الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

NºRef :……

Centre universitaire Abdelhafid Boussouf –Mila-

Institut des Mathématiques et Informatique **Département d'Informatique**

Mémoire préparé pour l'obtention du diplôme de Master

Spécialité : Intelligence Artificielle et ces Applications (I2A)

Une Approche évolutionnaire pour la génération automatique de l'emploi du temps du Collège d'Enseignement Moyen (CEM) El Amir Abdelkader –Mila-

Préparé par :

- **KERKOUCHE Affaf**
- **TOUTA Meriem**

Jury d'examen :

Dr. Zekiouk Mounira	**……**	**Président**
Dr. Bouzehzeh Mounira	**……**	**Examinateur**
Dr. Meghzili Said	**……**	**Superviseur**

Année académique: 2023/2024


بسم الله الرحمن الرحيم

Remerciements

Tous nos remerciements s'adressent tous d'abord à tout puissantALLAH, d'avoir guidé nos pas vers le chemin du savoir.Nous tenons à remercier Mr S.MEGHZILI, enseignant à le centre universitaire Abdelhafid Boussouf De MILA, pour son encadrement et ses encouragements à traversson attention, sa patienté, et ses conseils sérieux. Enfin, que toutes les personnes qui ont participé de prés ou de loin à la réalisation de ce travail trouvent ici l'expression de nos sincères remerciements.

Dédicaces

A mon père

Dans l'ombre de ton absence, chaque succès est teinté de tristesse, car tu n'es pas là pour le partager. Mais aujourd'hui, plus que jamais, je sens ta présence à mes côtés. Ce projet, qui marque la fin de mes études, est un hommage à toi, mon guide silencieux, dont l'amour et la force continuent de m'inspirer chaque jour.

A ma mère

source de mon courage et de ma persévérance, je dédie ce projet avec tout mon amour.

À mes sœurs Amina, Wafa, Sarra, et à mon frère Abderrahmane, vous avez été ma force et mon inspiration à chaque étape de ce parcours.

À ma binôme Affaf Kerkouche, je te remercie pour ton soutien et ta complicité tout au long de ce projet.

À tous mes amis et collègues de la ***première promo Intelligence Artificielle****, je vous remercie pour les moments partagés, les défis surmontés ensemble, et l'esprit d'équipe qui a rendu cette aventure inoubliable.*

Meriem

Dédicaces

ALa mémoire de mon père, qui restera toujours dans mon cœur et mon esprit.

A ma mère,pour tout ce qu'elle a fait et ce qu'elle fait encore pour moi.

A mon frère Omar, qui m'a guidée vers la joie de la réussite.

A mon mari Islam, qui m'encourage toujours.

A mes sœurs : Assia, Nadia, Siham et Asma, pour leur tendresse et leurs soutien.

A mon fils Mohamed Racim, la plus belle chose de ma vie, qui m'apporte chaque jour bonheur et joie.

A mesnièces.

A Mes neveux.

A mon amie et ma sœur Meriem Touta, pour son précieuse collaboration pour succès ce travail.

A tous mes collègues au CEM El AmirAbdelkader.

A tous mes amies et mescollègues du master 2 Intelligence Artificielle et ces Applications.

Affaf

ملخص

تُعد إدارة الوقت من التحديات الرئيسية في العديد من المجالات، خاصة في بيئات العمل. يتطلب كل مجال تنظيم العديد من المهام الأساسية بشكل متزامن، مما يجعل من الصعب جدًا، إن لم يكن مستحيلًا، إنشاء جدول زمني مثالي. لمواجهة هذا التحدي، اخترنا استخدام الخوارزميات الجينية كوسيلة لتطوير برنامج يُنشئ جدولًا زمنيًا يأخذ في الاعتبار جميع القيود، بهدف تحقيق توزيع مثالي للوقت يُقلل من الجهد البشري ويُعزز الكفاءة.

الكلمات المفتاحية: (الخوارزمية الجينية، CSP، الجدولة الزمنية، تحسين، قيود، جدول زمني.)

Abstract

The problem of timetabling is one of the problems that occupy many people, especially in the field of work. Each particular area has many bases that need to be organized at the same time. For someone to come up with a schedule that organizes these various tasks without any flaws, it is considered hardly impossible.

For this reason, we have chosen genitic algorithms as a way to develop a program that sets a schedule taking into account all the constraints in order to obtain an ideal time distribution that saves human effort and time.

Keywords: (Genetic algorithm, CSP, Timetabling, Optimisation, constraints, schedule)

Résumé

Le problème de la gestion du temps fait partie des problèmes qui occupent beaucoup de monde, notamment dans le domaine du travail. Chaque domaine particulier a beaucoup de bases qu'il faut organiser en même temps. Pour que quelqu'un propose un emploi du temps qui organise ces diverses tâches sans aucun défaut, il est considéré comme difficilement impossible.

Pour cette raison, nous avons choisi les algorithmes génétiques comme moyen de développer un programme qui fixe un échéancier prenant en compte toutes les contraintes afin d'obtenir une répartition temporelle idéale qui économise l'effort humain et le temps.

Mots clés : (Algorithme génétique, CSP, Emploi du temps, Optimisation, Contraintes, calendrier)

Table des matières

Table des figures

Chapitre 01 : Etude De L'existant

Chapitre 02 : Les problèmes de satisfaction des contraintes et les algorithmes génétiques

Chapitre 04 :Approche évolutionnaire pour le problème d'emploi du temps, Cas d'étude CEM El Amir Abd El Kader

Chapitre 05: Réalisation

Liste des tableaux

Chapitre 01 : Etude De L'existant

Chapitre 05: Réalisation

Liste des algorithmes

Chapitre 02 : Les problèmes de satisfaction des contraintes et les algorithmes génétiques

Chapitre 04 :Approche évolutionnaire pour le problème d'emploi du temps, Cas d'étude CEM El Amir Abd El Kader

Introduction générale

Un grand nombre de problèmes en intelligence artificielle et dans d'autres domaines de l'informatique peuvent être interprétés comme des problèmes de satisfaction de contraintes. L'exemple typique de cette catégorie de problèmes est la conception et la maintenance d'un emploi du temps dans divers domaines.

Pour résoudre ce type de problème, plusieurs approches en étés développées. Certaines utilisent la propagation des contraintes pour simplifier le problème, d'autres utilisent les différentes techniques de recherche sur le problème pour trouver la solution et les plus récentes utilisent des algorithmes évolutionnaires. Toutes ces techniques tendent vers le même but: résoudre les problèmes posés en vue de trouver des solutions les plus optimales possibles et en un minimum de temps.

Le problème de satisfaction de contraintes (CSPs) est un sous domaine de l'intelligence artificielle défini par un triplet (V, D, C) ou V représente un ensemble de variables, D représente un ensemble de domaines qui peuvent être finis ou non et C représente un ensemble de contraintes.

Parmi les techniques utilisées pour résoudre les CSPs, on trouve les algorithmes génétiques ou les algorithmes évolutionnaires qui sont inspirés de la nature. Ces techniques sont basées sur les opérations génétiques : le croisement, la mutation et la sélection. Pour résoudre n'importe quel problème de satisfaction des contraintes par les algorithmes génétiques, on doit d'abord trouver la représentation du problème par un chromosome et définir les critères de la sélection (la fonction de fitness). Cette approche consiste à exécuter les opérations génétiques de façon répétitive afin de générer des nouveaux individus qui se rapprochent au fils des générations de la solution idéale.

Dans le cadre de notre projet nous allons appliquer l'approche évolutionnaire (précisément les algorithmes génétiques) pour résoudre le problème d'emploi du temps d'un Collège d'Enseignement Moyen (CEM) Amir Abdelkader –Mila-.

Le problème d'emploi du temps est un problème classique qui consiste a faire affecter des cours, des classes d'étudiants et des enseignants a des créneaux et a des salles avec la satisfaction au maximum d'un ensemble de contraintes. On trouve trois type d'emploi du temps : l'emploi du temps des examens, l'emploi du temps des université et l'emploi du temps des écoles. Nous allons nous intéresser dans ce travail au dernier type d'emploi du temps.

Organisation du mémoire :

Notre mémoire est organisé en cinq (05) chapitre que nous les résumons comme suis :

Chapitre 01 : Dans ce chapitre, nous commencerons par présenter une étude générale du systèm pédagogique éducatif, en examinant les ressources humaines et matérielles pertinentes. Ensuite, nous nous concentrerons sur l'application existante de gestion des emplois du temps. Dans un premier temps nous présenterons le collège El Amir Abd El Kader.

Chapitre 02 : Ce chapitre est organisé comme suit, La première section commence par une définition de l'intelligence artificielle . Ensuite, la deuxième section introduit les problèmes de satisfaction de contraintes (CSPs) en explicitant les concepts clés . La troisième section illustre ces concepts à travers des exemples pratiques afin d'améliorer la compréhension. La quatrième section se concentre sur les différents types de problèmes CSPs , tandis que la cinquième section examine les diverses techniques de résolution de ces problèmes . La sixième section présente les origines des algorithmes génétiques, et enfin, la septième section aborde en détail le fonctionnement des algorithmes génétiques.

Chapitre 03 : Ce chapitre est organisé comme suit, dans la première section, le problème de timetabling (TTP) et ses caractéristiques sont exposés. La deuxième section aborde l'application des méthodes de recherche opérationnelle pour résoudre le TTP. La troisième section traite de l'interaction homme-machine dans le cadre de la résolution du TTP. Enfin, la quatrième section explore l'utilisation de l'intelligence artificielle pour aborder le TTP.

Chapitre 04 : dans ce chapitre,nous présenterons le processus génétique appliqué à la planification des emplois du temps scolaires. Ladeuxième section traite de la représentation du chromosome, tandis que la troisième aborde la représentation des

gènes. La quatrième section est consacrée à l'initialisation de la population, en détaillant les contraintes à respecter. ensuite, nous explorons la fonction de fitness. Les sections 6, 7 et 8 sont dédiées aux opérations génétiques,(le croisement, la mutation et la sélection).

Chapitre 05 : Ce chapitre débute par la présentation du langage de programmation choisi (Java), suivie de l'introduction de l'environnement de développement NetBeans, la troisième section présente la classe principale de gestion d'emploi du temps, en finaliser par le résultat de processus et une discussion de résultat.

CHAPITRE 1

Etude De L'existant

Introduction

L'étape d'analyse de l'existant est cruciale dans le cycle de vie d'un système, car ellepermet de comprendre la situation actuelle de l'organisation pour pouvoir prendre des décisionséclairées. L'analyse du système existant doit fournir toutes les informations nécessaires pour établirune bonne conception et proposer des solutions appropriées.

Dans ce chapitre, nous commencerons par présenter une étude générale du systèmepédagogique éducatif, en examinant les ressources humaines et matérielles pertinentes. Ensuite,nous nous concentrerons sur l'application existante de gestion des emplois du temps. Dans unpremier temps nous présenterons collège El Amir Abd El Kader.

1.1 Présentation de l'organisme d'accueil

1.1.1 Historique

Le CEM El Amir Abd El Kader une institution éducative indépendante sur le plan financier, relevant du ministère de l'Éducation nationale et sous la supervision de la direction de l'éducation de la wilaya de Mila. Fondée en 1962 puis il est devenu une institution en 1977, elle a été nommée en l'honneur du prince Abd El Kader. Le CEM fonctionne selon un régime demi-pension, accueillant plus de 250 élèves chaque année. Initialement réservée aux garçons, elle est devenue mixte en 1990. Il garantit à chaque élève une base de compétences essentielles dans les domaines de l'éducation, de la

culture et de la qualification, lui permettant de poursuivre ses études et sa formation au-delà de l'obligation scolaire ou de s'intégrer dans la vie professionnelle.

1.1.2 Organisation académique

L'organisation des études au niveau du cycle moyen s'étendent sur une période de quatre ans, où les élèves sont enseignés par des enseignants spécialisés dans différentes matières éducatives.

1.1.3 Présentation du collège

Le collège El Amir El Kader est situé au cœur de la ville de Mila, dans la wilaya de Mila, sur une superficie d'environ 76.77 hectares dont 26.23 hectares sont construits et 50.54 hectares sont non construits. Il peut accueillir jusqu'à 300 élèves selon un régime demi-pension.

Il est doté de six blocs suivants :

1- Bloc administratif et pédagogique.

2- Bloc des classes (16 classes).

3- Bloc des laboratoires (physique et science).

4- Bibliothèque et salle informatique.

5- terrain de sport.

6- résidence.

1.1.4 Organigramme pédagogique

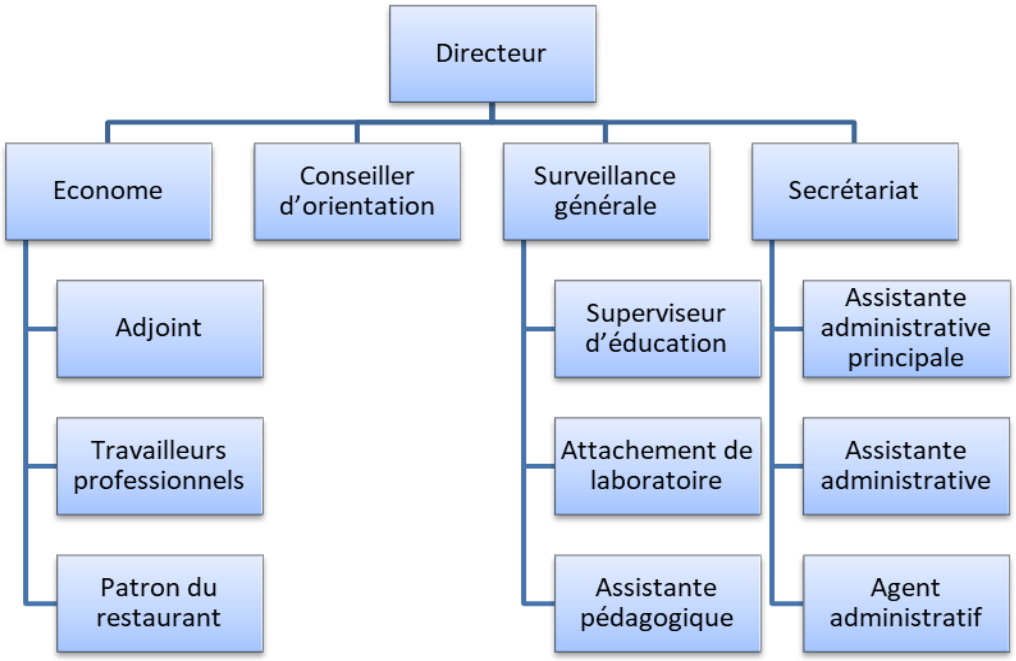


Figure 1.1 : l'organigramme pédagogique

1.1.5 Infrastructures et moyens humains

Pour assurer son bon fonctionnement, il est nécessaire de fournir tous les moyens nécessaires, à savoir :

- 16 classes.
- 04 laboratoires (physique et science).
- Une salle informatique (16 machines reliées et connectées au réseau local).

Les enseignants sont assurés par 32 enseignants répartis par matières comme suit :

Matières	Nombre d'enseignants
Mathématique	5
Science	3
Physique	3
Arabe	6
Histoire géo	3
Français	5
Anglais	4
Sport	2
Informatique	1

Tableau 1.1 : Les moyens humains.

1.2 Analyse de l'existant

1.2.1 Etude du système pédagogique

Le système pédagogique représente un élément fondamental d'un domaine global lié à la gestion des institutions éducatives, nécessitant la coopération de toutes les parties prenantes pour assurer le bon fonctionnement et l'efficacité de l'établissement. Cette organisation implique les responsabilités du directeur de l'établissement, notamment la formation des classes et leur répartition en fonction des matières et de l'utilisation du temps. Elle constitue ainsi l'un des piliers du travail éducatif, où tous les aspects pédagogiques, organisationnels et matériels se conjuguent de manière cohérente.

L'emploi du temps vise à atteindre plusieurs objectifs, parmi lesquels figurent notamment : assurer le bon fonctionnement de l'institution, permettre à son équipe de gérer et d'organiser le temps, ainsi que d'exploiter les ressources humaines et matérielles disponibles dans l'établissement selon un plan rigoureux qui favorise une interaction dynamique continue tout au long de l'année scolaire.

1.2.1.1 L'activité pédagogique

L'activité pédagogique est représentée par : les enseignants, les groupes et les matériaux sont considérés comme les ressources associées à l'enseignement. Les matériaux sont les ressources qui seront utilisées lors des séances d'enseignement. Il peut arriver qu'un enseignement soit dispensé simultanément par plusieurs enseignants.

1.2.1.2 Les ressources

Les ressources nécessaires à l'élaboration des emplois du temps comprennent les salles, les enseignants, les groupes, les élèves et les matériels.

Dans la suite nous décrivons les caractéristiques spécifiques des ressources considérées dans l'étude.

- **Les ressource de type « salle »**

  Une salle désigne un lieu réservé à la tenue des cours. Le type d'une salle indique le genre d'enseignement qui peut y être dispensé. Dans notre étude, collège El Amir Abd El Kader contient 16 salles, 4 laboratoires et une salle informatique.

- **Les ressources de type « enseignant »**

L'enseignant désigne une personne pouvant assurer des enseignements. Chaque enseignant est caractérisé par : son nom et son prénom, son grade, sa spécialité. La spécialité renseigne sur la matière que peut enseigner un enseignant.

- **Les ressources de type « groupes »**

  Un groupe est un ensemble des élèves. En général les élèves sont décomposer en niveau et chaque niveau est décomposé en plusieurs groupes. Le collège El Amir Abd El Kader se compose de :

  - 4 groupes de 1 er année.
  - 4 groupes de 2eme année.
  - 3 groupes de 3eme année.
  - 4 groupes de 4eme année.

- **Les ressources de type « matériel »**

  Pour qu'un enseignement puisse se dérouler, l'enseignant utilise un matériel pédagogique. Généralement ce matériel est limité à un tableau, éventuellement un rétroprojecteur et parfois un vidéo projecteur. Pour des enseignements particuliers, l'enseignant peut avoir besoin de matériel spécifique.

1.2.1.3 La notion de temps

L'emploi temps est une activité éducative et pédagogique importante qui permet à un groupe de classes de l'établissement de mener des activités pédagogiques et éducatives sous la supervision d'un groupe d'enseignants désignés à cet effet. Cela se fait à des moments et des endroits spécifiques, donc la planification doit être le résultat de l'interaction de trois éléments essentiels.

- **Les enseignants**
- **Les entités temporelles:***:* le début des cours commence à 8 heures et se termine à 16 heures**.**
- **Les séances et les réservations:**Une séance correspond à une instance temporelle d'un enseignement à une heure donnée. Une réservation correspond à une option posée sur l'occupation de cette ressource.

1.2.1.4 Contraintes

L'analyse pratique démontre que les données gérées doivent respecter des contraintes spécifiques pour maintenir leur cohérence. De façon abstraite ; les contraintes à respecter peuvent être classées en deux groupes.

- **Les contraintes physiques**

Ces contraintes doivent être observées afin d'éviter les conflits éventuels. Un "conflit physique de ressources" se produit entre deux séances, lorsque celles-ci partagent une ressource pour une période donnée. Voici les contraintes physiques:

- une ressource ne peut pas être occupée en même temps dans deux séances différentes.
- on ne peut pas mettre plus d'élevé qu'il n'y a de places dans une salle.
- le volume horaire total des séances d'un enseignement ne peut pas dépasser le volume prévu.
- les calendriers sont respectés pour toutes les ressources.

