MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS DE MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique
Département de Mathématiques et d'Informatique
Filière Informatique

MEMOIRE DE FIN D'ETUDES
Pour l'Obtention du Diplôme de Master en Informatique
Option : Ingénierie des Systèmes d'Information

THEME :

Programmation par contrainte des emplois du temps

Etudiante :

- ✓ **BOUNAB Yasmin**

Résumé

La génération automatique d'emploi du temps universitaire peut être décrite comme l'allocation des ressources (Créneaux Horaires, Salle …) aux événements (Séances de cours, TD et TP), tout en essayant de satisfaire un ensemble de contraintes. Le cœur du problème se réside dans les contraintes qui existent au sein de chaque ressource et entre les ressources mêmes. Il existe différentes approches et méthodes pour résoudre le problème d'emploi du temps. Ce projet se concentre sur l'élaboration d'un modèle de problème de satisfaction de contraintes pour un problème d'emploi du temps universitaire. La solution d'un problème de satisfaction de contraintes est l'affectation de toutes les variables à des valeurs de telle sorte que toutes les contraintes soient satisfaites.
En effet notre système est devenu intéressant et rapide grâce aux méthodes stochastiques, qui permettent de résoudre les problèmes fortement combinatoires et de générer des emplois du temps automatique juste en cliquant sur un bouton.

Mots Clés : problèmes combinatoires, planification, programmation par contraintes, problème de satisfaction de contraintes, emploi du temps, algorithme génétique .

Abstract

Automatic generation of university Timetabling can be described as the allocation of resources (time slots, rooms...) to the events (lesson, TD and TP), while trying to meet a set of constraints. The heart of the problem is the constraints that exist within each resource and between resources. There are various solution approaches to solve the timetabling problem. This project focuses on developing a constraint satisfaction problem model for a university timetabling problem. A solution of a constraint satisfaction problem is a consistent assignment

TABLE DES MATIERES

CHAPITRE III : Méthodes et techniques de résolution du problème d'emploi du temps

CHAPITRE IV : Conception et réalisation

Introduction générale

Les entreprises, les écoles, les universités et les hôpitaux sont régulièrement amenés à résoudre un problème d'emploi du temps. Construire un bon emploi du temps est souvent une tâche longue qui exige un effort considérable afin de répondre aux exigences et aux besoins.
Les problèmes d'optimisation combinatoire sont des problèmes difficiles à résoudre. Ainsi, la nature discrète des variables forme un espace non dérivable qui rend inutiles les techniques classiques. Le problème de la production des emplois du temps appartient effectivement à la famille des problèmes combinatoires.
La difficulté de cette tache fastidieuse et répétitive, qui fait généralement intervenir de nombreux éléments d'information, est liée à la nature des contraintes qu'il s'agit de satisfaire.
A celle-ci s'ajoutent des caractéristiques bien particulières de l'institution ou l'organisation concernée. Parce qu'il y a plusieurs modèles du problème, des contraintes qui changent, et des caractéristiques particulières pour chaque institution ou organisation, il devient de plus en plus difficile de trouver une solution générale pour les problèmes d'emploi du temps, et c'est pour cela que ce domaine a besoin de plus en plus de recherche.
Dans ce projet, nous nous concentrerons sur le PET des cours. Ce problème se modélise généralement par un réseau de contraintes, encore appelé problème de satisfaction de contraintes (CSP), la résolution d'un CSP consiste à trouver l'assignation consistante de valeurs à des variables prenant leurs valeurs dans des domaines discrets et finis. Un PET

Notre projet est structuré comme suit :

- Chapitre I fournit des connaissances de base sur le PET et la planification en générale.
- Chapitre II, décrit la PPC ainsi que le principe d'un CSP avec quelques exemples.
- Dans le chapitre III, nous citons quelques méthodes et techniques utilisées, pour résoudre un problème de satisfaction de contraintes, un modèle CSP pour le PET est présenté aussi.
- Chapitre IV, nous aborderons les différentes phases nécessaires à la réalisation de cette application, de la conception jusqu'à l'implémentation.

I.1 Introduction

L'emploi du temps est l'un des exemples les plus fréquents des problèmes d'optimisation et de planification dans un établissement. Il peut se manifester sous plusieurs formes et particulièrement selon la spécificité de l'environnement où l'on en a besoin.

Ce chapitre met en scène la problématique de la planification des horaires et de l'emploi du temps.

En effet, la question de l'aménagement du temps de travail et de ses enjeux préoccupe toute société ou établissement actif, ce qui a incité les chercheurs à proposer des méthodes et des techniques pour aider à gérer au mieux les horaires de travail. Pour cela nous définissons les différents types de plannings dans différents domaines de travail et plus particulièrement dans le domaine pédagogique.

I.2 Etude de l'existant

I.2.1 Présentation de FSEI

La Faculté des Sciences Exactes et de l'Informatique est une filiale de l'université d'Abdelhamid Ibn Badis Mostaganem.

Cette faculté est composée de quatre départements « Mathématiques, Informatique, Physique et Chimie », elle est dotée de cinq blocs :

Blocs administratif, pédagogique, bibliothèques, laboratoires, et blocs des amphis.

I.2.2 Ressources

A. Définition

- ✓ Moyens pécuniaires dont dispose une personne pour assurer son existence. (finances).
- ✓ Moyens matériels dont dispose un pays, une région, une collectivité. (Ressources hydrauliques, industrielles, minières, pétrolières d'un pays).
- ✓ Ensemble constitué des éléments biotiques et non biotiques de la terre, ainsi que des diverses formes d'énergies reçues (énergie solaire) ou produites sans intervention de l'homme. (ressources naturelles).
- ✓ Ensemble des moyens financiers mis effectivement à la disposition de l'entreprise (économie).
- ✓ Ensemble des moyens dont dispose un ordinateur pour exécuter un

C. Ressources de la faculté

Les ressources considérées sont les entités physiques nécessaires à l'élaboration des emplois du temps. Il s'agit des salles, enseignants et étudiants.

I.3 Problématique de la planification d'horaires de travail

La planification est une étape indispensable à la réussite et à une bonne gestion du temps. Planifier consiste à organiser logiquement les différentes étapes à suivre, et à déterminer les ressources essentielles pour atteindre un but défini [2].

I.3.1 Définition de la planification

La planification est l'organisation dans le temps de la réalisation d'objectifs :

- dans un domaine précis ;
- avec différents moyens mis en œuvre ;
- sur une durée (et des étapes) précise(s).

La planification vise à affecter les ressources humaines pour chaque intervalle de temps sur un horizon donné, de telle manière que les besoins par intervalle soient couverts et que les différentes contraintes soient satisfaites.

I.3.2 Définition d'un planning

C'est un document qui permet de représenter les relations entre les tâches à effectuer et les ressources disponibles au fil du temps.

Les plannings peuvent être journaliers (spécifiant les pauses et périodes de travail de la journée de chaque employé), hebdomadaires (utilisés pour une paie hebdomadaire), mensuels (utilisés pour le calcul des coûts pour les besoins de la paie mensuelle) ou annuels (permettant de gérer les congés annuels des employés).Le planning, permet aussi de partager la vision des travaux à réaliser avec l'ensemble des intervenants. Il permet à chacun de visualiser ce qu'il doit faire, de vérifier si des tâches n'ont pas été oubliées et si tout le monde a confiance dans le fait de mener à bien les tâches dans les délais prévus[3].

I.3.3 Objectif d'un planning

I.3.4 Domaines d'applications de planning

- **Le domaine de la santé**

  Les plannings dans le domaine de la santé sont des calendriers de travail pour chaque équipe, où figurent à la fois le temps, et l'affectation des personnels (jours et horaires de travail, congés et repos).

- **Le domaine de transport**

  • Le transport routier : il est toujours nécessaire de gérer aux mieux les ressources existantes en optimisant les investissements. Comme les clients exigent toujours plus de flexibilité, il faut offrir des services sur mesure, replanifier en permanence et en temps réel.

  • Le transport maritime : la planification des travaux (gestion des escales, gestion du personnel docker) est complexe. En effet les navires doivent rester à quai un temps minimum et les équipes docker doivent être disponibles. Cette activité représente un enjeu économique majeur.

  • Le transport aérien : la gestion dans ce domaine correspond aussi à des problèmes d'optimisation combinatoire dont la résolution est très complexe. En effet, la gestion du flux de trafic aérien permet d'assurer la sécurité des vols(en terme de risque de collision), et d'améliorer la capacité du réseau de routes sur lequel les avions se déplacent.

- **Le domaine de la pédagogie**

  Le planning dans les établissements scolaires, est un problème de résolution de contraintes dont la solution n'est pas, a priori connue. C'est un travail très important, et difficile à réaliser.

  Fournir une solution, nécessite d'être capable de s'adapter aux changements dynamiques de l'environnement en tenant compte de la diversité des contraintes telles que la multitude des matières étudiées, la durée des cours,

I.4.1 Définition de l'emploi du temps

L'emploi du temps est un plan qui représente l'organisation et la distribution des activités pour une période donnée, il consiste à gérer les charges de travail dans le temps tout en prenant compte des ressources humaines et matérielles disponibles.

I.4.2 Domaines d'application

Le problème d'emploi du temps se pose en général dans tous les secteurs: économiques, industriels, administratifs, les établissements scolaires, les instituts, les universités, les entreprises…etc.

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

I.4.3 Classification des problèmes d'emplois du temps

Selon son cycle de vie, un emploi du temps peut être quotidien, hebdomadaire, mensuel ou annuel et selon le domaine de l'application

I.4.4 Classes d'un emploi du temps pédagogique

Les problèmes d'emploi du temps peuvent être classés en trois catégories principales [4] :

A) Emploi du temps des écoles

Etablir un programme hebdomadaire pour toutes les classes d'une école en évitant que deux professeurs enseignent une même classe en même temps et vice versa.

Parmi les problèmes rencontrés dans les écoles, un enseignant qui enseigne une variété de cours et reste dans la même salle avec la même classe toute la journée.

B) **Emploi du temps des cours**

Etablir le programme hebdomadaire dont l'objectif est d'assigner les cours et les groupes à des créneaux horaires de manière à ce qu'aucun conflit ne se produise à aucune période (éviter le chevauchement des cours ayant des étudiants communs). Le nombre d'étudiants affectés à une pièce ne doit pas dépasser la capacité maximale de la pièce. Nous discuterons en détail l'emploi du temps des cours dans la section I.4.6.

I.4.5 Problème d'emploi du temps (PET)

Parmi l'ensemble des problèmes de planification d'horaire, on repère celui de l'emploi du temps qui semble être au premier moment très difficile. Car la réalisation manuelle de ce dernier, demande la participation de plusieurs personnes aptes et capables de prendre des décisions cruciales, afin de mettre en œuvre un emploi du temps acceptable. Or, la modification d'une donnée du problème initial peut changer complètement la solution trouvée, et parfois ça demande de recommencer à zéro le même travail.

