3.1 Conflits de groupes	7
1.3.2 Conflits de salles	7
1.3.3 Conflits d'enseignants	8
1.3.4 Erreurs de saisie et oublis	8
1.3.5 Temps excessif de génération	8
1.4 Problème de la planification des emplois du temps universitaires	8
1.4.1 Définition du problème de planification (Timetabling)	8
1.4.2 Ressources impliquées	9
1.4.3 Classification des contraintes (Hard vs Soft)	9
1.4.4 Complexité théorique : nature NP-difficile et explosion combinatoire	10

1.5	Méthodes d'optimisation dans la littérature	11
1.5.1	Les méthodes exactes	11
1.5.2	Méthodes heuristiques	13
1.5.3	Méthodes métaheuristiques	16
1.6	Analyse comparative des approches et outils existants	18
1.6.1	Tableau comparatif des approches algorithmique	18
1.6.2	Les approches existantes	19
1.7	Positionnement du projet IMS-Scheduler	23
1.7.1	Justification du développement d'un système dédié	23
1.7.2	Architecture à double mode de fonctionnement	23
1.7.3	Approche algorithmique retenue et justification	24
2	Modélisation et Spécification des Besoins	25
2.1	Cadre théorique -Maintenance améliorative	25
2.1.1	Définition de la maintenance logicielle	25
2.1.2	Maintenance améliorative (Perfective)	26
2.1.3	IMS-Scheduler comme projet de maintenance améliorative	27
2.2	Identification des ressources du système	27
2.2.1	Enseignants et disponibilités	28
2.2.2	Salles, types et capacités	28
2.2.3	Groupes d'étudiants et effectifs	29
2.2.4	Matières et volumes horaires réglementaires	29
2.3	Recensement et classification des contraintes	29
2.3.1	Contraintes strictes (formalisation concrète)	29
2.3.2	Contraintes souples	30
2.4	Modélisation mathématique du problème	31
2.4.1	Définition formelle des ensembles	31
2.4.2	Variable de décision	32
2.4.3	Paramètres du système	32
2.4.4	Contraintes enseignants formalisées	33
2.4.5	Contraintes salles formalisées	34
2.4.6	Contraintes groupes formalisées	35
2.5	Formulation complète du problème d'optimisation	36
2.5.1	Fonction d'évaluation et score de pénalité	36
2.5.2	Application au système IMS-Scheduler	36
2.6	Spécification des besoins	37
2.6.1	Besoins fonctionnels du mode manuel	37
2.6.2	Besoins fonctionnels du mode automatique	38
2.6.3	Besoins non fonctionnels	38

3	Architecture du Moteur de Génération des Emplois du Temps	40
3.1	Architecture générale de l'IMS Scheduler	40
3.1.1	Architecture fonctionnelle globale	40
3.1.2	Architecture logicielle	42
3.1.3	Interaction entre Mode Manuel et Mode Automatique	43
3.2	Stratégie de génération automatique	43
3.2.1	Phase 1- Algorithme glouton	43
3.2.2	Phase 2 - Réparation locale	46
3.2.3	Phase 3 - Algorithme génétique	48
3.2.4	Flux global et transitions entre les phases	52
3.3	Stratégie de génération manuelle	53
3.3.1	Affectation manuelle ressources	53
3.3.2	Principe de l'affectation séquentielle manuelle	53
3.3.3	Détection des conflits en temps réel	54
3.4	Conclusion	54
4	Conception de l'Outil et Analyse Comparative	55
4.1	Environnement de développement et base de données	55
4.1.1	Langages utilisés	55
4.1.2	Frameworks pour l'interface utilisateur	56
4.1.3	Justification des choix technologiques	56
4.2	Implémentation de l'interface utilisateur	57
4.2.1	Diagramme de gestion des utilisateurs	57
4.2.2	Interface de saisie manuelle des créneaux	59
4.2.3	Interface de lancement de la génération automatique	60
4.2.4	Gestion des Interactions Utilisateur-Système	60
4.3	Conception de la base de données	62
4.3.1	Modèle Conceptuel des Données (MCD)	62
4.3.2	Modèle Logique des Données (MLD)	62
4.3.3	Structure des tables	63
4.4	Analyse du mode manuel du système IMS-Scheduler	63
4.4.1	Analyse de la gestion des conflits	64
4.4.2	Évaluation des performances opérationnelles	64
4.5	Analyse du mode automatique	65
4.5.1	Analyse des configurations algorithmiques	65
4.5.2	Paramétrage de l'Algorithme Génétique dans le pipeline hybride	66
4.5.3	Analyse des Performances de l'Approche Hybride	68
4.6	Analyse comparative manuel VS automatique	70
4.6.1	Comparaison du temps de génération	71

4.6.2	Comparaison des conflits détectés	71
4.6.3	Comparaison de l'efficacité	71
4.6.4	Analyse comparative finale	72
4.7	Tests et validation	74
4.7.1	Tests fonctionnels - Mode Manuel	74
4.7.2	Tests fonctionnels - Mode Automatique	75
4.7.3	Validation par les utilisateurs finaux	76
4.8	Analyse des résultats	77
4.8.1	Qualité des solutions et fonction de fitness	78
4.8.2	Stabilité du système	78
4.8.3	Influence du nombre de séances sur le temps d'exécution	78
4.8.4	Influence des contraintes sur la qualité des résultats	79
4.9	Limites et perspectives	79
4.9.1	Limites actuelles du système IMS-Scheduler	79
4.9.2	Perspectives futures	80
	Conclusion Générale	81
	Bibliographie	84

TABLE DES FIGURES

1.1	Principales limites du système actuel de planification	7
1.2	Organigramme de l'algorithme génétique.	18
1.3	Architecture générale du système HyperPlanning [24].	21
1.4	Processus de génération dans FET [23].	22
2.1	Type de maintenance [30].	26
3.1	Architecture de IMS-Scheduler.	41
3.2	Architecture logicielle.	42
3.3	Architecture de l'algorithme glouton.	44
3.4	Construction initiale par l'algorithme glouton.	46
3.5	Cycle de réparation	48
3.6	Étapes de l'Algorithme Génétique pour l'Optimisation des Emplois du Temps	52
3.7	Pipeline algorithmique du projet IMS-Scheduler.	52
3.8	Pipeline du mode manuel.	54
4.1	Diagramme de gestion des utilisateurs du système IMS-Scheduler.	58
4.2	Interface de connexion du système IMS-Scheduler.	59
4.3	Interface de saisie manuelle.	60
4.4	Interface de lancement de la génération automatique.	60
4.6	Comparaison des performances entre le mode manuel et le mode automa- tique.	73
4.7	Évaluation multicritère des approches (manuel vs automatique).	73
4.8	Détection d'un conflit de salle en mode manuel.	74
4.9	Rapport de vérification affichant l'absence de conflits après génération au- tomatique.	75
4.10	Interface de visualisation et de filtrage des emplois du temps.	76

4.11	Interface de consultation de l'emploi du temps d'un enseignant.	77
4.12	Interface de consultation de l'emploi du temps d'un groupe d'étudiants. . .	77

LISTE DES TABLEAUX

1.1	Organisation académique de l'Institut IMSI	4
1.2	Feuilles du fichier Excel utilisées dans la procédure actuelle de planification.	6
1.3	Tableau comparatif des méthodes d'optimisation appliquées à la planification.	19
2.1	Vue d'ensemble des ressources du système IMS-Scheduler.	27
2.2	Contraintes strictes (<i>Hard Constraints</i>) appliquées au système IMS Scheduler	30
2.3	Contraintes souples (<i>Soft Constraints</i>) appliquées au système IMS Scheduler.	31
2.4	Définition des ensembles du modèle.	32
2.5	Paramètres fixes du modèle.	33
2.6	Contraintes liées aux enseignants.	33
2.7	Contraintes liées aux salles.	34
2.8	Contraintes des groupes.	35
2.9	Description des termes de la fonction de fitness	37
3.1	Paramètres de l'algorithme génétique.	51
4.1	Justification des choix technologiques du système IMS-Scheduler.	57
4.2	Interactions utilisateur-système et réponses associées.	61
4.3	Entités et relations du MCD d'IMS-Scheduler.	62
4.4	Structure des tables principales de la base de données IMS-Scheduler.	63
4.5	Résultats de la session de planification manuelle.	64
4.6	Résultats globaux de la configuration G+R+AG.	66
4.7	Configurations de paramétrage de l'algorithme génétique	67
4.8	Valeurs de fitness par filière selon les configurations du Tableau4.7.	67
4.9	Résultats détaillés de la configuration hybride G+R+AG.	68
4.10	Comparaison des performances des configurations algorithmiques.	70
4.11	Cas de test du mode manuel.	74

4.12 Cas de test du mode automatique.	75
4.13 Validation par profil d'utilisateur.	76

INTRODUCTION GÉNÉRALE

Dans un contexte universitaire en constante évolution, la gestion des emplois du temps constitue l'une des tâches administratives les plus complexes auxquelles sont confrontés les établissements d'enseignement supérieur, en raison de la multiplicité des contraintes à satisfaire simultanément : disponibilités des enseignants, capacités des salles, volumes horaires réglementaires et répartition des groupes d'étudiants.

À l'Institut de Maintenance et de Sécurité Industrielle (IMSI) de l'Université Oran 2 Mohamed Ben Ahmed, ce processus est actuellement réalisé manuellement à l'aide de fichiers Excel non interconnectés, engendrant des conflits récurrents, des erreurs de saisie fréquentes et un temps de génération excessif pouvant dépasser plusieurs heures par semestre.

Face à ces limites, ce projet propose de concevoir et de développer IMS-Scheduler, un système dédié à la génération automatique et manuelle des emplois du temps de l'IMSI. Sur le plan scientifique, ce problème est de nature NP-difficile, ce qui justifie le recours à une approche hybride combinant un algorithme glouton, une phase de réparation locale et un algorithme génétique.

Ce mémoire est organisé en quatre chapitres :

- Le **premier chapitre** présente l'analyse de l'existant, l'état de l'art et le positionnement du projet.
- Le **deuxième chapitre** est consacré à la modélisation mathématique du problème et à la spécification des besoins du système.
- Le **troisième chapitre** détaille l'architecture du moteur de génération automatique et du mode manuel.
- Le **quatrième chapitre** présente la conception de l'outil, les résultats expérimentaux et la validation du système.

Ce travail est conclu par une synthèse des résultats obtenus et une présentation des perspectives futures du système IMS-Scheduler.

Domaines techniques d'intervention de l'IA

Dans le cadre de ce projet, l'intelligence artificielle a été utilisée comme outil d'assistance pour certaines tâches techniques et rédactionnelles.

1. Amélioration de l'interface utilisateur

Nous avons proposé des améliorations ergonomiques pour l'interface web, optimisé l'expérience utilisateur (UX), amélioré la navigation ainsi que la manière de présenter les données et de visualiser les emplois du temps.

2. Génération des diagrammes

Nous avons créé des diagrammes UML ainsi que des schémas techniques au format SVG ou Mermaid afin de compléter la documentation du projet.

3. Amélioration de la fonction de fitness

Nous avons analysé et optimisé la fonction de fitness utilisée par l'algorithme génétique, proposé de nouveaux coefficients de pondération pour les contraintes souples (C_{soft}) et exploration de formulations alternatives permettant d'améliorer la qualité des emplois du temps générés.

Le système développé constitue une première version fonctionnelle (version 0.1). Bien qu'elle réponde aux principaux objectifs du projet, certaines améliorations et corrections restent envisageables et pourront être intégrées dans les futures versions de l'application.

CHAPITRE 1

ÉTUDE DE L'EXISTANT ET ÉTAT DE L'ART

Comprendre un problème avant de le résoudre constitue le principe fondateur de toute démarche d'ingénierie. C'est pourquoi ce travail débute par un examen minutieux de l'IMSI, de son fonctionnement académique et des pratiques de planification qui y sont actuellement déployées. Les insuffisances ainsi révélées trouvent un écho dans la littérature scientifique consacrée au problème du timetabling, dont la complexité algorithmique a fait l'objet de nombreuses contributions. Ce chapitre en propose une synthèse critique, avant d'examiner les outils du marché susceptibles de répondre aux besoins de l'établissement.

1.1 Présentation de l'IMSI

Dans le but de circonscrire le périmètre de cette étude, il apparaît nécessaire, en premier lieu, de procéder à une présentation rigoureuse du cadre institutionnel au sein duquel s'inscrit ce travail de recherche.

1.1.1 Présentation de l'établissement

L'Institut de Maintenance et de Sécurité Industrielle (IMSI) de l'Université d'Oran 2 Mohamed Ben Ahmed est une structure qui assure des prestations en formation et en recherche dans les domaines de maintenance industrielle, de sécurité industrielle et de l'environnement. Depuis son rattachement au Ministère de l'Enseignement Supérieur et de la Recherche Scientifique en 1993, l'institut a connu une évolution continue dans le but d'améliorer la qualité de ses formations afin de répondre aux besoins du secteur industriel et de l'environnement socio-économique.

L'Institut de Maintenance et de Sécurité Industrielle (IMSI) a pour vocation d'assurer des formations de niveau gradué et post-gradué afin de préparer des cadres de haut niveau

dans des domaines variés, notamment la maintenance industrielle, l'exploitation, l'instrumentation, l'automatisation et la sécurité des procédés.

Ces formations visent à doter les diplômés de compétences à la fois théoriques et pratiques, en adéquation avec les exigences du secteur industriel et du milieu académique. [1].

1.1.2 Organisation académique

L'Institut IMSI est organisé en départements spécialisés afin d'offrir une formation professionnalisante adaptée aux besoins du secteur industriel. Il est structuré autour de trois départements principaux :

Département	Sigle	Domaines principaux	Secteurs d'application
Maintenance en Instrumentation	GI	Mesure, contrôle, automatisation, capteurs, API, supervision, calibration, diagnostic	Sidérurgie, pétrochimie, énergie
Maintenance en Électromécanique	ELM	Machines électriques, hydraulique, pneumatique, entraînements, maintenance préventive/corrective, GMAO	Lignes de production automatisées
Sécurité Industrielle et Environnement	HSE	Prévention des risques, sécurité des procédés, réglementation HSE, gestion des urgences, conformité	Installations industrielles tous secteurs

TAB. 1.1. Organisation académique de l'Institut IMSI

1.1.3 Ressources disponibles

Au niveau de la conception des emplois du temps universitaires, les ressources constituent un facteur déterminant dans l'organisation des activités pédagogiques. L'Institut IMSI dispose d'un ensemble structuré de ressources pédagogiques et techniques destinées à assurer une formation de qualité et en adéquation avec les exigences du milieu industriel. L'établissement dispose notamment des infrastructures suivantes :

- Amphithéâtres et salles de cours destinés à l'enseignement théorique.
- Salles de travaux dirigés permettant l'application et l'approfondissement des notions étudiées.

- Laboratoires pédagogiques spécialisés dans plusieurs domaines techniques et industriels.
- Laboratoire de recherche orienté vers la sécurité industrielle, l'environnement et le développement durable.
- Outils pédagogiques et techniques modernes tels que les logiciels de simulation et les équipements expérimentaux.

1.2 Analyse de la méthode manuelle actuelle

L'environnement académique et les ressources propres à l'IMSI ayant été exposés, l'analyse se portera désormais sur le processus de planification actuellement en vigueur au sein de l'établissement, afin d'en saisir les mécanismes et les fondements opérationnels.

1.2.1 Procédure actuelle de génération des emplois du temps

La génération des emplois du temps à l'Institut IMSI s'effectue actuellement de manière manuelle assistée par des feuilles Excel. Ce processus s'appuie sur une planification centralisée, placée sous la responsabilité du coordinateur pédagogique en collaboration avec les différentes parties prenantes de l'établissement, en coordination avec les chefs de département, les enseignants, le service de scolarité et les étudiants.

1.2.2 Méthode actuelle de planification

Le processus de planification des emplois du temps s'articule autour des étapes suivantes :

- **Collecte des données relatives aux étudiants** : identification des groupes pédagogiques, des filières et des années universitaires concernées, constituant la base de la structure horaire.
- **Recueil des informations sur les enseignants** : spécialité, charge pédagogique minimale et maximale, disponibilités horaires, ainsi que les modules qu'ils sont appelés à assurer (cours, travaux dirigés et travaux pratiques).
- **Répartition des charges pédagogiques** : affectation des cours, TD et TP entre les enseignants en veillant à une répartition équilibrée de la charge pédagogique entre les enseignants.
- **Affectation des modules aux créneaux horaires** : attribution de chaque module à un créneau spécifique et à une salle adaptée, en tenant compte :

- De la disponibilité des enseignants.
 - De la disponibilité et de la capacité des salles.
 - De la nature de la séance (cours, TD ou TP).
 - De l'absence de chevauchement entre les groupes d'étudiants.
- **Vérification des contraintes** : chaque modification entraîne une révision systématique des feuilles afin de détecter les conflits liés aux enseignants, aux groupes et aux salles.

Si cette approche offre une certaine souplesse dans la gestion des cas particuliers, elle demeure fortement tributaire de l'intervention humaine et se révèle difficile à maintenir à mesure que le volume des contraintes augmente .

1.2.3 Outils utilisés

La procédure actuelle repose sur un fichier Excel composé de quatre feuilles distinctes, chacune contribuant à l'élaboration des emplois du temps, comme le présente le tableau 1.2.

Feuille	Données détenues	Objectif
Enseignants	Nom, prénom, nombre de cours/TD/TP effectués, volume horaire enseigné, volume minimum requis, heures complémentaires, modules co-enseignés	Suivre les volumes effectués, éviter les surcharges et assurer une répartition équilibrée des tâches
Étudiants	Nom, prénom, groupe, formation, année d'étude	Organiser les groupes de travail et alimenter la construction des emplois du temps
C-TD-TP	Formation, module, type d'enseignement (Cours/TD/TP), chargé de cours, salle, créneau	Base de travail pour l'élaboration de l'emploi du temps; association formation/module/type de cours/enseignant
ET-Groupe	Emploi du temps visualisé par groupe d'étudiants	Visualisation de l'emploi du temps par groupe, contrôle de la cohérence des heures de cours et repérage des conflits (chevauchements, doubles créneaux)

TAB. 1.2. Feuilles du fichier Excel utilisées dans la procédure actuelle de planification.

Ces quatre feuilles ne sont que partiellement interconnectées : des macros établissent un lien entre la feuille *C-TD-TP* et la feuille *ET-Groupe*, permettant la détection automatique de certains conflits d'affectation. En revanche, les feuilles *Enseignants* et *Étudiants* demeurent totalement indépendantes, ce qui oblige le responsable pédagogique à effectuer manuellement les croisements de données entre ces sources.

1.3 Difficultés et limites du système actuel

Malgré l'utilisation de macros permettant la détection de certains conflits, le système actuel présente plusieurs limites liées à l'absence de synchronisation complète entre les différentes feuilles Excel et au caractère principalement manuel du processus de planification. Les principales difficultés identifiées sont résumées dans la figure 1.1.

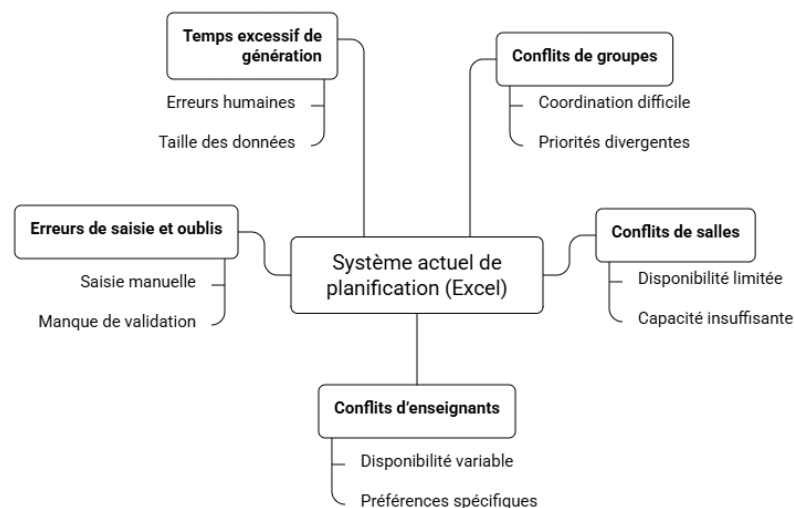


FIG. 1.1. Principales limites du système actuel de planification

1.3.1 Conflits de groupes

Les conflits de groupes surviennent lorsqu'un même groupe d'étudiants est affecté à plusieurs séances durant le même créneau horaire. Bien que certaines incohérences puissent être détectées par les macros existantes, la vérification reste partielle en raison de l'absence d'une synchronisation complète entre les différentes feuilles Excel. Cette situation nécessite souvent des corrections manuelles afin de garantir la cohérence des emplois du temps.

1.3.2 Conflits de salles

L'affectation des salles constitue une contrainte importante dans le processus de planification. La disponibilité limitée de certaines salles, notamment les laboratoires et les

salles spécialisées, peut engendrer des conflits d'occupation. En l'absence d'un système centralisé de réservation, des ajustements manuels sont parfois nécessaires pour résoudre ces situations.

1.3.3 Conflits d'enseignants

La gestion des disponibilités des enseignants demeure complexe en raison de la séparation des informations dans plusieurs feuilles indépendantes. Cette organisation rend les vérifications plus difficiles et peut entraîner des chevauchements d'horaires ou des incompatibilités dans les affectations. Les corrections doivent alors être effectuées manuellement par le responsable de la planification.

1.3.4 Erreurs de saisie et oublis

La manipulation manuelle des données augmente le risque d'erreurs de saisie, d'oublis ou de doublons. Ces incohérences peuvent concerner les horaires, les groupes, les enseignants ou les salles. L'absence d'un mécanisme global de validation limite la fiabilité des données utilisées lors de la génération des emplois du temps.

1.3.5 Temps excessif de génération

L'élaboration des emplois du temps nécessite de nombreuses opérations manuelles telles que la saisie des données, la vérification des contraintes et la correction des conflits détectés. Toute modification importante peut entraîner une révision partielle ou complète de la planification. Cette situation allonge considérablement le temps nécessaire à la génération et à la mise à jour des emplois du temps.

1.4 Problème de la planification des emplois du temps universitaires

Les déficiences du système actuel étant ainsi mises en évidence, il s'avère indispensable de recourir aux apports de la littérature scientifique afin d'appréhender ce problème dans toute sa complexité théorique.

1.4.1 Définition du problème de planification (Timetabling)

Le problème de l'élaboration des emplois du temps universitaires consiste à affecter des plages horaires à des cours, des enseignants et des salles, tout en respectant un grand nombre de contraintes. Il est généralement modélisé comme un problème de satisfaction

de contraintes (CSP) et d'optimisation combinatoire [2, 3]. Sa nature NP-difficile et l'explosion combinatoire qui en résulte sont analysées en détail à la section 1.4.4. En pratique, sa résolution repose sur des méthodes heuristiques et métaheuristiques, notamment les algorithmes génétiques [4, 5].

1.4.2 Ressources impliquées

La planification universitaire s'appuie sur deux catégories de ressources essentielles : humaines et matérielles.

- **Les ressources humaines :** Les principales ressources humaines du système sont les enseignants et les groupes d'étudiants. Les enseignants sont définis par leur spécialité, leur grade et leurs disponibilités, tandis que les groupes sont organisés selon la filière, le niveau et la spécialité. La prise en compte de ces caractéristiques permet d'assurer une planification cohérente et équilibrée des séances d'enseignement.
- **Les ressources matérielles :** peuvent être assimilées à trois grands ensembles :
 - Les salles d'enseignement dont le type (salle de cours, salle de travaux dirigés, salle informatique, laboratoire, etc.) et la capacité conditionnent le type de séances ainsi que le nombre de participants pour lesquels il est possible d'organiser ces séances.
 - Les équipements pédagogiques (matériels informatiques, laboratoires, équipements audiovisuels, etc.), dont la mise à disposition est indispensable pour le bon déroulement de certaines séances spécifiques.