- **Les contraintes pédagogiques**

Les contraintes pédagogiques sont utilisées pour exprimer ce que doit être un bon emploi du temps.

Voici quelques exemples de ces contraintes :

- homogénéiser la durée des séances d'un enseignement.
- éviter les séances creuses dans l'emploi du temps.
- éviter de placer plusieurs séances d'un même enseignement dans une journée.
- éviter de finir après 17h.

Conclusion

Au cours de l'analyse de l'état actuel, nous avons collecté toutes les informations requises pour notre projet, à savoir la conception et la mise en œuvre d'un système de génération automatique et de gestion des emplois du temps pour le collège El Amir Abd El Kader. Ces informations, obtenues notamment grâce à l'étude du système pédagogique éducatif, nous ont permis de définir les objectifs principaux et de choisir les techniques à adopter.

CHAPITRE 2

Les problèmes de satisfaction des contraintes et les algorithmes génétiques

Introduction

Un grand nombre de problèmes en intelligence artificielle et dans d'autres domaines de l'informatique peuvent être interprétés comme des cas spéciaux de problèmes de satisfaction de contraintes(CSP).[1] Quelques exemples d'applications sont la conception et la maintenance de d'emploi du temps dans divers domaines de l'industrie et de l'économie, les problèmes de graphe, le design des plan de construction et les problèmes de satisfiabilité. Ces problèmes complexes, appartenant à la catégorie des NP-Complets,[3] nécessitent des approches sophistiquées pour être résolus efficacement. Plusieurs méthodes ont été développées pour y parvenir, allant de la propagation des contraintes à la recherche exhaustive et aux algorithmes probabilistes. Certaines d'entre elles utilisent la propagation des contraintes pour simplifier le problème. D'autres utilisent différentes techniques de recherche sur le problème pour trouver la solution. D'autres encore utilisent des algorithmes probabilistes. Enfin, l'utilisation des algorithmes génétiques permet de trouver des solutions encore plus performantes. Toutes ces techniques tendent vers le même but: résoudre les problèmes posés en vue de trouver des solutions le plus optimales possible et en un minimum de temps.[2]

Ce chapitre est organisé comme suit : La première section commence par une définition de l'intelligence artificielle . Ensuite, la deuxième section introduit les problèmes de satisfaction de contraintes (CSPs) en explicitant les concepts clés . La troisième section illustre ces concepts à travers des exemples pratiques afin d'améliorer la compréhension. La quatrième section se concentre sur les différents types de problèmes CSPs , tandis que la cinquième section examine les diverses techniques de

résolution de ces problèmes . La sixième section présente les origines des algorithmes génétiques, et enfin, la septième section aborde en détail le fonctionnement des algorithmes génétiques, en expliquant des aspects tels que le codage, la sélection, le croisement, la mutation et le remplacement .

2.1 Définitions des CSPs

Dans cette section, nous présentons quelques définitions fondamentales relatives aux Problèmes de Satisfaction de Contraintes (CSPs), afin de mieux comprendre les concepts sous-jacents à leur modélisation et leur résolution.

2.1.1 Définition 1

Un CSP est **m-ary** est un CSP qui contient **m** variables et chacune des contraintes dans le problème peut nécessiter un nombre quelconque de variables qui est entre un et la taille du problème **m**.[4]

2.1.2 Définition 2

Une fonction de pénalité est une fonction qui prend une affectation comme son entrée et elle retourne zéro pour une affectation satisfaisable, ou elle retourne une valeur plus grande que zéro si l'affectation non satisfaisable.[4]

2.2 Exemples

Un Problème de Satisfaction des Contraintes (CSP) peut revêtir une simplicité élémentaire, tel qu'un problème impliquant une seule variable arithmétique, ou une complexité plus élevée, comme le cas de la planification des emplois du temps scolaires avec des centaines de variables en jeu. Quels sont les avantages de modéliser un problème avec les CSP ? Et comment peut-on le faire ? Cela devient plus clair avec les exemples suivants. Il est important de noter que la représentation d'un problème n'est pas unique, et que différentes représentations peuvent influencer la recherche de solutions efficaces.

2.2.1 Exemple 1

Soit le problème CSP défini comme suit :

- Ensemble de variables V = {X1, X2, X3}
- Un domaine pour chaque variable D1= D2 = D3 ={1,2,3}.
- Une contrainte spécifiée par l'équation linéaire X1+ X2 = X3.

Il y a trois solutions possibles :

- {X1=1, X2=1, X3=2}
- {X1=1, X2=2, X3=3}
- {X1=2, X2=1, X3=3}

Ces 3 assignations sont complètes et compatibles.

2.2.2 Exemple 2 : Colorier une carte

Carte des provinces de l'Australie.

Figure 2.1 : Une carte géographique pour le problème du Colorier une carte.

- On vous demande d'utiliser seulement trois couleurs (rouge, vert et bleu) de sorte que deux provinces frontalières n'aient jamais les mêmes couleurs.
- On peut trouver une solution à ce problème en le formulant comme un problème CSP et en utilisant des algorithmes généraux pour CSP.
- Formulation du problème CSP :
- ***Lesvariables*** sont les provinces : V = { WA, NT, Q, NSW, V, SA, T }
- ***Le domaine*** de chaque variable est l'ensemble des trois couleurs : {R, G, B}
- ***Contraintes binaires*** : Les provinces frontalières doivent avoir des couleurs différentes
    - C={ WA≠ NT, …, NT≠ Q, … }

Solution: { WA=R, NT=G, Q= R, NSW=G, V=R, SA=B, T=G}

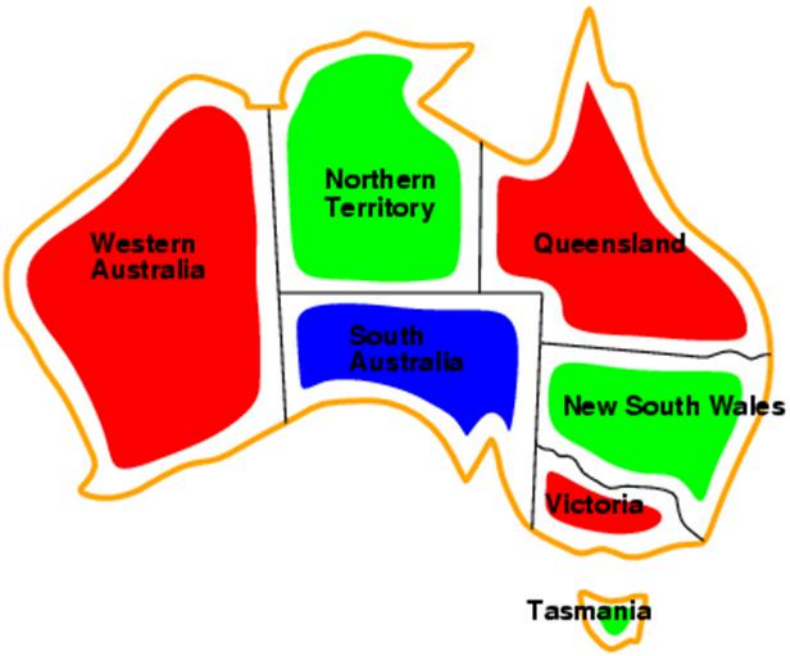


Figure 2.2 : **Une des solutions possibles pour le problème du Colorier une carte.[4]**

2.3 Types de problèmes CSP

- CSP avec **des domaines finis** (et discrets).
- CSP **Booléens**: les variables sont vraies ou fausses.
- CSP avec **des domaines continus** (et infinis). Par exemple, problèmes d'ordonnancement avec des contraintes sur les points (début/fin) dans le temps.
- CSP avec **descontraintes linéaires.**
- CSP avec **des contraintes non linéaires**.

2.4 Les techniques de résolution

Plusieurs méthodes de résolution ont été développées pour les CSPs (Problèmes de Satisfaction de Contraintes). Certaines de ces méthodes sont généralisées pour résoudre différents types de problèmes, tandis que d'autres sont spécifiquement conçues pour les CSPs. Les approches fondamentales pour la résolution des CSPs incluent la réduction du problème et l'utilisation d'algorithmes heuristiques. Dans cette section, nous examinerons quelques-unes de ces techniques en détail.

2.4.1 Algorithmes de Recherche de Base (Basic Search Algorithms)

La recherche constitue une méthode fondamentale dans la résolution des problèmes en intelligence artificielle (IA). Les algorithmes de recherche de base (BSA) sont simples à utiliser et servent de base pour des méthodes de recherche plus sophistiquées. Dans cette section, nous suivrons la convention qui divise les BSA en deux classes : la recherche systématique et la recherche stochastique. Des exemples de recherche systématique comprennent Generate-And-Test et le simple BACKTRACKING. Quant à la recherche stochastique, elle inclut des techniques telles que Random Guessing et Random-Walk

2.4.1.1 Les algorithmes systématiques de base (Basic Systematic Algotithms)

a) Generateand Test :

La méthode Generate-And-Test (GT) attribue des valeurs aux variables et obtient des affectations complètes. Ensuite, elle teste si l'affectation satisfait toutes les contraintes. Cette approche, connue sous le nom de Brute Force, consiste à essayer toutes les valeurs pour toutes les variables successivement et à vérifier la consistance de l'affectation. Comme aucune valeur n'est éliminée du domaine pendant la vérification de la consistance, la complexité de l'algorithme dans le pire des cas est |D1 * D2 * D3 * ... * Dn| = O(dn), où D1, D2,..., Dn représentent les domaines des variables. Étant donné que GT explore toutes les combinaisons possibles, il s'agit d'un algorithme complet qui fournit toutes les solutions (quel que soit le temps) s'il en existe. De plus, grâce à sa nature exhaustive, il peut être utilisé pour déterminer si un CSP est non satisfaisable.

Le Backtracking chronologique (Chronological Backtracking):

De manière systématique, cet algorithme explore l'espace de recherche en suivant une approche en profondeur. À chaque étape, une variable est instanciée, et l'algorithme continue jusqu'à ce qu'une solution soit trouvée ou que toutes les instances aient été explorées, démontrant ainsi qu'aucune solution n'existe. L'algorithme est plus efficace que l'approche gloutonne (GT) car il est capable de revenir en arrière dès qu'il détecte une incohérence dans l'affectation des variables. Concrètement, s'il constate qu'une variable instanciée ne viole pas une contrainte, il conserve cette affectation et passe à la variable suivante. En revanche, s'il rencontre une violation de contrainte, il rétrograde à la variable précédemment affectée et essaie une autre valeur pour cette variable, non encore explorée. Bien que cette méthode de recherche arrière soit fiable et exhaustive,

elle ne permet pas d'identifier précisément la source de l'incohérence, ce qui peut engendrer des faiblesses et des itérations répétées dans le processus de recherche.

2.4.1.2 Les algorithmes stochastiques de base (Basic stochastic Algotithms)

a) Generate and Test :

Cet algorithme représente une approche de recherche stochastique rudimentaire, comparable à la méthode de lancer des fléchettes à l'aveugle. Il procède en faisant plusieurs tentatives d'affectation aléatoire des variables, vérifiant chaque fois si cette affectation satisfait les contraintes, jusqu'à ce qu'une solution soit trouvée ou qu'un délai maximum soit atteint (ou un nombre maximal d'itérations). Étant donné que les variables sont affectées de manière non systématique, il est possible que certaines affectations soient vérifiées plusieurs fois, et il n'y a aucune garantie que toutes les possibilités seront explorées. En conséquence, l'algorithme n'est pas complet, ce qui signifie qu'il ne peut pas garantir la découverte d'une solution ou déterminer l'impossibilité de trouver une solution.

D'une autre part, la méthode est simple est rapide. Donc, elle peut être utilise pour un problème avec plusieurs solutions et toutes les solutions sont acceptées[4]

b) L'algorithme de la marche aléatoire (Random Walk Algorithm) :

RW, ou Random Walk, est une approche de recherche locale de base. Elle commence par initialiser toutes les variables, puis "marche" dans l'espace de recherche en priorisant la consistance, dans l'espoir de satisfaire toutes les contraintes ou jusqu'à ce qu'un délai maximum soit atteint. Tout comme dans le cas du Random Guessing, cette méthode est incomplète et ne garantit ni la découverte d'une solution, ni la détermination de l'impossibilité de trouver une solution. Pour pallier cette limitation, des méthodes heuristiques ont été développées afin d'améliorer l'efficacité de cette démarche.

2.4.2 La recherche stratégique et les heuristiques

Cette section met en évidence l'utilisation d'heuristiques et d'algorithmes hybrides pour améliorer les performances de résolution des CSP. Ces approches sont utilisées pour explorer de manière efficace l'espace de recherche, à la fois de manière systématique et stochastique.

2.4.2.1 La propagation des contraintes

Les techniques de propagation des contraintes telles que Backmarking(Marquage en arrière),, Backtracking(Retour en arrière),, Backjumping(Saut en arrière) et Forward Checking(Vérification en avant) sont discutées. Ces techniques permettent de réduire l'espace de recherche en éliminant les valeurs incompatibles pour les variables non affectées.

2.4.2.2 Les stratégies hybrids

Il est mentionné que divers algorithmes peuvent explorer plusieurs branches [1]et que des techniques hybrides combinent la force de différents algorithmes pour améliorer la recherche. Des exemples tels que Backmarking et Backjumping.[4]

2.4.3 La recherche stochastique

Cette section souligne que les méthodes systématiques ne sont pas toujours suffisantes pour résoudre efficacement les CSP difficiles, et que les méthodes stochastiques offrent une approche alternative en explorant de manière non déterministe l'espace du problème.[5]

La recherche locale

La recherche locale est présentée comme une méthode basée sur la stratégie de réparation, où une affectation initiale est générée et améliorée itérativement jusqu'à ce qu'une solution soit trouvée. Cependant, cette méthode n'est pas garantie pour trouver une solution et peut être incomplète.[4]

2.4.4 L'informatique Evolutionnaire

Cette approche, qui inclut les algorithmes génétiques, la programmation évolutive, etc., est mentionnée comme un paradigme populaire pour la résolution de problèmes, y compris les CSP. Elle utilise une population de solutions potentielles et des opérations de reproduction pour évoluer vers des solutions optimales.[4]

2.5 Les origines des algorithmes génétiques

C'est en 1860 que Charles Darwin publie son livre intitulé L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature. Dans ce livre, Darwin rejette l'existence «de systèmes naturels figés», déjà adaptés pour toujours à toutes les conditions extérieures, et expose sa théorie de l'évolution des espèces : sous

l'influence des contraintes extérieurs, les êtres vivants se sont graduellement adaptés à leur milieu naturel au travers de processus de reproductions.[6]

Darwin proposa une théorie qui clarifie l'évolution des espèces en mettant en avant quatre lois :

- La loi de croissance et de reproduction.
- La loi d'hérédité qu'implique quasiment la loi de reproduction
- La loi de variabilité, résultant des conditions d'existence.
- La loi de multiplication des espèces qui amène la lutte pour l'existence et qui a pour conséquence la sélection naturelle.[6]

Presque simultanément, en 1866, Mendel (le moine des poix) publie l'article retraçant dix années d'expériences d'hybridation chez les végétaux (recombinaison des gênes) et l'adresse aux sociétés scientifiques des quatre coins du monde. Les réactions sont mitigées, voire inexistantes. Le monde scientifique n'est pas prêt à reconnaître la qualité de ses résultats. Ce n'est seulement en1900, que la publication de trois nouveaux articles signés Hugo de Vries, Carl Correns et Erich vonTschermark révèle des résultats similaires à ceux de Mendel,et feront que ces premiers serontreconnus.

C'est alors à partir du 20èmesiècle que la mutation génétique a été mise en évidence.Les problèmes de traitement de l'information sont résolus de manières figés : lors de sa phase de conception, le système reçoit toutes les caractéristiques nécessaires pour les conditionsd'exploitations connues au moment de sa conception, ce qui empêche une adaptation à desconditions d'environnement inconnues, variables ou évolutives. Les chercheurs en informatiqueétudient donc des méthodes pour permettrent aux systèmes d'évoluer spontanément en fonction denouvelles conditions : c'est l'émergence de la programmation évolutionnaire.[6]

Dans les années 1960, John Holland étudie les systèmes évolutifs et, en 1975, il introduit le premier modèle formel des algorithmes génétiques (the canonical genetic algorithm AGC) dans son livre Adaptation in Natural and Artificial Systems. Il expliqua comment ajouter de l'intelligence dans un programme informatique avec les croisements (échangeant le matériel génétique) et la mutation (source de la diversité génétique). Ce modèle servira de base aux recherches ultérieures et sera plus particulièrement repris par Goldberg qui publiera en 1989, un ouvrage de vulgarisation des algorithmes génétiques, et ajouta à la théorie des algorithmes génétiques les idées suivantes :

- un individu est lié à un environnement par son code d'ADN.

- une solution est liée à un problème par son indice de qualité.

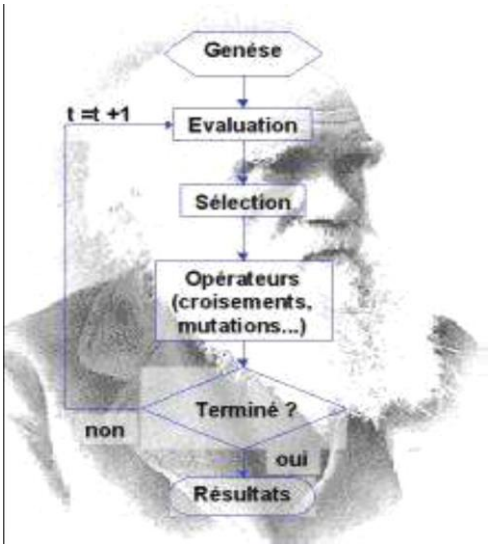


Figure 2.3 : Une carte géographique pour le problème du Colorier une carte.

Ci-dessus est présenté l'organigramme d'un algorithme évolutionnaire. Il s'agit de simuler l'évolution d'une population d'individus divers (généralement tirée aléatoirement au départ) à laquelle on applique différents opérateurs (recombinaisons, mutations…) et que l'on soumet à une sélection, à chaque génération. Si la sélection s'opère à partir de la fonction d'adaptation, alors la population tend à s'améliorer. Un tel algorithme ne nécessite aucune connaissance du problème: on peut représenter celui-ci par une boîte noire comportant des entrées (les variables) et des sorties (les fonctions objectif). L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc.[6]

Les algorithmes évolutionnaires constituent une approche originale: il ne s'agit pas de trouver une solution analytique exacte, ou une bonne approximation numérique, mais de trouver des solutions satisfaisant au mieux à différents critères, souvent contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes classiques, pour un même temps de calcul. [6]

Ils font parti du champ de la vie artificielle. La vie artificielle est l'étude des systèmes conçus par l'homme, qui présentent des comportements similaires aux systèmes vivants naturels. Elle complète l'approche traditionnelle de la biologie, définie étymologiquement par étude des êtres vivants, en essayant de synthétiser leurs

comportements sur support artificiel. La modélisation, s'ajoutant à l'observation, à la théorie et à l'expérience, est un nouvel outil scientifique qui s'est fait valoir depuis l'avènement de l'informatique. Celle-ci peut contribuer à la biologie théorique en la plaçant dans un contexte plus vaste. [6]

L'objectif est double: d'une part, la modélisation de ces phénomènes permet de mieux les comprendre, et ainsi mettre en évidence les mécanismes qui sont à l'origine de la vie; d'autre part, on peut exploiter ces phénomènes de façon libre et peuvent donc être diverses.