I.4.6 Emploi du temps des universités

Dans cette partie nous allons citer quelques contraintes d'un emploi du temps universitaires, elles sont généralement divisées en deux catégories « dure » et « souple » :

- ✓ Les contraintes dures : ces contraintes doivent être satisfaites :
  - Aucune ressource (étudiant ou enseignant) ne peut demander d'être dans plus d'un endroit à un moment donné.
  - Un et seulement un cours doit être planifié dans une salle.
- ✓ Les contraintes souples : sont celles qui sont souhaitables mais pas absolument essentielles.
  - Le cours A doit être planifier après le cours B.
  - Les professeurs préfèrent avoir toutes leurs conférences en un certain nombre de jours et se libérer le reste de la semaine.
  - Les professeurs préfèrent enseigner toujours dans une salle X.
  - Certains Cours ne doivent avoir lieu que dans des pièces particulières.

I.5 Conclusion

Dans ce chapitre, nous avons présenté la faculté des sciences exactes et d'informatique. Nous avons vu la définition d'un emploi du temps, ses domaines d'applications et ses types. Nous avons conclu que la génération d'emploi du temps est une tâche difficile, qui ne s'agit pas seulement d'affecter des ressources à des évènements mais aussi à respecter des

LISTE DES ACRONYMES

AG : Algorithmes génétiques.

BJ : Backjumping.

CBJ : Conflict-directed Backjumping.

CSP : Problème de satisfaction de contraintes.

GBJ : Graph-based Backjumping .

PET : Problème d'emploi du temps.

POC : Problème d'optimisation combinatoire.

PPC : Programmation par contraintes.

TS : Tabu Search.

MCD : Modèle conceptuel de données.

MLD : Modèle logique de données.

AG : Algorithme génétique

SGBD : système de gestion de base de données

LISTE DES FIGURES

LISTE DES TABLES

II.1 Introduction

À la frontière entre programmation mathématique et intelligence artificielle, la programmation par contraintes se démarque par son paradigme et sa polyvalence. L'approche est basée sur la réduction du domaine de décision par la propagation des contraintes dans un arbre de branchement.
Chaque variable peut prendre un certain nombre de valeurs qui constituent son domaine. Les contraintes créent des relations entre les variables et limitent donc leurs domaines.

La programmation mathématique et la programmation par contraintes, la recherche locale et les algorithmes génétiques, sont autant de branches de l'informatique qui ont vocation à résoudre des problèmes construits autour de la notion de contrainte. Chacune de ses disciplines dispose d'un formalisme particulier et propose des méthodes différentes pour tenter de résoudre de manière générique un problème de la satisfaction des contraintes [5].

Dans ce chapitre, nous traitons une partie de la programmation par contraintes, celle qui s'intéresse à la résolution de problèmes combinatoires, en définissant un problème de satisfaction de contraintes.

II.2 Problèmes combinatoires

Un problème combinatoire est un problème où il s'agit de trouver la meilleure combinaison possible de solutions. Il peut être un problème de décision, un problème de recherche ou un problème d'optimisation, selon la question laquelle on est censé répondre.

II.2.1 Problème de décision

La réponse à la question : Est-ce-qu'une solution x est la meilleure solution dans un domaine de recherche ? Engendre un type de problèmes dits problèmes de décision. Un problème de décision est un problème où la résolution se limite à la réponse par «oui » ou par « non » à la question de savoir s'il existe une solution au problème.

II.2.2 Problème de recherche

La résolution du problème ne s'arrête plus au point de savoir si le problème admet ou non une solution. L'algorithme doit être en mesure de fournir la solution si elle existe. Par conséquent, tout problème de décision peut être étendu à un problème de

- **Caractéristiques des POC**

  Le premier type des problèmes d'optimisation combinatoire est le problème d'optimisation sans contraintes. Dans ce type de problème, l'optimisation peut s'effectuer en tout point de l'espace de recherche puisqu'il y a absence de contraintes.

  Parfois un problème n'a pas de critère à optimiser. Les problèmes qui n'ont pas de critère d'optimisation, mais qui possèdent un ensemble de contraintes sont des problèmes de satisfaction de contraintes. L'ajout au problème précédent un critère d'optimisation, crée un autre type de problème appelé problème d'optimisation combinatoire avec contraintes.

Ces problèmes sont généralement classés en fonction des ressources (temps de calcul, espace mémoire…etc.) nécessaires à leur résolution algorithmique, on parle alors de la complexité :L'enjeu est donc de discerner entre les problèmes qui ont une solution réalisable et ceux qui ne peuvent pas avoir une telle solution, quelque soit les améliorations futures des machines.

II.3 Programmation par contraintes (PPC)

C'est un paradigme de programmation apparu dans les années 1970 et 1980 permettant de résoudre des problèmes combinatoires de grandes tailles. Cette approche consiste à modéliser le problème par un ensemble de variables prenant leur valeur dans un ensemble fini et liées par un ensemble de contraintes mathématiques ou symboliques.

On sépare la partie modélisation à l'aide de problème de satisfaction de contraintes (CSP), de la partie résolution dont la particularité réside dans l'utilisation active des contraintes du problème pour réduire la taille de l'espace des solutions à parcourir (on parle de propagation de contraintes).

L'efficacité de ce paradigme repose sur de puissants algorithmes de propagation de contraintes [5].

En effet, la solution nécessite de trouver des affectations de valeurs pour les variables de manière à satisfaire toutes les contraintes. En général, ces problèmes peuvent être représentés

- $X = \{X_1, X_2, ..., X_n\}$ est l'ensemble des variables (les inconnues) du problème ;
- D est la fonction qui associe à chaque variable Xi son domaine D(Xi), c'est-à-dire l'ensemble des valeurs que peut prendre Xi ;
- $C = \{C_1, C_2, ..., C_k\}$ est l'ensemble des contraintes. Chaque contrainte C_j est une relation entre certaines variables de X, restreignant les valeurs que peuvent prendre simultanément ces variables[4].

Par exemple, on peut définir le CSP (X, D, C) suivant :

- X = {a,b,c,d}
- D(a) = D(b) = D(c) = D(d) = {0,1}
- C = { $a \neq b$, $c \neq d$, $a+c < b$ }

Ce CSP comporte 4 variables a, b, c et d, chacune pouvant prendre 2 valeurs (0 ou 1). Ces variables doivent respecter les contraintes suivantes : a doit être différente de b ; c doit être différente de d et la somme de a et c doit être inférieure à b.

II.4.1 Définition d'une contrainte

Une contrainte est une relation entre différentes variables. Cette relation peut être définie en extension ou en intension :

- En extension :

si les domaines des variables x et y contiennent les valeurs 0, 1 et 2, alors on peut définir la contrainte "x est plus petit que y" en extension par "(x=0 et y=1) ou (x=0 et y=2) ou (x=1 et y=2)" ,

ou encore par "(x,y) élément de {(0,1),(0,2),(1,2)}"

- En intension : on utilise des propriétés mathématiques connues.

Par exemple : "x < y" ou encore "A et B => non(C)"

II.4.2 Arité d'une contrainte

L'arité d'une contrainte est le nombre de variables sur lesquelles elle porte.

- Unaire si son arité est égale à 1 (elle ne porte que sur une variable),

  Par exemple "x*x = 4" ou encore "est-un-triangle(y)"

II.5 Type de CSP

Lorsqu'un CSP n'a pas de solution, on dit qu'il est *surcontraint : il y* a trop de contraintes et on ne peut pas toutes les satisfaire. Dans ce cas, on peut souhaiter trouver l'affectation totale qui viole le moins de contraintes possibles. Un tel CSP est appelé max-CSP (max car on cherche à maximiser le nombre de contraintes satisfaites).

Inversement, lorsqu'un CSP admet beaucoup de solutions différentes, on dit qu'il est *sous-contraint.*

II.5 Exemples

Nous allons présenter certains problèmes connus dans ce domaine tel que « le problèmes des N_reines » ,puis le modéliser sous forme d'un CSP.

II.5.1 Problème des N_reines

A. Description du problème

Il s'agit de placer n reines sur un échiquier qui est un carré divisé en $n_x n$ petits carrés, de telle manière à ce qu'aucune reine ne soit en prise. Deux reines sont en prise si elles sont sur la même ligne, colonne ou diagonale [6].

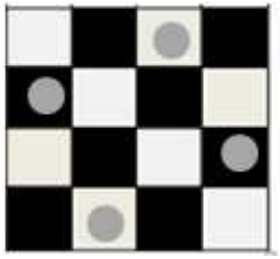


Figure II.1 : Problème des N_reines

B. Modélisation sous la forme d'un CSP

La modélisation est une phase qui permet de spécifier notre problème

Les "inconnues" du problème sont les positions des reines sur l'échiquier. Prenant un échiquier de taille 4*4, En numérotant les lignes et les colonnes de l'échiquier de la façon suivante :

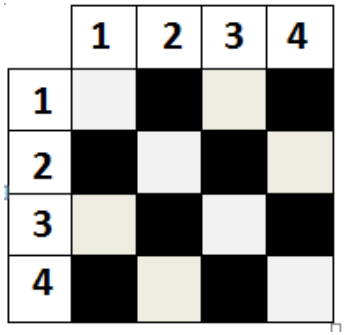


Figure II.2 : Numérotation des lignes et des colonnes d'un échiquier 4*4

Le problème consiste à placer les reines de façons qu'elles ne seront pas en prises, alors on associe à chaque reine i deux valeurs Li et Ci (ligne et colonne).

Les contraintes sont : les reines doivent être sur des lignes différentes, des colonnes différentes et des diagonales différentes ; notons que lorsque 2 reines sont sur une même diagonale montante, la somme de leurs numéros de ligne et de colonne est égale, tandis que lorsqu'elles sont sur une même diagonale descendante, la différence de leurs numéros de ligne et de colonne est égale.

On en déduit le CSP suivant :
Variables : X = {L1, L2, L3, L4, C1, C2, C3, C4}
Domaines : D(L1) = D(L2) = D(L3) = D(L4) = D(C1) = D(C2) = D(C3) = D(C4) = {1,2,3,4}
Contraintes : on identifie 4 types de contraintes

- Les reines doivent être sur des lignes différentes.
  Clig = {L1≠L2, L1≠L3, L1≠L4, L2≠L3, L2≠L4, L3≠L4 }
- Les reines doivent être sur des colonnes différentes.
  Ccol = {C1≠C2, C1≠C3, C1≠C4, C2≠C3, C2≠C4, C3≠C4 }
- Les reines doivent être sur des diagonales montantes différentes.
  Cdm={C1+L1≠C2+L2,C1+L1≠C3+L3,C1+L1≠C4+L4,C2+L2≠C3+L3,C2+L2≠C4+L4, C3+L3≠C4+L4}
- Les reines doivent être sur des diagonales descendantes différentes

II.6 Conclusion

Dans ce chapitre nous avons présenté une petite introduction sur les problèmes combinatoires. Nous avons vu que la programmation par contraintes est également une discipline dont le but est, comme son nom l'indique, de résoudre des problèmes posés sous la forme de contraintes. Nous avons aussi essayé de faire un tour d'horizon sur les concepts de base de la théorie des CSPs, ainsi que le formalisme d'évaluation des contraintes.