1.4.3 Classification des contraintes (Hard vs Soft)

Avant de présenter les méthodes de résolution du problème de planification, il est nécessaire de distinguer les différentes catégories de contraintes intervenant dans la construction des emplois du temps. Ces contraintes sont généralement classées en deux catégories : les contraintes dures et les contraintes souples .

Contraintes dures (*Hard Constraints*)

Les contraintes dures, également appelées contraintes absolues, sont des conditions obligatoires qui doivent impérativement être respectées dans tout emploi du temps valide. Toute violation d'une contrainte dure rend l'emploi du temps irréalisable et inexploitable.

Dans un problème d'optimisation, une solution ne peut être considérée comme admissible que si la totalité des contraintes dures est respectée. Formellement, une solution x est réalisable si et seulement si :

$$\forall c_i \in C_{\text{dur}} : c_i(x) = 0.$$

Ces contraintes traduisent des impératifs physiques, organisationnels ou réglementaires non négociables : unicité d'une ressource à un créneau donné, respect des capacités, compatibilité entre les compétences de l'enseignant et le module affecté .

Contraintes souples (*Soft Constraints*)

Les contraintes souples, aussi appelées contraintes de préférence, ne sont pas obligatoires mais contribuent à améliorer la qualité globale de la solution. Leur violation n'invalide pas la solution, mais entraîne une pénalité dans la fonction objectif.

L'objectif est de minimiser le nombre ou l'intensité des violations des contraintes souples, tout en garantissant la satisfaction complète des contraintes dures :

$$\min C_{\text{soft}}(x) = \sum w_i \cdot v_i(x).$$

Où $v_i(x)$ représente la mesure de violation de la contrainte souple i et w_i son poids relatif.

1.4.4 Complexité théorique : nature NP-difficile et explosion combinatoire

La planification des emplois du temps (*timetabling*) est l'un des problèmes les plus difficiles en optimisation combinatoire et en recherche opérationnelle. Elle consiste à affecter des activités à des créneaux horaires tout en respectant un ensemble de contraintes liées aux ressources disponibles telles que les enseignants, les salles et les groupes d'étudiants [6].

Dans le contexte universitaire, la complexité du problème provient du grand nombre de variables à prendre en compte ainsi que de l'interdépendance des contraintes, rendant difficile l'obtention d'une solution à la fois réalisable et de qualité [7].

Cette difficulté repose principalement sur trois aspects : l'explosion combinatoire, les conflits de ressources et la nature NP-difficile du problème.

1.Explosion combinatoire

L'explosion combinatoire correspond à la croissance exponentielle du nombre de solutions possibles lorsque la taille du problème augmente [8]. Dans le cas du planification des emplois du temps (*timetabling*), chaque cours peut être affecté à plusieurs enseignants, salles et créneaux horaires, ce qui engendre un très grand nombre de combinaisons [9].

Par exemple, l'affectation de 20 enseignants, 10 salles et 15 groupes sur 25 créneaux horaires génère un espace de recherche extrêmement vaste, impossible à explorer de manière exhaustive [8].

Cette explosion combinatoire entraîne :

- Une augmentation importante du temps de calcul [9].
- L’impossibilité de tester toutes les solutions [8].
- Une difficulté à garantir l’optimalité [6].

Par conséquent, des méthodes heuristiques ou approximatives sont utilisées afin d’obtenir des solutions satisfaisantes dans un temps raisonnable [7].

2. Nature NP-difficile du problème

2.1 Définition de la classe NP

Un problème appartient à la classe NP (Nondeterministic Polynomial time) si la vérification d’une solution candidate peut être effectuée en temps polynomial à l’aide d’un algorithme déterministe [10]. Autrement dit, il est rapide de vérifier qu’une solution donnée est bien une solution du problème, même s’il peut être difficile de la trouver.

2.2 Appartenance à NP et nature NP-difficile du problème des emplois du temps

Le problème des emplois du temps appartient à la classe NP car une solution peut être vérifiée efficacement en s’assurant que les différentes contraintes sont respectées (enseignants, salles, groupes et organisation) [11].

Il est également considéré comme NP-difficile [12], car pour trouver une solution optimale, il faut explorer un très grand nombre de solutions possibles. Ce nombre s’accroît de manière exponentielle avec la taille du problème (le nombre de séances, de ressources et de créneaux horaires), ce qui entraîne une explosion combinatoire et rend impossible toute recherche exhaustive pour des cas réels. On retrouve plusieurs approches comme les méthodes semi-automatiques et automatisées.

1.5 Méthodes d’optimisation dans la littérature

Le cadre conceptuel du problème de planification étant désormais établi, l’attention se portera sur les différentes méthodes d’optimisation recensées dans la littérature, en vue d’identifier les approches les plus pertinentes pour sa résolution.

1.5.1 Les méthodes exactes

Les méthodes exactes visent à obtenir une solution optimale garantissant le respect rigoureux des contraintes du problème. Elles reposent généralement sur des modèles mathématiques précis et des techniques de résolution systématiques adaptées aux problèmes d’optimisation combinatoire.

1.1 Programmation linéaire

Dans le domaine industriel, les problèmes d'optimisation sont généralement résolus à l'aide de méthodes exactes, telles que la programmation linéaire et la programmation par contraintes, ou de méthodes approchées, comme les heuristiques et les métaheuristiques.

La **programmation linéaire (PL)** est une méthode de recherche opérationnelle utilisée pour optimiser l'allocation de ressources limitées. Son développement majeur date de 1947 grâce à George Dantzig, qui formula le problème général de programmation linéaire dans le cadre du projet *Computation of Optimum Program* de la U.S. Air Force et développa la méthode du simplexe [7].

La PL permet de déterminer les valeurs optimales des variables de décision afin de maximiser un profit ou de minimiser un coût, tout en respectant un ensemble de contraintes linéaires [13]. Elle est appliquée dans de nombreux domaines tels que la planification de la production, l'optimisation des transports, la logistique et, plus généralement, l'ingénierie et les sciences appliquées.

1. Principe et fonctionnement

Un modèle de programmation linéaire comporte trois éléments principaux :

- **Variables de décision** : les quantités à déterminer.
- **Fonction objectif** : le critère à maximiser ou à minimiser.
- **Contraintes** : les limites ou exigences imposées aux variables.

Une fois le problème modélisé, des méthodes de résolution, telles que l'algorithme du simplexe, permettent de rechercher la solution optimale parmi l'ensemble des solutions réalisables.

2. Formulation générale

La formulation générale d'un problème de programmation linéaire consiste à maximiser ou minimiser une fonction objectif exprimée en fonction des variables de décision. Elle peut s'écrire sous la forme suivante :

$$\text{Maximiser } Z = \sum_{i=1}^n c_i x_i. \quad (1.1)$$

où x_i représente la variable de décision i , c_i le coefficient associé à cette variable dans la fonction objectif, et n le nombre total de variables de décision.

1.2 Programmation par contraintes

La programmation par contraintes (PPC) est une approche déclarative permettant de modéliser et résoudre des problèmes combinatoires complexes en définissant des variables,

leurs domaines de valeurs et un ensemble de contraintes à respecter [14]. Développée depuis les années 1960-1970, elle vise à réduire l'explosion combinatoire en limitant l'espace de recherche. Son adoption a été facilitée par l'émergence de langages et de solveurs spécialisés. Aujourd'hui, elle est largement utilisée dans l'industrie grâce à des outils performants exploitant la propagation de contraintes et des heuristiques de recherche.

1. Principe et fonctionnement

La programmation par contraintes repose sur trois éléments principaux :

- **Les variables** : les éléments à déterminer.
- **Les domaines** : les valeurs possibles de chaque variable.
- **Les contraintes** : les règles que doivent respecter les variables.

Le solveur réduit d'abord les domaines des variables en éliminant les valeurs incompatibles grâce à la propagation de contraintes. Ensuite, il explore les différentes affectations possibles jusqu'à trouver une solution satisfaisant l'ensemble des contraintes. En cas de conflit, il utilise le retour arrière (*backtracking*) pour tester d'autres possibilités [14].

1.5.2 Méthodes heuristiques

Les **heuristiques** sont des méthodes de résolution de problèmes qui utilisent des règles empiriques ou des stratégies d'exploration pour guider efficacement la recherche de solutions. Contrairement aux méthodes exactes qui garantissent l'optimalité, les heuristiques fournissent des solutions réalisables sans garantie formelle d'optimalité [15].

Les méthodes exactes sont adaptées aux petites instances de problèmes difficiles, mais pour les problèmes NP-difficiles, la taille des instances solvables reste limitée. De plus, la structure du problème influence fortement sa complexité [16].

Ainsi, une heuristique peut être définie comme une technique qui améliore l'efficacité de la recherche en acceptant un compromis entre la qualité de la solution et le temps de calcul, en s'appuyant sur l'expérience, l'intuition et l'exploitation de résultats antérieurs.

1 Principes généraux des heuristiques

Les méthodes heuristiques reposent sur un ensemble de principes visant à simplifier la résolution de problèmes complexes en réduisant l'espace de recherche et le temps de calcul. Contrairement aux approches exactes, elles n'explorent pas toutes les solutions possibles, mais utilisent des stratégies intelligentes pour atteindre rapidement une solution satisfaisante. Le principe fondamental des heuristiques consiste à guider la recherche à l'aide de règles empiriques, souvent basées sur l'expérience ou sur des observations antérieures. Ces

méthodes privilégient une approche pragmatique en acceptant un compromis entre la qualité de la solution et le coût computationnel.

En général, le fonctionnement d'une heuristique suit un processus itératif composé des étapes suivantes :

- Initialisation : génération d'une solution initiale (souvent simple ou aléatoire).
- Application de règles heuristiques : utilisation de critères locaux pour améliorer la solution.
- Évaluation : mesure de la qualité de la solution obtenue.
- Amélioration : modification progressive de la solution en explorant son voisinage.
- Condition d'arrêt : arrêt lorsque la solution devient satisfaisante ou qu'un critère est atteint.

Ce processus permet d'obtenir rapidement une solution réalisable, même pour des problèmes de grande taille.

2 Gestion des contraintes

Dans les méthodes heuristiques, la gestion des contraintes constitue un défi majeur, car ces approches privilégient la rapidité et l'exploration efficace plutôt qu'une vérification exhaustive. Les contraintes se divisent en deux types :

- **Contraintes dures** : obligatoires, leur violation rend la solution non valide.
- **Contraintes souples** : facultatives, leur non-respect dégrade seulement la qualité de la solution.

Pour les intégrer, plusieurs stratégies sont utilisées :

- **Fonction de pénalité** : transformation du problème en ajoutant un coût aux violations.
- **Mécanismes de réparation** : correction des solutions non admissibles.
- **Génération de solutions faisables** : respect direct des contraintes, au risque de réduire la diversité.

Des approches hybrides combinent ces techniques pour améliorer la qualité des solutions et accélérer la convergence. Ainsi, la gestion des contraintes est un élément central de la recherche heuristique, permettant d'équilibrer faisabilité et performance dans les problèmes complexes [16].

3 Algorithmes Gloutons (Greedy)

Un algorithme glouton est une méthode de résolution qui construit une solution de manière **incrémentale**, en sélectionnant à chaque étape l'option la plus avantageuse selon un critère local, sans jamais remettre en question les choix précédents. Cette approche repose sur le principe que des décisions localement optimales mènent à une solution globalement acceptable [17, 18].

Fonctionnement : Le processus de construction suit trois règles fondamentales :

- **Sélection** : à chaque étape, choisir l'élément le plus prometteur parmi les candidats disponibles.
- **Faisabilité** : vérifier que l'élément choisi ne viole aucune contrainte stricte.
- **Irrévocabilité** : une fois un élément intégré à la solution, il n'est jamais retiré.

Avantages et Limites : L'atout majeur des algorithmes gloutons est leur **rapidité d'exécution** et leur **simplicité d'implémentation**. Ils sont particulièrement adaptés à la génération d'une solution de départ pour des méthodes d'optimisation plus avancées. Cependant, leur principale faiblesse réside dans l'absence de garantie d'optimalité : le choix localement optimal ne conduit pas toujours à la meilleure solution globale, et l'algorithme peut se retrouver bloqué dans une impasse sans possibilité de correction [19].

4 Méthodes de Réparation (Repair)

Les méthodes de réparation constituent une approche complémentaire aux algorithmes constructifs. Plutôt que de garantir la faisabilité à chaque étape de construction, elles acceptent une solution initiale potentiellement **infaisable** et appliquent ensuite des opérations correctives itératives pour éliminer progressivement ces violations [20, 17].

Fonctionnement : Le processus de réparation se déroule en quatre phases :

1. **Génération** : produire une solution initiale, même incomplète ou conflictuelle.
2. **Détection** : identifier et lister l'ensemble des violations de contraintes présentes.
3. **Sélection** : choisir le conflit prioritaire à corriger selon une stratégie définie.
4. **Correction** : appliquer une opération de réparation (déplacement, échange, réaffectation) pour éliminer le conflit sélectionné.

Ces phases sont répétées jusqu'à l'obtention d'une solution sans violation de contraintes strictes.

Avantages et Limites : Les méthodes de réparation offrent une grande **flexibilité** et s'intègrent naturellement dans des approches hybrides, notamment avec les algorithmes évolutionnaires où elles servent à corriger les individus infaisables générés par les opérateurs de croisement et de mutation. Toutefois, elles peuvent souffrir de **cycles de réparation** lorsque la correction d'un conflit en engendre un autre, et ne garantissent pas non plus une solution optimale, uniquement une solution faisable [19].

1.5.3 Méthodes métaheuristiques

Une métaheuristique est une stratégie générale d'optimisation destinée à guider une ou plusieurs heuristiques afin d'explorer efficacement un espace de recherche complexe. Le terme provient du grec meta (au-delà) et heuriskein (trouver), indiquant qu'il s'agit d'une approche de niveau supérieur permettant d'orienter les méthodes de recherche vers de bonnes solutions[17]. Les métaheuristiques sont particulièrement utilisées pour résoudre des problèmes d'optimisation difficiles, notamment les problèmes combinatoires de grande taille, pour lesquels les méthodes exactes deviennent souvent impraticables en raison du temps de calcul élevé. Elles offrent ainsi une alternative efficace permettant d'obtenir des solutions de bonne qualité dans un temps raisonnable [17].

1.3Caractéristiques des métaheuristiques

Les métaheuristiques présentent plusieurs caractéristiques fondamentales qui expliquent leur efficacité dans la résolution de problèmes complexes.

1. Flexibilité (approche boîte noire)

Les métaheuristiques sont généralement considérées comme des méthodes «boîte noire». Elles manipulent des solutions potentielles sans nécessiter une connaissance détaillée de la structure interne du problème. La seule information indispensable est une fonction d'évaluation (ou fonction de coût, appelée également fitness), qui permet de mesurer la qualité des solutions générées.

2. Équilibre entre exploration et exploitation

Le succès des métaheuristiques repose sur un compromis entre deux mécanismes fondamentaux :

- **Exploration (diversification)** : capacité de l'algorithme à parcourir différentes régions de l'espace de recherche afin d'éviter de se limiter à une zone restreinte et de manquer la solution optimale globale.
- **Exploitation (intensification)** : capacité à approfondir la recherche autour des solutions prometteuses afin d'améliorer progressivement leur qualité.

L'équilibre entre ces deux mécanismes constitue un facteur déterminant pour la performance des algorithmes métaheuristiques [17].

2.3 Algorithme génétique (GA)

Les algorithmes génétiques sont des méthodes d'optimisation inspirées de l'évolution naturelle, développées par *John Holland* dans les années 1970 [21]. Ils manipulent une population de solutions candidates, appelées chromosomes, et les font évoluer itérativement pour converger vers une solution optimale [22].

Fonctionnement

- **Population initiale** : une population de N solutions est générée aléatoirement, chacune représentant une solution candidate au problème.
- **Évaluation** : chaque individu est évalué par une fonction de *fitness* mesurant sa qualité par rapport à l'objectif visé.
- **Sélection** : les individus les plus performants sont sélectionnés pour la reproduction proportionnellement à leur *fitness* via la méthode de sélection par roulette .
- **Croisement** : deux parents échangent leurs caractéristiques pour produire de nouveaux individus potentiellement meilleurs.
- **Mutation** : une perturbation aléatoire est appliquée sur certains individus pour maintenir la diversité et éviter les optima locaux.
- **Mise à jour** : la population est renouvelée selon une stratégie élitiste, conservant les N meilleurs individus parmi les anciens et les nouveaux.
- **Critère d'arrêt** : le processus se répète jusqu'à convergence, atteinte du nombre maximal de générations, ou satisfaction de la valeur cible de *fitness*.

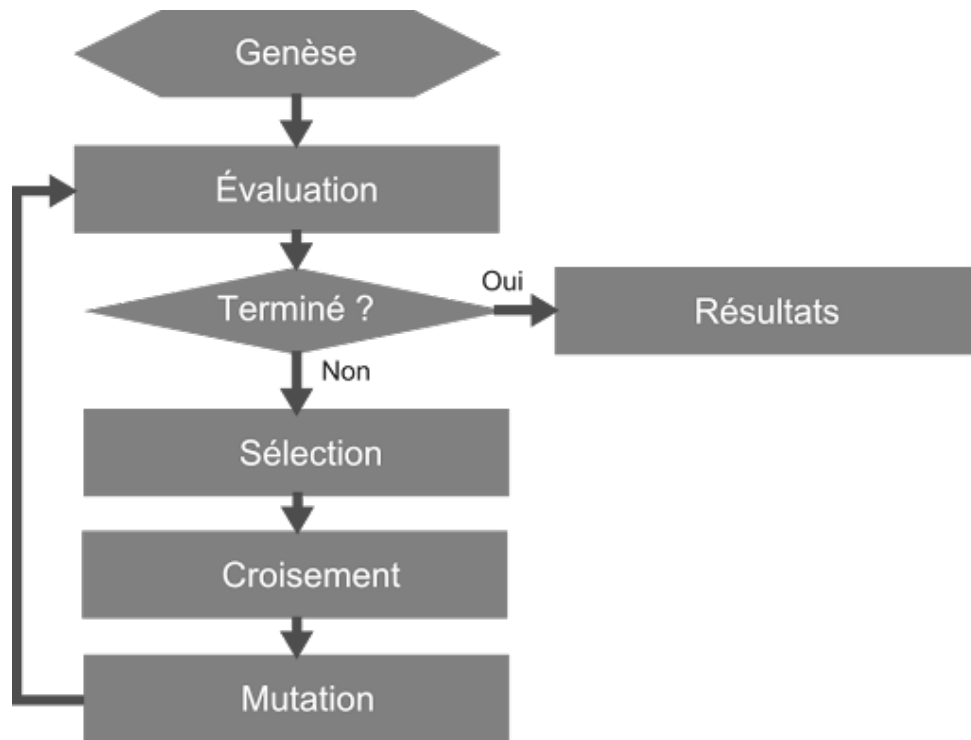


FIG. 1.2. Organigramme de l'algorithme génétique.

Avantages et Limites : Les GA sont robustes et explorent efficacement de grands espaces de recherche sans nécessiter d'information sur le gradient. Cependant, ils peuvent être coûteux en temps de calcul, sensibles au réglage des paramètres, et ne garantissent pas l'optimalité de la solution finale.

1.6 Analyse comparative des approches et outils existants

Cette section présente une analyse comparative des principales approches algorithmiques utilisées dans la littérature pour résoudre le problème de planification des emplois du temps, ainsi qu'une évaluation des outils logiciels existants les plus répandus.

1.6.1 Tableau comparatif des approches algorithmique

Le tableau suivant synthétise les caractéristiques des principales méthodes d'optimisation appliquées au problème *Timetabling*, en les évaluant selon plusieurs critères essentiels : complexité d'implémentation, qualité de la solution, temps d'exécution, gestion des contraintes et adaptabilité au contexte universitaire.

Méthode	Complexité d'implémenta- tion	Qualité de la so- lution	Temps d'exé- cution	Gestion des contraintes	Adaptabilité universitaire
Méthodes Exactes					
Programmation Linéaire (LP)	Élevée	Optimale (petite échelle)	Exponentiel	Forte	Faible
Programmation par Contraintes (PPC)	Très élevée	Optimale (petite échelle)	Exponentiel	Excellente	Faible
Méthodes Heuristiques					
Algorithme Glouton	Faible	Locale (pas optimale)	Très rapide	Partielle	Moyenne
Heuristique de Réparation	Modérée	Acceptable	Rapide	Moyenne	Moyenne
Méthodes Métaheuristiques					
Algorithmes Génétiques (GA)	Élevée	Très bonne	Moyen	Bonne	Excellente
Approches Hybrides					
Approche Hybride	Très élevée	État de l'art	Maîtrisé	Maximale	Maximale

TAB. 1.3. Tableau comparatif des méthodes d'optimisation appliquées à la planification.

L'analyse de ce tableau met en évidence plusieurs constats importants :

- Les méthodes **exactes** (Programmation Linéaire, Programmation par Contraintes) garantissent l'optimalité mais deviennent rapidement impraticables face à la taille réelle des instances universitaires en raison de leur complexité exponentielle.
- Les méthodes **heuristiques** (Glouton, Réparation) sont rapides et simples, mais ne produisent que des solutions localement acceptables sans garantie d'optimalité globale.
- Les méthodes **métaheuristiques** (Recuit Simulé, Recherche Tabou, Algorithmes Génétiques) offrent le meilleur compromis entre qualité de solution et temps d'exécution pour des problèmes de grande taille.
- Les **approches hybrides**, combinant les avantages de plusieurs méthodes, représentent l'état de l'art actuel .

1.6.2 Les approches existantes

Cette section présente les principales approches existantes de génération d'emplois du temps et met en évidence leurs limites .

1. HyperPlanning

HyperPlanning est un logiciel commercial français développé par la société *Index Éducation*, largement utilisé dans les établissements d’enseignement supérieur pour la gestion et la génération des emplois du temps. Il propose une interface graphique complète permettant la gestion des ressources pédagogiques telles que les enseignants, les salles, les groupes et les modules, tout en facilitant la planification des séances de cours, TD et TP[24].

Fonctionnement : Le système HyperPlanning repose sur un moteur de planification semi-automatique basé sur la résolution de contraintes. Lors de la génération des emplois du temps, le logiciel prend en compte les disponibilités des enseignants, les capacités des salles, les contraintes pédagogiques ainsi que les priorités de certaines séances. Afin d’éviter les conflits, HyperPlanning utilise plusieurs techniques algorithmiques telles que le *Constraint Satisfaction Problem (CSP)*, la propagation de contraintes et le *backtracking*. Le logiciel applique également des heuristiques de placement pour prioriser les séances critiques et optimiser l’affectation des ressources. Malgré l’automatisation proposée, l’administrateur peut intervenir manuellement à chaque étape afin de modifier ou ajuster les solutions générées[24].

Limites Malgré ses fonctionnalités avancées, HyperPlanning présente plusieurs limites notables :

- **Coût élevé :** solution propriétaire et payante, difficilement accessible pour les établissements disposant de budgets limités.
- **Personnalisation limitée :** l’intégration de règles métier spécifiques nécessite souvent des configurations complexes ou des développements supplémentaires.
- **Dépendance à l’éditeur :** les évolutions fonctionnelles et les mises à jour dépendent entièrement de la société *Index Éducation*.
- **Complexité pour les petites structures :** le logiciel est principalement conçu pour les grandes institutions et peut être surdimensionné pour des établissements comme l’IMSI.
- **Optimisation limitée :** contrairement aux approches modernes basées sur les méta-heuristiques (*algorithmes génétiques, colonies de fourmis, recuit simulé*), HyperPlanning reste principalement fondé sur des méthodes classiques de satisfaction de contraintes.