Le domaine de l'évolution artificielle n'a connu une réelle expansion qu'à partir de ces 15 dernières années. Pourtant, l'idée de simuler sur ordinateurs des phénomènes évolutionnaires remonte aux années 50. Des concepts tels que la représentation des chromosomes par des chaînes binaires étaient déjà présents. [6]

L'essor de l'évolution artificielle, depuis les années 80, peut s'expliquer par deux phénomènes concurrents. Premièrement, cet essor est principalement dû à l'accroissement exponentiel des moyens de calculs mis à la disposition des chercheurs, ce qui leur permet d'afficher des résultats expérimentaux pertinents et prometteurs. Le deuxième point est l'abandon du biologiquement plausible. [6]

Trois types d'algorithmes évolutionnaires ont été développé isolément et à peu prés simultanément, par différents scientifiques : la programmation évolutionniste, les Stratégies d'évolution et les Algorithmes Génétiques. [6]

Dans les années 90, ces trois champs ont commencé à sortir de leur isolement et ont été regroupés sous le terme anglo-saxon d'Evolutionnary Computation.

Nous traiterons seulement ici les algorithmes génétiques fondés sur le Néo-Darwinisme, c'est-à-dire l'union de la théorie de l'évolution et de la génétique moderne. Ils s'appuient sur différentes techniques dérivées de cette dernière : croisements, mutation, sélection... [6]

Un algorithme génétique recherche le ou les extrema d'une fonction définie 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. La qualité du codage des données conditionne le succès des algorithmes génétiques. Les codages binaires ont été très utilisés à l'origine. Les codages réels sont désormais

largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles. [6]

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 domaine de recherche. [6]
3. Une fonction à optimiser. Celle-ci retourne une valeur appelée fitness ou fonction d'évaluation de l'individu.
4. Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. L'opérateur de croisement recompose les gènes d'individus existant dans la population, l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'états. [6]
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. [6]

Nous savons maintenant sur quoi se basent les algorithmes génétiques.

Il est désormais temps d'approfondir les mécanismes de sélection de population et la notion de diversité qui en découle. Nous tacherons également de définir les opérateurs évoqués dans l'organigramme de l'algorithme évolutionnaire (figure 4). [6]

Donner une image à la fois globale et précise des outils principaux des algorithmes génétiques, tel sera notre objectif majeur au cours de notre seconde partie. [6]

2.6 Fonctionnement des algorithmes génétiques

Les algorithmes génétiques fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique. Selon cette méthode, des milliers de solutions (génotypes) plus ou moins bonnes sont crées au hasard puis sont soumises à un procédé d'évaluation de la pertinence de la solution mimant l'évolution des espèces : les plus "adaptés", c'est-à-dire les solutions au problème qui sont les plus optimales survivent davantage que celles qui le sont moins et la population évolue par générations successives en croisant les meilleures solutions entre elles et en les faisant muter, puis en relançant ce procédé un certain nombre de fois afin d'essayer de tendre vers la solution optimale. [6]

Le mécanisme d'évolution et de sélection est indépendant du problème à résoudre : seules changent trois fonctions :

- La fonction qui s'occupe de représenter le problème en codant chaque information caractérisant une solution possible selon un codage bien particulier, chaque information représente alors un gène et toutes les valeurs que peuvent prendre cette caractéristique représentent les allèles possibles pour ce gène, et en concaténant tous ces gènes pour obtenir un chromosome qui lui représente une solution dans son intégralité.
- La fonction inverse qui à partir d'un chromosome permet d'obtenir une solution par décodage du génome.[6]
- La fonction qui évalue l'adaptation d'une solution à un problème, sa pertinence.

En effet, quand on utilise les algorithmes génétiques, aucune connaissance de la manière dont résoudre le problème n'est requise, il est seulement nécessaire de fournir une fonction permettant de coder une solution sous forme de gènes (et donc de faire le travail inverse) ainsi que de fournir une fonction permettant d'évaluer la pertinence d'une solution au problème donné. Cela en fait donc un modèle minimal et canonique pour n'importe quel système évolutionnaire et pour n'importe quel problème pouvant être abordé sous cet angle, sous ce paradigme. [6]

Cette représentation nous permet donc d'étudier des propriétés quasiment impossibles à étudier dans leur milieu naturel, ainsi que de résoudre des problèmes n'ayant pas de solutions calculables en temps raisonnables si on les aborde sous d'autres paradigmes, avec des performances quantifiables, facilement mesurables et qu'on peut confrontés aux autres stratégies de résolution. [6]

Les algorithmes génétiques peuvent être particulièrement utiles dans les domaines suivants :

- Optimisation : optimisation de fonctions, planification, etc. ...
- Apprentissage : classification, prédiction, robotique, etc. ...
- Programmation automatique : programmes LISP, automates cellulaires, etc. ...
- Etude du vivant, du monde réel : marchés économiques, comportements sociaux, systèmes immunitaires, etc. ... [6]

Les principales différences des algorithmes génétiques par rapport aux autres paradigmes sont les suivantes :

- On utilise un codage des informations: on représente toutes les caractéristiques d'une solution par un ensemble de gènes, c'est-à-dire un chromosome, sous un

certain codage (binaire, réel, code de Gray, etc. ...), valeurs qu'on concatène pour obtenir une chaîne de caractères qui est spécifique à une solution bien particulière (il y a une bijection entre la solution et sa représentation codée.

- On traite une population "d'individus" de solutions : cela introduit donc du parallélisme.
- L'évaluation de l'optimalité du système n'est pas dépendante vis-à-vis du domaine.
- On utilise des règles probabilistes : il n'y a pas d'énumération de l'espace de recherche, on en explore une certaine partie en étant guidé par un semi-hasard : en effet des opérateurs comme la fonction d'évaluation permet de choisir de s'intéresser à une solution qui semble représenter un optimum local, on fait donc un choix délibéré, puis de la croiser avec une autre solution optimale localement, en général la solution obtenue par croisement est meilleure ou du même niveau que ses parents, mais ce n'est pas assuré, cela dépend des aléas du hasard, et cela et d'autant plus vrai pour l'opérateur de mutation qui ne s'applique qu'avec une certaine probabilité et dans le cas où il s'applique choisit aléatoirement sur quel(s) locus (loci) introduire des modifications. [6]

Un algorithme génétique générique à la forme suivante : [6]

1) Initialiser la population initiale P.
2) Evaluer P.
3) Tant Que (Pas Convergence) faire :
a) P ' = Sélection des Parents dans P
b) P ' = Appliquer Opérateur de Croisement sur P '
c) P ' = Appliquer Opérateur de Mutation sur P '
d) P = Remplacer les Anciens de P par leurs Descendants de P '
e) Evaluer P
Fin Tant Que

Figure2.4 : Format général d'un algorithme génétique.

Le critère de convergence peut être de nature diverse, par exemple :

- Un taux minimum qu'on désire atteindre d'adaptation de la population au problème.
- Un certain temps de calcul à ne pas dépasser.
- Une combinaison de ces deux points. [6]

2.6.1 Le codage

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).

Un chromosome est une suite de gène, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position : son locus sur le chromosome en question.

Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus. [6]

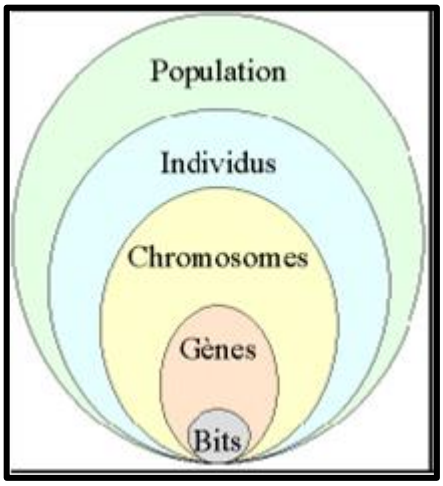


Figure 2.5 : Les cinq niveaux d'organisation d'un algorithme génétique.

Il y a trois principaux types de codage utilisables, et on peut passer de l'un à l'autre relativement facilement :

- ***Le codage binaire :*** c'est le plus utilisé.

Chaque gène dispose du même alphabet binaire **{0, 1}**

Un gène est alors représenté par un entier long (32 bits), les chromosomes qui sont des suites de gènes sont représentés par des tableaux de gènes et les individus de notre espace de recherche sont représentés par des tableaux de chromosomes.

Ce cas peut être généralisé à tout alphabet allénique n-aire permettant un codage plus intuitif, par exemple pour le problème du voyageur de commerce on peut préférer utiliser l'alphabet allénique **{c1, c2, c3, ..., cn}** où ci représente la ville de numéro **i**. [6]

- ***Le codage réel :*** cela peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle. [6]

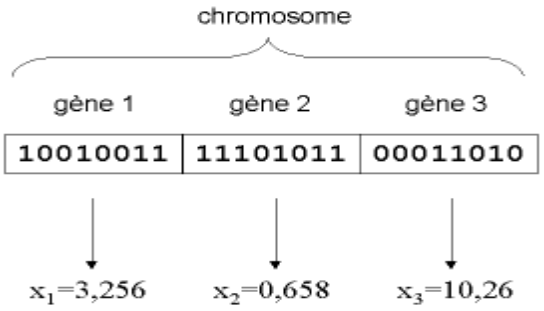


Figure 2.6: Illustration schématique du codage des variables réelles.

- ***Le codage de Gray :*** dans le cas d'un codage binaire on utilise souvent la "distance de Hamming" comme mesure de la dissimilarité entre deux éléments de population, cette mesure compte les différences de bits de même rang de ces deux séquences.

Et c'est la que le codage binaire commence à montrer ses limites. En effet, deux éléments voisins en terme de distance de Hamming ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "codage de Gray" : le codage de Gray est un codage qui a comme propriété que entre un élément **n** et un élément **n + 1**, donc voisin dans l'espace de recherche, un seul bit diffère. [6]

2.6.2 La sélection

Cet opérateur est chargé de définir quels seront les individus de P qui vont être dupliqués dans la nouvelle population P' et vont servir de parents (application de l'opérateur de croisement).

Soit n le nombre d'individus de P, on doit en sélectionner n/2 (l'opérateur de croisement nous permet de repasser à n individus).

Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population.

On trouve essentiellement quatre types de méthodes de sélection différentes :

- La méthode de la "loterie biaisée" (roulette Wheel) de GoldBerg,
- La méthode "élitiste",
- La sélection par tournois,
- La sélection universelle stochastique. [6]

2.6.3 Le croisement (Crossover)

Le Crossover utilisé par les algorithmes génétiques est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents.

Son rôle fondamental est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population. [6]

Cet opérateur est appliqué après avoir appliqué l'opérateur de sélection sur la population **P**; on se retrouve donc avec une population **P'** de **n/2** individus et on doit doubler ce nombre pour que notre nouvelle génération soit complète. [6]

On va donc créer de manière aléatoire **n/4** couples et on les fait se "reproduire".

Les chromosomes (ensembles de paramètres) des parents sont alors copiés et recombinés de façon à former deux descendants possédant des caractéristiques issues des deux parents.

Détaillons ce qui se passe pour chaque couple au niveau de chacun de leurs chromosomes :

Un, deux, voire jusqu'à **lg - 1** (où **lg** est la longueur du chromosome) points de croisements (loci) sont tirés au hasard, chaque chromosome se retrouve donc séparé en "segments". Puis chaque segment du parent 1 est échangé avec son "homologue" du parent 2 selon une probabilité de croisement **pc**. De ce processus résulte 2 fils pour chaque couple et notre population P' contient donc bien maintenant n individus. [6]

On peut noter que le nombre de points de croisements ainsi que la probabilité de croisement **pc** permettent d'introduire plus ou moins de diversité.

En effet, plus le nombre de points de croisements sera grand et plus la probabilité de croisement sera élevée plus il y aura d'échange de segments, donc d'échange de paramètres, d'information, et plus le nombre de points de croisements sera petit et plus la probabilité de croisement sera faible, moins le croisement apportera de diversité. [6]

Ci-dessous, un schéma illustrant un croisement en un point, un autre pour un croisement en deux points, et enfin un schéma représentant un croisement avec lg - 1 points de croisements (on notera d'ailleurs sur ce schéma que l'échange d'un segment avec son homologue ne se fait pas toujours) : [6]

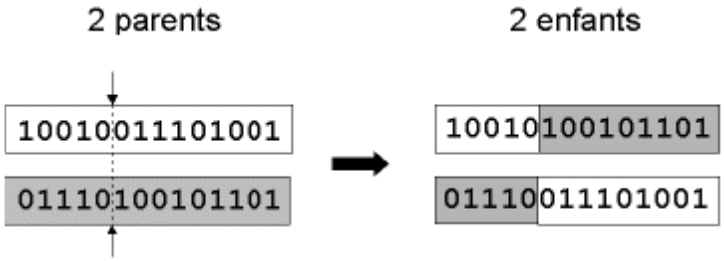


Figure 2.7: Croisement avec un point de Crossover.

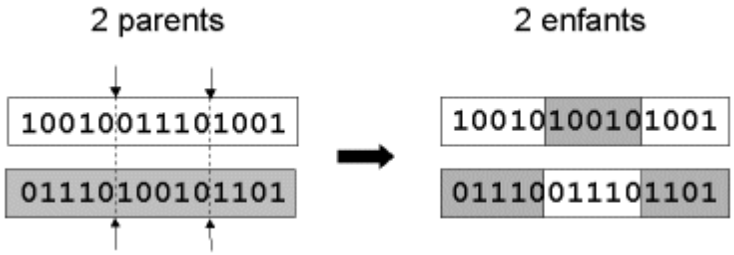


Figure 2.8: Croisement avec 2 point de Crossover.

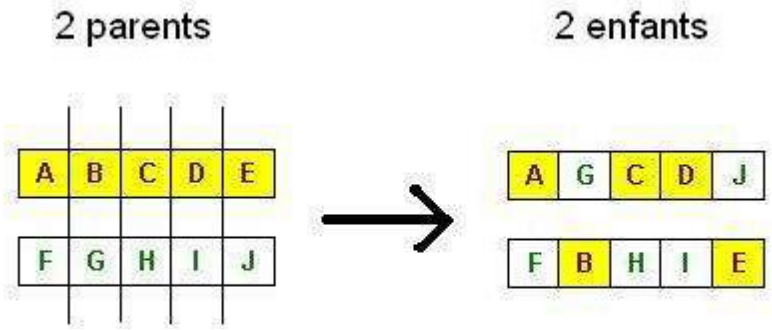


Figure 2.9: Le croisement uniform

On peut citer aussi une autre méthode très utilisée dans le cas des problèmes modélisés par un codage binaire, il s'agit du croisement uniforme. La mise en oeuvre de ce procédé est fort simple, elle consiste à définir de manière aléatoire un "masque", c'est-à-dire une chaîne de bits de même longueur que les chromosomes des parents sur lesquels il sera appliqué. Ce masque est destiné à savoir, pour chaque locus, de quel

parent le premier fils devra hériter du gène s'y trouvant; si face à un locus le masque présente un 0, le fils héritera le gène s'y trouvant du parent n° 1, si il présente un 1 il en héritera du parent n° 2. La création du fils n° 2 se fait de manière symétrique : si pour un gène donné le masque indique que le fils n° 1 devra recevoir celui-ci du parent n° 1 alors le fils n° 2 le recevra du parent n°2, et si le fils n° 1 le reçoit du parent n° 2 alors le fils 2 le recevra du parent n° 1. [6]

L'opérateur de croisement favorise l'exploration de l'espace de recherche. En effet, considérons deux gènes **A** et **B** pouvant être améliorés par mutation. Il est peu probable que les deux gènes améliorés **A'** et **B'** apparaissent par mutation dans un même individu. Mais si un parent porte le gène mutant **A'** et l'autre le gène mutant **B'**, l'opérateur de croisement permettra de combiner rapidement **A'** et **B'** et donc de créer un nouvel individu possèdent cette combinaison, combinaison grâce à laquelle il est possible qu'il soit encore plus adapté que ses parents. [6]

L'opérateur de croisement assure donc le brassage du matériel génétique et l'accumulation des mutations favorables. En termes plus concrets, cet opérateur permet de créer de nouvelles combinaisons des paramètres des composants.

Malgré tout, il est possible que l'action conjointe de la sélection et du croisement ne permette pas de converger vers la solution optimale du problème. [6]

En effet, imaginons que nous avons une population d'individus possédant un seul chromosome. Considérons un gène particulier de ce chromosome, on l'appellera **G**, gène ayant 2 allèles possibles :

0 et 1; si aucun individu de la population initiale ne possède l'allèle 1 pour ce gène, aucun croisement possible ne permettra d'introduire cet allèle pour notre gène **G**. Si la solution optimale au problème est telle que notre gène **G** possède l'allèle 1, il nous sera impossible d'atteindre cette solution optimale simplement par sélection et croisement. C'est pour remédier entre autre à ce problème que l'opérateur de mutation est utilisé. [6]

2.6.4 La mutation

Cet opérateur consiste à changer la valeur allélique d'un gène avec une probabilité **pm** très faible, généralement comprise entre 0.01 et 0.001.

On peut aussi prendre **pm = 1 / lg** où lg est la longueur de la chaîne de bits codant notre chromosome. Une mutation consiste simplement en l'inversion d'un bit (ou de plusieurs bits, mais vu la probabilité de mutation c'est extrêmement rare) se trouvant en

un locus bien particulier et lui aussi déterminé de manière aléatoire; on peut donc résumer la mutation de la façon suivante :

On utilise une fonction censée nous retourner true avec une probabilité **pm**.

Pour chaque locus faire

Faire appel à la fonction

Si cette fonction nous renvoie true alors

on inverse le bit se trouvant à ce locus

Fin Si

Fin Pour

Algorithme 1: L'algorithme général pour la mutation.

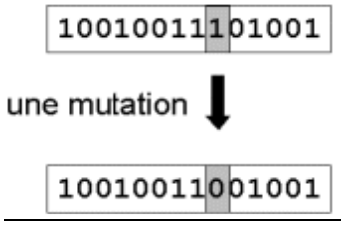


Figure 2.10: Une mutation

L'opérateur de mutation modifie donc de manière complètement aléatoire les caractéristiques d'une solution, ce qui permet d'introduire et de maintenir la diversité au sein de notre population de solutions. Cet opérateur joue le rôle d'un "élément perurbateur", il introduit du "bruit" au sein de la population. [6]

Cet opérateur dispose de 4 grands avantages :

- Il garantit la diversité de la population, ce qui est primordial pour les algorithmes génétiques.
- Il permet d'éviter un phénomène connu sous le nom de ***dérive génétique***. On parle de dérive génétique quand certains gènes favorisés par le hasard se répandent au détriment des autres et sont ainsi présents au même endroit sur tous les chromosomes. Le fait que l'opérateur de mutation puisse entraîner de manière aléatoire des changements au niveau de n'importe quel locus permet d'éviter l'installation de cette situation défavorable.
- Il permet de limiter les risques d'une convergence prématurée causée par exemple par une méthode de sélection élitiste imposant à la population une pression

sélective trop forte. En effet, dans le cas d'une convergence prématurée on se retrouve avec une population dont tous les individus sont identiques mais ne sont que des optimums locaux. Tous les individus étant identiques, le croisement ne changera rien à la situation. En effet, l'échange d'informations par Crossover entre des individus strictement identiques est bien sûr totalement sans conséquences; on aura beau choisir la méthode de croisement qu'on veut on se retrouvera toujours à échanger des portions de chromosomes identiques et la population n'évoluera pas. L'évolution se retrouvant bloquée on n'attendra jamais l'optimum global. La mutation entraînant des inversions de bits de manière aléatoire permet de réintroduire des différences entre les individus et donc de nous extirper de cette situation. Il est quand même utile de garder à l'esprit que ceci n'est pas une solution "miracle" et qu'il est bien entendu plus intelligent de ne pas utiliser de méthodes de sélection connues pour entraîner ce type de problème.