Dans le chapitre qui suit nous allons parler des techniques utilisées pour résoudre le PET, ainsi que nous allons proposer notre modèle.

III.1 Introduction

Le problème d'emploi du temps consiste à planifier une séquence de cours pour les enseignants et les étudiants dans une période définie (généralement une semaine), satisfaisant un ensemble de contraintes de différents types. Un grand nombre de variant de PET ont été proposés dans la littérature tout dépend le type d'institution concernée (université ou école) et le type de contraintes. La solution manuelle du problème d'emploi du temps nécessite des jours de travail. En outre, la solution obtenue peut être peu satisfaisante. Pour cette raison, une attention considérable a été consacrée à l'emploi du temps automatisé. Au cours des trente dernières années, de nombreux articles relatifs à l'emploi du temps automatisé ont été publiés dans des conférences et des revues. En outre, plusieurs applications ont été développées et employées avec succès.

Dans ce chapitre, les principales approches des problèmes de satisfaction sont présentées. Ces algorithmes recherchent une solution pour le CSP, sinon ils essayent de prouver l'inexistence de cette dernière. Nous décrivons aussi le problème de manière informelle ainsi que la modélisation.

III.2 Etat de l'art

Une des plus grandes questions dans l'étude d'un emploi du temps est de savoir s'il est unifiable, c'est-à-dire : pouvons-nous trouver un algorithme ou une méthode assez générale, pour produire un bon emploi du temps pour tous les problèmes avec toutes leurs contraintes?

Les premières techniques employées dans la résolution du problème d'emploi du temps ont été basées sur la simulation de l'approche humaine dans la résolution du problème, ces techniques ont été appelées « les heuristiques directes », elles sont basées sur l'idée de créer un emploi du temps partiel en planifiant d'abord le cours le plus contraint, ensuite, cette solution partielle est étendue jusqu'à ce que tous les cours seront planifiés.

L'intérêt de génération d'emploi du temps a augmenté dramatiquement dans les années 60 principalement en raison de la disponibilité d'ordinateurs pour exécuter les algorithmes développés. Autour de la fin des années 60 quelques tentatives qui ont traité le problème en considérant des études de cas commençaient à être publiées. Par exemple Appleby , blake Newman (1960), et Lewis (1961) ont utilisé la technique look-ahead. En 1969, Lawrie a développé un modèle pour le problème d'emploi du temps en employant l'approche de

des heuristiques directes, en particulier il a décrit les divers logiciels mis en œuvre et leur utilisation dans les divers établissements.

En 1994, Corne a fait une enquête sur l'application des algorithmes génétiques au PET et a discuté les futures perspectives de telles approches en comparant les résultats obtenus avec ceux obtenus avec d'autres approches.

En 1996, Carter et Laporte ont mis à jour les études pour résumer les approches algorithmiques. Les critères de discussion étaient que la méthode devrait être soit testée sur des données réelles, soit mise en œuvre dans les applications du monde réel. Ils ont catégorisé les méthodes en quatre types: méthodes de regroupement, méthodes séquentielles, recherche généralisée (méta-heuristiques) et contraintes. Ils ont conclu en suggérant que les chercheurs devraient reporter les résultats des tests sur un Benchmark afin de mieux comprendre les diverses approches adoptées lors de la construction d'un emploi du temps, et c'est ce qui s'est passé depuis 1996 [4].

David en 1998 a appliqué des techniques de satisfaction des contraintes pour le problème d'emploi du temps d'examens dans une école française, l'Ecole des Mines de Nantes, sa solution a été exécutée plusieurs fois et est utilisée avec succès à l'école [7].

Brailsford, Potts et Smith, en 1999 ont présenté plusieurs recherches sur les problèmes de satisfaction des contraintes CSP, et ont démontré que cette technique peut être appliquée à des problèmes d'optimisation.

Paquete et Stutzle, en 2002 ont développé une méthodologie de recherche tabou pour l'emploi du temps des examens [8].Dans la même année Sheibani, a construit un modèle mathématique spécial et a développé un algorithme génétique standard pour résoudre les problèmes d'emploi du temps d'examen dans les centres de formation [9].

En 2003,Merlot et al, ont utilisé la programmation par contraintes en utilisant l'OPL (an optimisation programming language), pour produire des solutions initiales. Ensuite, la méthode recuit simulé et hill climbing ont été utilisées pour améliorer les solutions [10].

Naji Azimi ,en 2004, a mis en place un système de colonies de fourmis et l'a comparé avec simulated annealing, tabu search (recherche tabou) et un algorithme génétique sous un framework unifié pour résoudre des problèmes d'emploi du temps, il a été constaté que les

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

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

Table III.1 : tableau récapitulatif des travaux effectués pour l'automatisation d'un emploi du temps.

III.3 Complexité

La confection d'emploi du temps dans les établissements scolaires est un travail très important, difficile à réaliser, c'est typiquement un problème de résolution de contraintes, NP-complet, dont la solution n'est pas a priori connue dans le cas général. Pour fournir une solution, nécessite d'être capable de s'adapter aux changements dynamiques de l'environnement en tenant compte de la diversité des contraintes .

D'une manière générale, pour résoudre le problème, on est appelé à trouver l'algorithme le plus efficace. Cette notion d'efficacité induit normalement toutes les ressources de calcul nécessaires pour exécuter un algorithme. Or, le temps d'exécution est généralement le facteur dominant pour déterminé si un algorithme est assez efficace pour être

III.3.1 Classes de complexité

Les classes en temps classiques sont :

- P, la classe des problèmes décidés en temps polynomial par une machine déterministe. Ces problèmes sont souvent considérés comme ceux pour lesquels il existe un algorithme efficace;
- NP, la classes des problèmes décidés en temps polynomial par une machine non déterministe;
- EXPTIME, la classe des problèmes décidés en temps exponentiel par une machine déterministe ;
- NEXPTIME, la classe des problèmes décidés en temps exponentiel par une machine non-déterministe.

III.4 Résolution d'un CSP

Le concept CSP permet de représenter un grand nombre de problèmes, pour le résoudre (CSP), diverses techniques ont été mises au point, et il n'y a pas de méthode universelle pour une résolution efficace, parmi ces méthodes on cite :

III.4.1 Méthodes exactes (complètes)

La plupart des algorithmes de recherche exacte pour les problèmes de satisfaction de contraintes sont basés sur l'algorithme Backtrack standard. De nombreuses améliorations majeures ont été proposées pour cet algorithme de base.
Ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité.

III.4.1.1 Backtrack

Cette méthode consiste à énumérer l'ensemble des solutions et à vérifier que chaque contrainte est satisfaite. Cette méthode donne une bonne réponse à tous les coups, mais conduit à développer un grand nombre de solutions potentielles rendant souvent les traitements très lourds. Pour trouver systématiquement une solution, l'algorithme affecte au fur et à mesure une valeur de D (domaines) à chaque variable correspondant dans X. Il

Le défaut de cet algorithme est qu'il peut tester toutes les valeurs possibles avant de trouver une solution. Si jamais la première bonne solution se trouve dans les dernières possibilités ou si jamais il n'y a pas de solutions, le temps de calcul sera très long [14].

```
fonction Backtrack(A,(X,D,C)) retourne un booléen
Précondition :
        A = affectation partielle
        (X,D,C) = un CSP sur les domaines finis
Postrelation :
        retourne vrai si A peut être étendue en une solution pour (X,D,C), faux sinon
début
        si A n'est pas consistante alors retourner faux finsi
        si toutes les variables de X sont affectées à une valeur dans A alors
        /* A est une affectation totale et consistante = une solution */  retourner vrai

        sinon /* A est une affectation partielle consistante */
                choisir une variable Xi de X qui n'est pas encore affectée à une valeur dans A
                pour toute valeur Vi appartenant à D(Xi) faire
                        si Backtrack(A U {(Xi,Vi)}, (X,D,C)) = vrai alors retourner vrai
                finpour
                retourner faux
        finsi
fin
```

Figure III.1: Algorithme backtrack [13]

A. Algorithmes avec retour arrière non chronologique

Quand il rencontre un échec, l'algorithme Backtrack remet en cause le dernier choix et essaie alors une nouvelle valeur pour la variable courante x. Lorsque toutes les valeurs ont été utilisées, il revient en arrière sur la variable précédente. Cependant, rien ne garantit que cette variable soit en cause dans les différents échecs rencontrés lors de l'affectation de x. Par exemple, si cette variable n'est pas liée à x par une contrainte, elle ne peut être impliquée dans ces échecs. Par conséquent, essayer de nouvelles valeurs pour cette variable conduira aux mêmes échecs au niveau de l'affectation de x. Pour pallier ce défaut du Backtrack, plusieurs méthodes dotées de retour arrière non chronologiques ont vu le jour. Le

- **L'algorithme Graph-based Backjumping (GBJ) :** le retour arrière repose sur le graphe de contraintes. Si l'affectation courante ne peut être étendue de façon consistante à une variable x, alors GBJ revient en arrière sur la variable la plus profonde du voisinage de x. soit y cette variable. Si toutes les valeurs du domaine de y ont été essayées, GBJ revient sur la variable la plus profonde qui appartienne soit au voisinage de y, soit au voisinage de toute variable instanciée après y qui soit une cause potentielle de l'échec d'une extension de l'affectation courante (x est en une), et ainsi de suite.

- **L'algorithme Conflict-directed Backjumping (CBJ) :** pour chaque variable instanciée x, CBJ maintient un ensemble dit ensemble des conflits. Cet ensemble contient toutes les variables instanciées avant x avec lesquelles x est en conflit pour au moins une de ses valeurs ainsi que les variables qui sont en cause dans l'échec d'une extension d'une affectation contenant x. lorsqu'un échec survient ou lorsque toutes les valeurs ont été essayées, on revient en arrière jusqu'à la variable la plus profonde de l'ensemble des conflits de la variable courante.

B. Algorithmes avec filtrage avant

- **look-back scheme :** L'algorithme teste la consistance de l'affectation courante en vérifiant qu'elle ne viole aucune contrainte liant x et une variable affectée. Autrement dit, le test de consistance s'effectue en fonction des variables déjà instanciées.
- **look-ahead scheme :** Suivant ce concept, à chaque affectation d'une variable x, les valeurs des variables non instanciées qui ne sont pas compatibles avec la valeur de x sont supprimées. Le calcul des valeurs à supprimer dépend alors du niveau de filtrage utilisé.

C. Algorithmes avec mémorisation

Les algorithmes avec retour arrière non chronologique permettent d'éviter certaines redondances dans l'arbre de recherche. Toutefois, il subsiste tout de même des redondances. Aussi, pour ne pas visiter plusieurs fois les mêmes sous arbres, une solution consiste à mémoriser des informations portant sur le travail déjà réalisé. Ces informations sont généralement mémorisées sous la forme de contraintes induites désignées généralement par le terme nogood. Un nogood correspond à une affectation qui ne peut être étendue en une solution. Leur mémorisation permet donc d'interdire certaines affectations et ainsi d'éviter de reproduire plusieurs fois les mêmes sous arbres. Parmi ces méthodes, on peut citer :

Si les méthodes de résolution exactes permettent d'obtenir une solution dont l'optimalité est garantie, dans certaines situations, on peut cependant chercher des solutions de bonne qualité, sans garantie d'optimalité, mais au profit d'un temps de calcul plus réduit. Pour cela, On applique des méthodes appelées métaheuristiques (méthodes approché), qui ont comme inconvénient de ne disposer en retour d'aucune information sur la qualité des solutions obtenues.

