Application des algorithmes génétiques sur un planning sous contraintes

Rakotomahefa A.[1], Randimbindrainibe F.[2], Robinson M.[3]

Ecole Doctorale en Science et Technique de l'Ingénierie et de l'Innovation (ED-STII)

Laboratoire de Sciences Cognitives et Applications (SCA)

Université d'Antananarivo

BP 1500, Ankatso – Antananarivo 101 – Madagascar

[1]andri.mirindra@gmail.com, [2]falimanana@mail.ru, [3]mat_robinson2000@yahoo.fr

Résumé

L'élaboration d'un planning, une tache habituelle, compliquée et lourde, dans une entreprise ou dans un établissement, est un problème NP difficile ; dans cette recherche, nous proposons une approche basée sur les algorithmes génétiques pour l'automatiser. Les tâches sont placées méthodiquement dans une grille de temps ; tout en respectant les contraintes liées aux agents : leurs disponibilités aussi bien que leurs nombres d'heures de travail, et les contraintes liées aux temps : les jours ouvrables et le nombre d'heures de travail dans une journée. De générations en générations, les algorithmes génétiques approchent progressivement les solutions optimales. Nous avons pu montrer que ces solutions seront atteintes après quelques centaines de générations et que le temps d'exécution du modèle proposé est raisonnable car sa complexité algorithmique est polynomiale.

Mots clés : Sciences cognitives, Intelligence Artificielle, Planification, Algorithmes Génétiques.

Abstract

Scheduling, a habitual, complicated and heavy task for an organization and for a company is a NP-hard problem; in this research, we propose an approach based on genetic algorithms to automate it. All tasks are methodically placed in a time grid, while respecting the constraints related to agents: their availability as well as their working hour's number, and the time constraints: the working days, the number of working's hours a day. From generation to generation, genetic algorithms approach gradually the optimal solutions. We have shown that these solutions will be reached after hundreds of generations, and the execution time of the proposed model is reasonable because its computational complexity is polynomial.

Keys words: Cognitive Science, Artificial Intelligence, Scheduling, Genetic Algorithms.

1. Introduction

La planification concerne chaque individu et spécialement tous les organismes et c'est l'une des clés de la réussite d'une entreprise. La planification, qui est l'allocation des ressources données dans un intervalle de temps, de façon à satisfaire un ensemble de contraintes – est un problème difficile, compliqué et de grande taille, dont la réalisation à la main mobilise plusieurs personnes et demande un certain nombre de jours de travail et surtout, plus de travaux intellectuels. Tout cela conduit à la question majeure suivante : *est-il possible de trouver une solution qui consiste à imiter le fonctionnement du cerveau humain afin d'élaborer, d'une manière automatique, un planning qui satisfait toutes contraintes imposées* ?

1.1 La planification

La planification est un processus d'aide à la décision qui vise à prévoir des ressources et des services requis pour atteindre des objectifs déterminés, selon un ordre de priorité établi, permettant ainsi le choix d'une solution préférable parmi plusieurs alternatives. Ce choix prend en considération le contexte et les contraintes.

Ces contraintes sont essentiellement :

- Les contraintes légales et réglementaires
- Les contraintes organisationnelles
- Les desiderata du personnel

La démarche typique de la planification consiste, en premier lieu à définir les actions à accomplir. Une fois que la liste des actions est établie, on commence à identifier et à ordonner les tâches correspondantes. Il est important de savoir qu'à chaque tâche, nous devons :

- Affecter un responsable
- Allouer des ressources
- Evaluer la durée
- Préciser les dates (début et fin)

L'étape suivante est de prévoir les ressources et moyens nécessaires pour l'accomplissement des actions. La dernière étape c'est de produire le plan de travail, appelé aussi le planning.

1.2 Le planning

Le planning est un calendrier de travail, où figurent à la fois le temps, l'affectation des ressources, les jours et les horaires de travail.