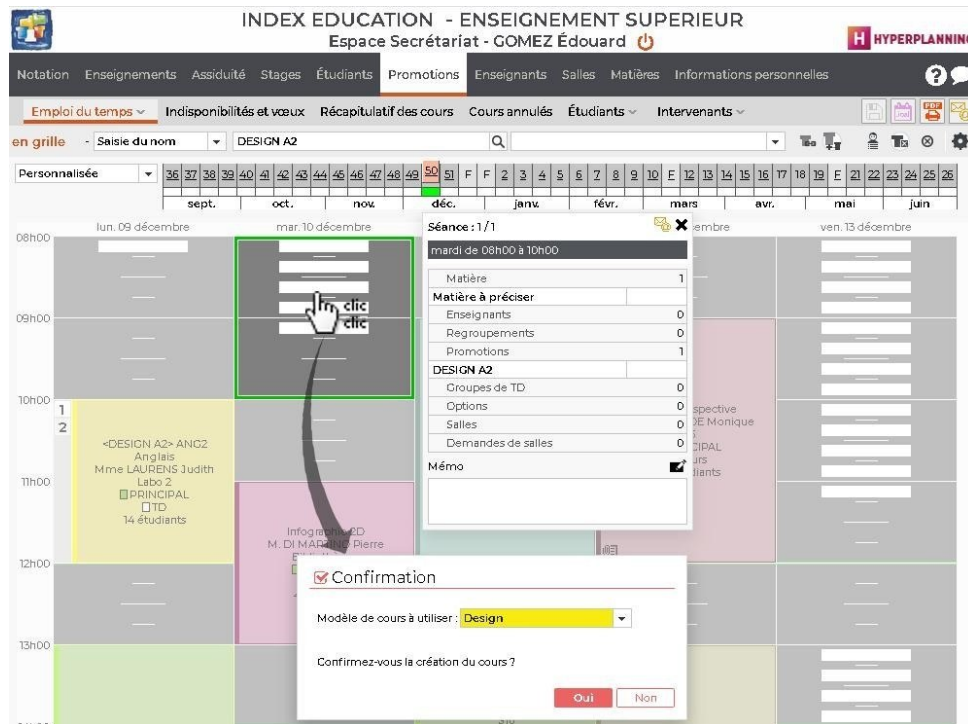


FIG. 1.3. Architecture générale du système HyperPlanning [24].

2.Free Evolutionary Timetabling (FET) :

FET est un logiciel open-source de génération automatique d'emplois du temps, développé par Liviu Lalescu. Disponible gratuitement, il est utilisé dans de nombreux établissements scolaires et universitaires à travers le monde grâce à sa flexibilité et à sa licence libre [23].

Fonctionnement : FET utilise un algorithme heuristique basé sur une **approche de placement séquentiel avec retour en arrière** (*backtracking*), qui fonctionne comme suit :

- Les activités (cours) sont placées **une par une** dans l'emploi du temps, en respectant un ordre de priorité défini.
- En cas de conflit, l'algorithme effectue un **retour en arrière** et tente une nouvelle affectation.
- Le processus intègre nativement un grand nombre de **types de contraintes** (disponibilités, salles, groupes, enseignants, intervalles horaires).
- La génération peut être **relancée** plusieurs fois pour obtenir différentes solutions valides[23].

Limites : Bien que FET soit une solution puissante et gratuite, il présente certaines limitations :

- **Interface peu ergonomique** : la prise en main est complexe et nécessite une formation préalable, ce qui freine son adoption par des utilisateurs non techniques.
- **Temps de génération variable** : pour des instances complexes, le temps de calcul peut devenir très long, voire conduire à un échec de génération.
- **Absence d'optimisation des contraintes souples** : FET privilégie la satisfaction des contraintes strictes mais optimise peu les préférences secondaires (contraintes souples).
- **Pas d'intégration native** : FET ne s'intègre pas facilement avec des systèmes d'information existants (bases de données, ERP universitaires).
- **Personnalisation limitée** : adapter FET aux règles spécifiques d'un établissement requiert une modification du code source, ce qui suppose des compétences en développement logiciel.

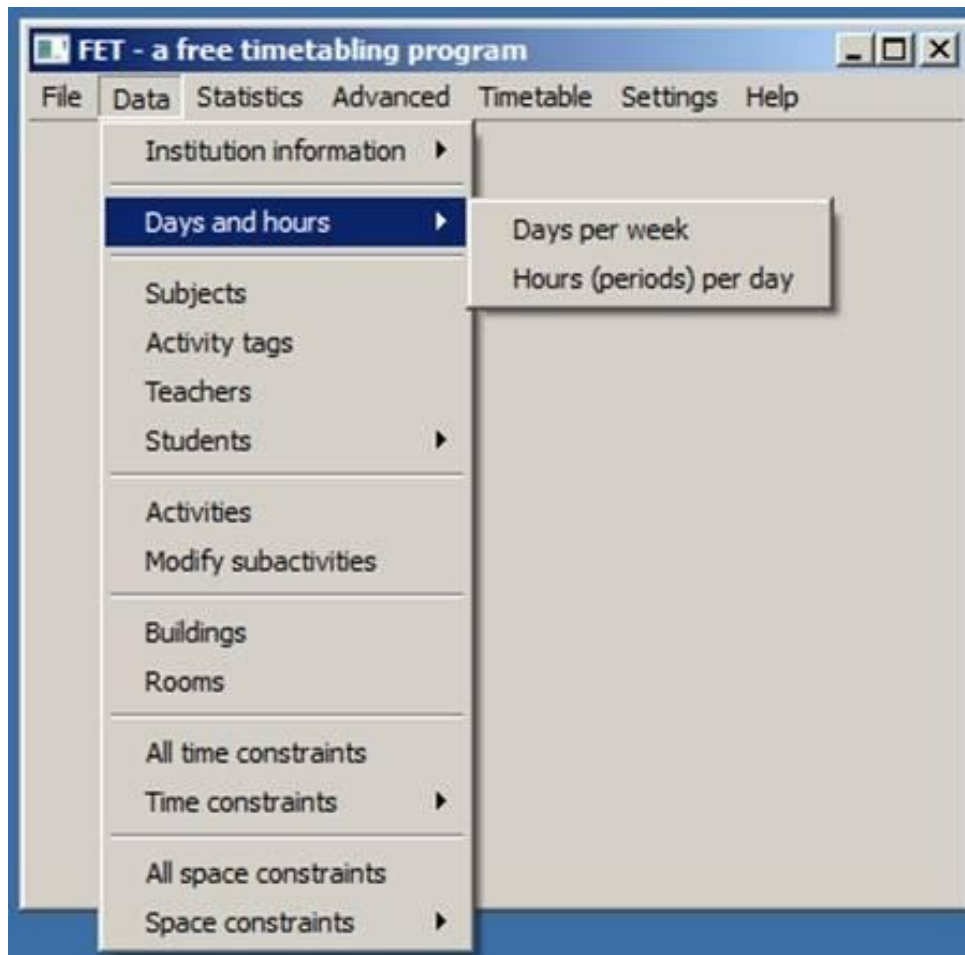


FIG. 1.4. Processus de génération dans FET [23].

1.7 Positionnement du projet IMS-Scheduler

L'étude de l'existant et l'analyse des solutions de planification universitaire ont montré que les méthodes actuellement utilisées à l'IMSI présentent plusieurs limites, notamment les conflits de salles, les chevauchements d'enseignants, les erreurs de saisie et le temps important nécessaire à la génération des emplois du temps.

De plus, les outils existants tels que HyperPlanning ou FET ne répondent pas totalement aux besoins spécifiques de l'établissement. Certaines solutions sont difficiles à adapter au fonctionnement interne de l'IMSI, tandis que d'autres nécessitent des configurations complexes ou des contraintes de licence.

Dans ce contexte, le projet IMS-Scheduler a été proposé afin de mettre en place un système dédié à la gestion et à la génération des emplois du temps universitaires de manière plus flexible et plus efficace.

1.7.1 Justification du développement d'un système dédié

Le développement d'un système se justifie par la complexité du processus de planification universitaire et par les limites des méthodes manuelles actuellement utilisées.

L'établissement gère un nombre important d'étudiants, d'enseignants, de groupes pédagogiques, de matières et de salles, ce qui rend la génération manuelle des emplois du temps difficile et source de nombreux conflits.

Le système IMS-Scheduler vise ainsi à automatiser une grande partie du processus de planification afin de réduire les erreurs, diminuer le temps de génération et améliorer l'organisation globale des emplois du temps.

1.7.2 Architecture à double mode de fonctionnement

Le système IMS-Scheduler propose deux modes de fonctionnement complémentaires : un mode manuel et un mode automatique.

- Le mode automatique permet de générer automatiquement un emploi du temps complet à partir des données disponibles et des contraintes définies par l'administration.
- Le mode manuel permet quant à lui aux responsables pédagogiques d'effectuer des modifications ou des ajustements spécifiques selon les besoins réels de l'établissement.

Le choix de cette approche à double mode permet de combiner les avantages de l'automatisation avec la flexibilité de l'intervention humaine. Cette solution offre ainsi un meilleur contrôle du processus de planification tout en réduisant la charge de travail administrative.

1.7.3 Approche algorithmique retenue et justification

Le problème de génération des emplois du temps universitaires est un problème complexe en raison du grand nombre de contraintes liées aux enseignants, aux salles, aux groupes et aux créneaux horaires.

Afin de répondre efficacement à cette complexité, le système IMS-Scheduler adopte une approche hybride combinant plusieurs techniques complémentaires : un algorithme glouton, une phase de réparation des conflits et un algorithme génétique.

L'approche gloutonne permet de générer rapidement une première solution tout en réduisant le temps de calcul. La phase de réparation intervient ensuite pour corriger les conflits pouvant apparaître lors de cette génération initiale. Enfin, l'algorithme génétique permet d'améliorer progressivement la qualité de l'emploi du temps obtenu en optimisant les contraintes souples.

Le choix de cette combinaison repose sur la complémentarité des méthodes utilisées. Elle permet d'obtenir un bon compromis entre rapidité, réduction des conflits et qualité globale de l'emploi du temps généré.

Conclusion

Ce chapitre a permis d'analyser le processus actuel de génération des emplois du temps à l'IMSI et de mettre en évidence les principales limites de la planification manuelle, notamment les conflits de ressources, les chevauchements et le temps important de traitement. L'étude du problème de planification universitaire a montré sa nature NP-difficile ainsi que la nécessité de recourir à des approches d'optimisation adaptées. Enfin, cette analyse a justifié le développement du système IMS-Scheduler, fondé sur une architecture à double mode et une approche hybride combinant méthodes gloutonnes, réparation locale et algorithmes génétiques. Le chapitre suivant sera consacré à la modélisation du problème et à la spécification des besoins du système.

CHAPITRE 2

MODÉLISATION ET SPÉCIFICATION DES BESOINS

Ce chapitre présente la modélisation du système IMS-Scheduler ainsi que l'analyse des besoins nécessaires à sa conception. Il décrit les ressources du système, les contraintes de planification, la modélisation du problème et les exigences fonctionnelles et non fonctionnelles de la solution proposée.

2.1 Cadre théorique -Maintenance améliorative

Afin de mieux situer le projet IMS-Scheduler dans son contexte conceptuel et méthodologique, il est nécessaire de présenter les notions fondamentales liées à la maintenance logicielle et à l'amélioration des systèmes d'information.

2.1.1 Définition de la maintenance logicielle

Selon la norme NF EN 13306, la maintenance désigne l'ensemble des actions techniques, administratives et de management réalisées durant le cycle de vie d'un bien, dans le but de le maintenir ou de le rétablir dans un état lui permettant d'accomplir la fonction requise. Avec le développement des systèmes informatiques, ce concept a été naturellement étendu au domaine du logiciel[25].

La maintenance logicielle est le processus de modification d'un système après sa livraison. Ces modifications peuvent aller de simples corrections d'erreurs de codage à des améliorations plus substantielles visant à intégrer de nouvelles exigences ou à adapter le système à un environnement technologique en évolution. Contrairement aux systèmes matériels qui se dégradent physiquement avec le temps, les logiciels ne s'usent pas, ils nécessitent néanmoins des interventions régulières pour corriger les anomalies, s'adapter aux évolutions et répondre aux besoins changeants des utilisateurs.

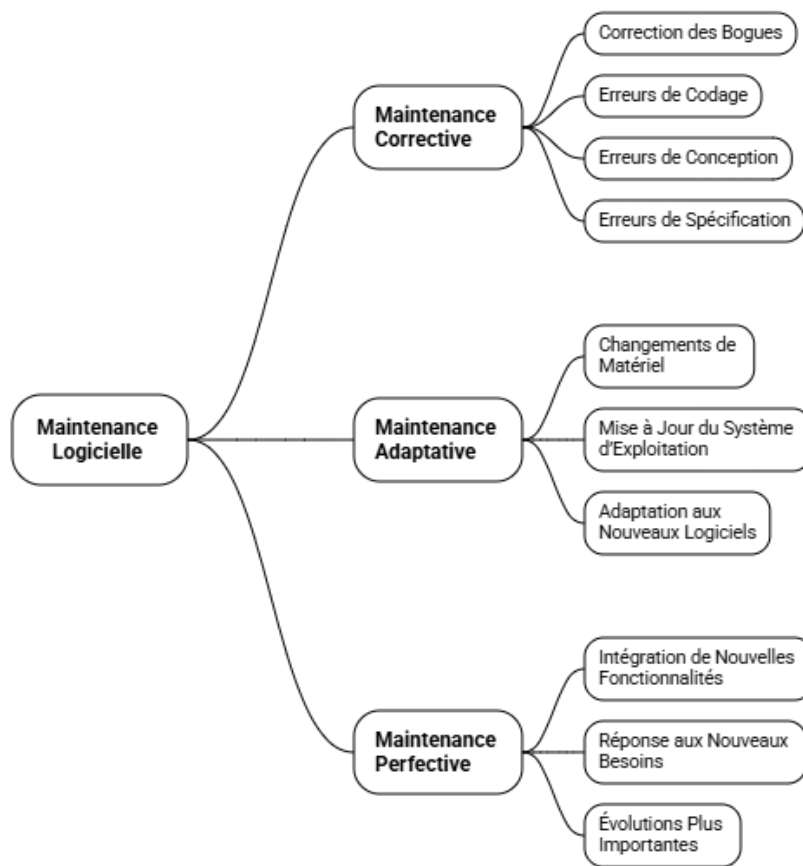


FIG. 2.1. Type de maintenance [30].

2.1.2 Maintenance améliorative (Perfective)

Dans le domaine des systèmes de gestion, la maintenance améliorative va au-delà de la simple correction de pannes : elle représente une stratégie proactive visant à faire évoluer un système existant pour en accroître l'efficacité et la fiabilité. Conformément à la norme NF EN 13306 [25], elle repose sur une démarche structurée en trois étapes diagnostic initial, amélioration et optimisation, suivi continu dont l'objectif est de transformer durablement le mode de fonctionnement d'un système plutôt que d'en corriger des défaillances ponctuelles [26].

Le passage d'un processus manuel à un système automatisé en constitue l'expression la plus concrète : lorsque la planification repose sur des feuilles non interconnectées et des vérifications humaines répétitives, les erreurs et les conflits deviennent structurels [29]. L'introduction d'un outil centralisé et automatisé capable de vérifier les contraintes en temps réel et de générer des solutions optimisées s'inscrit alors pleinement dans cette logique d'amélioration continue, telle que promue par les approches modernes de la Total Produc-

tive Maintenance (TPM) et du Lean Maintenance [27, 28].

2.1.3 IMS-Scheduler comme projet de maintenance améliorative

Le système IMS-Scheduler s'inscrit dans une démarche de maintenance logicielle améliorative appliquée au système de gestion des emplois du temps de l'établissement. Il ne s'agit pas de développer un système totalement indépendant, mais plutôt d'améliorer et de moderniser un processus déjà existant, basé sur une organisation manuelle.

L'objectif principal est d'apporter une évolution fonctionnelle du système actuel en automatisant une partie importante du processus de planification, tout en conservant la possibilité d'intervention humaine via le mode manuel. Cette approche permet de répondre aux limites identifiées dans l'existant, notamment les erreurs de planification, les conflits de ressources et la charge de travail élevée imposée aux responsables pédagogiques en début de chaque semestre.

Ainsi, IMS-Scheduler représente une amélioration concrète du système existant en introduisant des fonctionnalités avancées de génération automatique et de vérification des contraintes, tout en renforçant la fiabilité et l'efficacité du processus global de planification universitaire.

2.2 Identification des ressources du système

La génération d'emplois du temps universitaires repose sur la gestion cohérente de plusieurs catégories de ressources, chacune jouant un rôle précis dans le processus de planification. À l'Institut IMSI, ces ressources ont été identifiées, inventoriées et modélisées dans la base de données du système IMS-Scheduler.

Ressource	Nature	Rôle dans la planification	Exemples (IMSI)
Enseignants	Humaine	Assurer les séances selon disponibilité, spécialité et charge	Permanents, vacataires - toutes spécialités
Groupes d'étudiants	Humaine	Bénéficier des séances planifiées sans chevauchement	G1, G2 G0 = promotion entière
Salles	Matérielle	Accueillir les séances selon leur type et leur capacité	C10-C29, A1, A2, INF1-INF16, LAB1-LAB16
Modules / Cours	Pédagogique	Définir les unités d'enseignement à programmer	Thermodynamique, Automates
Créneaux horaires	Temporelle	Définir la grille de disponibilité hebdomadaire	08h30-10h00, 10h00-11h30, 14h30-16h00

TAB. 2.1. Vue d'ensemble des ressources du système IMS-Scheduler.

2.2.1 Enseignants et disponibilités

Les enseignants constituent la ressource humaine centrale du système. Chaque enseignant est caractérisé par son grade, ses spécialités disciplinaires, sa charge pédagogique minimale et maximale (exprimée en nombre de séances hebdomadaires), ainsi que ses disponibilités horaires.

La disponibilité d'un enseignant e au jour j sur le créneau c est modélisée par la variable binaire :

$$disp(e, j, c) \in \{0, 1\}.$$

Cette information est saisie en amont dans le profil enseignant et constitue un paramètre non modifiable lors de la phase de génération automatique.

À l'IMSI, le corps enseignant se compose d'enseignants permanents titulaires et de vacataires. La compétence d'un enseignant à assurer un module m est formalisée par le paramètre :

$$comp(e, m) \in \{0, 1\}.$$

Cette relation est définie lors de l'élaboration des affectations pédagogiques de chaque semestre.

2.2.2 Salles, types et capacités

Les salles représentent la ressource matérielle principale du système. Elles sont réparties en trois catégories fonctionnelles selon la nature des séances qu'elles peuvent accueillir :

- **Salles standards (C10 , C29, A1, A2)** : destinées aux cours magistraux (CM) et aux travaux dirigés (TD).
- **Salles informatiques (INF1-INF16)** : équipées de postes de travail informatique pour les TP numériques.
- **Laboratoires (LAB1-LAB16)** : dédiés aux travaux pratiques nécessitant des équipements techniques spécialisés.

Chaque salle est caractérisée par sa capacité (nombre de places assises) et son type . Ces deux attributs conditionnent directement les contraintes d'affectation C7 et C8 formalisées dans la section (2.3).

2.2.3 Groupes d'étudiants et effectifs

Les groupes d'étudiants organisent la population étudiante selon la filière, l'année universitaire et la subdivision pédagogique. L'IMS Scheduler distingue deux niveaux de regroupement :

- **Le groupe G0 (promotion entière)** : convoqué uniquement pour les cours magistraux (CM), il regroupe l'ensemble des étudiants d'une filière et d'une année. Deux séances de type CM pour le même groupe G0 ne génèrent pas de conflit.
- **Les groupes Gx (G1, G2, ...)** : sous-groupes définis pour les TD et TP. Deux groupes Gx et Gy appartenant à la même filière mais distincts peuvent être planifiés simultanément sans conflit entre eux.

L'effectif de chaque groupe conditionne le choix de la salle assignée (contrainte de capacité C7).

2.2.4 Matières et volumes horaires réglementaires

Les modules pédagogiques définissent les unités d'enseignement à planifier chaque semaine. Chaque module m est caractérisé par son intitulé, sa filière de rattachement et son volume horaire réglementaire $vol(m, t)$, exprimé en nombre de séances hebdomadaires selon le type de séance t .

où :

$$t \in \{C, TD, TP\}, \quad vol(m, t) \in \mathbb{N}.$$

Ces volumes sont extraits du programme pédagogique officiel de l'IMSI et constituent des paramètres fixes du modèle. La contrainte de couverture horaire C11 garantit que chaque séance est planifiée exactement une fois par semaine, conformément au volume réglementaire.

2.3 Recensement et classification des contraintes

La modélisation des contraintes du système IMS-Scheduler s'appuie sur la classification théorique présentée à la section 1.4.3 du chapitre précédent. Les contraintes identifiées lors de l'analyse du système actuel ont été formalisées et divisées en contraintes strictes (Hard Constraints) et en contraintes de préférence (Soft Constraints).

2.3.1 Contraintes strictes (formalisation concrète)

Les contraintes dures de l'IMS Scheduler découlent directement des règles organisationnelles et réglementaires de l'IMSI. Leur violation est interdite et conduit à rejeter auto-

matiquement la solution candidate lors de l'évaluation par la fonction de *fitness*.

Les contraintes dures identifiées et formalisées sont :

Réf.	Intitulé	Catégorie	Description
C1	Unicité de présence enseignant	Enseignant	Un enseignant ne peut pas être affecté simultanément à deux séances différentes, sauf dans le cas d'un regroupement de groupes pour un cours magistral au sein d'une même filière et année.
C2	Disponibilité	Enseignant	Toute affectation d'un enseignant doit respecter ses disponibilités horaires déclarées.
C3	Charge pédagogique	Enseignant	Le volume horaire hebdomadaire de chaque enseignant doit rester dans les bornes réglementaires $[ch_min(e), ch_max(e)]$.
C4	Compétence disciplinaire	Enseignant	Un enseignant ne peut être affecté qu'aux modules pour lesquels il est habilité ($comp(e, m) = 1$).
C6	Unicité salle	Salle	Une salle ne peut accueillir deux séances simultanées, sauf pour un CM regroupant toute la promotion.
C7	Capacité suffisante	Salle	La capacité de la salle assignée doit être supérieure ou égale à l'effectif du groupe convoqué.
C8	Type de salle adapté	Salle	Le type de salle utilisé doit correspondre au type requis par la séance (standard, informatique, laboratoire).
C10	Non-chevauchement des groupes	Groupe	Aucun groupe d'étudiants ne peut être convoqué à deux séances simultanées selon la logique de conflit strict définie.
C11	Couverture horaire	Groupe	Chaque séance d'un module doit être planifiée exactement une fois par semaine, conformément au volume réglementaire $vol(m, t)$.

TAB. 2.2. Contraintes strictes (*Hard Constraints*) appliquées au système IMS Scheduler

2.3.2 Contraintes souples

Les contraintes souples traduisent des préférences organisationnelles et pédagogiques qui améliorent la qualité de l'emploi du temps sans en conditionner la validité. Leur satisfaction est optimisée par la composante C_{soft} de la fonction de *fitness*.

Réf.	Intitulé	Catégorie	Description
C5	Regroupement des cours	Enseignant	Minimiser le nombre de jours de présence de chaque enseignant à l'institut pour réduire ses déplacements.
C9	Salle non surdimensionnée	Salle	Éviter d'affecter des salles dont la capacité dépasse largement l'effectif du groupe, afin d'optimiser l'utilisation des infrastructures.
C12	Ordre TD/CM	Groupe	Un TD ne doit pas précéder le cours magistral correspondant lors de la même journée, pour des raisons pédagogiques.
C13	Minimisation des trous	Groupe	Réduire les créneaux vides entre deux séances consécutives pour les enseignants et les groupes d'étudiants.

TAB. 2.3. Contraintes souples (*Soft Constraints*) appliquées au système IMS Scheduler.

2.4 Modélisation mathématique du problème

La modélisation formelle du problème de génération des emplois du temps constitue le fondement théorique sur lequel repose l'algorithme hybride implémenté dans le système IMS Scheduler. Elle repose sur la définition précise des ensembles, de la variable de décision, des paramètres et des contraintes formalisées.