III.4.2 Méthodes approchées

III.4.2.1 Les méthodes approchées

Réparent une configuration donnée en parcourant de manière aléatoire l'espace de recherche. La méthode min-conflits est la plus répandue. Elle est construite autour d'une heuristique de réparation locale avec minimisation de conflits. L'heuristique consiste à choisir une variable intervenant dans une contrainte non satisfaite. Et à choisir pour cette variable, une valeur qui minimise le nombre de contraintes violées .

III.4.2.2 Les méthodes stochastiques

Sont aussi des méthodes approchées, mais elles ont une approche différente de celles citées précédemment. Les méthodes stochastiques commencent par une instanciation initiale et elles essaient de la réparer en faisant des changements de valeurs de certaines variables. Contrairement aux autres méthodes approchées, celles-ci admettent une dégradation de la solution, d'une façon qui suit certaines règles qui font que, l'algorithme puisse sortir d'un optimum local. Parmi ces méthodes on cite le recuit simulé, la recherche tabou, les algorithmes génétiques et les colonies de fourmis [16].

- **Recuit simulé**

Le " recuit simulé" a été inventé par Kirkpatrick, Gelatt et Vecchi (1983) Ils s'inspirent de la méthode de simulation de Métropolis (1950) en mécanique statistique. L'analogie s'inspire du " recuit des métaux " en métallurgie. Un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un minimum local pour un problème d'optimisation combinatoire (POC). Si on le refroidit lentement, les atomes se

d'un minimum local. Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi il faut que l'heuristique ait de la mémoire : le mécanisme consiste à interdire (d'où le nom de tabou) de revenir sur les dernières positions explorées.

- **Les algorithmes génétiques**

Les algorithmes génétiques (AG) sont des algorithmes d'optimisation stochastique fondés sur les mécanismes de la sélection naturelle et de la génétique. Ils ont été adaptés à l'optimisation par John Holland, également les travaux de David Goldberg ont largement contribué à les enrichir. Le vocabulaire utilisé est le même que celui de la théorie de l'évolution et de la génétique, on emploie le terme individu (solution potentielle), population (ensemble de solutions), génotype (une représentation de la solution), gène (une partie du génotype), parent, enfant, reproduction, croisement, mutation, génération, etc.

Leur fonctionnement est extrêmement simple, on part d'une population de solutions potentielles (chromosomes) initiales, arbitrairement choisies. On évalue leurs performances (Fitness) relatives. Sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires simples : la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre. On recommence ce cycle jusqu'à ce qu'on trouve une solution satisfaisante. Nous le détaillons plus dans la section III.5.

- **Les algorithmes de colonies de fourmis**

Les algorithmes de colonies de fourmis sont des algorithmes itératifs à population où tous les individus partagent un savoir commun qui leur permet d'orienter leurs futurs choix et d'indiquer aux autres individus des choix à suivre ou à éviter. Le principe de cette métaheuristique repose sur le comportement particulier des fourmis, elles utilisent pour communiquer une substance chimique volatile particulière appelée phéromone grâce à une glande située dans leur abdomen. En quittant leur nid pour explorer leur environnement à la recherche de la nourriture, les fourmis arrivent à élaborer des chemins qui s'avèrent fréquemment être les plus courts pour aller du nid vers une source de nourriture. Chaque fourmi dépose alors une quantité de phéromones sur ces pistes qui deviendront un moyen de

III.5.1 Définition formelle :

Le PET consiste à concilier des ressources et les contraintes, ces contraintes portant sur :

- Des disponibilités temporelles ;
- Des compétences ;
- Le besoin des ressources spécifiques (matériels pédagogiques).

Formellement on peut représenter le problème de l'emploi du temps comme suit :

- un ensemble de Groupes : G1 …, G_m ; avec m est le nombre de groupes.
- un ensemble de professeurs P1 …, P_n ; avec n est le nombre de professeurs.
- un ensemble de périodes de 1..., H ; avec H est le nombre de périodes.

Une matrice (X_{ijk}) indique les contraintes entre les éléments (salles, professeurs, cours). La formulation mathématique est comme suit:

PET: consiste à Trouver X_{ijk} (i=1...m; j=1....n; k= 1...h)

$$X_{ijk} = \begin{cases} 1 \text{ si le groupe } G_i \text{ et le professeur } P_j \text{ se réunissent à la période k} \\ 0 \text{ dans le cas contraire} \end{cases}$$

- **Formulation des contraintes**

Un horaire peut être considéré comme satisfaisant s'il convient aux professeurs, aux élèves et s'il respecte la disponibilité des salles. Nous mentionnons ci-dessous les contraintes les plus fréquentes et celles dont nous allons tenir compte par la suite :

1. **Non chevauchement des professeurs** : un professeur ne peut pas enseigner plus d'une leçon à la fois.
2. **Non chevauchements des étudiants** : un étudiant ne peut suivre qu'une seule unité pédagogique à la fois.
3. **Contraintes d'affectation d'étudiants** : un étudiant est affecté à un seul groupe de cours, TD ou TP.
4. **Contraintes de pause :** l'enseignant ou l'étudiant ont au moins une pause de 45 minutes aux heures de midi.
5. **Disponibilité des professeurs :** l'emploi du temps essaie de respecter la journée de disponibilité des enseignants.
6. **Contraintes sur les horaires des cours :** on ne peut commencer une unité pédagogique si

III.5.2 Modélisation

Le PET se défini par huit ensembles :
Un ensemble Professeurs = $\{P_1, \ldots, P_P\}$, un ensemble Modules = $\{M_1,\ldots,M_M\}$, un ensemble Groupes = $\{G_1,\ldots, G_G\}$, un ensemble Salles = $\{S_1,\ldots,S_S\}$, Heures = $\{H_1,\ldots,H_H\}$,
Jour = $\{J_1,\ldots,J_J\}$, filière = $\{F_1,\ldots,F_F\}$, Section = $\{SC_1,\ldots,SC_{SC}\}$ et un ensemble Contraintes contenant l'ensemble des contraintes entre les variables des cinq ensembles précédents.
La connaissance du problème permet pour chaque professeur P_i de l'ensemble Professeurs, est fixé quel module et pour quel groupe appartenant à quel section. Il reste donc qu'à fixer pour chaque cours la salle et le créneau et le jour qui seront alloués et qui satisfassent les contraintes. Le CSP proposé se présente de la façon suivante :
V = { P, M, G, S, H, J, F, SC}
D = {D(P)=[1,p], D(M)=[1,m], D(G)=[1,g], D(S)=[1,s], D(H)=[1,h], D(J)=[1,j], D(F)=[1,f], D(SC)=[1,sc]}.
C= {les contraintes}.

III.6 Méthode utilisée

Pour résoudre les problèmes complexes ;il faut soit utiliser les méthodes qui s'inspire du cerveau humain ou bien des méthodes qui s'inspire de la théorie de l'évolution.
Et puisque notre problème est fortement contraint et possède un vaste espace de recherche , nous avons opté pour les méthodes évolutionnistes , et on a choisi l'algorithme génétique comme méthode.

III.6.1 Définition de l'algorithme génétique

Il est inventés par John Holland, puis développés par David Goldberg, les algorithmes génétiques ont gagné une renommée certaine. Ils utilisent une représentation génotypique binaire en chaînes de bit de longueur fixe qui représentent les solutions du problème. Au cours des générations, les individus sont sélectionnés par tirage à la roulette, donnant une plus grande probabilité aux individus les plus adaptés à être sélectionnés ; ceux-là sont ensuite croisés entre eux, par un croisement en un point, puis les nouveaux individus sont mutés avec une petite probabilité par une mutation d'un bit [18].

L'algorithme génétique (AG) est un algorithme de recherche basé sur les

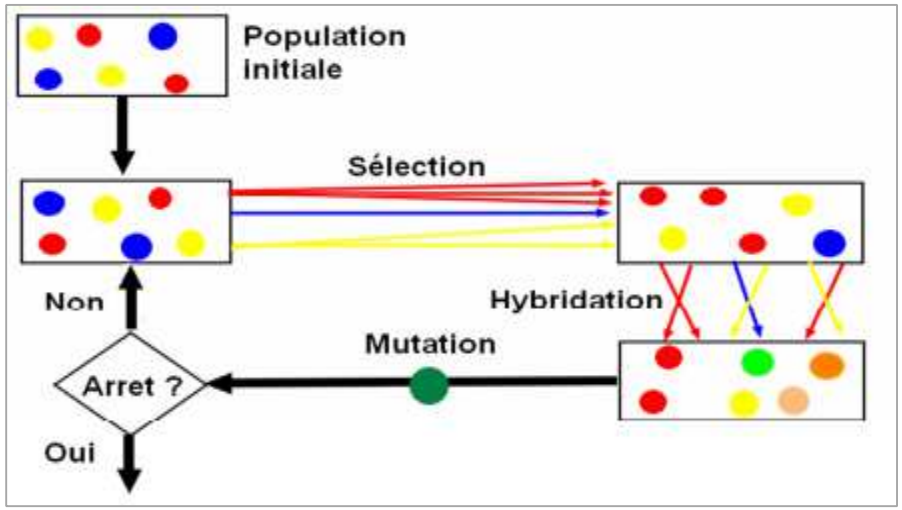


Figure III.2 : fonctionnement des AG [20].

III.6.2 Principe des AG

Un algorithme génétique recherche le ou les extrema d'une fonction défini sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

1. Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. Le choix du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très employés à l'origine. Les codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs, pour l'optimisation de problèmes à variables continues.

2. Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut rendre plus ou moins rapide la convergence vers l'optimum global. Dans le cas où l'on ne connaît rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le

5. Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt, probabilités d'application des opérateurs de croisement et de mutation.

- Une itération de l'algorithme général procède comme suit :
  1. à partir d'un ensemble de n parents
  2. produire une population de A enfants:
     a. choisir p parents
     b. recombiner les parents pour former un unique individu
     c. muter l'individu crée
  3. sélectionner les i meilleurs individus [21].