- La mutation permet d'atteindre la propriété d' ***ergodicité***. [6]

L'ergodicité est une propriété garantissant que chaque point de l'espace de recherche puisse être atteint. En effet, une mutation pouvant intervenir de manière aléatoire au niveau de n'importe quel locus, on a la certitude mathématique que n'importe quel permutation de notre chaîne de bits peut apparaître au sein de la population et donc que tout point de l'espace de recherche peut être atteint.

Grâce à cette propriété on est donc sûr de pouvoir atteindre l'optimum global.

On notera que la mutation règle donc le problème exposé à la fin du Section sur le croisement. [6]

2.6.5 Le remplacement

Cet opérateur est le plus simple, son travail consiste à réintroduire les descendants obtenus par application successive des opérateurs de sélection, de croisement et de mutation (la population **P**') dans la population de leurs parents (la population **P**).

On trouve essentiellement 2 méthodes de remplacement différentes :

2.6.5.1 Le remplacement stationnaire

Dans ce cas, les enfants remplacent automatiquement les parents sans tenir compte de leurs performances respectives, et le nombre d'individus de la population ne varie pas tout au long du cycle d'évolution simulé, ce qui implique donc d'initialiser la population initiale avec un nombre suffisant d'individus. Cette méthode peut être mise en oeuvre de 2 façons différentes :

- La première se contente de remplacer la totalité de la population **P** par la population **P'**, cette méthode est connue sous le nom de ***remplacement générationnel*** et on a donc unegénération gap qui vaut 1.
- La deuxième méthode consiste à choisir une certaine proportion d'individus de P' qui remplaceront leurs parents dans **P** (proportion égale à 100 % dans le cas du remplacement générationnel.

Ce type de remplacement engendre une population ayant une grande variation et de se fait favorise la dérive génétique qui se manifeste d'autant plus que la population est de petite taille. De plus dans bien des cas, étant donné que même un enfant ayant une faible performance remplace forcement un parent, on n'atteint pas la meilleure solution mais on s'en approche seulement. [6]

2.6.5.2 Le remplacement élitiste

Dans ce cas, on garde au moins l'individu possédant les meilleures performances d'une génération à la suivante. En général, on peut partir du principe qu'un nouvel individu (enfant) prend place au sein de la population que s'il remplit le critère d'être plus performant que le moins performant des individus de la population précédente. Donc les enfants d'une génération ne remplaceront pas nécessairement leurs parents comme dans le remplacement stationnaire et par la même la taille de la population n'est pas figée au cours du temps. Ce type de stratégie améliore les performances des algorithmes évolutionnaire dans certains cas. Mais présente aussi un désavantage en augmentant le taux de convergence prématuré.

Néanmoins, des implémentations plus fines procèdent de manière différente. Dans ce cas là, le taux de remplacement n'est pas de 100 %, la taille de la population augmente donc au cours des générations successives, on dit qu'il y a overcrowding. Il faut donc trouver un moyen pour sélectionner les parents qui seront supprimés, qui vont mourir. De Jong a proposé la solution suivante : imaginons qu'on veuille remplacer 30 % des parents, soit **np** le nombre de parents correspondants à ce pourcentage, on remplacera les **np** parents les plus proches de leurs descendants de **P'**. Cette méthode permet donc premièrement de maintenir la diversité et deuxièmement d'améliorer la fitness globale de la population. [6]

Conclusion

la résolution des problèmes de satisfaction de contraintes (CSP) joue un rôle central dans de nombreux domaines, allant de l'intelligence artificielle à l'optimisation de

systèmes complexes. Les approches classiques telles que la propagation des contraintes, les techniques de recherche et les algorithmes probabilistes ont permis d'apporter des solutions satisfaisantes à ces problèmes. Cependant, les algorithmes génétiques se distinguent par leur capacité à offrir des solutions encore plus performantes en exploitant les principes de l'évolution naturelle.

Ce chapitre a mis en lumière l'efficacité des algorithmes génétiques pour résoudre des problèmes tels que la planification d'emplois du temps, en utilisant des opérations comme la sélection, le croisement, la mutation et le remplacement. Ces mécanismes permettent d'explorer des espaces de solutions vastes et complexes de manière plus efficace.

Les algorithmes génétiques, bien qu'inspirés par la nature, sont adaptés aux défis techniques de la programmation moderne et peuvent être intégrés à des solutions informatiques robustes et évolutives.

CHAPITRE 3

Le problème d'emploi du temps (TimeTabling Problem)

Introduction

Le problème d'emploi du temps est un problème historique. Généralement, il est connu comme un problème d'allocation de ressources avec des caractéristiques de permutation et de combinaison. La seule caractéristique du problème de l'emploi du temps (**TTP**) est qu'il a des contraintes de nature hard et des contraintes de nature soft en même temps. Les contraintes hards sont des contraintes fonctionnelles, c'est-à-dire que chaque solution possible doit les satisfaire. Les contraintes soft sont des contraintes de préférence. On remarque qu'il est impossible de satisfaire toutes les contraintes, l'objectif est de satisfaire toutes les contraintes hard et de satisfaire le maximum de contraintes soft.

Ce chapitre est structuré comme suit : dans la première section, le problème de timetabling (TTP) et ses caractéristiques sont exposés. La deuxième section aborde l'application des méthodes de recherche opérationnelle pour résoudre le TTP. La troisième section traite de l'interaction homme-machine dans le cadre de la résolution du TTP. Enfin, la quatrième section explore l'utilisation de l'intelligence artificielle pour aborder le TTP.

3.1 Le problème d'emploi du temps (TTP)

Le problème d'emploi du temps est un problème simple à comprendre, mais -malheureusement- il est très difficile à résoudre. Tous les problèmes reliés à construire un emploi du temps appartient à la catégorie du problème NP-Complete.

Wren [7] a défini le TTP comme suit: "***Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to***

satisfy as nearly as possible a set of desirable objectives" [7]. Dans d'autre définition, TTP est l'affectation des créneaux de temps à un ensemble de sujets sous un ensemble de contraintes. Ces contraintes peuvent être hards qui doivent être toutes satisfaites, ou être softs qui peuvent être utilisées pour juger la qualité de la solution. En général, il est considéré comme un problème d'allocation de ressources dans la recherche Opérationnelle, où les ressources sont les étudiants, les classes et les cours ont pour but d'être allouer à des créneaux de temps d'un emploi du temps d'une semaine pour accomplir un sujet avec une fonction d'objective pour repositionner les ressources parmi les contraintes [7].

Un grand nombre de variété des TTPs ont été proposés, qui se diffèrent à cause de leurs type d'institution qui les implique et des différents types de contraintes. Un cas spécial des TTPs est l'emploi du temps des écoles ou des universités.

Des méthodes variées sont appliquées pour résoudre le TTP comme l'algorithme de graphes coloré , les techniques de programmation linéaire, le recuit simulé, la recherche taboue et les algorithmes génétique. Ces derniers semble les plus appropriés pour résoudre le problème d'emploi du temps éducationnel. Les sections suivantes discutent les différentes techniques employées pour résoudre le TTP [7].

3.2 La recherche opérationnelle

Historiquement, Graph Colouring Problem (GCP) est l'approche classique utilisé pour modéliser et résoudre le TTP. Autre recherche qui utilise les techniques de la programmation linéaire pour représenter le TTP on été appliquées [7].

GCP est un problème de recherche très connu. Plusieurs heuristiques sont drivées du Graph Colouring. GCP consiste à colorer des sommets d'un graphe G(X, U) où :

X= {x1, x2, …, xn} est un ensemble de sommets et

U= {u1, u2, …, un} est un ensemble d'arcs reliant ces sommets pour avoir le graphe coloré **C : V ➔ N** de telle sorte que les sommets auront des différentes couleurs [7].

Le TTP dans son simple forme, peut être vu comme un GCP, où chaque nœud représente une tache, chaque couleur représente un créneau de temps, et chaque arc **(xi, xj)** indique que xi et xj doivent être positionné dans le même créneau de temps [7].

Plusieurs recherches sont reliées à l'approche GCP pour avoir une solution faisable. De Werra [7], a conclu que la formulation de la programmation mathématique

est plus utilisé dans la résolution des cas généralement extensibles des problèmes NP-Complete. L'approche GCP peut résoudre plus précisément le TTP caractérisé par un modèle mathématique programmable. La méthode GCP est plus adaptative pour des problèmes de petite échelle mais il ne donne aucune solution pour les problèmes de grande échelle. Normalement, le TTP est un problème de grande échelle donc un TTP résolu par les GCP est plus loin d'une solution efficace [7].

3.3 L'interaction Homme Machine (HMI)

Cette approche propose une itération avec la machine qui propose une solution initiale, et l'utilisateur qui soit accepte cette solution ou soit l'améliore manuellement ou demande à la machine de refournir une autre solution. Le processus itératif se déroule jusqu'à ce que l'utilisateur soit satisfait de la solution ou l'implémentation soit impossible. Mulvey (1982) [7] propose le modèle d'interaction homme machine suivant : approximation, évaluation et modification. L'utilisateur demande un modèle d'approximation du problème mal définie. Après ça, ce model doit être évalué par un estimateur (***estimator***) pour évaluer qu'elle est une bonne solution de départ et comment elle est flexible pour aider l'utilisateur à améliorer la solution. L'utilisateur peut essayer à trouver la solution en modifiant par main la solution proposée. S'il sentit que la solution déterminée par le modèle d'approximation n'est pas un bon ordonnancement, il peut répéter cette alternative manuelle jusqu'à ce qu'une solution acceptable est obtenue. Finalement, l'utilisateur ou la machine parcourent le processus de modification comme il citer ci-dessus [7].

Voici un pseudo algorithme du processus d'interaction homme machine :

1. Générer les données nécessaire pour le modèle d'approximation, la machine demande à l'utilisateur d'introduire les données, comme les ressources. Ces données peuvent toutes être modifiées. Après ça, la machine va générer les données pour compléter les exigences du client.
2. (Approximation) après ça, la machine va générer des réseaux de mode et de solution.
3. (Evaluation) Déterminer si la solution candidate est faisable ou non, si non aller à l'étape 1.
4. (Evaluation) Déterminer si la solution candidate est acceptable ou non, si non aller à l'étape 6.
5. Imprimer la solution.

6. Décider de continuer à essayer d'améliorer la solution candidate ou non, si non aller à l'étape 8.
7. (Modification) Faire des modifications manuelles sur la solution candidate et aller à l'étape 3.
8. (Modification) Faire des modifications sur le réseau de formulation et aller à l'étape 3

Le grand inconvénient de cette approche est qu'elle n'est conseillée pour les grands systèmes, le processus sera laborieux, et avoir un modèle d'approximation sera plus chair [7].

3.4 L'Intelligence Artificielle (AI)

Il y a des méthodes méta-heuristiques variantes telle que le recuit simulé (Simulated Annealing), la recherche taboue (Tabu Search), et les algorithmes génétiques (Genetic Algorithms) qui sont utilisées pour résoudre des problème d'emploi du temps.

3.4.1 Le recuit simulé (Simulated Annealing)

Abramson (1991) [7] a utilisé la technique de recuit simulé pour résoudre le TTP. Le recuit simulé imite le processus du ***refroidissement*** d'une collection d'atomes de vibration chauds. Quand les atomes ont des températures hautes, ils sont libres pour se déplacer de manière aléatoire. Mais comme la masse refroidit les inter-particules forcent les atomes ensembles. Quand la masse refroidit, aucun déplacement est possible et la configuration est gelée. Si la masse a refroidit rapidement, la chance d'obtenir une solution moine coûteuse est basse que si elle a refroidit lentement. A n'importe quelle température donnée, une configuration des atomes est acceptable si l'énergie du système est basse. Par contre, si l'énergie du système est haute, la configuration ne sera acceptable seulement si la probabilité d'une augmentation est petit que celle espérée dans la température donnée. L'application du recuit simulé pour le TTP est relativement simple. Les atomes sont remplacés par des éléments. L'énergie du système est remplacé par le coût de l'emploi du temps. Une allocation initial est faite dans laquelle des éléments sont placés dans des périodes choisit aléatoirement. Le changement dans le coût est calculé à partir de deux composants [7]:

1. le coût d'enlever un élément d'une période.
2. le coût d'insérer un élément dans une période.

Le coût d'enlever un élément est le coût d'une classe, le coût d'un enseignant et le coût d'un groupe. Par conséquent, le coût d'insérer un élément est le coût d'une classe, le coût d'un enseignant et le coût d'un groupe. Si après l'enlèvement d'un élément à partit d'une période le nombre d'occurrence de cette classe est plus grand que zéro (0), alors le coût de classe qui sera sauvegardé sera un (1). Similairement, si le nombre d'occurrence d'un enseignant est supérieur ou égale à un (1) après la suppression de cet enseignant alors le coût qui sera sauvegardé sera un (1). Cette technique est aussi applique pour les groupes. Le coût d'insertion d'une classe est calculé de la même manière. Dans ce cas, il est possible de déterminer le changement dans le coût de manière incrémentale sans recalculé le coût total de l'emploi du temps. Les attributs sont utilisés particulièrement quand une version parallèle de l'algorithme est implémentée.

Comme avec les algorithmes génétiques, l'avantage majeur du recuit simulé est la flexibilité et la robustesse comme une méthode globale de recherche. Il peut être d'accord avec les problèmes non linéaires. Il est aussi préféré pour l'implémentation parallèle. Un inconvénient du recuit simulé est qui il pour le calcul intensif.

3.4.2 Les algorithmes génétiques

Les algorithmes génétiques appartiennent à des classes d'algorithmes utilisés pour trouver des solutions approximatives pour les problèmes difficiles en appliquant les principes de la biologie évolutionnaire comme l'héritage, le croisement et la sélection naturelle. L'algorithme génétique imite le processus de la sélection naturelle et peut être utilisé comme une technique pour résoudre des problèmes d'optimisation complexes, qui ont des espaces de recherche très larges. Au contraire de plusieurs schémas heuristiques, qui ont une solution sous optimal à la fois, l'algorithme génétique maintien plusieurs solution individuelles sous la forme d'une population. Les individus (les parents) sont choisis à partir d'une population et ils se reproduisent pour avoir de nouveaux individus (les fils). Les fils se reproduisent à son tour pour introduire la diversité à la population [7].

Les descriptions suivantes sont tenus du BURKE [7]. Un algorithme génétique commence par générer aléatoirement un ensemble (population) d'emploi du temps. Ensuite, ils sont évalués selon des critères. A la base de cette évaluation, les membres de la population sont choisis comme des parents pour les générations suivantes. Par la reproduction de la population, le processus de sélection faveur les bons emplois du

temps et élimine les mauvais emplois du temps et la recherche sera orientée dans les bons espaces de recherche [7].

Selon Abramson et Abela [7], les algorithmes génétiques nécessitent une mesure de fitness des individus pour déterminer quels sont les individus qui survivrent. L'individu qui a une valeur de fitness très grand a la chance de survivre [7].

3.5 Le problème d'emploi du temps scolaire comme un problème de satisfaction de contraintes

L'emploi du temps scolaire nécessite l'ordonnancement des classes, des éléves et des enseignants à un nombre fixe des créneaux de temps, tout en respectant certaines contraintes. Les contraintes dans ce problème peuvent être classées comme ***contraintes hard*** et des ***contraintes soft***. Les contraintes hard doivent être satisfaites par la solution du problème par contre les contraintes soft sont désirées d'être satisfaites.

3.5.1 Les contraintes hard

Les contraintes hard sont celles qui doivent être satisfaites par la solution donnée pour qu'elle sera acceptée. Un grand nombre de contraintes hard sont utilisées dans la recherche, dans ce qui suit nous allons présenter les contraintes hard qui sont commune pour la plupart des problèmes d'emploi du temps scolaires [6] :

1. Un éléve doit seulement avoir un cours par séance.
2. Un enseignant doit avoir seulement un cours par séance.
3. Les matières qui nécessitent deux séances consécutives, comme Sciences, Arabe, Physique et Mathématiques.

3.5.2 Les contraintes soft

Les contraintes soft sont, en général, de préférences et ne représentent aucun aspect physique. Elles doivent être satisfaites au maximum lorsqu'il est possible. Une solution faisable est acceptable même si elle ne satisfaite pas les contraintes soft, mais il n'est pas bon pour une bonne solution. La qualité d'une solution est proportionnellement inverse au nombre des contraintes soft violées. L'imposition au dessus des contraintes soft va produire un grand emploi du temps préféré, il va augmenter la complexité computationnelle du problème. Le nombre de telle contraintes

doit être plus petit le plus possible. Les trois types suivants des contraintes soft, qui sont imposées dans plusieurs écoles, sont considérés dans notre travail :

1. Les heures creuses dans l'emploi du temps des élèves.
2. Les heures creuses des enseignants.

3.6 Fonction d'évaluation

Puisque le problème d'emploi du temps est identifié comme un problème d'optimisation, on a besoin de fonction d'évaluation. Une fonction d'évaluation dans ce problème est simplement une mesure de la qualité de la solution.

Conclusion

Nous avons introduit le problème d'emploi du temps qui semble être facile a comprendre mais très difficile à résoudre. On a vu dans ce chapitre qu'il y a trois types d'emploi du temps : l'emploi du temps des universités, l'emploi du temps des écoles et l'emploi du temps des examens. On a détaillé le problème d'emploi du temps scolaire. Dans le chapitre qui suit, on va s'intéresser à la techniques basé sur les algorithmes génétiques pour résoudre le problème d'emploi du temps d'une établissement scolaire. Dans notre travail on va prendre l'emploi du temps de CEM El Amir Abd El Kader.

CHAPITRE 4

Approche évolutionnaire pour le problème d'emploi du temps, Cas d'étude CEM El Amir Abd El Kader

Introduction

Nous avons vu dans le chapitre précèdent que le problème d'emploi du temps peut être formalisé comme un CSPs et que parmi les techniques utilisées pour le résoudre on trouve les algorithmes génétiques (GAs). Les algorithmes génétiques sont des techniques heuristiques inspirés de la nature. Pour faire manipuler le problème d'emploi du temps par les GAs, on doit d'abord représenter le problème. La représentation va nous permettre d'avoir une structure qu'on peut manipuler par les opérations génétiques (croisement, mutation et sélection).

Dans ce chapitre, nous présenterons le processus génétique appliqué à la planification des emplois du temps scolaires. Nous détaillerons les principales composantes du système, y compris la représentation des chromosomes, la création des enseignants et des matières, ainsi que la génération et l'évaluation des solutions. Chaque étape du processus est conçue pour s'assurer que les emplois du temps respectent les contraintes pédagogiques tout en maximisant la satisfaction des critères de qualité définis.