2.4.1 Définition formelle des ensembles

Le problème est défini sur les ensembles suivants, dont les valeurs sont extraites des données réelles de l'IMSI :

Ensemble	Symbole	Valeurs issues des données IMSI
Modules	m	Thermodynamique, Analyse 2, Algèbre 2, Recherche opérationnelle, Machines électriques, Automates programmables, IA appliquée
Types de séances	t	C (Cours magistral), TD (Travaux Dirigés), TP (Travaux Pratiques)
Enseignants	e	Liste des enseignants permanents et vacataires de l'IMSI
Groupes	g	Groupes de licences et masters (G1, G2; groupe 0 = promotion entière pour les CM)
Jours	j	Samedi, Dimanche, Lundi, Mardi, Mercredi, Jeudi (6 jours)
Créneaux horaires	c	08h30–10h00 / 10h00–11h30 / 11h30–13h00 / 13h00–14h30 / 14h30–16h00
Salles	s	Salles standards (C10, C29, A1, A2), salles informatiques (INF1–INF16), laboratoires (LAB1–LAB16)

TAB. 2.4. Définition des ensembles du modèle.

2.4.2 Variable de décision

La variable de décision centrale du modèle est une variable binaire définie comme suit :

$$x(m, t, e, g, j, c, s) \in \{0, 1\}.$$

Cette variable vaut 1 si le module m , de type de séance t , assuré par l'enseignant e , est affecté au groupe g le jour j , dans le créneau horaire c , et dans la salle s . Elle vaut 0 sinon.

La dimension de l'espace de décision correspond au nombre total de variables binaires, soit :

$$|M| \times |T| \times |E| \times |G| \times |J| \times |C| \times |S|.$$

Cette combinatoire illustre directement la nature NP-difficile du problème et justifie le recours à des méthodes heuristiques et métaheuristiques.

2.4.3 Paramètres du système

Les paramètres fixes du système, extraits et calculés à partir des données de l'IMSI et stockés dans la base de données Django, sont définis dans le tableau suivant :

Paramètre	Notation	Description	Valeur / Source
Capacité salle	$cap(s)$	Nombre de places assises disponibles dans la salle s	Base de données Room
Effectif promotion	$eff_promo(f, a)$	Nombre total d'étudiants de la filière f en année a	PromotionStats
Effectif groupe	$eff_grp(f, a, g)$	Nombre d'étudiants du groupe g	GroupeStats
Disponibilité enseignant	$disp(e, j, c)$	1 si l'enseignant est disponible, 0 sinon	Profil enseignant
Charge minimale	$ch_min(e)$	Volume horaire hebdomadaire minimal	Règlement IMSI
Charge maximale	$ch_max(e)$	Volume horaire hebdomadaire maximal	Règlement IMSI
Type de salle requis	$type_req(t)$	Standard / INF / LAB	Attribut type_TP
Volume horaire	$vol(m, t)$	Nombre de séances hebdomadaires prévues	Programme pédagogique
Compétence	$comp(e, m)$	1 si l'enseignant est habilité à enseigner m	Affectation pédagogique
Max séances/jour	$MAX(J)$	Nombre maximal de séances par jour	$MAX_J = 5$

TAB. 2.5. Paramètres fixes du modèle.

2.4.4 Contraintes enseignants formalisées

Les contraintes liées aux enseignants garantissent la cohérence pédagogique, le respect des disponibilités ainsi qu'une répartition équilibrée de la charge d'enseignement au sein de l'établissement.

Réf.	Intitulé	Formulation mathématique	Implémentation	Type
C1	Unicité de présence	$\forall(j, c) : \sum x \leq 1$ sauf regroupement CM même filière/année/promotion	Vérification conflit horaire + autorisation regroupement CM	HARD
C2	Disponibilité	$x \leq disp(e, j, c)$	Filtrage via profil de disponibilité	HARD
C3	Charge pédagogique	$ch_min(e) \leq \sum x \leq ch_max(e)$	Vérification des volumes affectés	HARD
C4	Compétence disciplinaire	$x \leq comp(e, m)$	Validation affectation module-enseignant	HARD
C5	Regroupement des cours	$minPj\text{ours}(e)$	Réduction du nombre de jours de présence	SOFT

TAB. 2.6. Contraintes liées aux enseignants.

Exception formalisée : Un enseignant peut assurer un cours magistral simultanément pour plusieurs groupes appartenant à la même filière, à la même année et à la même promotion.

Listing 2.1. Fonction de détection des conflits d'enseignant

```

1 def _conflit_prof(self, seance, slot) -> bool:
2     occupants = self.profs[seance.teacher_id].get(slot)
3     if not occupants:
4         return False
5     promo = self._promo(seance)
6     tc = normalize_type_cours(str(seance.type_cours))
7     return self._check_liste(occupants, promo[0], promo[1], tc)

```

Le Code 2.1 présente la fonction de détection des conflits liés aux enseignants. Cette méthode vérifie si un enseignant est déjà affecté à une séance durant le même créneau horaire. En présence d'occupations existantes, une vérification supplémentaire est effectuée afin de prendre en compte les exceptions autorisées pour les cours magistraux appartenant à une même promotion.

2.4.5 Contraintes salles formalisées

Les contraintes liées aux salles assurent une utilisation cohérente des espaces pédagogiques en tenant compte des capacités d'accueil, des types de salles disponibles et des conflits d'occupation.

Réf.	Intitulé	Formulation mathématique	Implémentation	Type
C6	Unicité salle	$\forall(j, c) : \sum x \leq 1$ sauf CM même filière/année/promotion	Vérification conflit occupation + autorisation regroupement CM	HARD
C7	Capacité suffisante	$cap(s) \geq eff(g)$	Filtrage salles selon capacité requise	HARD
C8	Type de salle	$type(s) = type_req(t)$	Affectation selon type requis : standard, labo, info	HARD
C9	Salle non surdimensionnée	$\min P \max(0, cap(s) - eff(g))$	Réduction du gaspillage de capacité	SOFT

TAB. 2.7. Contraintes liées aux salles.

Exception formalisée : Une salle peut être partagée par deux séances de type CM appartenant à la même filière et année. L'ensemble de la promotion assiste alors au CM dans la même salle. Cette exception est gérée par la méthode de détection de conflits.

Listing 2.2. Fonction de détection des conflits de salle

```

1 def _conflit_salle(self, seance, slot, salle) -> bool:
2     occupants = self.salles[nom_salle(salle)].get(slot)
3     if not occupants:
4         return False
5     promo = self._promo(seance)
6     tc = normalize_type_cours(str(seance.type_cours))
7     return self._check_liste(occupants, promo[0], promo[1], tc)

```

Le Code 2.2 présente la fonction de détection des conflits liés aux salles. Cette méthode vérifie si une salle est déjà occupée pendant un créneau horaire donné. Lorsque la salle est libre, aucun conflit n'est signalé. Dans le cas contraire, la fonction récupère les informations relatives à la promotion ainsi qu'au type de cours, puis applique une vérification supplémentaire à l'aide de la méthode "check liste". Cette approche permet de prendre en compte les exceptions autorisées, notamment le partage d'une salle pour des séances de type cours magistral (CM) appartenant à une même filière et une même année.

2.4.6 Contraintes groupes formalisées

Les contraintes associées aux groupes d'étudiants visent à garantir la cohérence des emplois du temps, à éviter les chevauchements de séances et à améliorer le confort pédagogique des étudiants.

Réf.	Intitulé	Formulation mathématique	Implémentation	Type
C10	Non-chevauchement	$\forall(j, c),$ même filière/année : conflit si $groupe_en_conflit_strict(g1, g2)$ <i>Vrai</i>	Vérification via <code>groupes[slot][filière]</code> dans <code>is_libre()</code>	HARD
C11	Couverture horaire	$\sum x(m, t, e, g, j, c, s) = vol(m, t), \forall m, t, g$	Chaque séance planifiée exactement une fois	HARD
C12	Ordre TD/CM	TD(m) ne doit pas précéder CM(m) le même jour	Contrôle de l'ordre des séances (<code>td_avant_cm</code>)	SOFT
C13	Minimisation des trous	$\min P \text{ trous_par_groupe}$	Réduction des créneaux vides dans l'emploi du temps	SOFT

TAB. 2.8. Contraintes des groupes.

La logique de détection de conflit entre groupes distingue quatre cas : $G0$ vs $G0$ = pas de conflit (même promotion, regroupement CM autorisé), $G0$ vs Gx = conflit (groupe entier et sous-groupe ne peuvent être simultanément convoqués), Gx vs Gx = conflit (même sous-groupe), Gx vs Gy = pas de conflit (sous-groupes distincts de la même filière).

2.5 Formulation complète du problème d'optimisation

Afin d'évaluer la qualité des solutions générées, une fonction d'évaluation basée sur les contraintes du problème est définie.

2.5.1 Fonction d'évaluation et score de pénalité

La fonction d'évaluation est un élément fondamental de tout algorithme génétique. Elle joue le rôle d'un critère permettant de mesurer la qualité de chaque solution candidate (individu) au sein de la population. De manière générale :

$$f(x) = \frac{1}{1 + C(x)}. \quad (2.1)$$

où $C(x)$ représente le coût global associé à la solution x . Cette formulation présente plusieurs propriétés importantes :

- $f(x) \in]0, 1]$: la valeur de fitness est toujours positive et bornée ;
- $f(x) \rightarrow 1$ lorsque $C(x) \rightarrow 0$: plus le coût est faible, meilleure est la solution ;
- $f(x) \rightarrow 0$ lorsque $C(x) \rightarrow +\infty$: les solutions fortement pénalisées obtiennent une faible valeur de fitness.

Lorsque le problème comporte plusieurs types de contraintes, le coût global est décomposé en deux composantes pondérées :

$$C(x) = \alpha \times C_{\text{dur}}(x) + \beta \times C_{\text{soft}}(x), \quad \alpha \gg \beta \quad (2.2)$$

où $C_{\text{dur}}(x)$ représente le coût associé aux violations des contraintes dures et $C_{\text{soft}}(x)$ le coût associé aux violations des contraintes souples. Les paramètres $\alpha, \beta \in \mathbb{R}^+$ sont des coefficients de pondération strictement positifs. La condition $\alpha \gg \beta$ garantit qu'une violation de contrainte dure est beaucoup plus pénalisante qu'une violation de contrainte souple, assurant ainsi la priorité des contraintes dures dans le processus d'optimisation.

La fonction de fitness générale devient alors :

$$f(x) = \frac{1}{1 + \alpha \times C_{\text{dur}}(x) + \beta \times C_{\text{soft}}(x)}. \quad (2.3)$$

2.5.2 Application au système IMS-Scheduler

Chaque emploi du temps candidat ET est évalué selon cette même fonction, où $x = ET$, avec $\alpha, \beta \in [0, 1]$ et $\alpha \gg \beta$, garantissant ainsi la priorité des contraintes dures sur les contraintes souples.

Les différents termes intervenant dans cette fonction sont détaillés dans le Tableau 2.9.

Terme	Description	Poids
$C_{dur}(ET)$	Nombre de violations des contraintes dures	α
<i>surcharge</i>	Dépassement de la capacité des salles	w_{sur}
<i>td_avant_cm</i>	TD planifié avant le CM correspondant	w_{tdcm}
<i>trous</i>	Créneaux vides entre deux séances consécutives	w_{trou}
<i>jours_prof</i>	Nombre de jours de présence des enseignants	w_{prof}

TAB. 2.9. Description des termes de la fonction de fitness

La valeur optimale $f(ET) = 1$ est atteinte si et seulement si $C_{dur} = 0$ et $C_{soft} = 0$.

2.6 Spécification des besoins

La spécification des besoins constitue une étape essentielle dans le processus de développement du système IMS-Scheduler. Elle permet de définir clairement les fonctionnalités attendues ainsi que les critères de qualité que le système doit respecter. Cette spécification est basée sur l'analyse de l'existant présentée au chapitre 1 et sur la modélisation du problème exposée dans les sections précédentes.

2.6.1 Besoins fonctionnels du mode manuel

Le mode manuel permet à l'administrateur pédagogique de construire et d'ajuster l'emploi du temps de manière interactive. Les principaux besoins fonctionnels sont les suivants :

1. **Gestion des séances à planifier :** Le système doit permettre l'affichage de l'ensemble des séances à programmer en précisant les informations essentielles telles que la filière, l'année, le semestre, le module, l'enseignant, le type de cours et le groupe concerné.
2. **Affectation des créneaux et des salles :** L'administrateur doit pouvoir attribuer à chaque séance un jour, un créneau horaire et une salle disponible en fonction des contraintes pédagogiques et matérielles.
3. **Vérification des conflits :** Le système doit vérifier automatiquement les conflits potentiels liés aux enseignants, aux salles et aux groupes avant toute validation d'une affectation.

4. **Notification des erreurs :** En cas de conflit détecté, le système doit afficher des messages clairs indiquant la nature du problème afin de permettre sa correction immédiate.
5. **Modification des affectations :** L'utilisateur doit pouvoir modifier ou corriger une séance déjà planifiée à tout moment.

2.6.2 Besoins fonctionnels du mode automatique

Le mode automatique permet la génération complète de l'emploi du temps sans intervention humaine. Les besoins fonctionnels associés sont :

1. **Chargement des données :** Le système doit charger l'ensemble des données nécessaires à la planification (séances, enseignants, salles, groupes).
2. **Préparation des ressources :** Le système doit organiser les données en fonction des capacités des salles, des disponibilités et des contraintes pédagogiques.
3. **Génération automatique de l'emploi du temps :** L'emploi du temps doit être généré automatiquement en respectant les contraintes définies et en assurant une cohérence globale de la planification.
4. **Optimisation de la solution :** Le système doit améliorer progressivement la qualité de l'emploi du temps en réduisant les conflits et en optimisant l'utilisation des ressources.
5. **Sauvegarde des résultats :** L'emploi du temps généré doit être enregistré dans la base de données pour consultation et exploitation.

2.6.3 Besoins non fonctionnels

Le système IMS-Scheduler doit également respecter plusieurs exigences de qualité :

1. **Performance :** Le système doit être capable de générer un emploi du temps dans un temps raisonnable, même pour un volume important de données.
2. **Fiabilité :** Le système doit garantir la cohérence des emplois du temps générés en évitant les conflits majeurs.
3. **Maintenabilité :** L'architecture du système doit permettre l'ajout ou la modification des règles de gestion sans impact majeur sur le fonctionnement global.
4. **Accessibilité :** Le système doit être utilisable par différents profils d'utilisateurs (administrateurs, enseignants et étudiants).

5. **Sécurité :** L'accès aux fonctionnalités critiques doit être protégé et réservé aux utilisateurs autorisés.

Conclusion

Ce chapitre a présenté la modélisation du problème de planification ainsi que la spécification des besoins fonctionnels et non fonctionnels du système IMS-Scheduler. Cette étape permet de définir clairement les objectifs du système et de préparer la conception de son architecture.

Le chapitre suivant sera consacré à la conception et à l'architecture du moteur de génération automatique de l'emploi du temps.

CHAPITRE 3

ARCHITECTURE DU MOTEUR DE GÉNÉRATION DES EMPLOIS DU TEMPS

Une formulation mathématique, quelle que soit sa rigueur, demeure sans effet tant qu'elle n'est pas incarnée dans un outil fonctionnel. C'est l'ambition de ce chapitre, qui expose la manière dont le système IMS-Scheduler a été structuré pour produire, de façon automatisée, des emplois du temps conformes aux exigences établies. L'accent est mis sur le fonctionnement interne du moteur de génération, conçu comme une succession de traitements complémentaires capables de construire, corriger puis affiner une solution. Le dispositif de saisie manuelle, pensé comme complément du mode automatique, y est également abordé.

3.1 Architecture générale de l'IMS Scheduler

Ce chapitre décrit l'architecture fonctionnelle et logicielle du système IMS-Scheduler, ainsi que les stratégies adoptées pour la génération et l'optimisation des emplois du temps.

3.1.1 Architecture fonctionnelle globale

L'IMS Scheduler est une application web développée avec le framework Django, destinée à l'automatisation de la génération des emplois du temps universitaires. Le système repose sur une architecture en couches, assurant la séparation des responsabilités et la maintenabilité du code. Trois couches principales peuvent être distinguées :

1. **Couche de données** : elle regroupe les entités persistantes du système, notamment les salles (*Room*), enseignants (*Teacher*), modules (*Module*), affectations pédagogiques (*TeacherModule*), séances (*Seance*), ainsi que les objets de planification (*TimeTable*).

2. **Couche métier** : elle implémente la logique de génération et d'optimisation. Elle inclut la normalisation des données, la détection des conflits, ainsi que le mécanisme de planification basé sur une approche hybride combinant heuristique gloutonne, réparation locale et optimisation évolutionnaire (algorithme génétique).
3. **Couche de présentation** : expose une interface web permettant aux administrateurs de déclencher la génération, de visualiser les calendriers par étudiant ou par enseignant, de saisir manuellement des créneaux et de consulter les rapports de Vérification .

Les données d'entrée proviennent soit de fichiers d'importation (Excel), soit de l'interface d'administration Django. Le système produit en sortie des emplois du temps structurés, stockés sous forme d'instances *TimeTable* consultables par les étudiants ou par les enseignants.

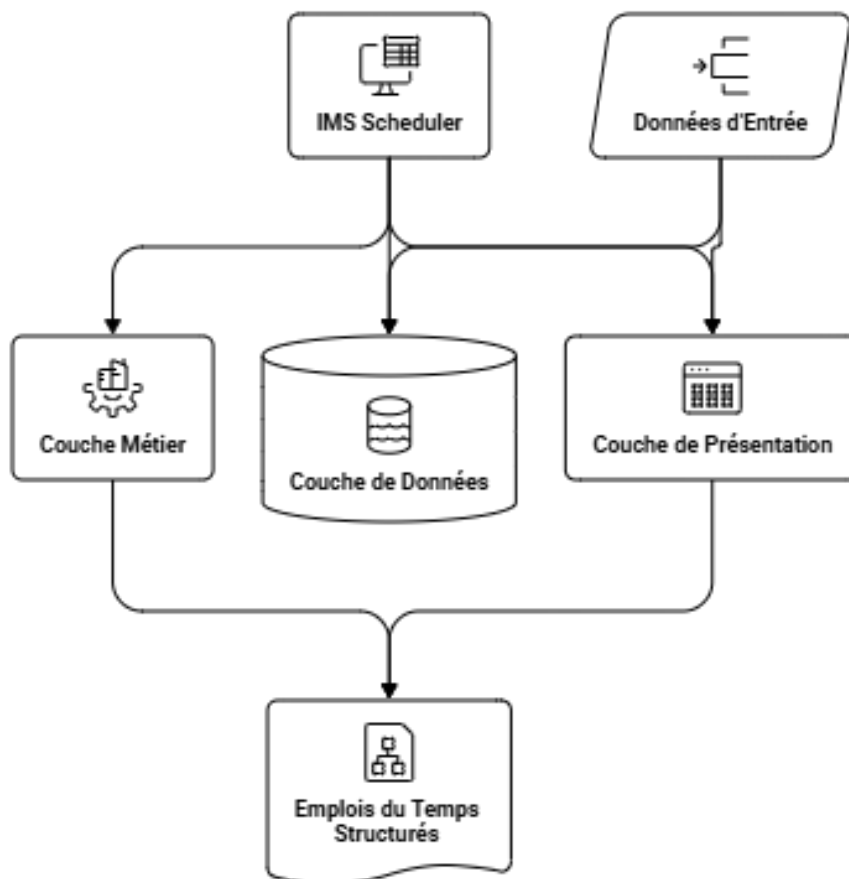


FIG. 3.1. Architecture de IMS-Scheduler.

3.1.2 Architecture logicielle

L'architecture logicielle suit le patron *Model-View-Template* (MVT) propre à Django. Le projet est découpé en six applications :

- **accounts** : authentification et gestion des rôles (administrateur, enseignant, étudiant).
- **dashboard** : tableau de bord administrateur avec déclenchement des commandes de génération.
- **room** : gestion du parc de salles (capacité, disponibilité).
- **scheduler** : noyau du système contient les modèles de planification, les vues de calendrier et toutes les commandes de gestion.
- **teacher** : modèles des enseignants, des modules et des attributions.
- **student** : modèles des étudiants et des statistiques agrégées par groupe et par promotion.

Le moteur de génération est entièrement implémenté dans l'application **scheduler**, notamment dans le fichier :

`scheduler/management/commands/generate_timetable.py` pour la génération automatique. Une variante séquentielle simplifiée, sans algorithme, est disponible dans `generate_manuelle.py` et constitue le *mode manuel* du système.

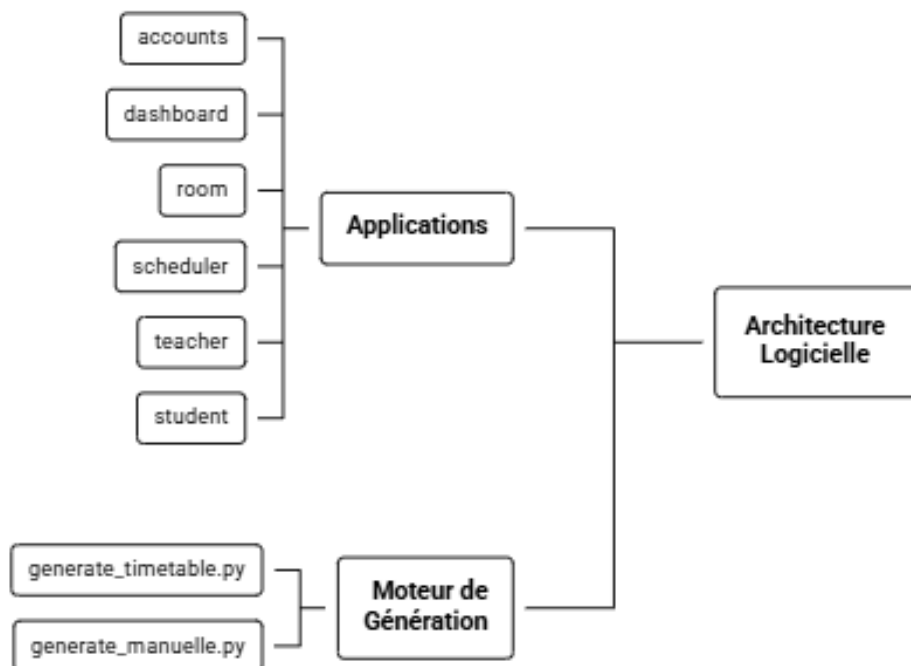


FIG. 3.2. Architecture logicielle.

3.1.3 Interaction entre Mode Manuel et Mode Automatique

Le système propose deux stratégies de planification complémentaires, accessibles depuis l'interface administrateur. Le mode manuel permet à l'administrateur d'effectuer des affectations directes et contrôlées, tandis que le mode automatique réalise une génération globale de l'emploi du temps en respectant les contraintes définies par le système.

Les deux modes partagent :

- Les mêmes modèles de données.
- Les mêmes fonctions utilitaires de normalisation et de détection de conflits exposées dans `scheduler/views.py`.
- La même logique d'allocation de salles.
- La même stratégie de modification.

3.2 Stratégie de génération automatique

Face à la complexité NP-difficile du problème de planification (définie au chapitre 1, section 1.4.4), le moteur automatique adopte une stratégie pipeline composée de trois phases successives et conditionnelles :

- **Phase 1** : Construction initiale par algorithme glouton.
- **Phase 2** : Réparation locale.
- **Phase 3** : Optimisation par algorithme génétique.

3.2.1 Phase 1- Algorithme glouton

La première phase se base sur une méthode gloutonne qui produit une solution de départ, par la suite bonifiée par des mécanismes d'optimisation et de réparation. Principe d'affectation séquentielle

L'algorithme glouton (`greedy_timetable`) parcourt les séances une par une et affecte à chacune à chaque étape, sans possibilité de retour en arrière, le couple (créneau, salle) présentant le score de pénalité minimal. Cette approche garantit une solution complète en temps linéaire et fournit au moteur un point de départ structuré pour les phases ultérieures.

Pour chaque séance, la sélection du meilleur créneau s'effectue en trois passes successives :

1. **Passe 1** : recherche d'un créneau libre respectant la contrainte de charge maximale journalière (`MAX_SEANCES_PAR_JOUR = 5`).