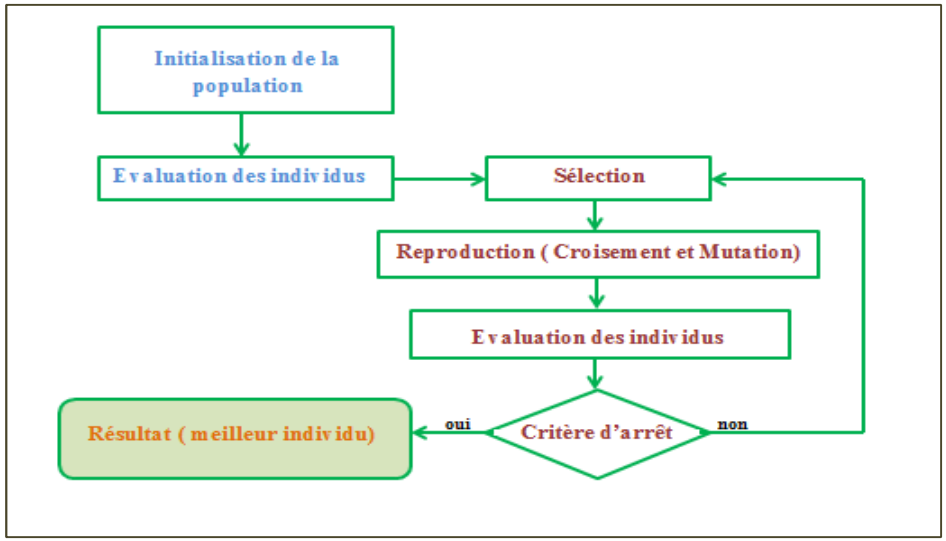


Figure III.3 : Organigramme d'un AG [21].

III.6.3 Opérateurs de variation

✓ **Les sélections:** Pour déterminer quels individus sont plus enclins à obtenir les meilleurs résultats, une sélection est opérée. Ce processus est analogue à un processus de sélection naturelle, les individus les plus adaptés gagnent la compétition de la reproduction tandis que les moins adaptés meurent avant la reproduction, ce qui améliore globalement

• Croisement uniforme: Il peut être vu comme un croisement multipoints dont le nombre de points de coupure n'est pas connu a priori. En pratique on utilise un masque binaire de la même longueur que les génotypes. Si à la nième position, il y a un 0, on conserve les symboles sinon on les échanges. Le masque est généré aléatoirement pour chaque couple.

✓ **La mutation:** De façon aléatoire, un gène peut, au sein d'un chromosome être substitué à un autre. De la même manière que pour les enjambements, on définit ici un taux de mutation lors des changements de population qui est généralement compris entre 0,1 et 0,01. Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution. La mutation sert à éviter une convergence prématurée de l'algorithme. Par exemple lors d'une recherche d'extremum la mutation sert à éviter la convergence vers un extremum local [22].

III.7 Algorithme génétique

Début

1. Générer une population aléatoire de n chromosomes.
2. Evaluer la fitness des chromosomes avec la fonction f(x).
3. appliquer l'opération de sélection.
4. Appliquer l'opération de croisement.
5. Appliquer l'opération de mutation.
6. Ajouter les nouveaux chromosomes à la nouvelle population.
7. Calculer la fonction fitness f(x), pour tout chromosome x
8. Revenir à 3
9. Jusqu'à la satisfaction des conditions de terminaison.

FIN

III.8 Conclusion

Dans ce chapitre nous avons mentionné que les méthodes proposées pour la résolution d'un CSP peuvent êtres classifiées principalement en deux grandes classes : les méthodes de

IV.1 Introduction

Dans ce chapitre nous décrivons les outils mis en œuvre pour la réalisation de notre application. Ce chapitre est composé de deux parties : la première partie présente l'environnement de développement et la seconde concerne les principales interfaces graphiques...

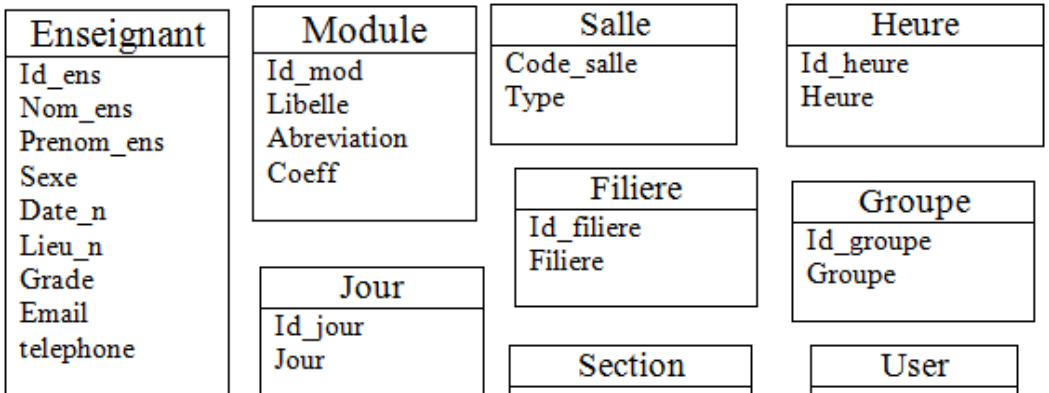
IV.2 Présentation de la méthode Merise

Durant la méthode d'analyse et de conception nous avons formalisé les étapes préliminaires du développement de notre système, cependant nous avons utilisé le modèle Entité-Association, En appliquant les règles de passage du MCD au MLD, nous avons construit notre base de données.

IV.2.1 Modèle Conceptuel des Données (MCD)

Le Modèle Conceptuel des Données (ou Modèle entité-association), permet de représenter la structure du système d'information, du point de vue des données, et définit également les dépendances ou relations entre ces différentes données [23].

- **Les entités déduites de la conception de notre système d'information sont :**



- **MCD**

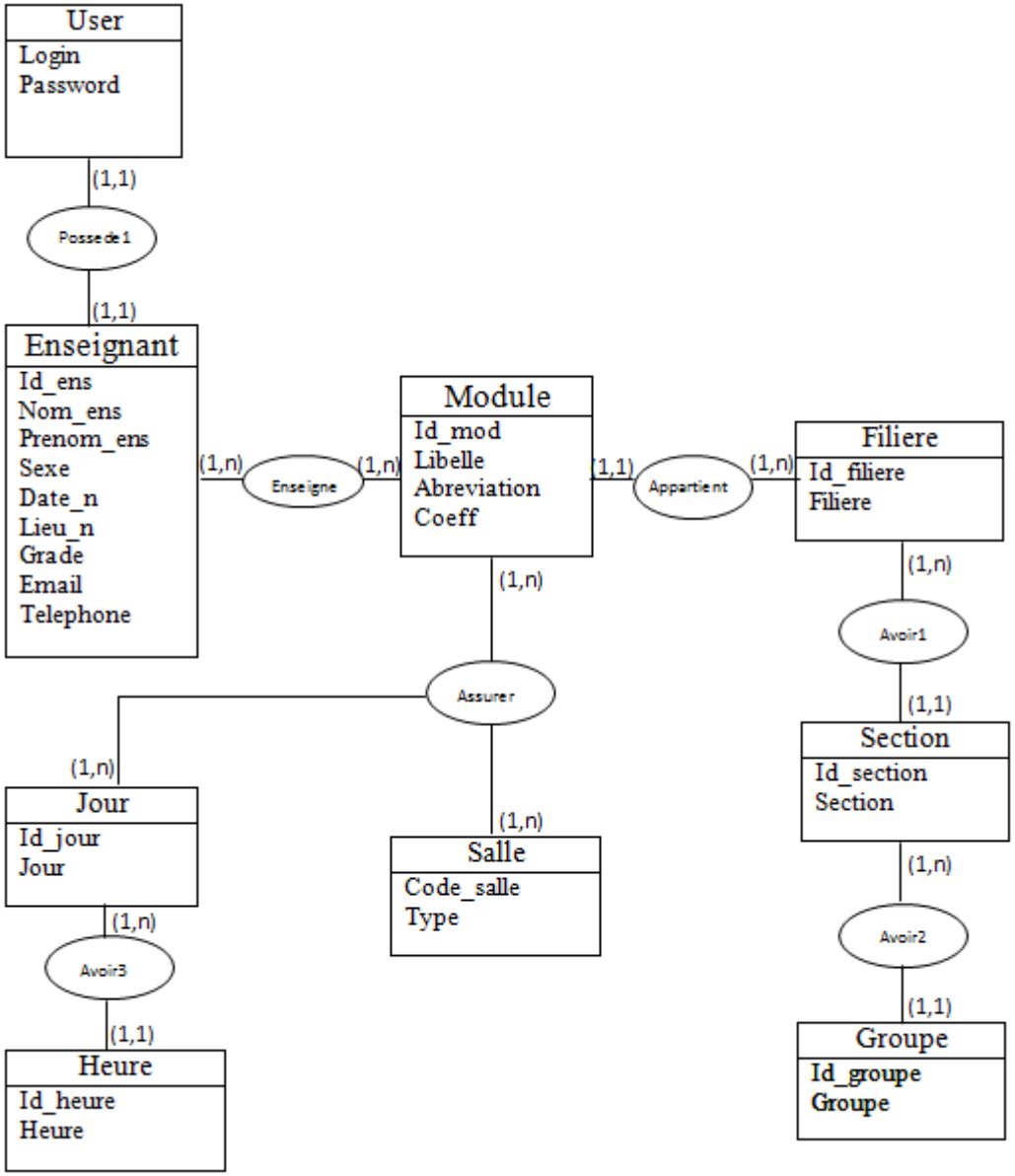


Figure IV.1 : Modèle conceptuel des données (MCD).

- User (Login, Password, Id_ens*)
- Enseignant (Id_ens, Nom_ens, Prenom_ens, Sexe, Date_n, lieu_n, Grade, Email, Telephone)
- Module (Id_mod, Libelle, Abreviation , Coeff, Id_filiere*)
- Enseigne (Id_mod*, Id_ens*)
- Salle (Code_salle, Type)
- Jour (Id_jour, Jour)
- Heure (Id_heure, heure, Id_jour*)
- Assurer (Id_jour*, Code_salle*, Id_module*)
- Section (Id_section , Section, Id_filiere*)
- Groupe (Id_groupe , Groupe, Id_section*)
- Enseigne(Id_ens*, Id_mod*)

IV.3 Environnement de développement

Lors du développement de cette application, on a utilisé les outils logiciels suivants :

- MySQL comme système de gestion de base de données.
- Netbeans 8.1 l'environnement de développement .
- WampServer pour coordonner le tout.

✓ NetBeans : NetBeans est à l'origine un EDI Java développé par une équipe d'étudiants à Prague, racheté ensuite par Sun Microsystems. C'est une plateforme, qui permet d'écrire des applications Swing. Ce qui fait de NetBeans une boite à outils facilement Améliorable ou modifiable.La licence de NetBeans permet de l'utiliser gratuitement à des fins commerciales ou non. Elle permet de développer tous types d'applications basées sur la plateforme NetBeans. Les modules que vous pourriez écrire peuvent être open-source comme ils peuvent être closed-source, Ils peuvent être gratuits, comme ils peuvent être payants.Il existe d'autres systèmes de développement rapide sous Windows mais Netbeans est particulièrement très bien placé grâce à ces propriétés [25].