4.1 le processus Génétique

Le schéma suivant résume le processus génétique qui sera détaillé plutôt :

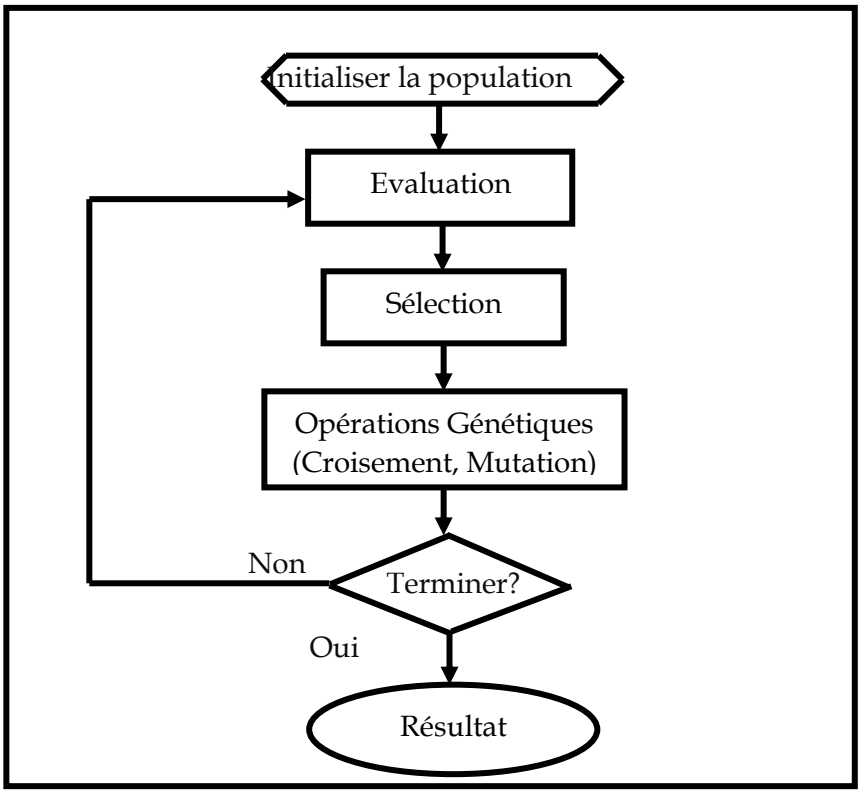


Figure 4.1: le processus évolutionnaire

Représentation de Chromosome

Un chromosome, aussi connu comme un individu dans la littérature évolutionnaire, est le composant de base d'un Algorithme Evolutionnaire (EA). Il est généralement utilisé pour représenter une solution dans l'espace de recherche d'un problème d'optimisation. Un chromosome est composé de gènes, dont chacun peut décrire un paramètre du problème. Un ensemble de chromosomes forme une population pour un EA, où l'évolution de la population consiste à appliquer les opérations évolutionnaires.

Dans le contexte de la planification, chaque emploi du temps est représenté comme un "chromosome". Ce chromosome est une structure de données qui contient toutes les informations nécessaires pour décrire la répartition des cours dans le temps. Dans notre modèle, un chromosome est représenté par une matrice à quatre dimensions qui simule l'emploi du temps de l'école pour une semaine. Chaque dimension de cette matrice représente une caractéristique spécifique de l'emploi du temps :

- La première dimension représente les jours de la semaine (Dimanche à Jeudi).
- La deuxième dimension représente les créneaux horaires disponibles pour chaque jour.
- La troisième dimension représente les différents niveaux scolaires.
- La quatrième dimension représente les groupes au sein de chaque niveau.

Chaque cellule de cette matrice contient une chaîne de caractères qui combine le nom de la matière et le nom de l'enseignant, ce qui nous permet de déterminer quelle matière est enseignée, par quel enseignant, à quel moment, dans quel niveau et pour quel groupe d'étudiants.

Supposonsque nous avons une matrice:

`emploi_du_temps[jour][créneau][niveau][groupe]`

- `jour` : Représente le jour.
- `créneau` : Représente le créneau horaire (la période ou l'heure).
- `niveau` : Représente le niveau d'enseignement (la classe).
- `groupe` : Représente le groupe d'étudiants.

Exemple d'une cellule :

emploi_du_temps[2][2][3][0] = "Arabe(Mr.Sahli)"

Ici :

- `2` : Le jour (mardi, par exemple).
- `2` : Le créneau horaire (la troisième période).
- `3` : Le quatrième niveau.
- `0` : Le groupe d'étudiants (groupe 1).

Cette cellule indique que la matière "Arabe" est enseignée par "Mr.Sahli" en quatrième année, groupe 1, pendant la troisième période du mardi.

Les tableaux conventionnels sont habituellement limités à trois dimensions (lignes, colonnes et pages). Par conséquent, manipuler un tableau à quatre dimensions demande une visualisation plus élaborée, telle que la combinaison de plusieurs tableaux ou l'utilisation de structures avancées comme les matrices multidimensionnelles en programmation.

Pour simplifier cela, nous pouvons imaginer un tableau bidimensionnel (lignes et colonnes) où chaque cellule contient un autre tableau (c'est-à-dire un tableau imbriqué). Ce tableau imbriqué représente la troisième et la quatrième dimension.

Nous représentons le tableau externe des niveaux et des groups, et dans chaque cellule, il y aura un tableau secondaire qui représente les jours et les créneaux horaires.

Chaque cellule du tableau secondaire contient une chaîne de caractères qui combine le nom de la matière et le nom de l'enseignant.

<table>
<tr><th></th><th colspan="4">Niveau 1</th><th>Niveau 2</th><th>…</th><th>…</th></tr>
<tr><td rowspan="5">Groupe 1</td><td colspan="4" rowspan="5"><table>
<tr><th></th><th>Période 1</th><th>Période 2</th><th>Période 3</th><th>.....</th></tr>
<tr><td>1 er jour</td><td>Free</td><td>francais(Mr. Harchouch)</td><td>History(Ms. Lehchili)</td><td>.....</td></tr>
<tr><td>2 éme jour</td><td>Free</td><td>Science(Ms. Zegrour)</td><td>Math(Ms. Belhain)</td><td>.....</td></tr>
<tr><td>3 éme jour</td><td>arabe(Ms. sahli)</td><td>sport(Mr. Kheloufi)</td><td>sport(Mr. Kheloufi)</td><td>.....</td></tr>
<tr><td>....</td><td>.....</td><td>.....</td><td>.....</td><td>.....</td></tr>
</table></td><td rowspan="5">…</td><td rowspan="5">…</td><td rowspan="5">…</td></tr>
<tr></tr><tr></tr><tr></tr><tr></tr>
<tr><td>Groupe 2</td><td colspan="4">…</td></tr>
<tr><td>…</td><td colspan="4">…</td><td>…</td><td>…</td><td>…</td></tr>
</table>

Figure 4.2: Exemple d'emploi du temps.

De cette manière, le chromosome est représenté comme une solution complète qui peut être améliorée au fil du temps pour obtenir le meilleur emploi du temps possible.

Représentation des gênes

Les gènes dans notre modèle représentent les plages horaires attribuées à chaque matière pour chaque niveau et groupe. Chaque gène est une chaîne de caractères qui

combine le nom de la matière avec le nom de l'enseignant, formant ainsi un emploi du temps complet pour chaque jour et chaque créneau horaire.

Figure 4.3:La représentation des gênes

Initialisation de la population

L'initialisation de la population consiste à générer un ensemble initial de chromosomes (emplois du temps) qui serviront de point de départ pour l'algorithme d'optimisation. Chaque chromosome est généré de manière aléatoire tout en respectant les contraintes de base telles que les matières à enseigner et les créneaux horaires disponibles.

Pour la génération des nombres aléatoires et le processus de l'initialisation passe par les étapes suivantes :

1. Créer une liste vide population
2. Créer un nouvel emploi du temps *schedule* avec dimensions (days, slotsPerDay, levels, groupsPerLevel).
3. Remplir le tableau de *schedule* avec des valeurs aléatoires

- Générer un nombre aléatoire pour le jour.
- Générer un nombre aléatoire pour le créneau horaire.
- Générer un nombre aléatoire pour le niveau.
- Générer un nombre aléatoire pour le groupe.

Si la case choisie est valide selon **les contraintes** alors affecter une matière au créneau choisi sinon choisir une autre case jusqu'à trouver une case valide.

4. Le processus est répété jusqu'à l'affectation de tous les emplois du temps.
5. Défini Un nombre maximal de tentatives pour éviter les boucles infinies et les échecs prolongés lors de la génération des emplois du temps.

Les contraintes respectées pour générer une population initiale :

- Les créneaux horaires sont vérifiés pour s'assurer que l'enseignant n'est pas déjà assigné à une autre tâche pendant cette période
- Les créneaux horaires sont vérifiés pour respecter les contraintes spécifiques à certains jours (par exemple, les restrictions pour le mardi où les créneaux après 4 ne sont pas considérés

- Les matières qui nécessitent deux séances consécutives, comme Sciences, Physique, Arabe et Mathématiques : vérifie la disponibilité de créneaux consécutifs et assure que l'enseignant n'est pas déjà assigné à d'autres créneaux consécutifs pour la même matière.
- les créneaux horaires sont disponibles avant d'affecter une matière à un créneau.
- Les heures de cours requises pour chaque matière sont respectées et distribuées en fonction de la disponibilité des créneaux horaires et des enseignants.

Algorithme 1 : Générer population initial

```
Début
Pour i de 1 à taille_population :
 Schedule ← créertableau(dimensions : days, slotsperday, levels, groupsperlevel)
          Pour chaque matière dans la liste_des_matières :
Tentative ← 0
  Affectation_réussie ← FAUX
Tant que tentative < max_tentatives ET affectation_réussie = FAUX :
 Jour ← générernombrealéatoire(1, nombre_jours)
  Créneau ← générernombrealéatoire(1, slotsperday)
  Niveau ← générernombrealéatoire(1, nombre_niveaux)
   Groupe ← générernombrealéatoire(1, groupsperlevel)
  // Vérifier les contraintes
  Si casevalide(jour, créneau, niveau, groupe, matière, enseignant) ALORS
Schedule[jour][créneau][niveau][groupe] ←matière+''(''+ enseignant'')''
Affectation_réussie ← VRAI
                SINON :
      Tentative ← tentative + 1
SI affectation_réussie = FAUX  ALORS :
     Retourner erreur "Impossible d'affecter cette matière"
                SI enseignantaffecté(enseignant, jour, créneau) ALORS :
Continuer à chercher un autre créneau
                SI jour = Mardi ET créneau > 12hALORS :
  Continuer à chercher un autre créneau
                SI matière nécessite des créneaux consécutifsALORS :
Vérifier disponibilité créneau+1
                Pour chaque matière dans la liste :
Répéter le processus d'affectation jusqu'à ce que toutes les matières soient
assignées.
Fin Pour chaque matière
                Population ← population + [schedule]
Fin Pour i
               Retourner population
Fin
```

La fonction de fitness

La fonction de fitness évalue la qualité de chaque emploi du temps en fonction de divers critères. Ces critères incluent le respect des horaires de cours, la répartition équilibrée des heures, et la disponibilité des enseignants.

Les contraintes respectées par la fonction fitness

- Minimiser le nombre des heures creuses dans l'emploi du temps des élèves, réduction de 100 points pour chaque heurs creuse.
- De préférence, les matières fondamentaux telles que "Math", "PE", "arabe", et "Science" cours seront à la matinée, ajout de 50 points.
- Si la matière "sport" est programmée (10h à 12h du matin) ou le soir, ajout de 50 points.
- Si une même matière est programmée plus de deux fois dans une journée, réduction de 50 points.
- Minimiser le nombre des heures creuses des enseignants :
  - réduction de 50 points pour chaque heure creuse.
  - si un enseignant a moins de deux périodes de cours dans la matinée réduction de 20 points.
  - si un enseignant a moins de deux périodes de cours après-midi réduction de 20 points.
- Journées Complètes de Repos pour les Enseignants : si un enseignant a une journée complète sans cours, ajout de 100 points.
- Minimiser les périodes libres en fin d'après-midi : Si un jour autre que le mardi a le créneau horaire 5 vide (15h-16h), ajout de 50 points.

Le **score de fitness** est une somme pondérée de plusieurs facteurs, chacun représentant une contrainte avec des pénalités ou des récompenses.

Algorithme 2 : La fonction de fitness

```
Début
fitness_score ← 0
 Pour chaque emploi_du_temps dans population :
  Pour chaque groupe dans emploi_du_temps :
 // Vérification des heures creuses
    heures_creuses_élèves ← calculerHeuresCreuses(emploi_du_temps, élève)
 fitness_score ← fitness_score - (heures_creuses_élèves * 100)
       // Vérification des matières fondamentales en matinée
```

```
9:   SI matière("Math", "PE", "Arabe", "Science") est programmée en matinée ALORS :
10:          fitness_score ← fitness_score + 50
11:       // Vérification de la matière "Sport"
12:  SI matière("Sport") est programmée entre 10h et 12h OU en soirée ALORS :
13:          fitness_score ← fitness_score + 50
14:       // Vérification des répétitions d'une même matière
15: SI même matière est programmée plus de deux fois dans une journée ALORS :
16:          fitness_score ← fitness_score –50
17: Pour chaque groue dans emploi_du_temps :
18:    // Minimiser les heures creuses des enseignants
19:    heures_creuses_enseignants←calculerHeuresCreuses(emploi_du_temps, enseignant)
20:    fitness_score ← fitness_score - (heures_creuses_enseignants * 50)
21:    // Vérifier si l'enseignant a moins de deux périodes de cours le matin
22: SI nombre_de_cours(enseignant, matinée) < 2 ALORS :
23:       fitness_score ← fitness_score - 20
24:    // Vérifier si l'enseignant a moins de deux périodes de cours l'après-midi
25: SI nombre_de_cours(enseignant, après_midi) < 2 ALORS :
26:       fitness_score ← fitness_score -20
27:    // Journées complètes de repos pour les enseignants
28: SI enseignant a une journée complète de repos ALORS :
29:       fitness_score ← fitness_score + 100
30: Pour chaque jour dans emploi_du_temps :
31:    SI jour ≠ Mardi ET créneau (15h-16h) est vide ALORS :
32:       fitness_score ← fitness_score + 100
33: Fin Pour chaque jour
34: Fin Pour chaque emploi_du_temps
35: Retourner fitness_score
36: Fin
```

L'opération de la Sélection

L'opération de sélection consiste à choisir les meilleurs chromosomes de la population pour la génération suivante. Les solutions les mieux notées sont conservées pour produire une nouvelle génération d'emplois du temps.

L'opération de croisement

L'opération de croisement est une étape cruciale dans les algorithmes génétiques, utilisée pour combiner les caractéristiques de deux individus parents afin de produire de nouvelles solutions enfants. Cette opération permet de mélanger et de transférer des

informations génétiques d'une génération à l'autre, favorisant ainsi la diversité et l'exploration de l'espace des solutions.

La procédure de croisement, pour produire le premier résultat, est exposée ci-dessous:

- Deux parents sont sélectionnés aléatoirement à partir de la population actuelle.
- Deux enfants sont créés en échangeant les jours d'emploi du temps entre les deux parents :
  - Pour les trois premiers jours de la semaine, les créneaux horaires de l'enfant 1 proviennent du parent 1, et ceux de l'enfant 2 proviennent du parent 2.
  - Pour les jours restants, les créneaux horaires de l'enfant 1 proviennent du parent 2, et ceux de l'enfant 2 proviennent du parent 1
- Après le croisement, il peut y avoir des incohérences dans les emplois du temps résultants. Pour résoudre ces problèmes, il doit ajuster les heures manquantes ou en surplusdans les emplois du temps enfants pour s'assurer que chaque matière est programmée exactement le nombre d'heures requis.
  - Suppression des heures supplémentaires: si une matière est programmée pour plus d'heures que nécessaire, les heures excédentaires sont identifiées et retirées.
  - Ajout des heures manquantes: si une matière a moins d'heures que nécessaire, des créneaux horaires sont ajoutés si cela ne viole aucune des contraintes hard jusqu'à ce que le nombre d'heures requis soit atteint.
- Création d'un troisième enfant est généré en mélangeant les emplois du temps des deux enfants de manière aléatoire.

Algorithme 3 : Le croisement
1: Début
2: parent1 ← sélectionnerAléatoirement(population)
3: parent2 ← sélectionnerAléatoirement(population)
4: enfant1 ← créerTableauVide(dimensions : jours, créneaux, niveaux, groupes)
5: enfant2 ← créerTableauVide(dimensions : jours, créneaux, niveaux, groupes)
6: enfant3 ← créerTableauVide(dimensions : jours, créneaux, niveaux, groupes)
7: **Pour** jour allant de 0 a 3 faire:
8: **Pour**créneau allant de 0 aNBRdeCréneau faire :

9: enfant1[jour][créneau] ← parent1[jour][créneau]
10: enfant2[jour][créneau] ← parent2[jour][créneau]
11: **Pour** jour allant de 3 a7 faire:
12: **Pour**créneau allant de 0 aNBRdeCréneau faire:
13: enfant1[jour][créneau] ← parent2[jour][créneau]
14: enfant2[jour][créneau] ← parent1[jour][créneau]
15: **Pour** chaque matière dans la liste_des_matières
16: **SI** nombre_heures(matière, emploi_du_temps) > heures_requises(matière) **ALORS**:
17: heures_excédentaires ← trouverHeuresExcédentaires(matière, emploi_du_temps)
18: supprimerHeures(heures_excédentaires, emploi_du_temps)
19: **Répéter** pour enfant2.
20: **Pour** chaque matière dans la liste_des_matières:
21: **SI** nombre_heures(matière, emploi_du_temps) < heures_requises(matière) **ALORS** :
22: ajouterHeuresManquantes(matière, emploi_du_temps, créneaux_disponibles)
23: **Répéter** pour enfant2.
24: **Pour** jour allant de 0 a NBRdeJour faire:
25: **Pour**créneau allant de 0 a NBRdeCréneau faire :
26: **SI** random() < 0.5 ALORS :
27: enfant3[jour][créneau] ← enfant1[jour][créneau]
28: SINON :
29: enfant3[jour][créneau] ← enfant2[jour][créneau]
30: Retourner enfant1, enfant2, enfant3
31: Fin

L'opération de Mutation

On va explorer l'espace de recherche. Si on trouve une heure creuse on va la changer ou la permuter avec une autre séance à condition que cette dernière ne produise pas une autre heure creuse et pour le processus on a les étapes suivantes :

Pour faire cette mutation on a procéder comme suit :

- Chercher une heure creuse.
- Prendre une séance aléatoire dans le premier ou dernier créneau du temps.
- Faire la permutation entre les deux créneaux de temps si cela ne viole aucune des contraintes hard.