2. **Passe 2** : si aucun créneau libre n'est trouvé en passe 1, la contrainte de charge est relâchée.
3. **Passe 3 (repli)** : si aucun créneau sans conflit n'existe, le créneau minimisant le nombre de conflits durs est sélectionné, afin de ne laisser aucune séance non affectée.

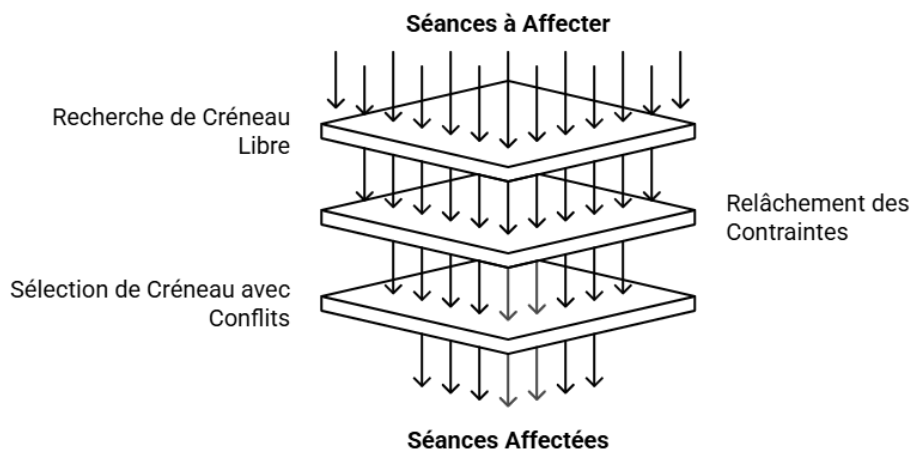


FIG. 3.3. Architecture de l'algorithme glouton.

Critère de priorisation par criticité

Les séances sont triées avant traitement selon la clé de tri multi-critères suivante :

1. **Type de cours** : les cours magistraux (C) sont traités en premier, suivis des travaux dirigés (TD), puis des travaux pratiques (TP). En effet, un cours magistral mobilise l'ensemble d'une promotion et bloque davantage de ressources, il est donc plus contraint.
2. **Nombre de salles compatibles** : une séance disposant de peu de salles éligibles (par exemple un TP en salle informatique) est placée avant une séance pouvant occuper n'importe quelle salle.
3. **Charge de l'enseignant** : un enseignant intervenant dans de nombreuses séances est plus susceptible de créer des conflits, ses séances sont donc prioritaires.
4. **Densité du groupe** : un groupe comportant de nombreuses séances dans la semaine est traité en priorité.

L'implémentation de cette stratégie de priorisation est présentée dans le Code [3.1](#).

Listing 3.1. Fonction de tri et de priorisation des séances.

```

1 def cle_tri(s):
2     priorite = {"C": 0, "TD": 1, "TP": 2}.get(normalize_type_cours(str(s
    ↪ .type_cours))), 3)
3     return (priorite,
4             len(rooms_cache.get(s.id, [])),
5             -nb_profs[s.teacher_id],
6             -nb_groupes[(s.filiere, s.annee, s.groupe)])
7
8 # ... suite du code ...
9
10 return emploi

```

Fonction de pénalité et détection des conflits

La fonction `penalite_slot` calcule un score pour chaque couple candidat (créneau, salle). Ce score intègre :

- une pénalité de 100 points par conflit dur (enseignant, salle ou groupe doublement affecté au même créneau).
- une pénalité proportionnelle à la charge journalière courante.
- une pénalité de 500 points si la limite journalière est dépassée.
- une pénalité de capacité si la salle sélectionnée est surdimensionnée par rapport à l'effectif du groupe.

L'implémentation correspondante est donnée dans le Code 3.2.

Listing 3.2. Fonction de calcul des pénalités.

```

1 def penalite_slot(seance, slot, salle, etat: Etat, charge_jour: dict) ->
    ↪ float:
2     score = etat.score_conflits(seance, slot, salle) * 100
3     charge = charge_jour[slot[0]]
4     score += charge * 4
5     if charge > MAX_SEANCES_PAR_JOUR: score += 500
6
7     # ... suite du code ...
8
9     return score

```

Les conflits sont détectés par la classe Etat, qui maintient trois index en mémoire : `profs[teacher_id][slot]`, `salles[nom_salle][slot]` et `groupes[slot][promo]`. La vérification de disponibilité (`est_libre`) consulte ces trois index simultanément avant toute affectation.

Règle de partage des cours magistraux : Une exception est prévue pour les cours magistraux : deux séances de type CM appartenant à la même filière et à la même année peuvent être affectées au même créneau et à la même salle, car elles correspondent à la même séance collective. Cette règle est implémentée dans la fonction `conflict_ressource` et dans le mécanisme de `cache_slot_cm_cache`.

À l'issue de la phase 1, le nombre de séances conflictuelles est compté par la fonction `seances_en_conflict`. Si ce nombre est nul, les phases 2 et 3 sont court-circuitées. Dans le cas contraire, le moteur déclenche automatiquement la phase 2.

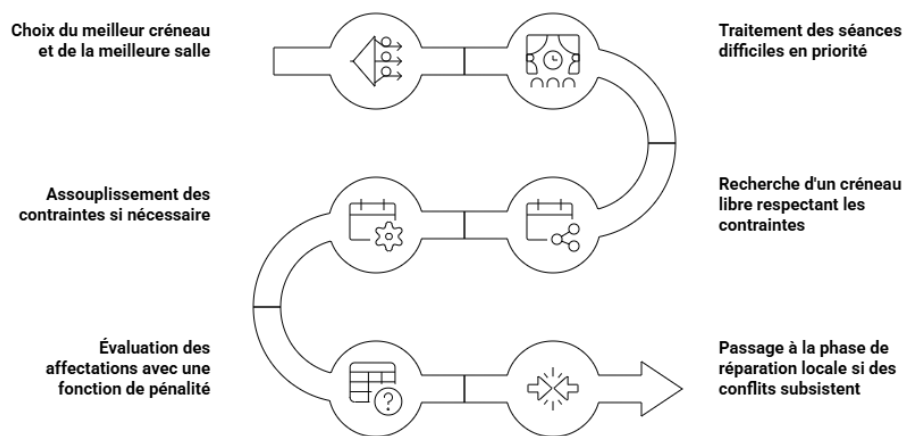


FIG. 3.4. Construction initiale par l'algorithme glouton.

3.2.2 Phase 2 - Réparation locale

La phase de réparation vise à éliminer les conflits résiduels issus de la construction initiale.

Détection des incohérences résiduelles

La fonction `seances_en_conflict` identifie l'ensemble des séances impliquées dans au moins un conflit (enseignant, salle ou groupe) après la phase 1. Elle retourne un ensemble d'identifiants de séances conflictuelles, distinguant les conflits internes au groupe traité et les conflits externes avec les filières déjà planifiées.

Mécanisme de réaffectation ciblée

La fonction `repair_timetable` effectue jusqu'à `REPAIR_PASSES = 25` passes itératives. À chaque passe :

1. L'état courant est reconstruit à partir des séances *sans conflit*, qui restent fixées.
2. Seules les séances conflictuelles sont remplacées, en appliquant la même logique de sélection par pénalité que la phase 1.
3. Si de nouveaux conflits apparaissent, ils seront traités lors de la passe suivante.

Listing 3.3. Fonction de réparation de l'emploi du temps.

```

1 def repair_timetable(emploi: dict, seances, profs_occupes=None,
  ↳ salles_occupes=None) -> dict:
2     par_id = {s.id: s for s in seances}
3     for _ in range(REPAIR_PASSES):
4         conflits = seances_en_conflit(emploi, seances, profs_occupes,
  ↳ salles_occupe)
5         if not conflits:
6             break
7         etat = Etat(profs_occupes, salles_occupes)
8         charge_jour = defaultdict(int)
9         slot_cm_cache: dict = {}
10        # ... suite du code ...
11    return emploi

```

La boucle s'arrête dès que l'ensemble des conflits est vide, ou après épuisement des passes allouées. Si des conflits persistent à l'issue des 25 passes, la phase 3 est déclenchée.

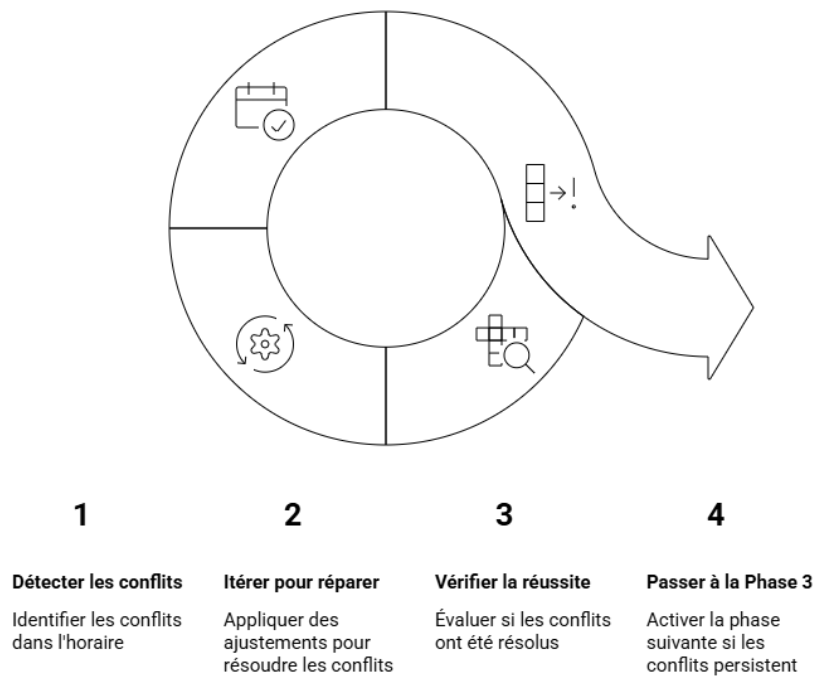


FIG. 3.5. Cycle de réparation .

3.2.3 Phase 3 - Algorithme génétique

Cette phase vise à améliorer la solution obtenue après la construction initiale et la réparation locale afin d'optimiser la qualité globale de l'emploi du temps.

Encodage chromosomique de l'emploi du temps

Chaque individu de la population est représenté comme un dictionnaire Python :

{seance_id : (jour, créneau, salle)}. Cette structure associe à chaque séance un triplet (jour, créneau horaire, salle), ce qui constitue une solution complète de l'emploi du temps. La population initiale est générée en appliquant la phase 1 suivie de la phase 2 sur GA_POPULATION = 150 perturbations aléatoires distinctes des créneaux.

Listing 3.4. Génération de la population initiale de l'algorithme génétique.

```

1 def genetic_algorithm(seances, profs_occupes, salles_occupes, stdout) ->
  ↳ tuple:
2     stdout.write(f"Population initiale ({GA_POPULATION} individus)")
3     population = []
4     for _ in range(GA_POPULATION):
5         ind = greedy_timetable(seances, profs_occupes, salles_occupes)
6         ind = repair_timetable(ind, seances, profs_occupes, salles_occupes)
7         population.append(ind)

```

```
# ... suite du code ...
```

L'initialisation de la population est réalisée selon l'algorithme présenté dans le Code 3.4. Les solutions générées servent de point de départ au processus d'évolution.

Opérateurs génétiques

Sélection par tournoi : La fonction `tournoi` sélectionne aléatoirement deux individus de la population et retient le plus performant selon la fitness. Ce mécanisme maintient une pression sélective modérée tout en préservant la diversité génétique. L'implémentation de cette procédure est présentée dans le Code 3.5.

Listing 3.5. Fonction de sélection par tournoi.

```
def tournoi(population: list, fitnesses: list) -> dict:
    i, j = random.sample(range(len(population)), 2)
    return population[i if fitnesses[i] > fitnesses[j] else j].copy()
```

Croisement : La fonction `croisement` produit un enfant (Un nouvelle emploi du temps) en choisissant aléatoirement, pour chaque séance, l'affectation de l'un ou l'autre des deux parents (probabilité 0,5). Ce croisement uniforme favorise la recombinaison de blocs d'affectations compatibles issus de solutions distinctes. L'implémentation de cet opérateur est présentée dans le Code 3.6.

Listing 3.6. Fonction de croisement uniforme.

```
def croisement( parent_a: dict, parent_b: dict) -> dict:
    return {
        sid: (parent_a[sid] if random.random() < 0.5 else parent_b.get
              ↪ (sid, parent_a[sid]))
        for sid in parent_a
    }
```

Mutation aléatoire : La fonction `mutation` modifie chaque gène (affectation d'une séance) avec une probabilité `GA_MUTATION = 0,05`. La nouvelle affectation est choisie aléatoirement parmi tous les créneaux et les salles compatibles.

L'implémentation de cet opérateur est présentée dans le Code 3.7.

Listing 3.7. Fonction de mutation.

```
def mutation(emploi: dict, seances) -> dict:
    mute = emploi.copy()
```

```

3   for sid in list(mute.keys()):
4       if random.random() > GA_MUTATION:
5           continue
6       salles = list(rooms_cache.get(sid, ["NO_ROOM"]))
7       new_slot = random.choice(ALL_SLOTS)
8       new_salle = random.choice(salles)
9       mute[sid] = (new_slot[0], new_slot[1], new_salle)
10  return mute

```

Réparation systématique : Après chaque opération de croisement ou de mutation, la fonction `repair_timetable` est appliquée à l'enfant afin de réduire le nombre de conflits durs avant son insertion dans la population suivante. Cette stratégie *Lamarckienne* accélère la convergence[32].

Fonction d'évaluation (fitness)

La fonction `calculate_fitness` évalue chaque individu selon la formule :

$$f = \frac{1}{1 + 30 \cdot C_{\text{dur}} + 0,05 \cdot S_{\text{soft}}}.$$

Les coefficients $\alpha = 30$ et $\beta = 0,05$ ont été fixés empiriquement à partir de plusieurs expérimentations afin d'assurer un équilibre entre la minimisation des conflits durs et l'optimisation des contraintes souples.

où C_{dur} représente le nombre de conflits durs détectés dans l'emploi du temps, tandis que S_{soft} correspond à un score de pénalité composite calculé comme suit :

$$S_{\text{soft}} = w_{\text{surchage}} \cdot P_{\text{surchage}} + w_{\text{trou}} \cdot P_{\text{trou}} + w_{\text{TD/C}} \cdot P_{\text{TD avant C}} + w_{\text{prof}} \cdot P_{\text{jours prof}}.$$

avec les poids suivants : $w_{\text{surchage}} = 1$, $w_{\text{trou}} = 2$, $w_{\text{TD/C}} = 1$, $w_{\text{prof}} = 2$.

- P_{surchage} : nombre de créneaux journaliers dépassant la limite de 5 séances par jour.
- P_{trou} : nombre de créneaux vides intercalés entre deux séances d'un même groupe dans une journée (calculé par `trous_par_groupe`).
- $P_{\text{TD avant C}}$: nombre de TD planifiés avant leur CM correspondant dans la même journée.
- $P_{\text{jours prof}}$: somme du nombre de jours de présence de chaque enseignant (calculé par `jours_par_prof`).

L'implémentation de cette fonction d'évaluation est présentée dans le Code 3.8.

Listing 3.8. Calcul de la fonction de fitness.

```

1 score_soft = (surcharge * POIDS_SURCHARGE+ td_avant_cm *
  ↳ POIDS_TD_AVANT_CM+ trous_par_groupe(emploi, seances) *
  ↳ POIDS_TROU+ jours_par_prof(emploi, seances) * POIDS_JOURS_PROF)
2 return 1.0 / ( 1.0+ nb_conflits * 30+ score_soft * 0.05)

```

Une valeur de fitness égale à 1.0 correspond à une solution parfaite (aucun conflit, aucune pénalité souple).

Paramètres et critères d'arrêt

Les paramètres de l'algorithme génétique sont résumés dans le tableau 3.1.

Paramètre	Valeur	Rôle
GA_POPULATION	150	Taille de la population.
GA_GENERATIONS	200	Nombre maximal de générations.
GA_MUTATION	0.05	Taux de mutation par gène.
GA_ELITE	6	Individus élitistes conservés.
REPAIR_PASSES	25	Passes de réparation par individu.

TAB. 3.1. Paramètres de l'algorithme génétique.

L'algorithme s'arrête soit après GA_GENERATIONS = 200 générations, soit dès qu'une solution sans conflit atteignant une fitness ≥ 0.90 est obtenue.

Le mécanisme d'arrêt correspondant est illustré dans le Code 3.9.

Listing 3.9. Condition d'arrêt de l'algorithme génétique.

```

1 if gen_fit > meilleur_fit:
2     meilleur_fit = gen_fit
3     meilleur_tt = population[gen_idx].copy()
4     meilleur_conf = gen_conf
5 if meilleur_conf == 0 and meilleur_fit >= 0.90:
6     stdout.write(f"Solution optimale trouvee a la generation {gen}")
7     break

```

L'élitisme conserve les GA_ELITE = 6 meilleurs individus à chaque génération, garantissant que la meilleure solution trouvée ne soit jamais perdue.

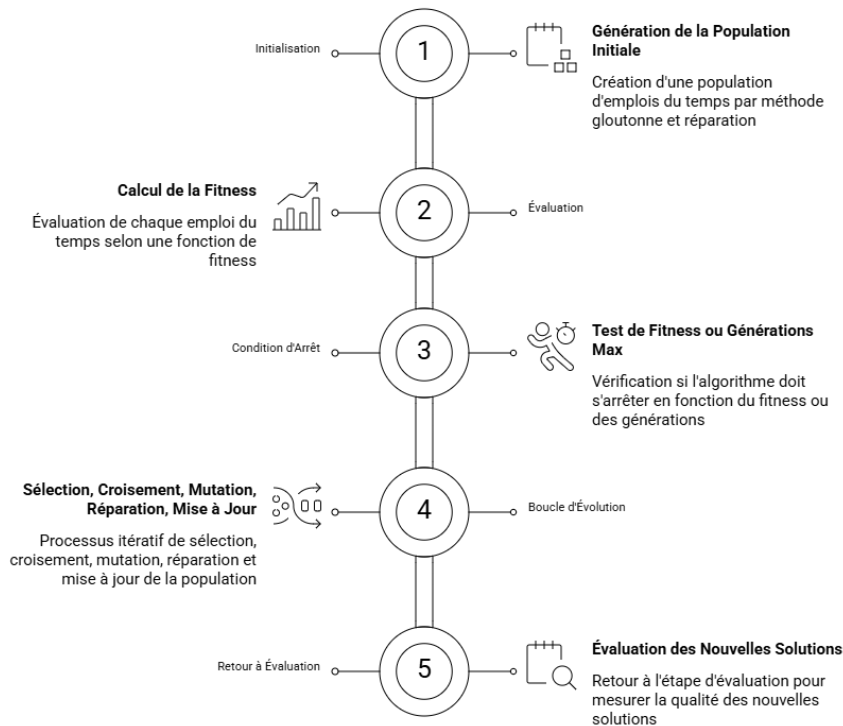


FIG. 3.6. Étapes de l'Algorithme Génétique pour l'Optimisation des Emplois du Temps .

3.2.4 Flux global et transitions entre les phases

Le schéma ci-dessous 3.7 illustre le pipeline algorithmique du projet IMS-Scheduler ainsi que l'enchaînement conditionnel des trois phases appliquées à chaque filière traitée.

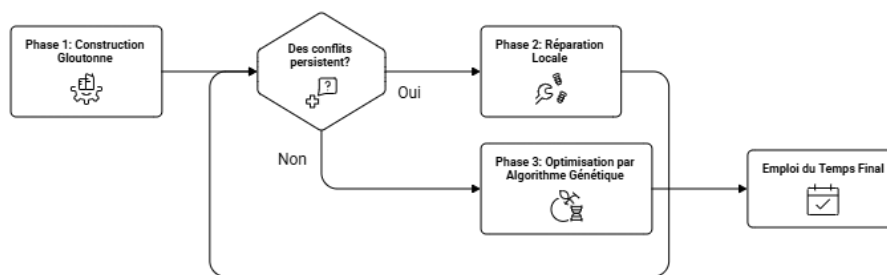


FIG. 3.7. Pipeline algorithmique du projet IMS-Scheduler.

Garanties du système Afin d'assurer la fiabilité et la qualité des emplois du temps générés, le système repose sur plusieurs garanties fonctionnelles et opérationnelles.

- **Complétude** : toutes les séances sont affectées à un créneau et une salle ; la passe 3 de l'algorithme glouton interdit tout refus d'affectation.

- **Réduction des conflits durs** : le pipeline triple réduit progressivement les conflits un rapport final (`rapport_conflits`) et liste les éventuels conflits résiduels.
- **Qualité soft** : la fonction de fitness pénalise les trous de planning, les surcharges journalières et la dispersion géographique des enseignants, favorisant des emplois du temps équilibrés.

3.3 Stratégie de génération manuelle

Le dispositif IMS-Scheduler prévoit également un mode manuel complémentaire où l'acteur peut réaliser une planification interactive avec la détection automatique des conflits.

3.3.1 Affectation manuelle ressources

Le processus d'affectation manuelle repose sur des index de disponibilité des enseignants, salles et groupes mis à jour en temps réel. Les séances impossibles à placer sont automatiquement enregistrées dans une liste de séances non affectées afin de faciliter leur traitement ultérieur. Cette liste permet d'identifier rapidement les séances nécessitant une intervention manuelle.

3.3.2 Principe de l'affectation séquentielle manuelle

Le mode manuel, implémenté dans `generate_manuelle.py`, repose sur le même algorithme glouton séquentiel que la phase 1 du mode automatique, mais *sans* les phases de réparation ni d'optimisation génétique. Les séances sont affectées une par une dans l'ordre de leur apparition, selon les créneaux et les salles disponibles au moment du traitement.

La classe `EtatManuel` maintient les mêmes trois index de disponibilité (profes, salles, groupes) et applique les mêmes règles de détection de conflits que la classe `Etat` du mode automatique. Les séances pour lesquelles aucun créneau libre n'existe sont consignées dans la liste `non_assignes` et signalées à l'administrateur dans le rapport final.

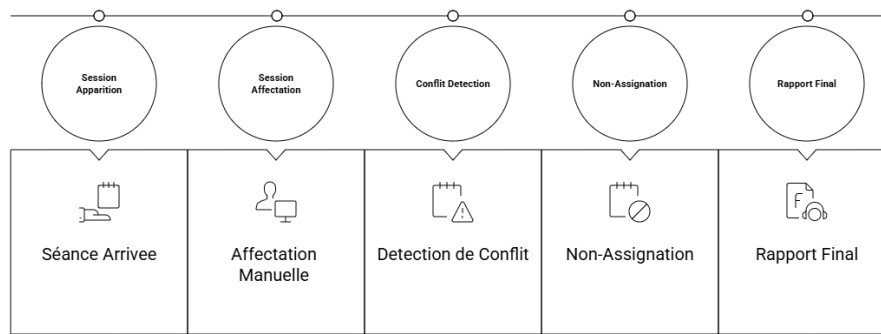


FIG. 3.8. Pipeline du mode manuel.

3.3.3 Détection des conflits en temps réel

La détection des conflits en mode manuel est effectuée lors de la sauvegarde des affectations. Le système vérifie automatiquement les contraintes liées aux enseignants, aux salles et aux groupes, puis affiche un rapport détaillant les conflits détectés, notamment les doubles affectations, les chevauchements horaires ou les indisponibilités de ressources.

3.4 Conclusion

Ce chapitre a présenté l'architecture complète du système IMS-Scheduler. La couche logicielle MVT de Django structure l'application en modules cohérents, tandis que le moteur de génération automatique met en œuvre un pipeline hybride en trois phases : construction gloutonne, réparation locale et optimisation génétique. Ce pipeline garantit qu'aucune séance ne reste non affectée, tout en minimisant les conflits durs et en améliorant la qualité globale de l'emploi du temps via la fonction de fitness multi-critère. Le mode manuel, quant à lui, offre à l'administrateur un contrôle direct sur les affectations, avec détection des conflits en temps réel. Le chapitre 4 présentera la conception détaillée du système IMS-Scheduler, la structure de la base de données ainsi que l'analyse et la validation des résultats obtenus.