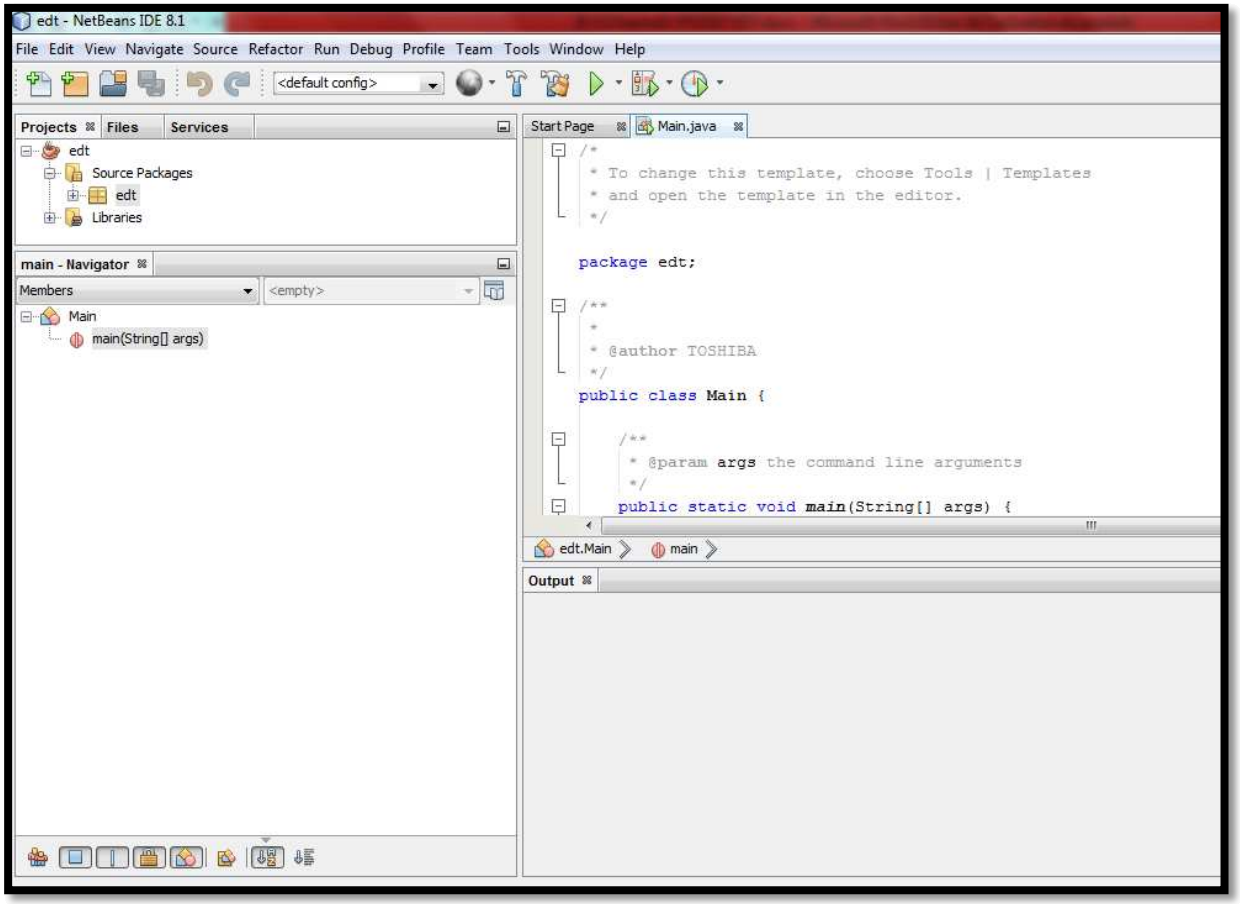


Figure IV.2 : Interface de NetBeans IDE 8.1

- ✓ MySQL : MySQL est un système de gestion de base de données (SGBD). Selon le type d'application, la licence est libre au propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications java principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server.

  MySQL est un serveur de bases de données relationnelles SQL développé dans un

- ✓ phpMyAdmin : est une application web qui permet de gérer un serveur de bases de données MySQL. Dans un environnement multiutilisateur, cette interface écrite en PHP permet également de donner à un utilisateur un accès à ses propres bases de données [27].

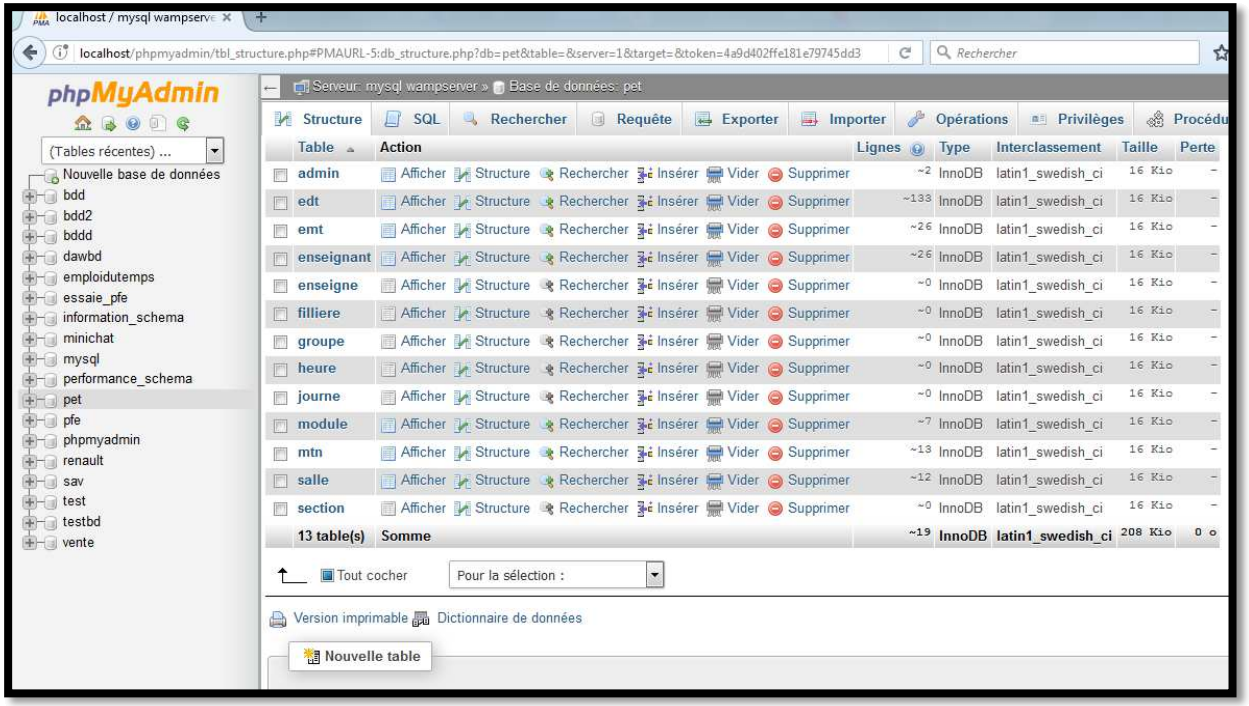


Figure IV.3: Interface de phpMyAdmin.

IV.4 Description de l'application

IV.4.1 Fenêtre principale

La fenêtre principale est devisée en deux parties (fenêtres) :

- La partie gauche : contient 5 onglets, les trois premiers représentent la liste des enseignants des modules et des salles et les deux derniers représente la relation entre ces entités

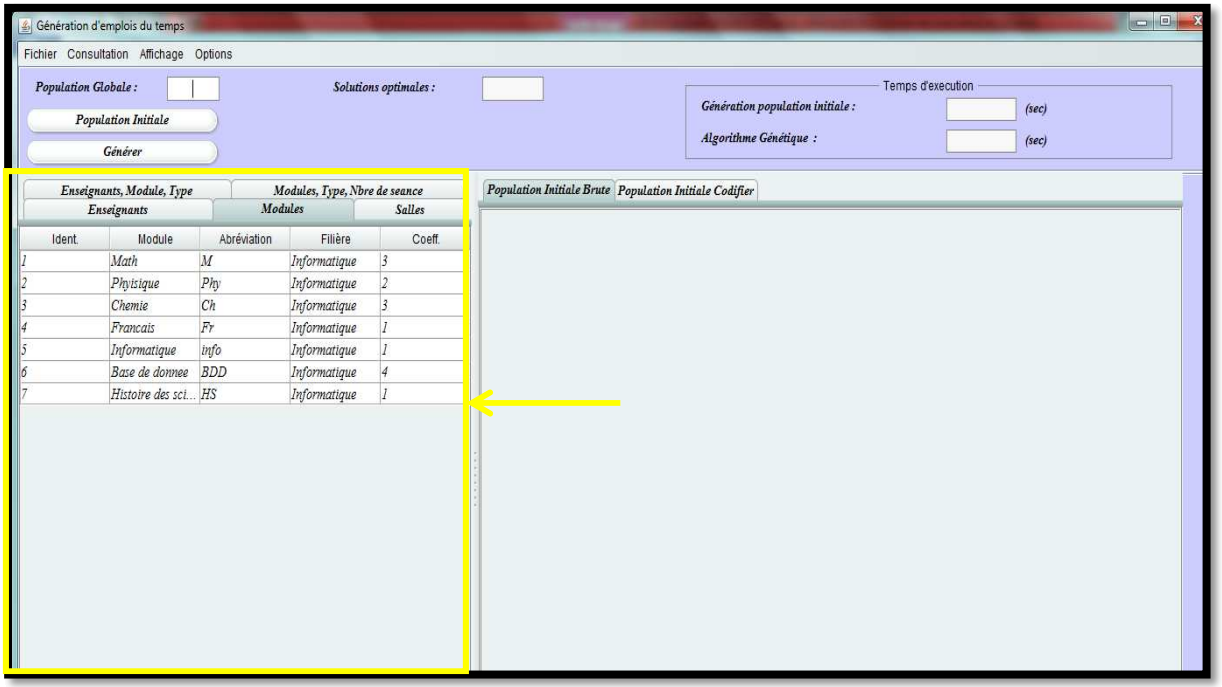
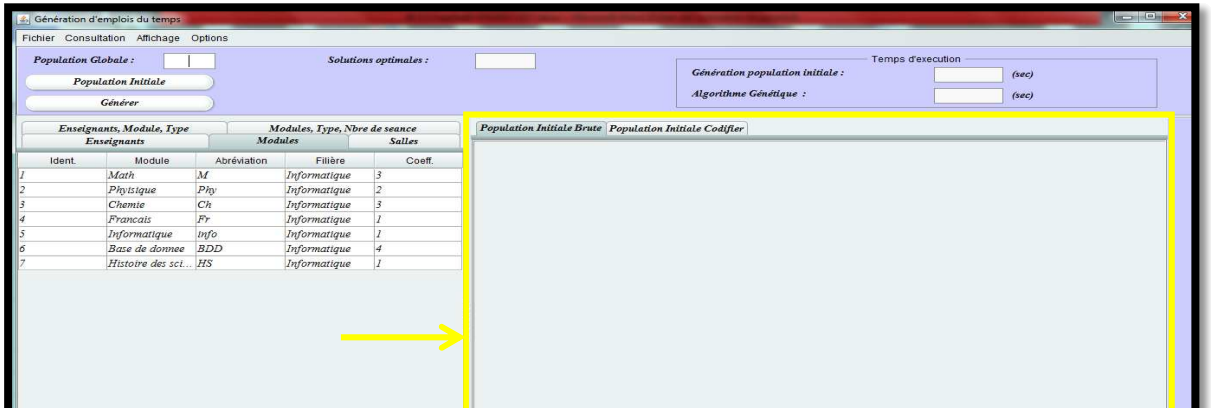


Figure IV.4: Liste des enseignants ,modules et salles.