Algorithme 4 : Mutation

1: Début
2: **Pour** chaque emploi_du_temps dans population :
3: créneau_aléatoire←générerNombreAléatoire(1,nombre_de_créneaux)

```
4:  SI (emploi_du_temps[Level][ Groupe][jour][ créneau_aléatoire] vide )ALORS :
        heure_creuse ← créneau_aléatoire
        Sortirdelaboucle
5:  Pour chaque emploi_du_temps dans population :
6:  //Sélectionner aléatoirement un créneau du premier ou dernier créneau du temps
7:  créneau_aléatoire←générerNombreAléatoire(1,nombre_de_créneaux)
8:  SI (créneau_aléatoire est du premier ou dernier créneau du temps)ALORS :
9:  SI permutation_valide(heure_creuse,emploi_du_temps,jour,créneau_aléatoire) ALORS
    emploi_du_temps[Level][Groupe][[jour][heure_creuse]←
    emploi_du_temps[jour][créneau_aléatoire]
    emploi_du_temps[jour][créneau_aléatoire]) ← vide
    Sortirdelaboucle
10: Retourner emploi_du_temps
11: Fin
```

Validation

Après le croisement et la mutation, chaque enfant est validé pour vérifier s'il respecte les contraintes de l'emploi du temps.

- Nombre d'heures par semaine pour chaque matière: chaque matière ne doit pas dépasser le nombre d'heures autorisées pour chaque niveau. Si une matière a plus d'heures attribuées qu'autorisé, l'emploi du temps est considéré invalide.
- Aucun enseignant ne doit être assigné à plus d'une matière au même moment: un enseignant ne peut pas enseigner deux classes différentes au même créneau horaire. Si un conflit est détecté où un enseignant est assigné à plusieurs cours au même moment, l'emploi du temps est invalide.

Conclusion

On a vu la représentation du chromosome pour notre problème, qui est une matrice dont les lignes représentent les salles et les colonnes présentes les créneaux de temps. Après avoir vu le processus d'initialisation de la population qui utilise la loi uniforme pour la génération des nombres aléatoire pour l'affectation des différentes classes. Ensuite on a vu les différentes opérations génétiques (croisement, mutation et sélection) qu'on peut appliquer pour faire évoluer la population des solution. Dans le chapitre suivant, on va présenter les détails d'implémentation de notre programme d'optimisation du problème d'emploi du temps.

CHAPITRE 5

Réalisation

Introduction

Avant la présentation des détails techniques de réalisation, nous commençons par la présentation du langage et l'outil de programmation utilisé. Dans notre projet, on a choisi de le faire à par la programmation orienté objet. Le langage utilisé est Java.connu pour sa robustesse et sa portabilité. Pour faciliter le développement, nous avons utilisé l'IDE **NetBeans**, qui offre des outils puissants pour la gestion de projet, l'auto-complétion du code, et le débogage, nous permettant d'optimiser le processus de création de notre application.

5.1 Environnement de développement

Pour développer notre application, nous avons utilisé le langage de programmation Java car c'est un langage orienté objet parmi les langages les plus utilisés dans le domaine del'intelligence artificielle et plus adapté pour l'implémentation des algorithmes génétiques.

5.1.1 Le langage java

Java est un langage de programmation orienté objet inspiré du langage C++, qui a été créépar Sun Microsystems en 1995 [9].Il permet de créer des programmes qui peuvent être exécutésdans un navigateur web, développer des applications côté serveur, combiner des applications oudes services basés sur le langage Java et écrire des applications efficaces pour les téléphonesportables. Java est disponible dans tous les

systèmes d'exploitation tels que Windows, Mac,Linux et dans les téléphones portables sous Android. Cette multiplicité de support estavantageuse, car cela permet aux développeurs de créer un programme et de le faire fonctionnersur plusieurs plateformes sans devoir recréer un nouveau programme [10].

5.1.2 NetBeans

NetBeans IDE est un environnement de développement intégré (IDE) gratuit et open source qui permet de développer des applications de bureau, mobiles et Web. L'IDE prend en charge le développement d'applications dans divers langages, notamment Java, HTML5, PHP et C++.

L'IDE fournit une prise en charge intégrée pendant tout le cycle de développement, de la création du projet au débogage, au profilage et au déploiement. L'IDE fonctionne sous Windows, Linux, Mac OS X et d'autres systèmes UNIX [11].

5.2 Représentation de Chromosome

Pour la représentation de chromosome, on a préféré de le codé se forme d'une matrice quatre dimensions où les indices représentent les jours, les créneaux horaires, les niveaux et les groupes. où les cases sont des chaînes de caractères.

/src/Schedule.java

```
16 public class Schedule {
17     String[][][][] timetable;
18     public   Schedule(intdays,   intslotsPerDay,   intnumTeachers,   intlevels,
19 intgroupsPerLevel) {
20         this.timetable=newString[days][slotsPerDay][levels][groupsPerLevel];
21     }
22
23
```

5.3 Classe Principale de Gestion d'emploi du temps

la classe principale orchestre l'exécution d'un algorithme génétique pour résoudre un problème de planification.

/src/ScheduleGenerator.java

```
28 public static void main(String[] args) {
29 List<Teacher>teachers = createTeachers();
30 List<Subject>subjects = createSubjects();
31 List<Solution>population=
32 generateInitialPopulation(teachers,subjects,POPULATION_SIZE);
33 List<Solution>archive = new ArrayList<>();
34 for (intgeneration = 0; generation<MAX_GENERATIONS; generation++) {
35 // Combiner population et archives
36 List<Solution>combined = new ArrayList<>(population);
37 combined.addAll(archive);
38 //  Évaluer Fitness de la population combinée
39 evaluateFitness(combined, teachers, subjects);
40 // Selection
41 population = selection(combined, POPULATION_SIZE);
42 //Croisement et mutation
43 population = crossoverAndMutation(population, teachers, subjects);
44 archive = updateArchive(combined, ARCHIVE_SIZE);
45         }
46       }
```

Ce code implémente un algorithme génétique pour optimiser un planning de cours. Voici les principales étapes :

5.3.1 Créationdes données de base

- **Création la Liste des Enseignants**

La méthode ***createTeachers()*** génère une liste d'enseignants avec leurs matières et les groupes auxquels ils sont assignés, facilitant ainsi la gestion des emplois du temps scolaires.

/src/ScheduleGenerator.java

```
71 static List<Teacher>createTeachers() {
72 List<Teacher>teachers = new ArrayList<>();
73 teachers.add(new  Teacher("Mr.  Belhain","Math",Arrays.asList("level  1  group
74 1","level 1 group 2", "level 3 group 3")));
```

```
75 teachers.add(new Teacher("Mr. Ben kerbi","Math",Arrays.asList("level 2 group
76 1","level 2 group 2", "level 4 group 4")));
77 teachers.add(new Teacher("Mr. Beldi","Math",Arrays.asList("level 1 group
78 3","level 1 group 4", "level 2 group 4")));
79 …
   }
```

- **Création et Initialisation des Matières Scolaires**

Le méthode ***createSubjects()*** crée et initialise une liste de matières scolaires avec des détails tels que le nom, le nombre d'heures hebdomadaires pour chaque niveau, et si nécessité d'heures consécutives . Ces informations sont utilisées pour générer des emplois du temps qui respectent les exigences pédagogiques.

/src/ScheduleGenerator.java

```
107 static List<Subject>createSubjects() {
108         List<Subject>subjects = new ArrayList<>();
109         subjects.add(new Subject("sport", new int[]{2, 2, 2, 2}, true));
110    subjects.add(new Subject("PE", new int[]{3, 3, 3, 3}, true));
111    …
112    }
113
```

5.3.2 Générer une population initiale

La méthode ***generateInitialPopulation*** crée une population initiale d'emplois du temps, en assignant aléatoirement des enseignants à des matières pour différents niveaux et groupes. Elle respecte plusieurs contraintes, comme le nombre d'heures par semaine pour chaque matière, la nécessité d'heures consécutives pour certaines matières, et l'absence de conflits d'horaires pour les enseignants. Si une solution ne respecte pas toutes les contraintes après un certain nombre de tentatives, une nouvelle tentative est faite jusqu'à ce qu'un emploi du temps valide soit généré.

/src/ScheduleGenerator.java

```
119 static List<Solution>generateInitialPopulation(List<Teacher>teachers,
120 List<Subject>subjects, int populationSize) {
121    …
122  Random rand = new Random();
123  for (intp = 0; p<populationSize; p++) {
```

```
Schedule schedule = new Schedule(DAYS, SLOTS_PER_DAY, teachers.size(),
LEVELS, GROUPS_PER_LEVEL);
   …
        for (intlevel = 0; level<LEVELS; level++) {
            for (intgroup = 0; group<GROUPS_PER_LEVEL; group++) {
                for (Subjectsubject : subjects) {
                    inthoursRemaining = subject.hoursPerWeek[level];
   …
                    while (hoursRemaining> 0 &&attempts <MAX_ATTEMPTS) {
intday = rand.nextInt(DAYS);
intslot = rand.nextInt(SLOTS_PER_DAY);
   …

if(subject.needsConsecutiveHours&&!schedule.hasConsecutivHours (subject.name,
teacher.name,level,group) ) {
..
 hoursRemaining -= 2;
          }
          }
              } else {
..
      hoursRemaining--;
        }}
                    }
    return population;}
```

5.3.3 La fonction fitness

La méthode ***evaluateFitness*** prend en compte plusieurs contraintes pour évaluer la qualité des emplois du temps générés.

/src/ScheduleGenerator.java

```
408 static List<Integer>evaluateFitness(List<Solution> solutions,
409   …
410 for (intlevel = 0; level<LEVELS; level++) {
411 for (intgroup = 0; group<GROUPS_PER_LEVEL; group++) {
412 for (intday = 0; day<DAYS; day++) {
413 //les heurs creuses
414   if(schedule.timetable[day][1][level][group]==null||
415 (schedule.timetable[day][2][level][group]==null)||
416 (schedule.timetable[day][5][level][group]==null)  )  {
417 solution.fitness -= 100; }
418    // les matieres fondamanteaux
419         for (intslot = 0; slot< 4; slot++) {
420  if(subject.equals("Math")||subject.equals("PE")
421 ||subject.equals("arabe") || subject.equals("Science")) {
422 solution.fitness += 50; }
423 }  //la matière "sport" est programmée après 10h de matin
424 for (intslot = 2; slot<SLOTS_PER_DAY; slot++) {
425  if (subject.equals("sport")) {
426  solution.fitness += 50; }
427 }//matière est programmée plus de deux fois dans une journée
428 int count = 0;
429 if(schedule.timetable[day][slot][level][group]!=null
430 &&schedule.timetable[day][slo][level][group]!=null&&
431 schedule.timetable[day][slot][level][group].equals
432 (schedule.timetable[day][slo][level][group])  ) {
433 count ++; }
434 if (count >2){
435    solution.fitness -= 100; }}
436 // les heurs creuses pour les enseignants
437 ...
438 if (entry != null && entry.contains(teacher.name) ) {
```

```
439 tab[slot]=1; }
440 else {tab[slot]=-1;
441 }
442                         }
443  for (int j = 1; j < 3; j++){
444    if ((tab[j-1]== 1)&& (tab[j+1]== 1) && (tab[j]== -1) ) {
445                               solution.fitness -= 150;}
446             if ((tab[4]== 1)&& (tab[6]== 1) && (tab[5]== -1) ) {
447 solution.fitness -= 150;}
448 // un enseignant a moins de deux périodes de cours  dans la matinée
449 ...
450                               if ((tab[j]== 1) ) {
451                                k+=1;}
452                            if (k<2)
453 {
454                               solution.fitness -= 180; }
455 //  un enseignant a moins de deux périodes de cours après-midi
456 ...
457       if ((tab[j]== 1) ) {
458   n+=1; }
459 if (n<2)
460 {solution.fitness -= 180;
461                                  }
462  //   Journées Complètes de Repos pour les Enseignants
463 for (Teacher teacher : teachers) {
464 ...
465 if(schedule.timetable[day][slot][level][group]!=null&&
466 schedule.timetable[day][slot][level][group].contains(entry)) {
467      hasClass = true;
468  break; }
469 ...
470      if (!hasClass) {
```

```
471 solution.fitness += 100; }
472  //Minimiser les périodes libres en fin d'après-midi
473 ...
474 if(schedule.timetable[day][5][level][group]==null&&
475 schedule.timetable[day][6][level][group] == null) {
476  solution.fitness += 100; }
477 ...
478     return fitnessValues;}
479
```

5.3.4 La sélection

Les meilleures solutions (les plannings les plus aptes) sont sélectionnées à l'aide de la méthode selection(). Cette étape imite la sélection naturelle où seuls les individus les plus forts sont conservés pour la prochaine génération

/src/ScheduleGenerator.java

```
238 static List<Solution> selection(List<Solution> combined, int populationSize)
239 {
240       combined.sort(Comparator.comparingInt(s -> -s.fitness));
241 // Créer une sous-liste contenant les premières populationSize solutions
442 return new ArrayList<>(combined.subList(0, populationSize));
443 }
```

5.3.5 Le croisement et la mutation

Pour le processus de croisement et de mutation qui sont décrits précédemment, l'intérêt principal est de produire de nouveaux individus et raffiner des solutions obtenues : ***Croisement*** (combinaison de deux solutions) et ***mutation*** (modification aléatoire d'une solution):

- **La méthode de croisement**

/src/ScheduleGenerator.java

```
246 static List<Solution> crossoverAndMutation(List<Solution> population,
247 ...
```

```
248     for (int i = 0; i < POPULATION_SIZE; i ++) {
249 ...
250         Schedule parent1 = population.get(index1).schedule;
251         Schedule parent2 = population.get(index2).schedule;
252 ...
253 for (int day = 0; day < 3; day++) {
254 ...
255 child1.timetable[day][slot][lvl][grp]=parent1.timetable[day][slot][lvl][grp];
256 child2.timetable[day][slot][lvl][grp]=parent2.timetable[day][slot][lvl][grp];
257 }
258 for (int day = 3; day < DAYS; day++) {
259 ...
260 child1.timetable[day][slot][lvl][grp]=parent2.timetable[day][slot][lvl][grp];
261 child2.timetable[day][slot][lvl][grp]=parent1.timetable[day][slot][lvl][grp];
262 }
263         RemoveandAddMissingHours(child1, teachers, subjects);
264         mutatSchedule(child1);
265  if(validateSchedule(child1, subjects))   {
266         newPopulation.add(new Solution(child1));
267    }
268 ...
269      if(validateSchedule(child2,subjects) && validateSchedule(child1, subjects))
270             {
271 ...
272 if (random.nextBoolean()){
273 ...
274  child3.timetable[day][slot][lvl][grp]=child1.timetable[day][slot][lvl][grp];
275 }
276 else{
277 ...
278 child3.timetable[day][slot][lvl][grp]=child2.timetable[day][slot][level][group];
279 }
```

```
280 }
281 ...
282  return newPopulation;   }
```

- **La méthode deMutation**

/src/ScheduleGenerator.java

```
662 static void mutatSchedule(Schedule schedule) {
663 ...
664         day1 = rand.nextInt(DAYS);
665         slot1 = rand.nextInt(SLOTS_PER_DAY);
666         level1 = rand.nextInt(LEVELS);
667         group1 = rand.nextInt(GROUPS_PER_LEVEL);
668 //Chercher une heure creuse
669 if ((slot1 == 0) || (slot1 == 3) || (slot1 == 4) || (slot1 == 6)) continue;
670 if (day1 == 2 && slot1>= 4) continue;
671 if (schedule.timetable[day1][slot1][level1][group1] == null ) {
672             foundEmptySlot = true;
673             break;
674         }
675     }
676     if (!foundEmptySlot) {
677 return;
678     }
679    //Prendre une séance aléatoire dans le premier ou dernier créneau du
680 temps
681 for (int attempts = 0; attempts < MAX_ATTEMPTS; attempts++) {
682         day2 = rand.nextInt(DAYS);
683         slot2 = rand.nextInt(SLOTS_PER_DAY);
684 ...
685         if (day2 == 2 && slot2 >= 4) continue;
686           if ((slot2 == 1) || (slot2 == 2) || (slot2 == 5) ) continue;
687   if (schedule.timetable[day2][slot2][level1][group1] != null &&
          !(schedule.isTeacherAssigned(day1, slot1, teacherName))) {
```

```
688 schedule.timetable[day1][slot1][level1][group1]=
689 schedule.timetable[day2][slot2][level1][group1];
690 schedule.timetable[day2][slot2][level1][group1] = null;
691  return; }  }
692
```

5.3.6 Miseà jour de l'archive

La méthode updateArchive met à jour l'archive des solutions en sélectionnant les meilleures solutions, trie par fitness décroissant et sélectionne les meilleures solutions uniques jusqu'à une taille maximale de l'archive

/src/ScheduleGenerator.java

```
586 static List<Solution>updateArchive(List<Solution>solutions,int archiveSize)
587 {
588     solutions.sort(Comparator.comparingInt(s -> -s.fitness));
    }
```

5.4 Le résultat de ce processus

Le code développé en **Java** a été appliqué au sein de le collège **El Amir Abd El Kader**. Toutes les contraintes ont été respectées, assurant ainsi une répartition correcte des horaires, la disponibilité des enseignants, et l'élimination des heures creuses, conformément aux exigences du projet.

Les figures suivantes montre le résultat obtenu après six cents (200) itération :

République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		informatique(...	francais(Mr. ...			PE(Mr. Zouhir)	PE(Mr. Zouhir)	
Lundi	sport(Mr. Khe...	sport(Mr. Khe...	arabe(Ms. sa...	Math(Mr. Bel...		arabe(Ms. sa...	anglais(Mr. B...	
Mardi	Math(Mr. Belh...	Math(Mr. Belh...	anglais(Mr. B...	arabe(Ms. sa...				
Mercredi	History(Ms. L...	francais(Mr. ...	History(Ms. L...	Science(Ms. ...		Math(Mr. Bel...	francais(Mr. ...	Math(Mr. Bel...
Jeudi	History(Ms. L...	arabe(Ms. sa...	arabe(Ms. sa...	Math(Mr. Bel...		Science(Ms. ...	Science(Ms. ...	PE(Mr. Zouhir)

Figure 5.1 : Résultat obtenu après 200 itérations de niveau 1 groupe 1

République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	History(Ms. L...	Math(Mr. Belh...	arabe(Ms. sa...	Math(Mr. Bel...			informatique(...	
Lundi	sport(Mr. Khe...	sport(Mr. Khe...	arabe(Ms. sa...			History(Ms. L...	francais(Mr. ...	Math(Mr. Bel...
Mardi	Math(Mr. Belh...	PE(Mr. Zouhir)	PE(Mr. Zouhir)	Science(Ms. ...				
Mercredi	Science(Ms. ...	Science(Ms. ...	anglais(Mr. B...	Math(Mr. Bel...		arabe(Ms. sa...	anglais(Mr. B...	francais(Mr. ...
Jeudi	Math(Mr. Belh...	francais(Mr. ...	arabe(Ms. sa...	arabe(Ms. sa...		PE(Mr. Zouhir)	History(Ms. L...	