CHAPITRE 4

CONCEPTION DE L'OUTIL ET ANALYSE COMPARATIVE

Un système n'a de valeur que s'il est capable de démontrer, par les faits, l'efficacité annoncée. Ce dernier chapitre s'attèle à cette démonstration en retraçant les conditions concrètes de réalisation du projet, des outils employés jusqu'aux résultats mesurés sur le terrain. Les performances du système y sont confrontées à celles de la méthode manuelle qu'il vise à remplacer, à travers des indicateurs objectifs et des retours d'utilisateurs. Ce bilan permet enfin de mesurer le chemin parcouru et d'esquisser les évolutions à venir.

4.1 Environnement de développement et base de données

Après avoir décrit l'architecture du système et les politiques de génération adoptées, ce chapitre présente la mise en œuvre réelle de la solution IMS-Scheduler.

4.1.1 Langages utilisés

Le développement du système **IMS-Scheduler** repose sur un ensemble de technologies soigneusement sélectionnées afin de répondre aux exigences fonctionnelles et non fonctionnelles du projet.

Python constitue le langage principal de développement. Il a été retenu pour sa lisibilité, la richesse de son écosystème scientifique ainsi que sa compatibilité native avec les bibliothèques d'optimisation combinatoire. La version **3.12** ou supérieure a été adoptée afin de bénéficier des dernières améliorations en termes de performance, de sécurité et d'évolution syntaxique.

SQL et ORM Django SQL est exploité de manière transparente à travers le mécanisme ORM (*Object-Relational Mapping*) fourni par le framework **Django**. Ce mécanisme permet

de traduire automatiquement les opérations réalisées sur les modèles Python en requêtes SQL optimisées. Cette abstraction garantit une indépendance vis-à-vis du système de gestion de base de données utilisé.

Dans le cadre du développement et des tests, le moteur **SQLite** a été retenu pour sa simplicité d'intégration, tandis qu'une migration vers **PostgreSQL** demeure envisageable en environnement de production afin de bénéficier de meilleures performances et d'une meilleure gestion de la concurrence.

Traitement des données L'importation et le prétraitement des données pédagogiques s'appuient sur les bibliothèques **Pandas** et **OpenPyXL**. Ces outils permettent la lecture, la transformation et la validation des fichiers `.xlsx` contenant les informations académiques de l'établissement, telles que les enseignants, les modules, les groupes pédagogiques et les volumes horaires.

4.1.2 Frameworks pour l'interface utilisateur

L'interface utilisateur du système **IMS-Scheduler** est développée à l'aide du framework **Django**, dans sa version **6**, reposant sur le patron architectural *MVT* (*Model-View-Template*).

Ce choix présente plusieurs avantages déterminants pour le projet. Tout d'abord, Django propose un système natif de gestion de l'authentification et des permissions, permettant la mise en place d'un contrôle d'accès basé sur les rôles des utilisateurs. Ensuite, son moteur de templates HTML offre une grande flexibilité pour la génération dynamique des interfaces, notamment pour l'affichage automatisé des grilles d'emploi du temps. Enfin, son interface d'administration intégrée (`/admin/`) facilite la gestion, la consultation et la modification des données de manière centralisée.

Les templates HTML sont enrichis par des feuilles de style **CSS** personnalisées, conférant à l'application une identité visuelle sombre et moderne, adaptée à un usage académique et professionnel.

Par ailleurs, l'application est entièrement accessible depuis un navigateur web standard, sans nécessiter l'installation de logiciels supplémentaires côté client. Cette approche garantit une accessibilité multiplateforme et simplifie considérablement le déploiement ainsi que l'utilisation quotidienne du système.

4.1.3 Justification des choix technologiques

Le choix de la stack technologique **Django/Python/SQLite** repose sur plusieurs critères complémentaires. Sur le plan de la performance algorithmique, Python offre une intégration directe avec les bibliothèques de calcul scientifique (`random`, `collections`, `copy`) nécessaires à l'implémentation de l'algorithme génétique et de l'heuristique gloutonne. Sur le plan de la maintenabilité, la structure modulaire de Django en applications indépendantes

(accounts, scheduler, teacher, student, room, dashboard) favorise la séparation des responsabilités et facilite les évolutions futures du système.

Critère	Choix retenu	Justification
Langage principal	Python 3.12+	Performance, lisibilité et richesse de l'écosystème.
Framework web	Django 6	Architecture MVT, ORM intégré, interface d'administration et gestion des rôles.
Base de données	SQLite	Solution légère, sans configuration complexe, adaptée au développement et aux tests.
Import des données	Pandas + OpenPyXL	Lecture, traitement et validation des fichiers Excel IMSI.
Frontend	HTML/CSS (Templates Django)	Interface responsive accessible sans dépendances côté client.
Algorithme	Python	Portabilité, maintenabilité et intégration native avec Django.

TAB. 4.1. Justification des choix technologiques du système IMS-Scheduler.

4.2 Implémentation de l'interface utilisateur

Après avoir présenté l'architecture du système, l'étude s'intéresse aux interfaces qui ont été développées pour permettre à l'utilisateur d'interagir avec le système.

4.2.1 Diagramme de gestion des utilisateurs

Cette sous-section présente le diagramme de gestion des utilisateurs du système développé. Ce diagramme illustre le processus d'authentification ainsi que les différentes fonctionnalités accessibles selon le type d'utilisateur connecté. Après la vérification des informations de connexion, le système redirige automatiquement l'utilisateur vers l'interface correspondant à son rôle.

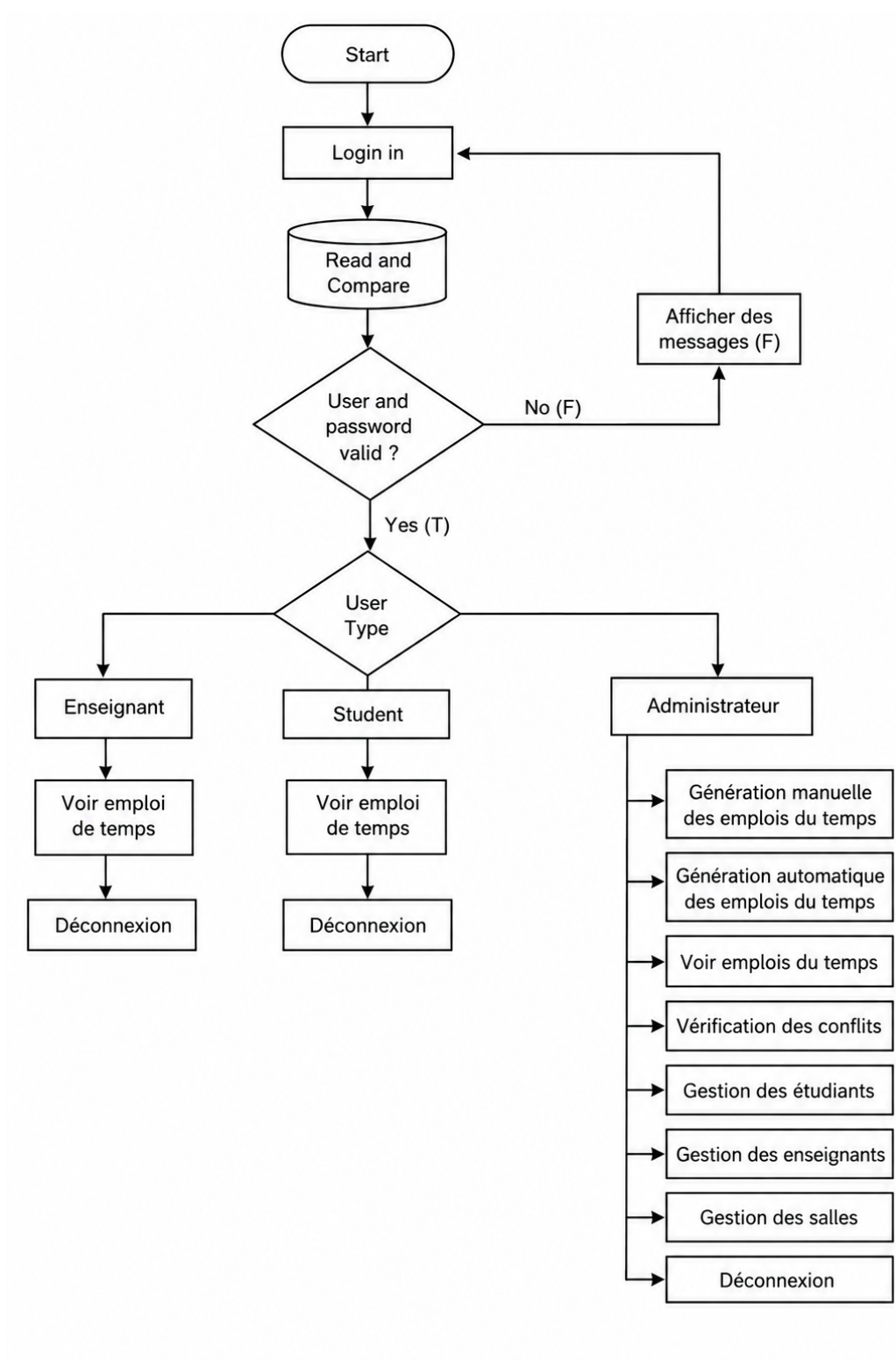


FIG. 4.1. Diagramme de gestion des utilisateurs du système IMS-Scheduler.

Le système distingue principalement trois types d'utilisateurs :

- **Enseignant** : consultation des emplois du temps.
- **Étudiant** : consultation des emplois du temps de la classe.

- **Administrateur** : gestion complète du système, incluant la génération automatique et manuelle des emplois du temps, la vérification des conflits ainsi que la gestion des étudiants, enseignants et salles.

Afin d'illustrer l'implémentation réelle du système, la figure suivante présente l'interface de connexion développée pour l'application.

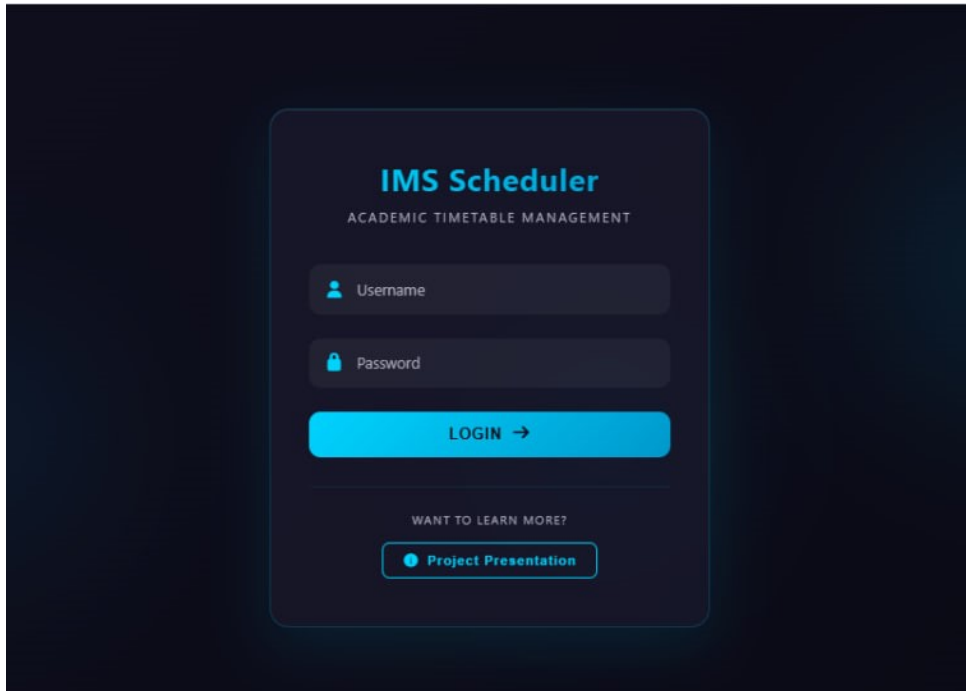


FIG. 4.2. Interface de connexion du système IMS-Scheduler.

4.2.2 Interface de saisie manuelle des créneaux

L'interface de saisie manuelle, accessible depuis le tableau de bord administrateur via le bouton «Générer manuellement», permet de planifier les séances à l'aide d'un formulaire tabulaire. Les informations essentielles de chaque séance sont affichées en lecture seule, tandis que les champs modifiables permettent de sélectionner le jour, le créneau horaire et la salle. Cette interface adopte une approche de minimalisme cognitif afin de simplifier la prise de décision de l'administrateur. La figure illustre cet écran de saisie.

N°	Module	Professeur	Étudiant	Année	Sem	Groupe	Type	Type TP	Jour	Créneau	Salle
1	Thermodynamique	BELOURA	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
2	Initiation à la programmation	REQUIEL_YSSAD	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
3	Analyse 2	HEBAL	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
4	Algèbre 2	HEBAL	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
5	Logiciels opensource	KATAB	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
6	Electricité et magnétisme	HAOUKADDOUR	ELM - Licence	L1	S2	0	C	-	-- Jour --	-- Créneau --	-- Salle --
7	Thermodynamique	TALBI	ELM - Licence	L1	S2	1	TD	-	-- Jour --	-- Créneau --	-- Salle --
8	Initiation à la programmation	REQUIEL_YSSAD	ELM - Licence	L1	S2	1	TP	INF	-- Jour --	-- Créneau --	-- Salle --
9	Logiciels opensource	REQUIEL_YSSAD	ELM - Licence	L1	S2	1	TP	INF	-- Jour --	-- Créneau --	-- Salle --
10	Analyse 2	BOULEKHRAH	ELM - Licence	L1	S2	1	TD	-	-- Jour --	-- Créneau --	-- Salle --

FIG. 4.3. Interface de saisie manuelle.

4.2.3 Interface de lancement de la génération automatique

Le tableau de bord administrateur constitue le point d'entrée principal du système de génération des emplois du temps. Il fournit une vue globale des ressources disponibles, notamment le nombre d'étudiants, d'enseignants, de modules et de salles. Cette interface permet à l'administrateur de vérifier les données avant le lancement de la génération. Plusieurs actions sont disponibles, telles que la génération automatique, la vérification, la consultation et la génération manuelle des emplois du temps.

IMS-SCHEDULER		
<p>Générez et vérifiez votre emploi du temps</p> <p>Appuyez sur Générer pour exécuter et sur Vérifier pour analyser les conflits.</p> <p> Générer automatiquement Vérifier l'emploi du temps Voir l'emploi du temps Générer manuellement </p>		
<p>Étudiants</p> <p>1347</p> <p>Nombre total d'étudiants</p>	<p>Enseignants</p> <p>90</p> <p>Modules associés: 71</p>	<p>Salles</p> <p>34</p> <p>Disponibles: 34</p>
<p>Rapports et vérification</p> <p>Analyse des conflits de salle, professeurs, capacité et créneaux vides.</p>		

FIG. 4.4. Interface de lancement de la génération automatique.

4.2.4 Gestion des Interactions Utilisateur-Système

L'interface de l'application est conçue autour d'une logique de dialogue continu entre l'utilisateur et le système. Chaque action réalisée par l'utilisateur déclenche une réponse immédiate du système, qu'il s'agisse d'une validation, d'une alerte ou d'une mise à jour visuelle de l'emploi du temps.

Flux d'interaction général

Le système gère principalement trois catégories d'interactions destinées à l'administrateur :

- **Interactions de saisie** : l'utilisateur renseigne ou modifie des données telles que les créneaux ou les salles. Le système réagit en temps réel par une validation ou un signal d'alerte.
- **Interactions de validation** : après chaque saisie, IMS Scheduler vérifie automatiquement les contraintes dures et affiche immédiatement le résultat de la vérification.
- **Interactions de génération automatique** : l'utilisateur configure les paramètres du moteur algorithmique, lance la génération automatique et consulte les résultats progressifs ou finaux.

Action utilisateur	Traitement système	Réponse affichée
Saisie d'un créneau manuel et enregistrer	Vérification des contraintes dures	Confirmation verte / alerte rouge.
Modification d'une séance existante et enregistrer	Recalcule local des conflits	Mise à jour instantanée de la grille.
Suppression d'une séance et enregistrer	Libération du créneau et de la salle	Slot remis à disponible.
Lancement de la génération automatique	Déclenchement du pipeline Gloutonne → réparation → AG	Génération automatique de l'emploi du temps.
Consultation des résultats	Lecture de la base de données résultat	Emploi du temps par filière + métriques.
Suppression de l'ancien emploi du temps	Suppression les créneaux ,salles	Aucun emploi du temps affiché.

TAB. 4.2. Interactions utilisateur-système et réponses associées.

Le système adopte une logique d'interaction dynamique entre l'utilisateur et le moteur de planification. Chaque action effectuée par l'administrateur déclenche une vérification immédiate des contraintes ainsi qu'une mise à jour visuelle de l'interface. Cette approche améliore la réactivité du système et facilite la correction progressive des affectations lors des opérations de planification manuelle ou automatique.

4.3 Conception de la base de données

Afin de permettre une gestion des données cohérente et centralisée, il a été nécessaire de concevoir un modèle pour la base de données qui servira de support à IMS-Scheduler.

4.3.1 Modèle Conceptuel des Données (MCD)

Le **Modèle Conceptuel des Données (MCD)** du système représente l'ensemble des entités métier ainsi que leurs associations, indépendamment de toute considération liée à l'implémentation technique [31].

Il constitue une étape essentielle de la conception, permettant de formaliser la structure logique des données manipulées par le système et de garantir la cohérence des interactions entre les différentes composantes fonctionnelles.

Les associations principales se structurent comme suit :

Entité / Relation	Description
ENSEIGNANT / MO-DULE	Relation entre enseignants et modules selon le type de cours, le semestre, la filière et le groupe.
SÉANCE	Association d'un enseignant, d'un module et d'un groupe pour une séance donnée.
PLAN	Ensemble des créneaux générés pour une filière, une année, un semestre et un groupe.
ENTRÉE D'EMPLOI DU TEMPS	Planification d'une séance dans un créneau horaire et une salle.
ÉTUDIANT	Étudiant rattaché à une filière et un groupe avec gestion des effectifs.

TAB. 4.3. Entités et relations du MCD d'IMS-Scheduler.

4.3.2 Modèle Logique des Données (MLD)

Le Modèle Logique des Données (MLD) traduit le MCD en tables relationnelles concrètes telles qu'elles sont implémentées dans les modèles Django et matérialisées dans la base SQLite [32]. Chaque entité du MCD correspond à une table, tandis que l'association plusieurs-à-plusieurs entre enseignant et module est représentée par la table de jonction `teacher_teachermodule`.

Un constat important de l'implémentation réelle est que les entités **FILIÈRE**, **ANNÉE** et **GROUPE** ne sont pas matérialisées sous forme de tables dédiées. Ces informations sont stockées directement dans les tables concernées sous forme de champs texte ou entier. Ce choix de dénormalisation simplifie les requêtes et réduit la complexité des jointures, au prix d'une certaine redondance des données.

Les clés étrangères définies dans Django assurent l'intégrité référentielle entre les différentes entités du système, notamment les enseignants, modules, séances et plans.

4.3.3 Structure des tables

La base de données est implémentée sous forme de tables correspondant aux entités et relations du système, afin d'assurer la gestion des données et la planification des emplois du temps.

Table	Clé primaire	Clés étrangères	Attributs notables
teacher_teacher	id	user_id → auth_user	nom, prenom, nom_dans_EDT, grade (MAA/MAB/MCA/MCB/Pr), charge_affectee, charge_minimale, heures_sup
teacher_module	id	—	nom (intitulé du module)
teacher_teachermodule	id	teacher_id → teacher_teacher module_id → teacher_module	type_cours (C/TD/TP), type_TP (LAB/EAD/INF), nb, semestre, filiere, annee, groupe
room_room	id	—	nom_salle, capacite, disponible
scheduler_seance	id	teacher_id → teacher_teacher module_id → teacher_module	filiere, annee, groupe, semestre, type_cours, type_TP
scheduler_timetableplan	id	—	filiere, annee, semestre, groupe, created_at
scheduler_timetable	id	seance_id → scheduler_seance plan_id → scheduler_timetableplan	jour, heure_debut, heure_fin, salle
student_promotionstats	id	—	filiere, annee, total_etudiants
student_groupestats	id	—	filiere, annee, groupe, total_etudiants
student_student	id	user_id → auth_user	nom_complet, situation (D/N/T), groupe, filiere, annee

TAB. 4.4. Structure des tables principales de la base de données IMS-Scheduler.

4.4 Analyse du mode manuel du système IMS-Scheduler

Après la présentation des composantes techniques du système, cette section analyse le mode manuel afin d'évaluer son efficacité et d'identifier ses limites dans la planification académique.

4.4.1 Analyse de la gestion des conflits

Le mécanisme de détection des conflits du mode manuel représente une amélioration notable par rapport au processus traditionnel basé sur des fichiers Excel. En vérifiant automatiquement les contraintes liées aux enseignants, aux salles et aux groupes à chaque affectation, le système réduit efficacement le risque d'incohérences dans la planification.

Cependant, cette gestion reste **corrective et non préventive**. Les conflits sont détectés après leur apparition, obligeant l'administrateur à effectuer des corrections successives. De plus, la résolution d'un conflit peut en créer de nouveaux, générant un phénomène de propagation difficile à contrôler.

Ce comportement confirme qu'une approche manuelle atteint rapidement ses limites face à un problème de planification fortement contraint.

4.4.2 Évaluation des performances opérationnelles

Une session de planification complète a été réalisée sur un semestre entier afin d'évaluer les performances du mode manuel. Les résultats sont présentés dans le tableau suivant :

Itération	Durée	Conflits résiduels
Saisie initiale	2h 56min	70
Correction 1	1h 13min	43
Correction 2	45min	13
Correction 3	10min	5
Correction 4	3min	0
Total	⌘ 5h 07min	0

TAB. 4.5. Résultats de la session de planification manuelle.

Trois constats ressortent de cette analyse. D'abord, la saisie initiale produit 70 conflits, confirmant qu'il est impossible pour un opérateur humain d'anticiper toutes les contraintes simultanément. Ensuite, la correction suit une **progression décroissante** : les premiers ajustements éliminent les conflits les plus simples, tandis que les conflits résiduels deviennent de plus en plus complexes à résoudre. Enfin, le temps total de **5 heures et 7 minutes** pour un seul semestre représente une charge administrative importante.

Au-delà du temps, le mode manuel ne permet pas d'optimiser les **contraintes souples** comme l'équilibrage des charges ou la minimisation des trous dans les emplois du temps. Il produit une solution valide, mais rarement une solution optimale.

4.5 Analyse du mode automatique

Après l'étude du mode manuel, le mode automatique sera analysé et on tentera d'évaluer les performances des différentes approches algorithmiques implantées dans IMS-Scheduler.

4.5.1 Analyse des configurations algorithmiques

Avant de présenter l'analyse de la configuration hybride finale adoptée par le système, plusieurs configurations algorithmiques ont été évaluées séparément afin d'étudier :

- leur capacité à supprimer les conflits.
- leur rapidité d'exécution.
- leur stabilité face aux variations aléatoires.
- leur aptitude à améliorer la qualité globale des emplois du temps.
- leur comportement face aux contraintes complexes.

Cette démarche progressive permet d'identifier les limites propres à chaque approche et de justifier scientifiquement le choix de la solution hybride intégrée dans application.

Configurations des algorithmiques étudiées

a) Algorithme génétique

L'algorithme génétique, testé seul sur la filière GI -Licence L1 S2 (44 séances, 30 individus, 80 générations et un taux de mutation de 8 %), parvient à réduire les conflits de 12 à 6 en 34 secondes, soit une amélioration de 50 %. Toutefois, l'absence d'initialisation intelligente, la vastitude de l'espace de recherche ainsi que la convergence prématurée de la population empêchent l'élimination totale des conflits. Ces résultats mettent en évidence les limites intrinsèques de l'algorithme génétique lorsqu'il fonctionne sans modules complémentaires.

b) Approche basée sur l'algorithme glouton

L'algorithme gloutonne constitue la première phase du pipeline. Il planifie l'ensemble des 425 séances en 11.29 secondes grâce à sa stratégie locale sans retour arrière, ce qui en fait l'approche la plus rapide. Cependant, cette rapidité se fait au détriment de la qualité des solutions obtenues. Les résultats révèlent la présence de 4 conflits résiduels, 117 trous et 354 jours professeurs, traduisant une optimisation insuffisante.