- La partie droite : présente le traitement de l'algorithme génétique, elle contient deux onglets :un onglet qui montre la population initiale brute et l'autre qui présente la population initiale codifier.



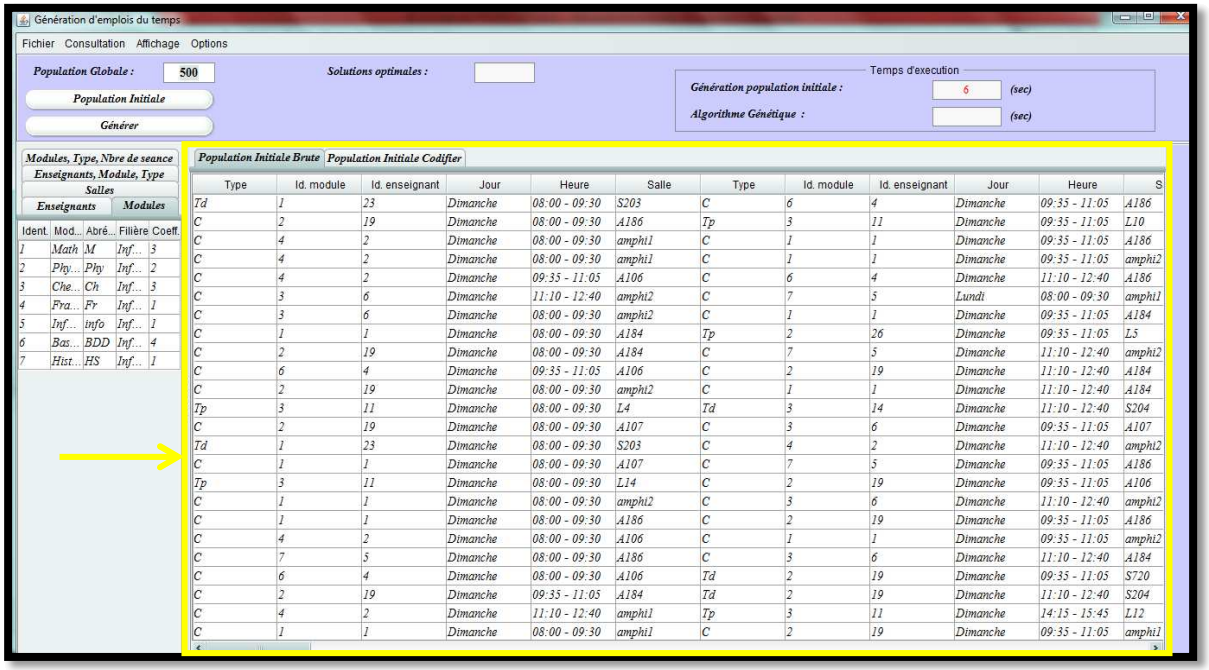
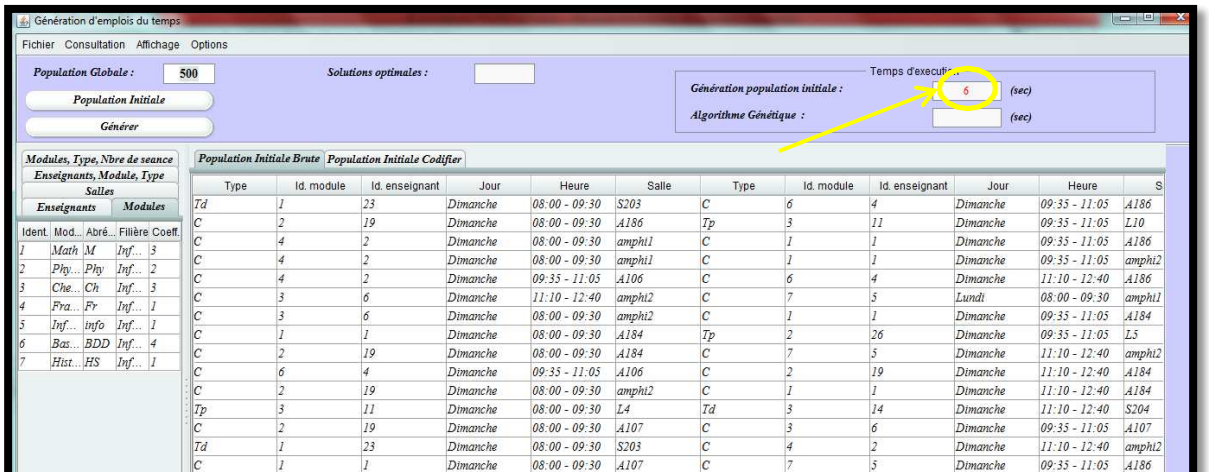


Figure IV.6: La population générée par l'algorithme génétique.

On constate en haut que l'algorithme a mis 6 secondes pour produire une population de 500 individus.



IV.4.2 Fenêtre paramètre

Pour lancer le traitement de l'algorithme génétique , on clique sur Option puis Paramètres.

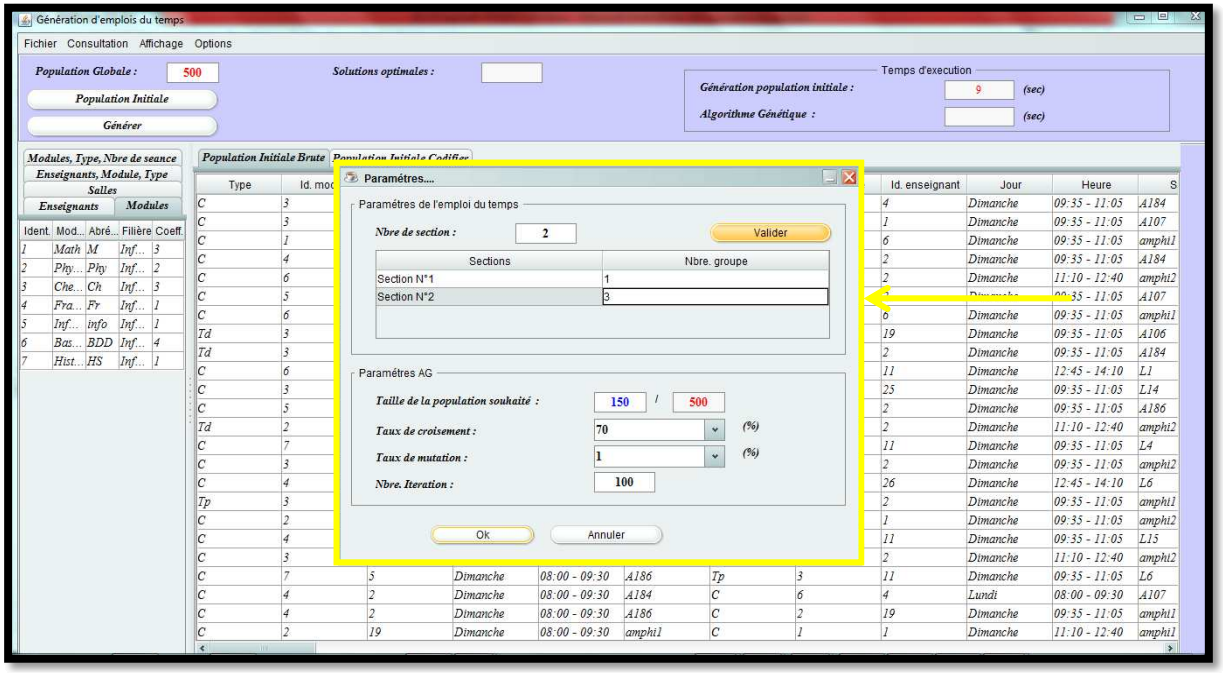


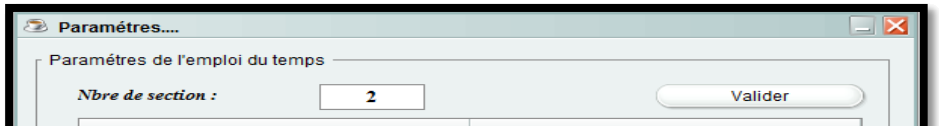
Figure IV.8: Paramètres.

Dans cette fenêtre l'utilisateur va pouvoir introduire des paramètres pour l'algorithme génétique.

D'abord il fixe le nombre de section ,puis en cliquant sur « Valider » il fixera le nombre de groupe pour chaque section.

Passant au paramètres de l'AG :

- On fixe le nombre d'individus que l'on souhaite survivre.



- On fixe après le taux du croisement.

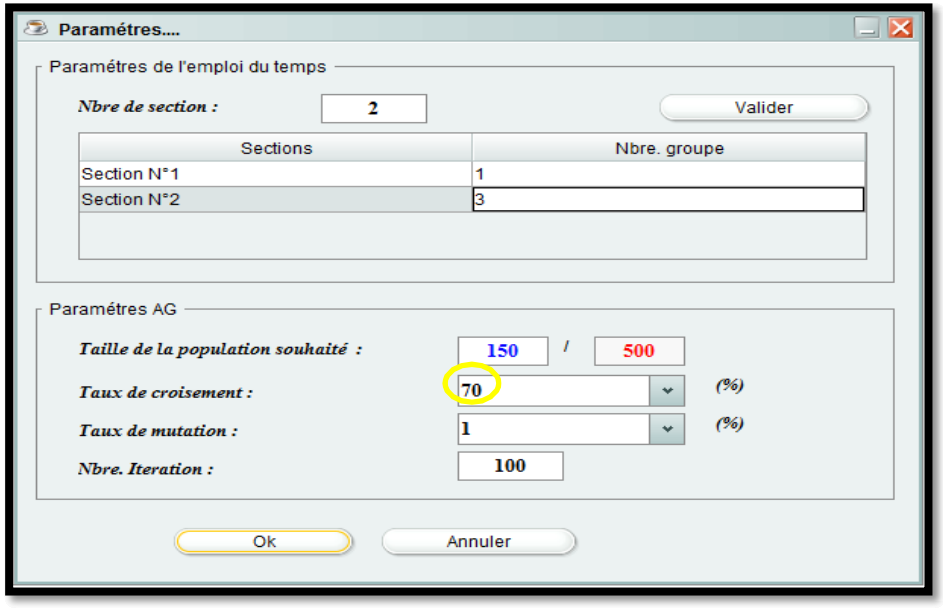
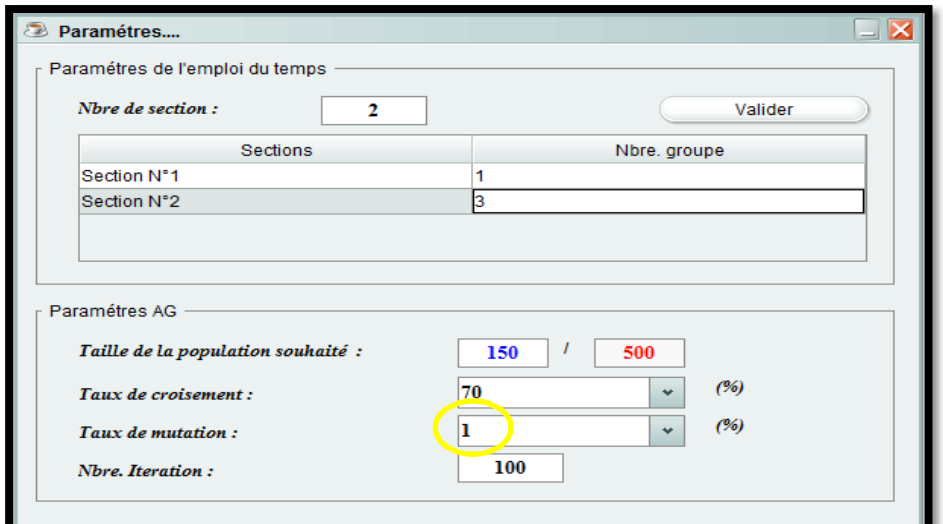


Figure IV.10: Le croisement.