Figure 5.2 : Résultat obtenu après 200 itérations de niveau 1 groupe 2

République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	Math(Mr. Beldi)	Math(Mr. Beldi)	anglais(Mr. B...	arabe(Ms. sa...			History(Ms. K...	arabe(Ms. sa...
Lundi	History(Ms. K...	arabe(Ms. sa...	PE(Mr. Zouhir)				anglais(Mr. B...	Math(Mr. Beldi)
Mardi	arabe(Ms. sa...	History(Ms. K...	Math(Mr. Beldi)	francais(Mr. ...				
Mercredi	informatique(...	Math(Mr. Beldi)	Science(Ms. ...	Science(Ms. ...		francais(Mr. ...	Math(Mr. Beldi)	
Jeudi	sport(Mr. Khe...	sport(Mr. Khe...	Science(Ms. ...	francais(Mr. ...		arabe(Ms. sa...	PE(Mr. Zouhir)	PE(Mr. Zouhir)

Figure 5.3 : Résultat obtenu après 200 itérations de niveau 1 groupe 3

République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

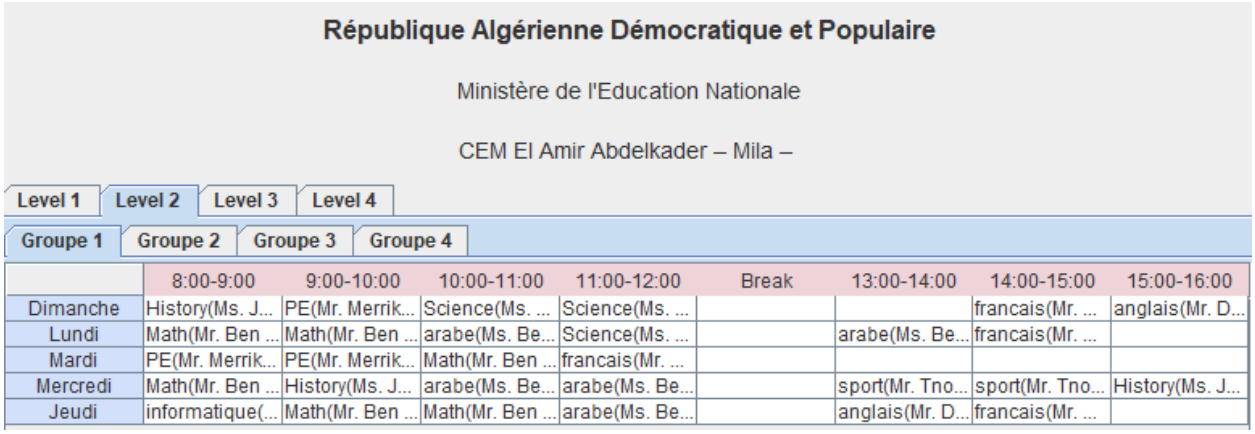
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	History(Ms. K...	PE(Mr. Zouhir)	PE(Mr. Zouhir)	francais(Mr. ...			arabe(Mr. Ber...	
Lundi	anglais(Mr. B...	arabe(Mr. Ber...	francais(Mr. ...	History(Ms. K...		PE(Mr. Zouhir)	Math(Mr. Beldi)	
Mardi	Math(Mr. Beldi)	sport(Mr. Khe...	sport(Mr. Khe...	Science(Ms. ...				
Mercredi	arabe(Mr. Ber...	Math(Mr. Beldi)	informatique(...	francais(Mr. ...		History(Ms. K...	arabe(Mr. Ber...	
Jeudi	anglais(Mr. B...	Science(Ms. ...	Science(Ms. ...	Math(Mr. Beldi)		Math(Mr. Beldi)	arabe(Mr. Ber...	Math(Mr. Beldi)

Figure 5.4 : Résultat obtenu après 200 itérations de niveau 1 groupe 4



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

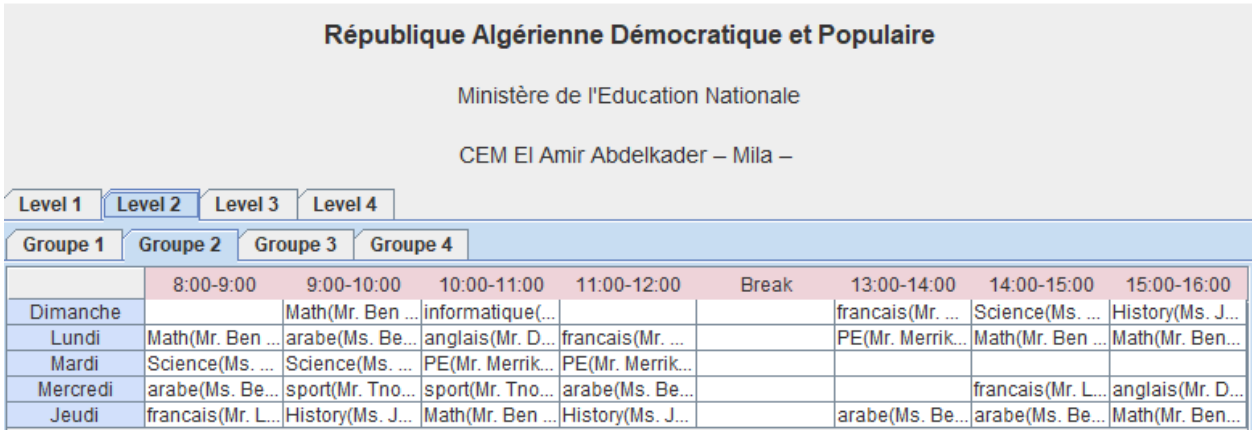
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	History(Ms. J...	PE(Mr. Merrik...	Science(Ms. ...	Science(Ms. ...			francais(Mr. ...	anglais(Mr. D...
Lundi	Math(Mr. Ben ...	Math(Mr. Ben ...	arabe(Ms. Be...	Science(Ms. ...		arabe(Ms. Be...	francais(Mr. ...	
Mardi	PE(Mr. Merrik...	PE(Mr. Merrik...	Math(Mr. Ben ...	francais(Mr. ...				
Mercredi	Math(Mr. Ben ...	History(Ms. J...	arabe(Ms. Be...	arabe(Ms. Be...		sport(Mr. Tno...	sport(Mr. Tno...	History(Ms. J...
Jeudi	informatique(...	Math(Mr. Ben ...	Math(Mr. Ben ...	arabe(Ms. Be...		anglais(Mr. D...	francais(Mr. ...	

Figure 5.5 : Résultat obtenu après 200 itérations de niveau 2 groupe 1



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

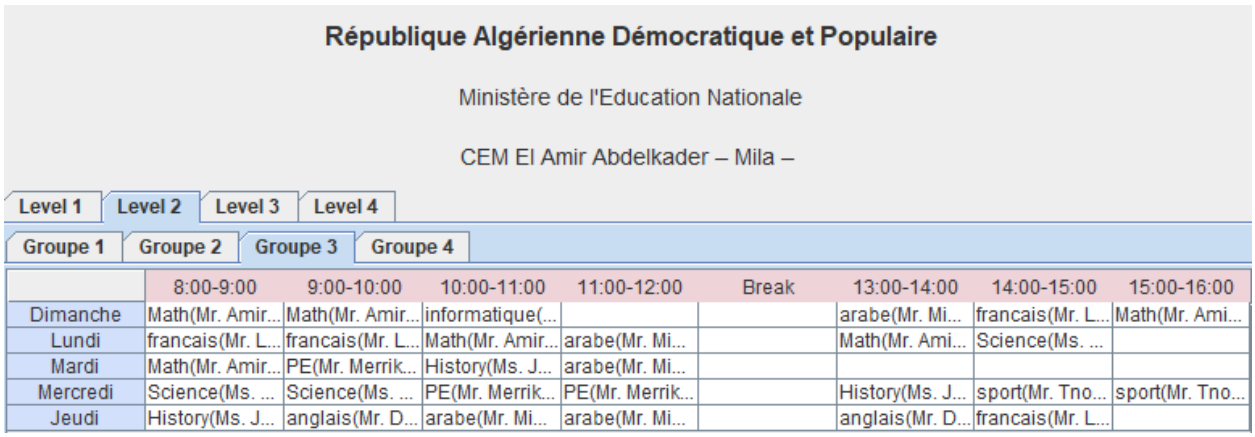
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		Math(Mr. Ben ...	informatique(...			francais(Mr. ...	Science(Ms. ...	History(Ms. J...
Lundi	Math(Mr. Ben ...	arabe(Ms. Be...	anglais(Mr. D...	francais(Mr. ...		PE(Mr. Merrik...	Math(Mr. Ben ...	Math(Mr. Ben...
Mardi	Science(Ms. ...	Science(Ms. ...	PE(Mr. Merrik...	PE(Mr. Merrik...				
Mercredi	arabe(Ms. Be...	sport(Mr. Tno...	sport(Mr. Tno...	arabe(Ms. Be...			francais(Mr. L...	anglais(Mr. D...
Jeudi	francais(Mr. L...	History(Ms. J...	Math(Mr. Ben ...	History(Ms. J...		arabe(Ms. Be...	arabe(Ms. Be...	Math(Mr. Ben...

Figure 5.6 : Résultat obtenu après 200 itérations de niveau 2 groupe 2



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

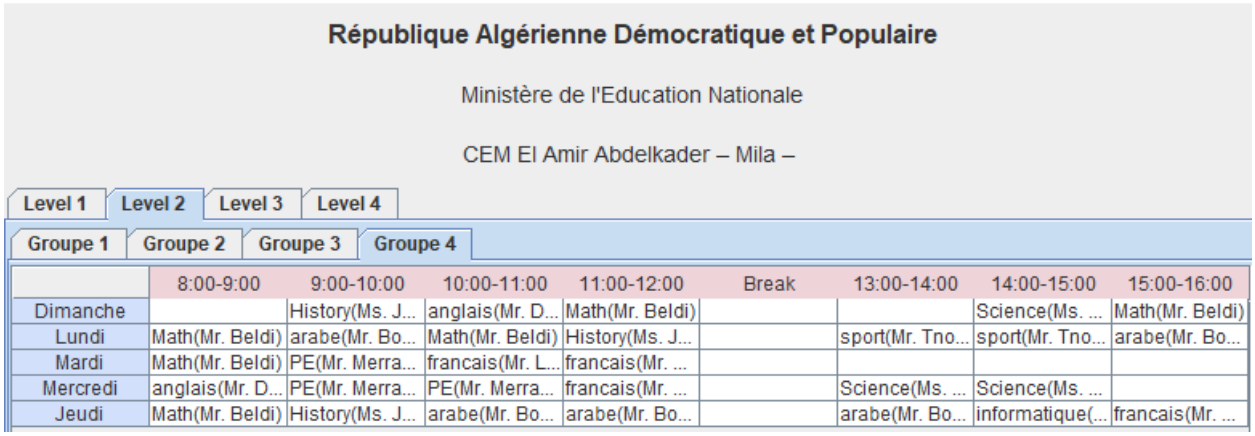
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	Math(Mr. Amir...	Math(Mr. Amir...	informatique(...			arabe(Mr. Mi...	francais(Mr. L...	Math(Mr. Ami...
Lundi	francais(Mr. L...	francais(Mr. L...	Math(Mr. Amir...	arabe(Mr. Mi...		Math(Mr. Ami...	Science(Ms. ...	
Mardi	Math(Mr. Amir...	PE(Mr. Merrik...	History(Ms. J...	arabe(Mr. Mi...				
Mercredi	Science(Ms. ...	Science(Ms. ...	PE(Mr. Merrik...	PE(Mr. Merrik...		History(Ms. J...	sport(Mr. Tno...	sport(Mr. Tno...
Jeudi	History(Ms. J...	anglais(Mr. D...	arabe(Mr. Mi...	arabe(Mr. Mi...		anglais(Mr. D...	francais(Mr. L...	

Figure 5.7 : Résultat obtenu après 200 itérations de niveau 2 groupe 3



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

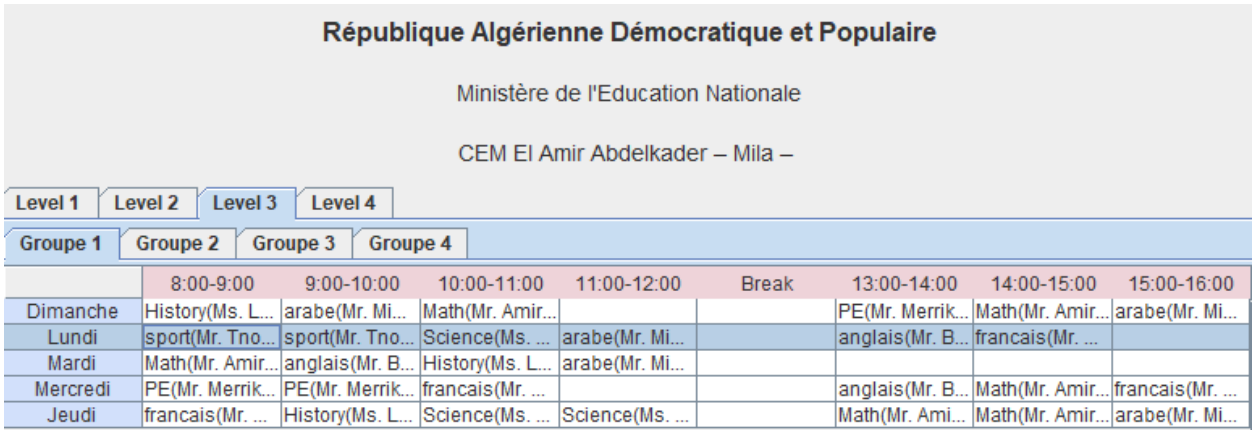
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		History(Ms. J...	anglais(Mr. D...	Math(Mr. Beldi)			Science(Ms. ...	Math(Mr. Beldi)
Lundi	Math(Mr. Beldi)	arabe(Mr. Bo...	Math(Mr. Beldi)	History(Ms. J...		sport(Mr. Tno...	sport(Mr. Tno...	arabe(Mr. Bo...
Mardi	Math(Mr. Beldi)	PE(Mr. Merra...	francais(Mr. L...	francais(Mr. ...				
Mercredi	anglais(Mr. D...	PE(Mr. Merra...	PE(Mr. Merra...	francais(Mr. ...		Science(Ms. ...	Science(Ms. ...	
Jeudi	Math(Mr. Beldi)	History(Ms. J...	arabe(Mr. Bo...	arabe(Mr. Bo...		arabe(Mr. Bo...	informatique(...	francais(Mr. ...

Figure 5.8 : Résultat obtenu après 200 itérations de niveau 2 groupe 4



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

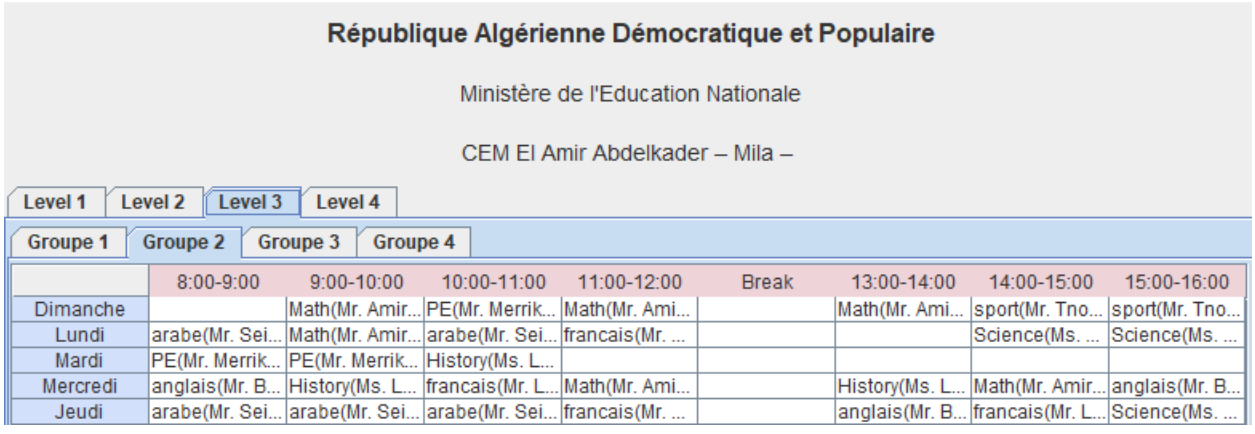
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	History(Ms. L...	arabe(Mr. Mi...	Math(Mr. Amir...			PE(Mr. Merrik...	Math(Mr. Amir...	arabe(Mr. Mi...
Lundi	sport(Mr. Tno...	sport(Mr. Tno...	Science(Ms. ...	arabe(Mr. Mi...		anglais(Mr. B...	francais(Mr. ...	
Mardi	Math(Mr. Amir...	anglais(Mr. B...	History(Ms. L...	arabe(Mr. Mi...				
Mercredi	PE(Mr. Merrik...	PE(Mr. Merrik...	francais(Mr. ...			anglais(Mr. B...	Math(Mr. Amir...	francais(Mr. ...
Jeudi	francais(Mr. ...	History(Ms. L...	Science(Ms. ...	Science(Ms. ...		Math(Mr. Ami...	Math(Mr. Amir...	arabe(Mr. Mi...

Figure 5.9 : Résultat obtenu après 200 itérations de niveau 3 groupe 1



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

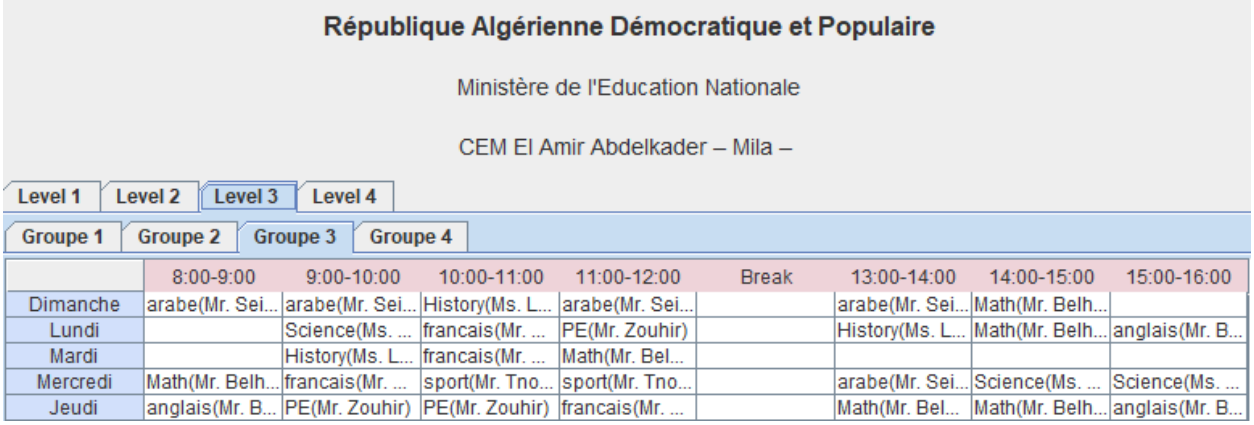
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		Math(Mr. Amir...	PE(Mr. Merrik...	Math(Mr. Ami...		Math(Mr. Ami...	sport(Mr. Tno...	sport(Mr. Tno...
Lundi	arabe(Mr. Sei...	Math(Mr. Amir...	arabe(Mr. Sei...	francais(Mr. ...			Science(Ms. ...	Science(Ms. ...
Mardi	PE(Mr. Merrik...	PE(Mr. Merrik...	History(Ms. L...					
Mercredi	anglais(Mr. B...	History(Ms. L...	francais(Mr. L...	Math(Mr. Ami...		History(Ms. L...	Math(Mr. Amir...	anglais(Mr. B...
Jeudi	arabe(Mr. Sei...	arabe(Mr. Sei...	arabe(Mr. Sei...	francais(Mr. ...		anglais(Mr. B...	francais(Mr. L...	Science(Ms. ...