Les conflits touchent surtout les filières aux ressources partagées ou fortement contraintes. La stabilité du gloutonne se dégrade avec la complexité, ce qui, malgré sa rapidité, rend son utilisation seule insuffisante et justifie l'ajout des phases réparation et AG.

c) Approche gloutonne + réparation

La configuration gloutonne + réparation permet la planification des 425 séances en 11.26 secondes, soit un temps d'exécution pratiquement identique à celui du gloutonne seul. L'apport du module réparation est particulièrement visible sur la qualité des résultats : le nombre de conflits est réduit de 4 à un seul conflit résiduel, tandis que le nombre de trous diminue à 110 et que le total des jours professeurs atteint 358.

Ces résultats confirment que le module réparation améliore efficacement la solution initiale produite par le gloutonne sans dégrader les performances temporelles. Toutefois, la présence d'un conflit résiduel ainsi qu'un nombre encore important de trous montrent que cette configuration ne permet pas d'obtenir une solution totalement admissible, ce qui justifie l'intégration de l'algorithme génétique comme troisième phase du pipeline.

d) Approche hybride (gloutonne + réparation + Algorithme Génétique)

Le pipeline hybride G+R+AG repose sur une répartition claire des responsabilités entre ses trois phases. Le gloutonne assure la construction rapide de la solution initiale, le module réparation prend en charge la correction des conflits résiduels, tandis que l'algorithme génétique intervient en dernière phase afin d'optimiser les contraintes souples sur un espace de recherche déjà réduit.

cette organisation séquentielle favorise une convergence plus rapide vers des solutions de meilleure qualité tout en maintenant un temps d'exécution global maîtrisé.

Critère	Valeur
Séances planifiées	425
Conflits	0
Trous	114
Jours enseignants	352
Temps d'exécution	12,51 s
Fitness globale	0,2594

TAB. 4.6. Résultats globaux de la configuration G+R+AG.

4.5.2 Paramétrage de l'Algorithme Génétique dans le pipeline hybride

Plusieurs configurations des paramètres principaux –GA_POPULATION, GA_GENERATIONS et GA_MUTATION – ont été testées afin d'étudier leur impact sur les performances globales du système.

Les sept *runs* expérimentaux couvrent un large spectre de configurations, allant d'une configuration légère (Run 3 : 10 individus, 20 générations) à une configuration maximale

(Run 7 : 150 individus, 200 générations), en passant par la configuration standard retenue (Run 1 : 30 individus, 80 générations et 8 % de mutation).

Le tableau 4.7 récapitule l'ensemble des configurations testées.

Run	GA_POPU	GA_GENERA	GA_MUTA	Profil
Run 1	30	80	8%	Configuration par défaut
Run 2	30	80	8%	Exécution aléatoire (random.shuffle)
Run 3	10	20	5%	Configuration légère
Run 4	50	80	15%	Mutation élevée
Run 5	30	80	8%	Retour à la configuration par défaut
Run 6	100	150	8%	Configuration lourde
Run 7	150	200	5%	Configuration maximale

TAB. 4.7. Configurations de paramétrage de l'algorithme génétique

Les résultats obtenus montrent que l'augmentation excessive de la taille de la population ainsi que du nombre de générations entraîne une hausse importante du temps de calcul sans amélioration proportionnelle de la qualité finale des solutions produites. Ainsi, la configuration standard (Run 1) constitue le meilleur compromis entre performances de génération et coût computationnel.

Fonction de fitness

L'évaluation des solutions générées repose sur une fonction de *fitness* permettant de mesurer la qualité globale des emplois du temps produits. Cette fonction agrège les pénalités associées aux violations des contraintes dures et souples, pondérées selon leur niveau de criticité. Plus la valeur de fitness est faible, meilleure est la solution obtenue.

Le tableau 4.7 présente les valeurs de fitness obtenues par filière pour chacune des configurations définies dans le tableau 4.5.

Filière (Séances)	Run 1	Run 2	Run 3	Run 4	Run 5
GI - L1 S2 (44)	0,0043	0,0038	0,0196	0,0164	0,0179
ELM - L2 S4 (43)	0,0039	0,0038	0,0149	0,0182	0,0167
HSI - L1 S2 (43)	0,0038	0,0038	0,0177	0,0164	0,0174
ELM - L1 S2 (40)	0,0043	0,0049	0,0213	0,0189	0,0213
GI - L2 S4 (39)	0,0046	0,0047	0,0196	0,0158	0,0185
ELM - M1 S2 (39)	0,0048	0,0048	0,0158	0,0187	0,0215

TAB. 4.8. Valeurs de fitness par filière selon les configurations du Tableau 4.7.

L'analyse des résultats par filière confirme la supériorité de la configuration standard. Les Runs 1 et 2 produisent systématiquement les valeurs de fitness les plus faibles, comprises entre 0,0038 et 0,0049, traduisant une très bonne qualité de planification.

À l'inverse, les configurations légère (Run 3) et à mutation élevée (Run 4) génèrent des valeurs significativement plus élevées, atteignant jusqu'à 0,0215 pour la filière ELM M1 S2. Ces résultats mettent en évidence une convergence insuffisante ou une instabilité provoquée par un taux de mutation trop important.

Par ailleurs, la variabilité observée entre les différents runs pour une même filière illustre la sensibilité de l'algorithme à l'ordre aléatoire de traitement initial introduit par le gloutonne.

4.5.3 Analyse des Performances de l'Approche Hybride

Cette section présente les résultats détaillés obtenus par la configuration hybride G+R+AG pour l'ensemble des 13 filières planifiées. L'analyse porte principalement sur les indicateurs liés aux conflits, à la compacité des emplois du temps ainsi qu'aux temps d'exécution observés lors de la génération automatique.

Filière / Promotion	Séances	Conflits	Trous	Jours profs	Temps (s)
GI - Licence L1 S2	44	0	16	36	1,34
ELM - Licence L2 S4	43	0	14	37	1,54
HSI - Licence L1 S2	43	0	7	35	1,11
ELM - Licence L1 S2	40	0	13	32	1,16
GI - Licence L2 S4	39	0	11	33	1,07
ELM - Master M1 S2	39	0	9	35	1,17
ELM - Licence L3 S6	37	0	7	29	0,93
HSI - Licence L2 S4	37	0	5	33	1,30
HSI - Master M1 S2	26	0	9	23	0,57
HSI - Licence L3 S6	25	0	5	20	0,97
GI - Master M1 S2	24	0	5	21	0,52
GI - Licence L3 S6	27	0	4	24	0,71
SYNTHÈSE GLOBALE	425	0	114	352	12,51

TAB. 4.9. Résultats détaillés de la configuration hybride G+R+AG.

Synthèse

Les résultats expérimentaux obtenus confirment la supériorité de l'approche hybride G+R+AG, qui permet de générer des emplois du temps totalement admissibles et sans conflit grâce à la complémentarité de ses trois composantes.

Cette performance est obtenue avec un coût computationnel limité, le temps d'exécution passant seulement de 11,29 à 12,51 secondes par rapport à l'approche gloutonne seule, tout en assurant une meilleure stabilité entre les différents *runs* ainsi qu'une répartition plus équilibrée des charges enseignantes.

Ces résultats montrent que la qualité d'un emploi du temps ne dépend pas uniquement de l'absence de conflits, mais également de l'optimisation des contraintes souples, élément essentiel pour améliorer le confort pédagogique et garantir l'opérationnalité du système dans un contexte universitaire réel.

Comparaison globale des configurations algorithmiques

Cette section présente une comparaison globale des différentes configurations algorithmiques évaluées dans le cadre du système IMS-Scheduler. L'objectif est d'analyser les performances respectives de chaque approche selon plusieurs critères essentiels, notamment le nombre de conflits générés, le taux d'admissibilité des solutions, la compacité des emplois du temps, la stabilité des exécutions ainsi que les temps de calcul observés.

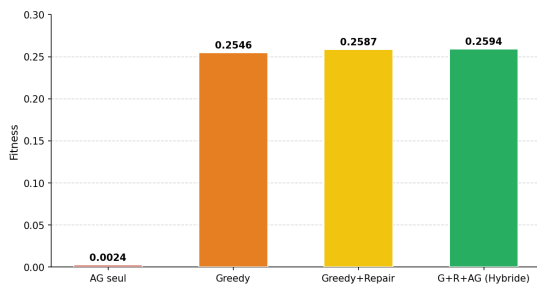
L'analyse comparative permet ainsi de mettre en évidence les avantages et les limites de chaque méthode, tout en justifiant le choix de l'approche hybride G+R+AG comme solution finale retenue pour la génération automatique des emplois du temps universitaires. Afin d'évaluer objectivement les performances des différentes configurations algorithmiques proposées, une analyse comparative multicritère a été menée. Les critères retenus couvrent trois dimensions fondamentales de la qualité d'un emploi du temps universitaire : l'admissibilité des solutions générées, la qualité d'optimisation mesurée à travers la fonction de *fitness*, ainsi que la compacité des plannings produits.

Les résultats obtenus sont présentés dans le tableau 4.10, puis illustrés à l'aide des figures correspondantes afin de faciliter l'interprétation comparative des performances de chaque configuration. Le tableau suivant confronte les différentes configurations algorithmiques selon l'ensemble des critères d'évaluation retenus. La colonne *Meilleur* identifie la configuration dominante pour chaque critère analysé.

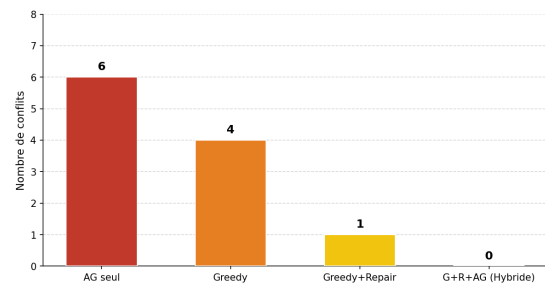
Critère d'évaluation	AG seul	gloutonne seul	gloutonne + réparation	G + R + AG	Meilleur
Conflits totaux (nb)	6 pour 44 séances	4	1	0	G+R+AG
Taux d'admissibilité (%)	0 % (44 séances)	77 %	92 %	100 %	G+R+AG
Trous totaux (nb)	–	117	110	114	G+réparation
Jours profs cumulés (nb)	–	354	358	352	G+R+AG
Fitness globale moyenne	0.0024	0.2546	0.2587	0.2594	G+R+AG
Temps d'exécution total (s)	34	11.29	11.26	12.51	G+réparation
Filières sans conflits	0/1	10/13	12/13	13/13	G+R+AG
Stabilité inter-exécutions	Variable	Variable	Variable	Stable	G+R+AG

TAB. 4.10. Comparaison des performances des configurations algorithmiques.

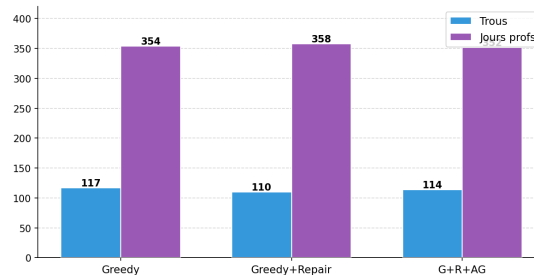
L'approche hybride G+R+AG s'impose comme la meilleure configuration : le gloutonne est rapide mais conflictuel, le réparation améliore sans garantir l'admissibilité, et le GA seul peine face à la complexité. Leur combinaison compense mutuellement leurs faiblesses. Résultat : 13 filières planifiées sans conflit, bonne compacité et charges enseignantes équilibrées, validant le déploiement en environnement universitaire réel.



(a) Fitness globale par configuration algorithmique .



(b) Conflits totaux par configuration algorithmique .



(c) Trous et jours professeurs cumulés par configuration (contraintes souples).

4.6 Analyse comparative manuel VS automatique

Cette section présente une comparaison globale entre le mode manuel et le mode automatique .

4.6.1 Comparaison du temps de génération

L'analyse finale montre une différence très importante entre les deux modes de planification. Le processus manuel a nécessité environ 5 h 07 min pour obtenir un emploi du temps sans conflit, en passant par plusieurs phases successives de correction. À l'inverse, le mode automatique a généré directement une solution admissible en seulement 12,51 secondes.

Ainsi, le système automatique permet une réduction du temps de traitement d'environ 99,93 %, avec un facteur d'accélération proche de $1\,472\times$ par rapport au mode manuel. Ces résultats mettent clairement en évidence l'efficacité du mode automatique, non seulement en termes de rapidité, mais aussi en termes de réduction de l'effort humain et de simplification du processus de planification.

4.6.2 Comparaison des conflits détectés

En mode manuel, la première génération de l'emploi du temps a produit 70 conflits en raison de la gestion séquentielle et non assistée des affectations. L'absence d'une vision globale des contraintes rend les collisions presque inévitables, et la correction d'un conflit peut entraîner l'apparition de nouveaux conflits dans d'autres parties du planning. Cette dépendance entre les contraintes explique les quatre phases successives de correction nécessaires avant d'obtenir une solution valide.

En revanche, le mode automatique a généré dès la première exécution un emploi du temps avec 0 conflit. Cette performance s'explique par l'intégration des contraintes directement dans le processus de génération. Le système vérifie automatiquement les contraintes dures avant chaque affectation, empêchant ainsi toute apparition de conflit. Contrairement au mode manuel, qui nécessite plusieurs corrections successives, le mode automatique produit immédiatement une solution admissible et cohérente.

4.6.3 Comparaison de l'efficacité

L'efficacité d'un système de planification universitaire se mesure à trois niveaux complémentaires : le coût temporel engagé, la qualité intrinsèque des emplois du temps produits et la reproductibilité des résultats obtenus.

Coût temporel. Le mode manuel mobilise plus de cinq heures de travail qualifié pour un seul semestre. À l'échelle d'un établissement traitant plusieurs promotions simultanément, ce coût devient difficilement soutenable. Le mode automatique ramène ce coût à 12,51 secondes, libérant intégralement le responsable pédagogique pour des tâches à plus forte valeur ajoutée.

Qualité des emplois du temps. Le mode manuel produit une solution admissible, mais sans optimisation systématique des contraintes souples. La compacité, l'équilibrage de la charge enseignante et la minimisation des trous dépendent exclusivement de l'expérience de l'opérateur. Le mode automatique, grâce à la phase génétique, optimise simultanément l'ensemble de ces critères de manière quantifiée, comme en témoigne la fitness globale de 0,2594 obtenue.

Reproductibilité. Le processus manuel introduit une variabilité non maîtrisée : deux responsables différents produiront des emplois du temps potentiellement très différents en qualité. Le mode automatique garantit une reproductibilité totale pour un jeu de paramètres donné, offrant une base stable pour la prise de décision institutionnelle.

4.6.4 Analyse comparative finale

L'analyse comparative globale confirme clairement la supériorité du système automatique par rapport à la méthode manuelle traditionnelle. Comme illustré dans les figures 4.6 et 4.7, les différences observées concernent non seulement le temps de génération, mais également la qualité globale des emplois du temps produits.

Le mode manuel nécessite plusieurs phases successives de correction afin d'éliminer les conflits et d'équilibrer les contraintes pédagogiques. Cette approche dépend fortement de l'expérience humaine et reste sensible aux erreurs, particulièrement dans les environnements universitaires complexes comportant un grand nombre de groupes, d'enseignants et de salles.

À l'inverse, le système automatique proposé permet de générer directement une solution admissible sans conflit, tout en optimisant simultanément plusieurs contraintes académiques. L'intégration combinée de l'algorithme glouton, du module de réparation et de l'algorithme génétique permet d'améliorer considérablement la qualité de planification tout en réduisant drastiquement le coût temporel.

Les résultats expérimentaux montrent également que le système assure une meilleure reproductibilité des solutions obtenues. Contrairement au processus manuel, dont les résultats peuvent varier selon l'opérateur, le système automatique garantit une stabilité des performances pour un même ensemble de paramètres.

Ainsi, cette étude comparative valide l'efficacité, la robustesse et la pertinence du système IMS-Scheduler pour l'automatisation intelligente de la génération des emplois du temps universitaires.

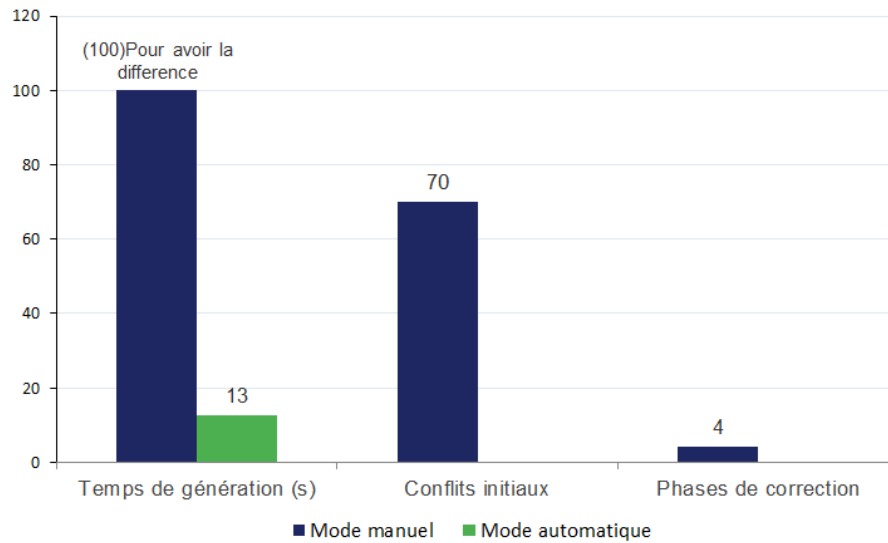


FIG. 4.6. Comparaison des performances entre le mode manuel et le mode automatique.

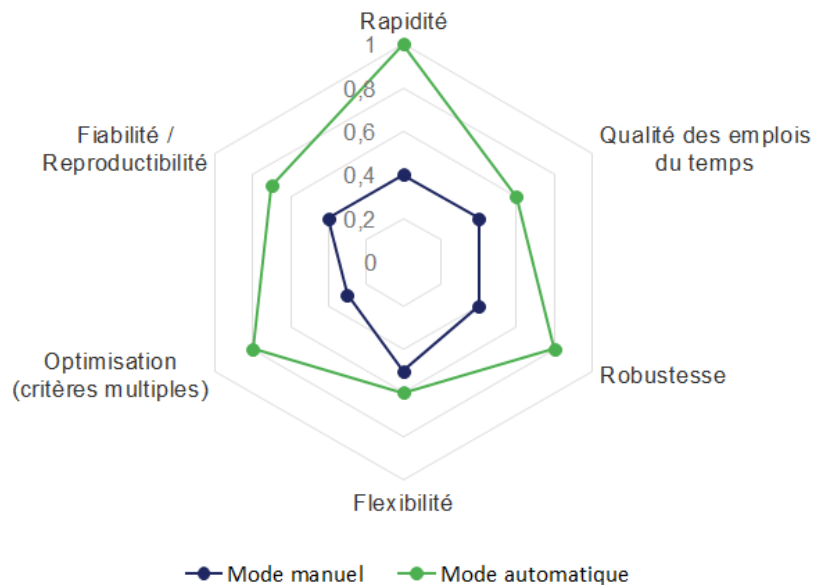


FIG. 4.7. Évaluation multicritère des approches (manuel vs automatique).

Le système automatique génère une solution admissible sans conflit en un temps extrêmement réduit, tout en optimisant plusieurs critères académiques. Il assure une meilleure qualité, une plus grande stabilité et une reproductibilité totale des résultats.

Le mode manuel, bien que capable d'atteindre une solution valide, reste chronophage, dépendant fortement de l'expérience humaine et moins performant sur l'optimisation multicritère.

4.7 Tests et validation

Après l'analyse comparative entre les modes manuel et automatique, il est nécessaire de vérifier concrètement le comportement du système à travers une série de tests fonctionnels et de validations utilisateurs.

4.7.1 Tests fonctionnels - Mode Manuel

Les tests du mode manuel portent sur la détection des conflits en temps réel lors de la saisie des affectations par l'administrateur. Le tableau suivant récapitule les cas de test réalisés :

Cas de test	Action	Résultat attendu	Résultat obtenu
Saisie normale	Affecter une séance à un créneau libre	Affectation acceptée	Conforme
Conflit de salle	Affecter deux séances à la même salle au même créneau	Ligne surlignée en rouge	Conforme
Champ non renseigné	Laisser jour/créneau/salle vide	Séance non affectée	Conforme
Correction du conflit	Modifier la salle ou le créneau conflictuel	Alerte disparaît	Conforme

TAB. 4.11. Cas de test du mode manuel.

La figure 4.10 illustre concrètement la détection d'un conflit de salle en mode manuel : les séances 17 et 18 ont été affectées toutes les deux à la salle C28 le mercredi sur le créneau 13:00–14:30, ce qui a provoqué le surlignage en rouge de la ligne 18 par le système. Ce mécanisme permet à l'administrateur d'identifier et de corriger immédiatement toute incohérence avant la sauvegarde finale.

11	Physique 1	BOULEKHRA	ELM - Licence	L1	S2	1	TP	LAB	Mercredi	11:30 - 13:00	C13
12	TP Elément de mécanique	SENHADJI	ELM - Licence	L1	S2	1	TP	LAB	Samedi	11:30 - 13:00	C13
13	TP Structure de la matière	TALBI	ELM - Licence	L1	S2	1	TP	LAB	Dimanche	11:30 - 13:00	C10
14	Thermodynamique	TALBI	ELM - Licence	L1	S2	1	TD	-	Lundi	10:00 - 11:30	C15
15	Analyse 2	BOULEKHRA	ELM - Licence	L1	S2	1	TD	-	Mardi	14:30 - 16:00	C12
16	Thermodynamique	BELOUFA	ELM - Licence	L1	S2	2	TD	-	Mercredi	14:30 - 16:00	C25
17	Initiation à la programmation	REGUIEG, YSSAD	ELM - Licence	L1	S2	2	TP	INF	Mercredi	13:00 - 14:30	C28
18	Logiciels opensource	REGUIEG, YSSAD	ELM - Licence	L1	S2	2	TP	INF	Mercredi	13:00 - 14:30	C28
19	Algebre 2	HEBAL	ELM - Licence	L1	S2	2	TD	-	-- Jour --	-- Créneau --	-- Salle --
20	TP Elément de mécanique	YAHIAOUI	ELM - Licence	L1	S2	2	TP	LAB	-- Jour --	-- Créneau --	-- Salle --

FIG. 4.8. Détection d'un conflit de salle en mode manuel.

4.7.2 Tests fonctionnels - Mode Automatique

Les tests du mode automatique visent à vérifier que le pipeline hybride G+R+AG génère des emplois du temps valides et cohérents pour l'ensemble des filières. Les cas de test suivants ont été exécutés :

Cas de test	Action	Résultat attendu	Résultat obtenu
Génération complète	Lancer la génération automatique sur les 13 filières	425 séances planifiées, 0 conflit	Conforme
Vérification des conflits	Consulter le rapport après génération	"Aucun conflit détecté"	Conforme
Affichage de l'emploi du temps et filtrage	Consulter l'emploi du temps et filtrer d'un groupe	Grille hebdomadaire correcte	Conforme

TAB. 4.12. Cas de test du mode automatique.