- Puis on choisis le taux de mutation.



- Enfin on définit le nombre d'itérations.

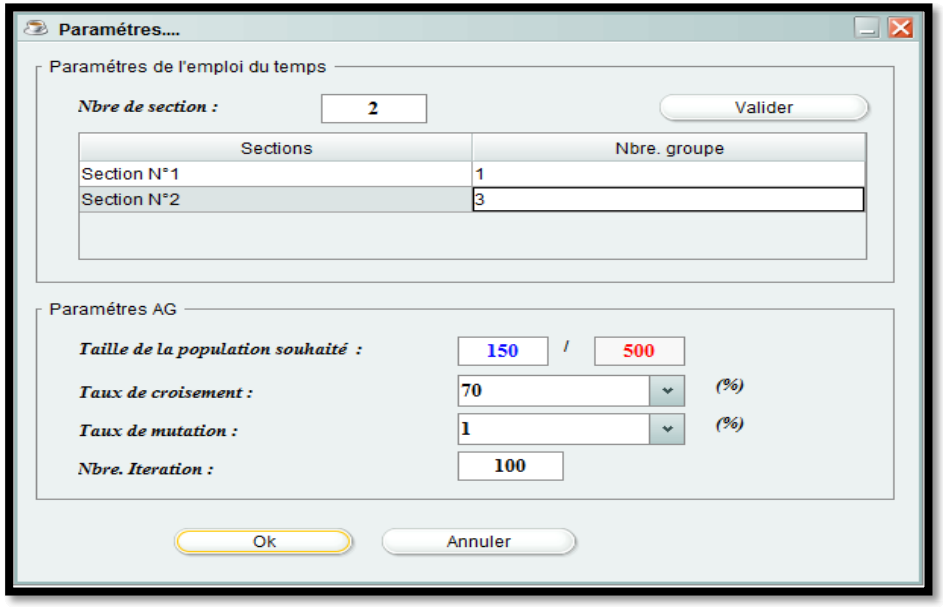
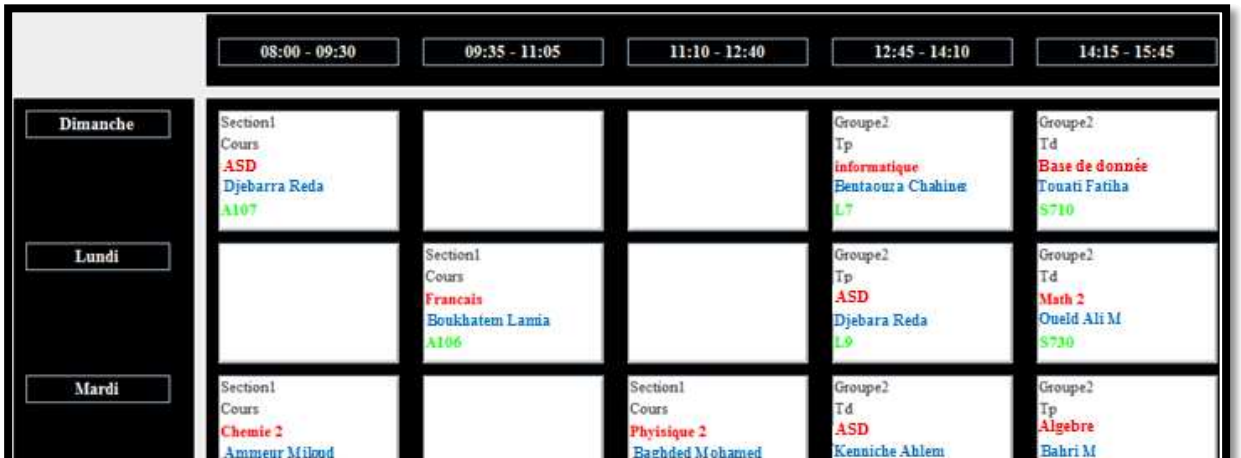


Figure IV.12: Nombre d'itérations.

IV.4.3 Fenêtre Emploi du temps :

Cette fenêtre est le résultat final de l'algorithme génétique qui est l'emploi du temps.



IV.5 Conclusion

Tout d'abord, on a commencé la conception en utilisant la méthode Merise, on a conçu le MCD puis le MLD, pour aboutir à notre base de données.
Ensuite, nous avons commencé la programmation en utilisant des technologies récentes: WampServer, Java , phpMyAdmin…
A la fin de la conception, le résultat sera une application qui fait la génération des emplois du temps en utilisant les méthodes stochastiques.

Conclusion générale

Le problème de planification des emplois du temps est considéré comme étant le problème académique, le plus contraignant dans le système de gestion universitaire, ce qui nécessite non seulement une intégration d'une solution au niveau du système globale, mais aussi un traitement préalable des donnés, afin de garantir et d'optimiser la solution.

Durant les derniers cinq mois, nous avons analysé et réalisé un projet de fin d'étude relatif à l'information de la génération d'emplois du temps au sein du département MI de l'université Abdelhamid Ibn Badis.

Après avoir analysé les données et les ressources dont un emploi du temps a besoin, on a introduit les problèmes d'optimisation liés à ce problème. Nous avons aussi présenté un modèle CSP.

Modéliser un problème ne consiste pas uniquement à trouver une manière de l'exprimer, mais plutôt à définir le modèle aboutissant à la résolution la plus efficace. Il n'existe malheureusement pas de méthodologie qui soit infaillible pour obtenir un bon modèle.

La résolution d'un tel problème nécessite l'application de certaines méthodes et stratégies que nous avons présentées et à tester dans notre projet.

Nous avons utilisé les metaheuristiques , qui sont des méthodes de résolution rapide et efficace pour les problèmes d'optimisation combinatoire a forte complexité.

Ainsi, nous avons appris les différentes étapes de conception d'un système d'information en utilisant la méthode Merise coté conception, Et WampServer, l'IDE Netbeans, Et le langage JAVA du coté programmation.

La plupart des nouveaux langages sont orientés objet. Le passage de la programmation fonctionnelle à l'orienté objet n'était pas facile. L'un des soucis était d'avoir une idée globale

Dans ce projet, nous avant exploité nos efforts et nos connaissances pour pouvoir bien analyser le sujet de la génération d'emplois du temps. Nous espérons que ce travail peut ramener un plus au département MI.

En désirants que nous aurons l'occasion de le développer et l'enrichir pour plus d'efficacité, selon les perspectives suivantes :

· L'extension du system aux autres départements;

· Utiliser les systèmes multi agents pour gérer plusieurs niveau en parallèle ;

· Améliorer l'interface du système;

· Héberger le système dans le serveur web de l'université Abdelhamid Ibn Badis Mostaganem.

Références :

[1] « Ressources », http://www.cnrtl.fr/definition/ressources, Date de consultation : 06/12/2016.

[2] « Planification », http://ekladata.com/VkWVgncXOy0FPlJv9MlpN1UHUmY/ress-planning-4.pdf, Date de consultation : 08/12/2016.

[3] **C.Y. Chéong**, **Peter**, « La planification du personnel : acteurs, actions et termes multiples pour une planification opérationnelle des personnes », Thèse de doctorat, Institut IMAG, Université Joseph Fourier-Grenoble, 1 octobre 2002.

[4] **A. Schaerf**, « A Survey of Automated Timetabling »,article, université de Rome,1 février 1999.

[5] **N. Berger**, « Modélisation et résolution en programmation par contraintes de problèmes mixtes continu/discret de satisfaction de contraintes et d'optimisation », Thèse de doctorat ,Université de Nantes, 31 Janvier 2011.

[6] « N_queens »,www.myuuu.fr/cours/PPC/nqueens.pdf, Date de consultation : 18/01/2017.

[7] **P. David**, **«** A constraint-based approach for examination timetabling using local repair techniques »,article, Springer lecture, 1998.

[8] **L. Paquete** et **T. Stutzle**, « Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem », article de conference,2002.

[9] **K. Sheibani** ,« An evolutionary approach for the examination timetabling problems », Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling, 21/23 Aout 2002.

[10] **L. Merlot, N. Boland, B.D. Hughes** et **P.J. Stucke**, « A hybrid algorithm for the examination timetabling problem »,article,Springer lecture,2003.

[11]**Z. Naji Azimi** , « Comparison of metaheuristic algorithms for examination timetabling problem », article Springer lecture, 2004.

[13]**C.ZheLiu,** «Approaches for consraint programming »,These de doctorat, Canada,1998.

[14] **S. Abdullah, S. Ahmadi,** « Investigating Ahuja-Orlins large neighbourhood search for examination timetabling », Livre: OR Spectrum,351- 372, 2007.

[15] **S. Minton, M. Johnston, A. Philips, P. Laird**, « Minimizing conflicts : a heuristic repair method for constraint satisfaction and scheduling problems », Livre: Artificial intelligence, 161-205, December 1992.

[16] **D.Johann, S.Patrick**, « Métaheuristiques pour l'optimisation difficile », Ouvrage, Éditions Eyrolles, 2003

[17] **J. Dréo , A. Pétrowski ,P. Siarry**, « Métaheuristiques pour l'optimisation difficile »,Ouvrage ,Éditions Eyrolles, 2003.

[18] **J.H.Holland**, « Adaptation In Natural And Artificial Systems », Ouvrage, University of Michigan Press,1975.

[19] **Goldberg, D.E**, «Genetic Algorithms in Search, Optimization and Machine Learning», Livre, Boston , 1989.

[20] «Algorithme génétique» , http://sis.univ-tln.fr/~tollari/TER/AlgoGen1/node3.html, Date de consultation : 19/03/2017.

[21] **Lamia Benameur** , « Contribution à l'optimisation complexe par des techniques de swarm intelligence », thèse de doctorat , Université Mohamed V Agdal Rabat Maroc, 13 Mai 2010.

[22] **Dejong K.A, W.M.Spears**, «A formel analysis of the role of multi-point crossover in genetic algorithms», Livre: Annals of Mathematics and Artificial Intelligence, 1-26,1992.

[23] «Merise», https://merise.developpez.com/faq/?page=MCD, Date de consultation : 15/04/2017.