Figure 5.10 : Résultat obtenu après 200 itérations de niveau 3 groupe 2



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

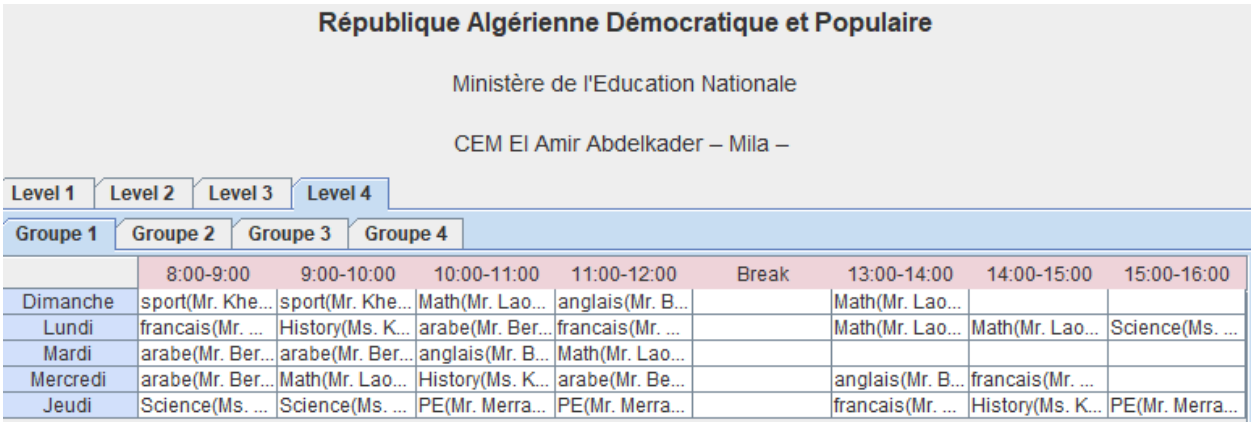
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	arabe(Mr. Sei...	arabe(Mr. Sei...	History(Ms. L...	arabe(Mr. Sei...		arabe(Mr. Sei...	Math(Mr. Belh...	
Lundi		Science(Ms. ...	francais(Mr. ...	PE(Mr. Zouhir)		History(Ms. L...	Math(Mr. Belh...	anglais(Mr. B...
Mardi		History(Ms. L...	francais(Mr. ...	Math(Mr. Bel...				
Mercredi	Math(Mr. Belh...	francais(Mr. ...	sport(Mr. Tno...	sport(Mr. Tno...		arabe(Mr. Sei...	Science(Ms. ...	Science(Ms. ...
Jeudi	anglais(Mr. B...	PE(Mr. Zouhir)	PE(Mr. Zouhir)	francais(Mr. ...		Math(Mr. Bel...	Math(Mr. Belh...	anglais(Mr. B...

Figure 5.11: Résultat obtenu après 200 itérations de niveau 3 groupe 3



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

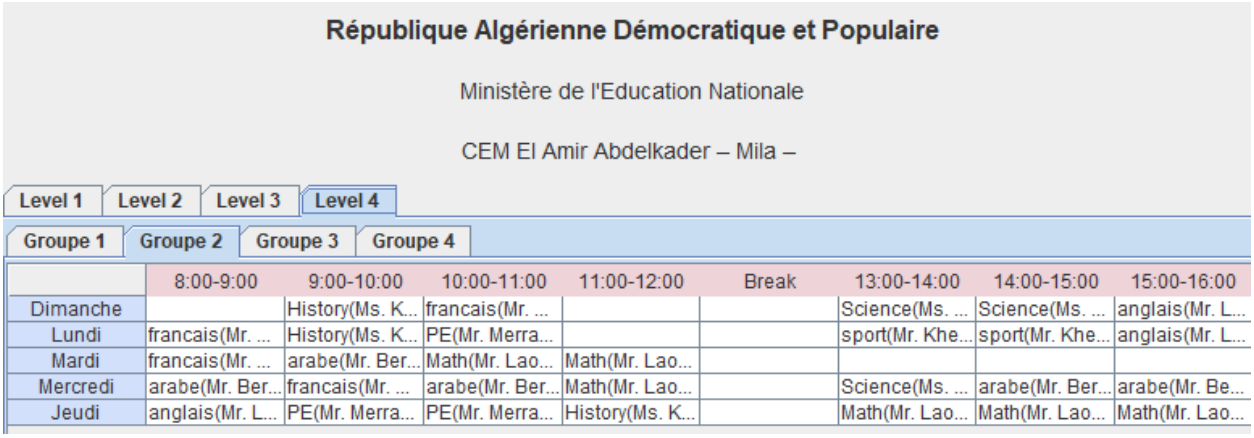
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche	sport(Mr. Khe...	sport(Mr. Khe...	Math(Mr. Lao...	anglais(Mr. B...		Math(Mr. Lao...		
Lundi	francais(Mr. ...	History(Ms. K...	arabe(Mr. Ber...	francais(Mr. ...		Math(Mr. Lao...	Math(Mr. Lao...	Science(Ms. ...
Mardi	arabe(Mr. Ber...	arabe(Mr. Ber...	anglais(Mr. B...	Math(Mr. Lao...				
Mercredi	arabe(Mr. Ber...	Math(Mr. Lao...	History(Ms. K...	arabe(Mr. Be...		anglais(Mr. B...	francais(Mr. ...	
Jeudi	Science(Ms. ...	Science(Ms. ...	PE(Mr. Merra...	PE(Mr. Merra...		francais(Mr. ...	History(Ms. K...	PE(Mr. Merra...

Figure 5.12 : Résultat obtenu après 200 itérations de niveau 4 groupe 1



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

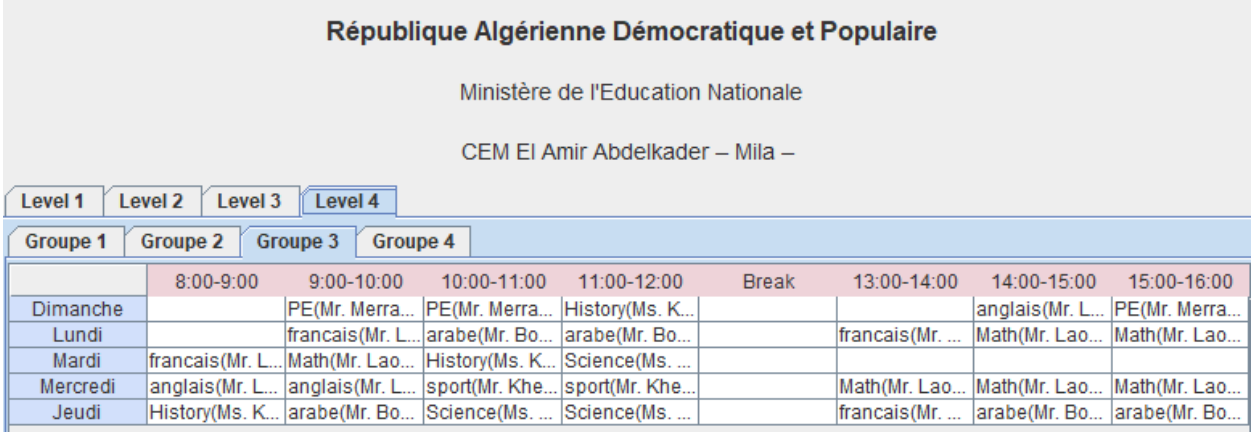
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		History(Ms. K...	francais(Mr. ...			Science(Ms. ...	Science(Ms. ...	anglais(Mr. L...
Lundi	francais(Mr. ...	History(Ms. K...	PE(Mr. Merra...			sport(Mr. Khe...	sport(Mr. Khe...	anglais(Mr. L...
Mardi	francais(Mr. ...	arabe(Mr. Ber...	Math(Mr. Lao...	Math(Mr. Lao...				
Mercredi	arabe(Mr. Ber...	francais(Mr. ...	arabe(Mr. Ber...	Math(Mr. Lao...		Science(Ms. ...	arabe(Mr. Ber...	arabe(Mr. Be...
Jeudi	anglais(Mr. L...	PE(Mr. Merra...	PE(Mr. Merra...	History(Ms. K...		Math(Mr. Lao...	Math(Mr. Lao...	Math(Mr. Lao...

Figure 5.13 : Résultat obtenu après 200 itérations de niveau 4 groupe 2



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

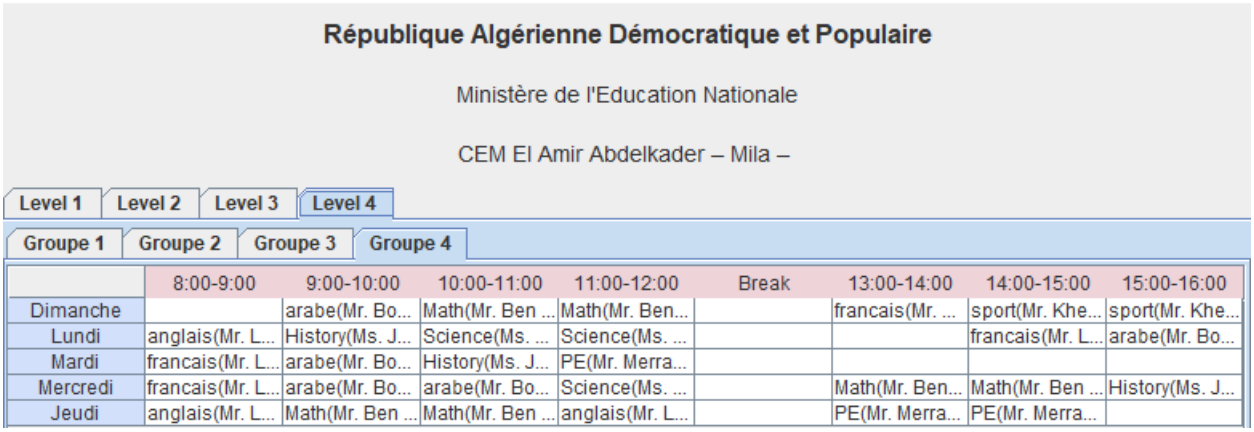
CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		PE(Mr. Merra...	PE(Mr. Merra...	History(Ms. K...			anglais(Mr. L...	PE(Mr. Merra...
Lundi		francais(Mr. L...	arabe(Mr. Bo...	arabe(Mr. Bo...		francais(Mr. ...	Math(Mr. Lao...	Math(Mr. Lao...
Mardi	francais(Mr. L...	Math(Mr. Lao...	History(Ms. K...	Science(Ms. ...				
Mercredi	anglais(Mr. L...	anglais(Mr. L...	sport(Mr. Khe...	sport(Mr. Khe...		Math(Mr. Lao...	Math(Mr. Lao...	Math(Mr. Lao...
Jeudi	History(Ms. K...	arabe(Mr. Bo...	Science(Ms. ...	Science(Ms. ...		francais(Mr. ...	arabe(Mr. Bo...	arabe(Mr. Bo...

Figure 5.14 : Résultat obtenu après 200 itérations de niveau 4 groupe 3



République Algérienne Démocratique et Populaire

Ministère de l'Education Nationale

CEM El Amir Abdelkader – Mila –

Level 1 | Level 2 | Level 3 | Level 4

Groupe 1 | Groupe 2 | Groupe 3 | Groupe 4

	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	Break	13:00-14:00	14:00-15:00	15:00-16:00
Dimanche		arabe(Mr. Bo...	Math(Mr. Ben ...	Math(Mr. Ben...		francais(Mr. ...	sport(Mr. Khe...	sport(Mr. Khe...
Lundi	anglais(Mr. L...	History(Ms. J...	Science(Ms. ...	Science(Ms. ...			francais(Mr. L...	arabe(Mr. Bo...
Mardi	francais(Mr. L...	arabe(Mr. Bo...	History(Ms. J...	PE(Mr. Merra...				
Mercredi	francais(Mr. L...	arabe(Mr. Bo...	arabe(Mr. Bo...	Science(Ms. ...		Math(Mr. Ben...	Math(Mr. Ben ...	History(Ms. J...
Jeudi	anglais(Mr. L...	Math(Mr. Ben ...	Math(Mr. Ben ...	anglais(Mr. L...		PE(Mr. Merra...	PE(Mr. Merra...	

Figure 5.15 : Résultat obtenu après 200 itérations de niveau 4 groupe 4

5.5 Discussion de résultat

Le résultat de notre travail a produit de bons emplois du temps, notamment en évitant les heures creuses et garantissant ainsi la planifiant des matières principales dans les créneaux du matin, la figure suivante représente la courbe de meilleur individu pour chaque génération. Cependant, la convergence observée de la fonction de fitness après environ 60 générations peut êtrejustifiée par la simplicité de cette fonction. Une fonction de fitness plus élaborée aurait probablement permis d'obtenir des résultats plus diversifiés et optimisés.

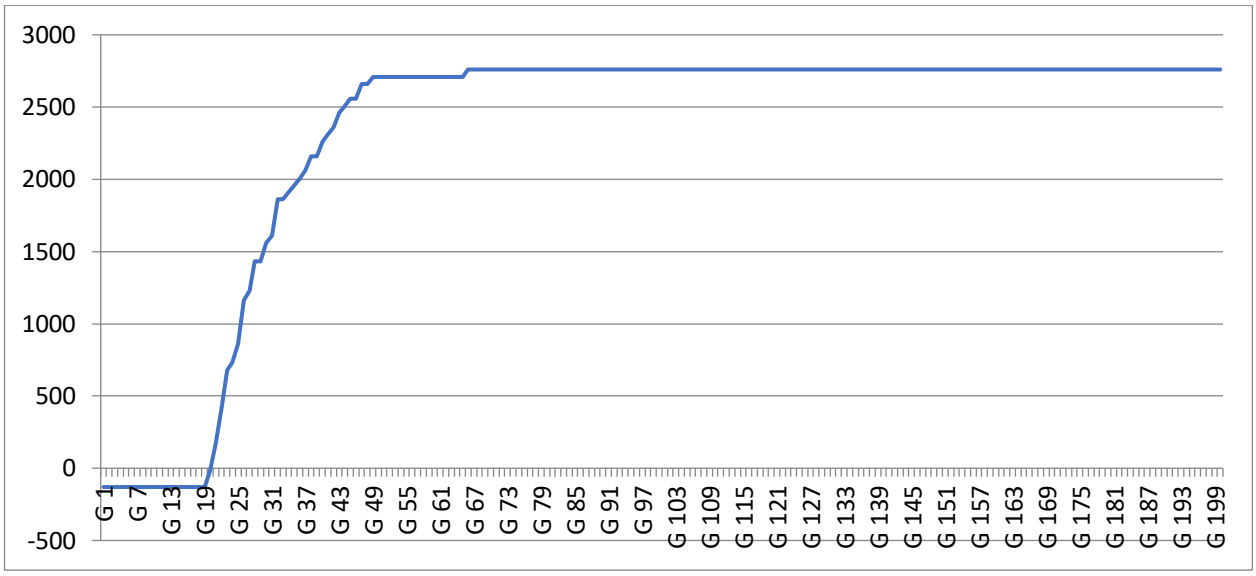


Figure 5.16 : les meilleurs individu pour chaque génération.

Les résultats en statistique

Le tableau représente les résultats de l'accomplissement de la fonction Fitness pour les matières fondamentaux ainsi que pour le sporttel que :

Séances Réalisées : Le nombre de séances réalisées avecla fonction Fitness.

Matière	Séances Réalisées	Total des Séances	Pourcentage
Mathématique	**57**	**90**	**63.33%**
Science	**30**	**45**	**66.66%**
Pysique	**33**	**45**	**73.33%**
Arabe	**50**	**75**	**66.66%**
Sport	**26**	**30**	**86.66%**

Tableau 5.1 : les résultats en statistique.

Conclusion

Tout au long de ce chapitre, nous avons détaillé les aspects techniques de notre projet, depuis le choix des outils de développement jusqu'à l'implémentation de l'algorithme de génération d'emploi du temps. La combinaison des fonctionnalités

puissantes de la programmation orienté objet en Java et des outils offerts par NetBeans nous a permis de créer une application efficace et évolutive, répondant aux exigences de la génération d'emploi du temps scolaire.et nous avons présenté les résultats obtenus lors de l'implémentation de notre approche.

Conclusion générale

Conclusion générale

De nombreux problèmes d'intelligence artificielle, pour lesquels on ne connaît souvent pas d'algorithme déterministe, relèvent du problème de Satisfaction de Contraintes. La résolution automatique de ces problèmes consiste à essayer successivement toutes les possibilités jusqu'à trouver une solution. Cette recherche est fortement combinatoire, l'" intelligence " consiste à trouver une solution acceptable en premier lieu et essayer par la suite de la raffiner pour qu'elle soit optimale.

Nous avons présenté dans ce modeste travail une étude sur l'application des algorithmes génétiques pour résoudre les problèmes de Satisfaction de Contraintes. Plus précisément, nous avons appliqué l'algorithme génétique pour l'optimisation du problème d'emploi du temps de collège el Amir Abdkader.

Les algorithmes génétiques ce sont des techniques inspirées de la nature basé sur des opérations génétiques telle que: la sélection, le croisement, la mutation… .L'importance de ces algorithmes est qu'ils sont heuristiques donc pas de problème de l'explosion combinatoire.

Perspective

À travers ce travail, nous avons démontré l'efficacité des algorithmes génétiques dans la résolution de problèmes de planification, en particulier pour la génération d'emplois du temps respectant des contraintes complexes. Cependant, plusieurs perspectives d'amélioration peuvent être envisagées pour étendre et perfectionner cette approche.

l'algorithme ait été appliqué à un établissement spécifique, il serait pertinent de l'adapter à d'autres niveaux d'enseignement ou types d'établissements, tels que les universités, en prenant en compte des contraintes supplémentaires comme les laboratoires pratiques.

Bibliographie

[1] : ***Edward Tsang. Foundations of Constraint Satisfaction, Academic Press Inc. San Diego, California, USA, 1993***

[2] : ***Pascal Van Hentenryck et Vijay Saraswat, Strategic directions in Constraint programming, ACM Computing surveys (CSUR), 1996.***

[3] : ***Peter G, D.A. Cohen, ET M. Cooper, Constraint, Consistency and closure, Artificial Intelligence, 101(1-2) Mars 1998.***

[4] : ***I-Ling Lin, Practical Swarm optimisation for solving Constraint Satisfaction Problems, B. Sc, Simon Fraser University, 2002.***

[5] : ***Jano Van Hemert, Constraint Satisfaction Problems and Evolutionary Computation: A reality check, edition ven den Bosch and H. Du 12ème conference de l'intelligence artificielle aux Pays-bas. Pages 267-274, Tilburg, Pays-Bas, Novembre 2000.***

[6] : ***Souquet Amédée, Radet Francois-Gérard, Algorithmes génétique, thèse de magistère, soutenu le 21/06/2004.***

[7] : ***Lixi Zhang, Solving the Timetabling Using Problem Satisfaction Programming, School of Economics and Information Systems, University of WOLLONGONG, 2005***

[8] : ***Roman Barteck, Constraint programming: In pursuit of the Holy Grail. In proceedings of week of Doctoral Students (WDS99), partie 04, page 555-564, Prague, Juin 1999.***

[9] : Logiciel java.sokeo.[En ligne]https://sokeo.fr/logiciel-java-definition/.

[10] : java.greelane.[En ligne]https://www.greelane.com/fr/science.

[11] : NetBeans.wikipedia.[En ligne]https://fr.wikipedia.org/wiki/NetBeans.