Comme en témoigne la figure 4.9, le tableau de bord administrateur affiche après génération le message "Aucun conflit détecté", confirmant que l'ensemble des contraintes dures ont été satisfaites

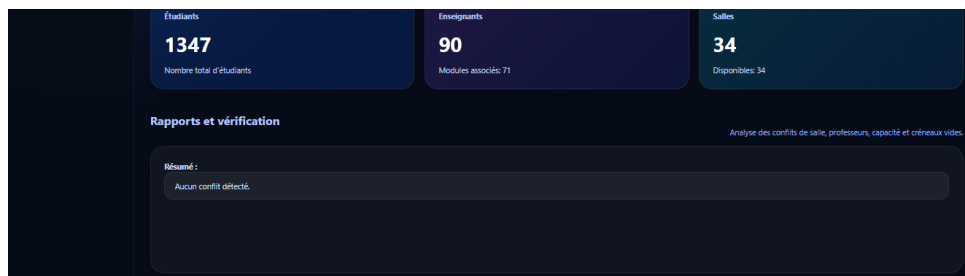


FIG. 4.9. Rapport de vérification affichant l'absence de conflits après génération automatique.

Par ailleurs, la figure 4.10 présente l'interface d'affichage et de filtrage des emplois du temps, permettant de consulter rapidement les séances d'un groupe donné à travers une grille hebdomadaire claire et correctement organisée.

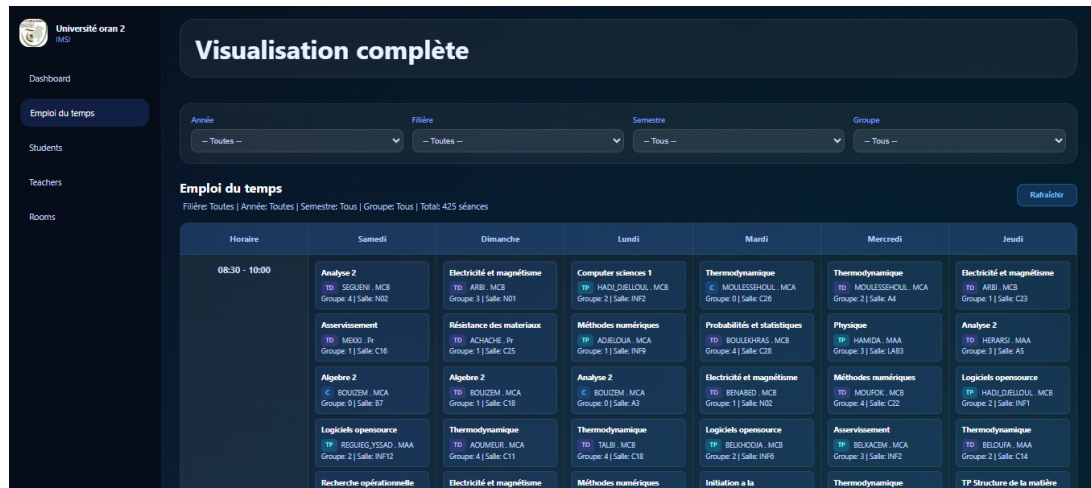


FIG. 4.10. Interface de visualisation et de filtrage des emplois du temps.

4.7.3 Validation par les utilisateurs finaux

La validation du système a été effectuée en testant les fonctionnalités accessibles à chaque profil d'utilisateur. Le tableau suivant présente les résultats obtenus pour les trois types d'utilisateurs du système :

Profil	Fonctionnalité testée	Résultat attendu	Résultat obtenu
Étudiant	Connexion et consultation de l'emploi du temps du groupe	Affichage de la grille hebdomadaire avec modules, enseignants et salles	Conforme
Enseignant	Connexion et consultation des séances assignées	Affichage des informations personnelles et du planning détaillé	Conforme
Administrateur	Génération manuelle avec détection de conflits	Surlignage rouge des lignes conflictuelles	Conforme
Administrateur	Génération automatique et vérification	Solution admissible et rapport sans conflit	Conforme

TAB. 4.13. Validation par profil d'utilisateur.

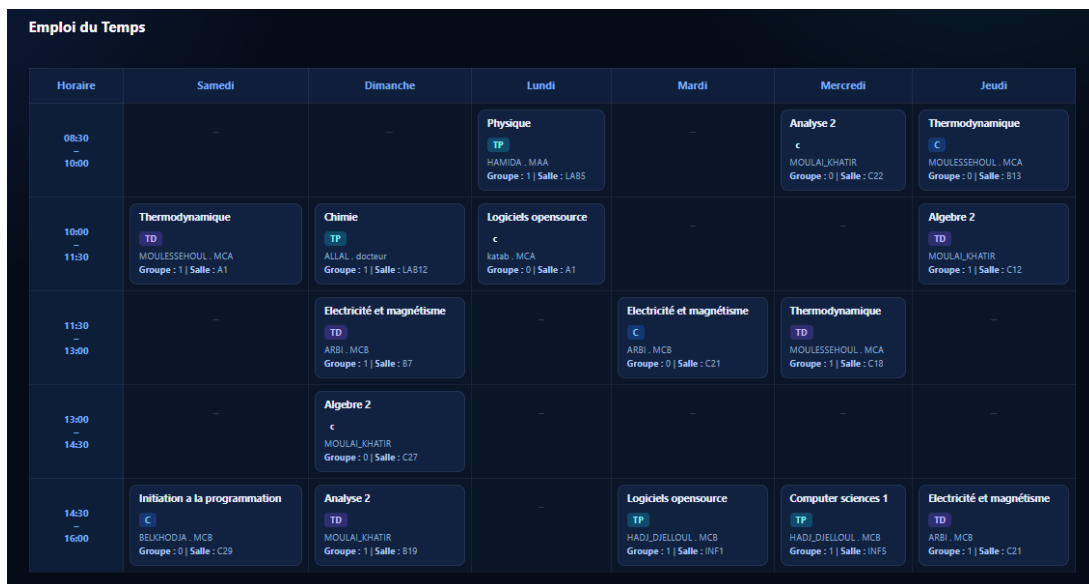
La figure 4.11 présente l'interface de consultation de l'emploi du temps de l'enseignant, avec sa charge affectée de 12.0h, ses modules (Algèbre 2, Analyse 2) et l'ensemble de ses créneaux hebdomadaires correctement affichés.



Jour	Heure	Module	Type	Groupe	Filière	Année	Salle
Samedi	1300-1430	Algebre 2	TD	1	GI - Licence	L1	C21
Samedi	1430-1600	Analyse 2	c	0	GI - Licence	L1	B5
Lundi	1300-1430	Algebre 2	TD	3	GI - Licence	L1	B13
Mardi	1130-1300	Algebre 2	TD	4	GI - Licence	L1	C29
Mercredi	1000-1130	Algebre 2	TD	2	GI - Licence	L1	C16
Jeudi	1130-1300	Analyse 2	TD	1	GI - Licence	L1	C29
Jeudi	1430-1600	Algebre 2	c	0	GI - Licence	L1	N01

FIG. 4.11. Interface de consultation de l'emploi du temps d'un enseignant.

La figure 4.12 illustre la grille hebdomadaire accessible aux étudiants, présentant l'ensemble des séances planifiées avec le module, le type de cours, l'enseignant, le groupe et la salle pour chaque créneau.



Horaire	Samedi	Dimanche	Lundi	Mardi	Mercredi	Jeudi
08:30 - 10:00			Physique TP HAMIDA, MAA Groupe : 1 Salle : LAB5		Analyse 2 c MOULAL_KHATIR Groupe : 0 Salle : C22	Thermodynamique C MOULESSEHOUL, MCA Groupe : 0 Salle : B13
10:00 - 11:30	Thermodynamique TD MOULESSEHOUL, MCA Groupe : 1 Salle : A1	Chimie TP ALLAL, docteur Groupe : 1 Salle : LAB12	Logiciels opensource c katab, MCA Groupe : 0 Salle : A1			Algebre 2 TD MOULAL_KHATIR Groupe : 1 Salle : C12
11:30 - 13:00		Electricité et magnétisme TD ARBI, MCB Groupe : 1 Salle : B7		Electricité et magnétisme C ARBI, MCB Groupe : 0 Salle : C21	Thermodynamique TD MOULESSEHOUL, MCA Groupe : 1 Salle : C18	
13:00 - 14:30		Algebre 2 c MOULAL_KHATIR Groupe : 0 Salle : C27				
14:30 - 16:00	Initiation a la programmation C BELKHODJA, MCB Groupe : 0 Salle : C29	Analyse 2 TD MOULAL_KHATIR Groupe : 1 Salle : B19		Logiciels opensource TP HADI, DJELLOUL, MCB Groupe : 1 Salle : INF1	Computer sciences 1 TP HADI, DJELLOUL, MCB Groupe : 1 Salle : INF5	Electricité et magnétisme TD ARBI, MCB Groupe : 1 Salle : C21

FIG. 4.12. Interface de consultation de l'emploi du temps d'un groupe d'étudiants.

L'ensemble de ces tests confirme que le système IMS-Scheduler répond correctement aux besoins fonctionnels identifiés dans le chapitre 2, aussi bien en mode manuel qu'en mode automatique, et pour les trois profils d'utilisateurs concernés.

4.8 Analyse des résultats

L'analyse des résultats porte sur trois dimensions complémentaires : la qualité des solutions générées mesurée par la fonction de fitness, la stabilité du système face aux variations

aléatoires, et l'influence du nombre de séances sur le temps d'exécution.

4.8.1 Qualité des solutions et fonction de fitness

La fonction de fitness adoptée dans le système IMS-Scheduler permet d'évaluer simultanément le respect des contraintes dures et la qualité des contraintes souples. Une valeur de fitness égale à 1,0 correspond à une solution parfaite, sans aucun conflit ni pénalité souple.

Les résultats obtenus par la configuration hybride G+R+AG montrent une fitness globale de 0,2594, ce qui s'explique par la présence résiduelle de contraintes souples non entièrement optimisées, notamment les trous dans les plannings (114 au total) et le nombre de jours de présence des enseignants (352 jours cumulés). Toutefois, la valeur de fitness par filière, comprise entre 0,0038 et 0,0049 pour la configuration standard (Run 1), traduit une très bonne qualité de planification à l'échelle individuelle de chaque filière.

Il est important de souligner que la fitness globale de 0,2594 ne reflète pas une mauvaise qualité du système, mais résulte de l'agrégation des pénalités souples sur l'ensemble des 13 filières et 425 séances planifiées. L'objectif prioritaire du système, à savoir l'élimination totale des conflits durs, a été pleinement atteint avec un taux d'admissibilité de 100%.

4.8.2 Stabilité du système

L'analyse des différents runs expérimentaux révèle une bonne stabilité du système pour la configuration standard. Les Runs 1 et 2, exécutés avec les mêmes paramètres (GA_POPULATION = 30, GA_GENERATIONS = 80, GA_MUTATION = 8%), produisent des valeurs de fitness très proches pour l'ensemble des filières. Par exemple, pour la filière GI L1 S2, les valeurs obtenues sont respectivement 0,0043 et 0,0038, soit un écart inférieur à 0,001.

Cette stabilité confirme que le pipeline hybride G+R+AG converge de manière reproductible vers des solutions de qualité comparable, indépendamment de l'ordre aléatoire de traitement initial introduit par l'algorithme glouton. En revanche, les configurations légère (Run 3) et à mutation élevée (Run 4) présentent une variabilité plus importante, avec des valeurs pouvant atteindre 0,0215 pour certaines filières, ce qui confirme la sensibilité de l'algorithme génétique au réglage de ses paramètres.

4.8.3 Influence du nombre de séances sur le temps d'exécution

Les résultats détaillés par filière présentés dans le tableau 4.9 mettent en évidence une corrélation directe entre le nombre de séances à planifier et le temps d'exécution observé. Les filières comportant un nombre important de séances, telles que GI L1 S2 (44 séances, 1,34 s) et ELM L2 S4 (43 séances, 1,54 s), nécessitent un temps de traitement plus élevé que

les filières de taille réduite, comme GI M1 S2 (24 séances, 0,52 s) et GI L1 S1 (1 séance, 0,05 s).

Toutefois, cette augmentation reste parfaitement maîtrisée : le temps d'exécution global pour l'ensemble des 13 filières et 425 séances s'établit à 12,51 secondes, ce qui représente une performance largement suffisante pour un déploiement en environnement universitaire réel. Cette efficacité temporelle s'explique par la complémentarité des trois phases du pipeline : L'algorithme glouton assure une construction rapide en temps linéaire, le module de réparation corrige localement les conflits résiduels, et l'algorithme génétique optimise les contraintes souples sur un espace de recherche déjà fortement réduit.

4.8.4 Influence des contraintes sur la qualité des résultats

L'analyse des résultats par filière révèle que la qualité des emplois du temps générés est directement influencée par la densité des contraintes propres à chaque filière. Les filières partageant un grand nombre d'enseignants et de salles avec d'autres filières présentent naturellement un nombre de trous plus élevé, en raison de la réduction de l'espace de créneaux disponibles lors de l'affectation. Ainsi, la filière GI L1 S2 affiche 16 trous, contre seulement 4 pour la filière GI L3 S6, qui comporte moins de séances et donc moins de contraintes croisées.

Ces résultats confirment que le système IMS-Scheduler gère efficacement la complexité combinatoire du problème de planification universitaire, en produisant des solutions admissibles et de bonne qualité pour l'ensemble des filières de l'IMSI, dans un temps d'exécution très raisonnable.

4.9 Limites et perspectives

Si les résultats obtenus sont satisfaisants, ils ne sont cependant pas exhaustifs. En effet, certaines limites restent à franchir et ouvrent la voie à de nombreuses perspectives d'amélioration et d'évolution pour le système IMS-Scheduler.

4.9.1 Limites actuelles du système IMS-Scheduler

Malgré les résultats obtenus, le système IMS-Scheduler présente encore certaines limites. La base de données SQLite utilisée actuellement reste adaptée au développement mais limitée pour une utilisation en production. Le système ne permet pas encore la gestion simultanée de plusieurs semestres universitaires. De plus, l'algorithme génétique peut produire des résultats légèrement différents à chaque exécution en raison de son fonctionnement aléatoire. Enfin, l'application ne dispose pas encore d'une interface mobile dédiée.

4.9.2 Perspectives futures

Plusieurs améliorations peuvent être envisagées pour les prochaines versions du système. La migration vers PostgreSQL permettrait d'améliorer les performances et la gestion multi-utilisateurs. L'ajout d'un module d'export PDF et Excel faciliterait le partage des emplois du temps. Une intégration avec le système de scolarité de l'université pourrait automatiser la synchronisation des données académiques. Enfin, l'utilisation de techniques d'apprentissage automatique permettrait de prédire certaines contraintes et d'optimiser davantage la génération des emplois du temps.

Conclusion

Ce chapitre a présenté la conception générale du système IMS-Scheduler ainsi que l'évaluation de ses performances en modes manuel et automatique. Les analyses réalisées ont montré l'efficacité de l'approche hybride adoptée pour produire des emplois du temps cohérents tout en respectant les contraintes de planification. Les différents tests effectués ont également confirmé la fiabilité fonctionnelle du système pour les trois profils d'utilisateurs. Enfin, les limites identifiées ouvrent plusieurs perspectives d'amélioration pour les futures versions de la plateforme.

CONCLUSION GÉNÉRALE

Ce travail avait pour objectif de concevoir et de développer un système informatique dédié à la génération automatique des emplois du temps universitaires de l'Institut de Maintenance et de Sécurité Industrielle (IMSI) de l'Université Oran 2 Mohamed Ben Ahmed. Ce système, dénommé IMS-Scheduler, a été développé dans une démarche de maintenance logicielle améliorative, visant à remplacer le processus manuel actuel, basé sur des fichiers Excel non interconnectés, par une solution automatisée, fiable et efficace.

Le premier chapitre a permis d'analyser en détail le processus de planification actuel de l'IMSI et d'en identifier les principales limites, notamment les conflits de ressources récurrents, les chevauchements entre enseignants et groupes, les erreurs de saisie et le temps excessif de génération. L'étude de l'état de l'art a confirmé la nature NP-difficile du problème de planification universitaire et a justifié le recours à des approches hybrides combinant méthodes heuristiques et métaheuristiques. L'analyse comparative des outils existants, tels que HyperPlanning et FET, a mis en évidence leur inadaptation aux besoins spécifiques de l'établissement, renforçant ainsi la pertinence du développement d'un système dédié.

Le deuxième chapitre a établi la modélisation mathématique formelle du problème, en définissant les ensembles, la variable de décision, les paramètres et l'ensemble des contraintes dures et souples propres au contexte de l'IMSI. La fonction de fitness multi-critère adoptée permet d'évaluer simultanément le respect des contraintes dures et la qualité des contraintes souples, constituant ainsi le fondement théorique du moteur de génération automatique.

Le troisième chapitre a présenté l'architecture complète du système IMS-Scheduler, reposant sur le framework Django selon le patron MVT. Le moteur de génération automatique met en œuvre un pipeline hybride en trois phases successives et conditionnelles : une construction initiale par algorithme glouton, une phase de réparation locale des conflits résiduels, et une optimisation finale par algorithme génétique. Ce pipeline garantit qu'aucune séance ne reste non affectée, tout en minimisant les conflits durs et en améliorant la qualité globale des emplois du temps. Le mode manuel, quant à lui, offre à l'administrateur

un contrôle direct sur les affectations avec détection des conflits en temps réel.

Le quatrième chapitre a détaillé la conception de l'outil, les choix technologiques retenus et les résultats expérimentaux obtenus. Les tests fonctionnels ont validé le bon fonctionnement du système pour les trois profils d'utilisateurs : étudiant, enseignant et administrateur. L'analyse comparative entre le mode manuel et le mode automatique a mis en évidence des résultats particulièrement significatifs : le mode automatique génère un emploi du temps complet pour l'ensemble des 13 filières et 425 séances en seulement 12,51 secondes et sans aucun conflit, contre plus de cinq heures de travail et 70 conflits initiaux pour le mode manuel. Cette réduction du temps de traitement de 99,93% confirme l'efficacité et la pertinence de l'approche hybride G+R+AG adoptée.

En définitive, le système IMS-Scheduler répond pleinement aux objectifs fixés en début de projet. Il automatise une grande partie du processus de planification, réduit considérablement la charge de travail administrative des responsables pédagogiques, et garantit la cohérence et la qualité des emplois du temps générés pour les 1347 étudiants, 90 enseignants et 34 salles de l'IMSI.

Ce travail ouvre plusieurs perspectives d'amélioration futures, notamment :

- la migration vers PostgreSQL pour un déploiement en production.
- l'intégration d'un module d'exportation aux formats PDF et Excel.
- le développement d'une interface mobile.
- l'intégration institutionnelle avec le système d'information de l'Université Oran 2.

Ces évolutions permettraient de transformer IMS-Scheduler en un système de gestion pédagogique centralisé et pleinement intégré au fonctionnement quotidien de l'établissement.

Ressources externes

Les documents détaillés sont disponibles via les liens suivants :

- **Analyse de génération de l'emploi du temps**

<https://drive.google.com/file/d/1ITnb17lsBrdCF9oZRY0--6ZQ-V-K5wKt/view?usp=sharing>

- **Gestion des utilisateurs et mots de passe**

<https://drive.google.com/file/d/1KgDA0x8WexoqtlhgQPsqVHSQltBkn5Jg/view?usp=sharing>

- **Documentation complète du projet**

<https://drive.google.com/file/d/1nN7YlxBDdgG83z2hvmOM9S5s2aMBhRnS/view?usp=sharing>

- **Documentation technique de generate-timetable**

https://drive.google.com/file/d/1GvX0jfu_3xha8VAdUct8hAEg_BYuejWq/view?usp=sharing

BIBLIOGRAPHIE

- [1] Institut de Maintenance et de Sécurité Industrielle (IMSI), *Site officiel de l'Institut IMSI - Université Oran 2 Mohamed Ben Ahmed*. [En ligne]. Disponible : <https://institutimsi.univ-oran2.dz/>
- [2] Z. Mallem, M. K. Kholadi et S. Sahraoui, «Conception d'un système pour l'emploi du temps universitaire», *Actes de conférence*, Novembre 2012.
- [3] A. Bashab, A. O. Ibrahim, I. A. T. Hashem, K. Aggarwal, F. Mukhlif, F. A. Ghaleb et A. Abdelmaboud, «Optimization Techniques in University Timetabling Problem : Constraints, Methodologies, Benchmarks, and Open Issues», *Computers, Materials & Continua*, vol. 74, no. 3, pp. 6461–6484, 2023.
- [4] A. Nurul Liyana Abdul et A. Nur Aidya Hanum, «A brief review on the features of university course timetabling problem», *AIP Conference Proceedings*, 2018.
- [5] C. H. Papadimitriou et K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*, Dover, 1998.
- [6] M. Pinedo, *Scheduling : Theory, Algorithms, and Systems*, 5e éd., Springer, 2016.
- [7] F. S. Hillier et G. J. Lieberman, *Introduction to Operations Research*, 10e éd., McGraw-Hill Education, 2015.
- [8] A. S. Asratian et D. de Werra, «A generalized class–teacher model for some timetabling problems», *European Journal of Operational Research*, vol. 143, no. 3, pp. 531-542, 2002.
- [9] M. Assi, B. Halawi et R. A. Haraty, «Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem», *Procedia Computer Science*, vol. 126, pp. 899–906, 2018.

- [10] M. R. Garey et D. S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.
- [11] R. Dechter, *Constraint Processing*, Morgan Kaufmann, 1997.
- [12] E. K. Burke et S. Petrovic, «Recent Research Directions in Automated Timetabling», *European Journal of Operational Research*, vol. 140, no. 2, pp. 266-280, 2002.
- [13] W. L. Winston, *Operations Research : Applications and Algorithms*, 4e éd, Brooks/Cole, 2004.
- [14] F. Rossi, P. Van Beek et T. Walsh, *Handbook of Constraint Programming*, Elsevier, 2006.
- [15] A. A. T. J. Nicholson, *Optimization in Industry : Industrial Applications*, London Business School Series, 1971.
- [16] S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi, «Optimization by Simulated Annealing», *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
- [17] E. K. Burke, J. Mareček, A. J. Parkes et H. Rudová, «A Survey of Approaches for University Course Timetabling Problem», *Computers & Industrial Engineering*, vol. 86, pp. 43-59, 2015.
- [18] B. M. Coşar, B. Say et T. Dökeroğlu, «A New Greedy Algorithm for the Curriculum-Based Course Timetabling Problem», *Duzce University Journal of Science and Technology*, vol. 11, no. 2, pp. 1121-1136, 2023.
- [19] H. Alghamdi, T. Alsubait, H. Alhakami et A. Baz, «A Review of Optimization Algorithms for University Timetable Scheduling», *Engineering, Technology & Applied Science Research*, vol. 10, no. 6, pp. 6410-6417, 2020.
- [20] S. Abdennadher et M. Marte, «University Course Timetabling Using Constraint Handling Rules», *Applied Artificial Intelligence*, vol. 14, no. 4, pp. 311-325, 2000.
- [21] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, USA, 1975.
- [22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, USA, 1989.
- [23] L. Lalescu, *FET : Free Timetabling Software*, [Logiciel]. Université de Timișoara, Roumanie.
- [24] Index Éducation, *Hyperplanning - Solution de gestion des emplois du temps et de planification universitaire*, [Logiciel]. France.

- [25] AFNOR, *NF EN 13306 - Terminologie de la maintenance*, Association Française de Normalisation, Paris, 2018.
- [26] J. Moubray, *Reliability-Centered Maintenance*, 2e éd., Industrial Press Inc., New York, USA, 1997.
- [27] S. Nakajima, *Introduction to TPM : Total Productive Maintenance*, Productivity Press, Cambridge, USA, 1988.
- [28] T. Wireman, *Total Productive Maintenance*, Industrial Press Inc., New York, USA, 2004.
- [29] R. S. Pressman, *Software Engineering : A Practitioner's Approach*, 7e éd., McGraw-Hill, New York, USA, 2010.
- [30] R. S. Pressman et B. R. Maxim, *Software Engineering : A Practitioner's Approach*, 8e éd., McGraw-Hill Education, 2015.
- [31] H. Tardieu, A. Rochfeld et R. Colletti, *La méthode MERISE : Principes et outils*, Éditions d'Organisation, Paris, 1983.
- [32] R. Elmasri et S. B. Navathe, *Fundamentals of Database Systems*, 7e éd., Pearson, 2016.