Pour représenter un planning, plusieurs façons existent mais les plus connues sont :

- La technique GANTT *(planning à barres)*
- La technique PERT (Program Evaluation and Review Technic – *méthode des potentielles étapes et planning des tâches.*
- Le réseau des antécédents *(méthode des potentielles tâches)*

2. Approches utilisées dans la littérature

La planification fait partie des problèmes appartenant à la classe de complexité NP difficile. Dans la littérature, de nombreuses

études et recherches sont déjà faites et plusieurs techniques et approches algorithmiques sont proposées. Ce qui se suivent ne sont qu'une partie seulement et à titre indicatif.

a. La programmation mathématique :

C'est une approche quantitative où l'on s'intéresse à maximiser ou à minimiser une fonction objective qui mesure la performance ou la qualité de la décision.

Christian Liebchen [1] a utilisé cette technique pour planifier les arrivées et les départs de train.

b. La recherche taboue :

C'est une méthode de recherche locale combinée avec un ensemble de techniques permettant d'éviter d'être piégé dans un minimum local ou la répétition d'un cycle.

Cette méthode a été utilisée par **Glover** [2] pour résoudre le problème de trafic qui consiste à livrer des colis dans plusieurs destinations tout en minimisant la distance parcourue alors que tous les clients seront servis.

c. Techniques inspirées de la nature

D'autres techniques sont inspirées par des systèmes naturels dans de nombreux domaines tels que :

- L'éthologie (l'algorithme de colonies de fourmis : on imite le mécanisme qui permet aux fourmis de résoudre collectivement des problèmes complexes)
- La physique (le recuit simulé : basée sur une technique utilisée par les métallurgistes qui, pour obtenir un alliage sans défaut, faisant alterner les cycles de réchauffage (ou de recuit) et de refroidissement lent des métaux)
- La biologie (l'algorithme évolutionnaire et l'algorithme génétique)

3. Méthode adoptée

Les méthodes proposées peuvent êtres classifiées principalement en deux grandes catégories : les méthodes de recherche exacte et les méthodes de recherche approchée. Ces premières tentent d'atteindre la solution optimale, mais elles sont trop gourmandes en termes de temps de calcul et d'espace mémoire requis [3]. Les méthodes approchées se sont illustrées comme une alternative intéressante même si elles ne garantissent pas l'optimalité de la solution. Ces méthodes peuvent être partagées en deux classes : les méthodes heuristiques et les méthodes métaheuristiques. Une méthode heuristique est applicable sur un problème donné tandis qu'une méthode métaheuristique est plus générique et elle peut être appliquée sur une panoplie de problèmes d'optimisation. On retrouve également deux classes de métaheuristiques : les méthodes de recherche à base de voisinage et celles à base de population de solutions. Nous avons adopté une approche appartenant à cette dernière classe qui est l'algorithme génétique.

3.1 Les Algorithmes Génétiques

Les algorithmes génétiques sont des algorithmes fondés sur les mécanismes de la génétique et des principes évolutifs de la sélection naturelle de Charles Darwin. Ces techniques énonçaient que les individus les plus aptes à survivre se reproduiront plus souvent et auront plus de descendants.
Ils ont été adaptés à l'optimisation par John Holland dans les années 1970 [4].

Le fonctionnement des algorithmes génétiques est simple : on part d'une population d'individus de solutions initiales ; on évalue leur performance relative ; sur la base de ces performances on crée une nouvelle population de solutions potentielles en utilisant des opérateurs évolutionnaires tels que la sélection, le croisement et la mutation. Quelques individus se reproduisent, d'autres disparaissent et seuls les individus les mieux adaptés sont supposés survivre. En répétant ce cycle plusieurs fois, on obtient une population composée de solutions meilleures.

La mise en œuvre de l'algorithme génétique se résume par les neuf éléments suivants :

- Le Codage des individus
- La Génération de la population initiale
- La Fonction d'adaptation (Fitness)
- La Sélection
- Le Croisement
- La Mutation
- L'Optimisation
- Le Remplacement
- Le Critère d'arrêt

Algorithme génétique générique

1. Générer une population d'individus de taille n : $x_1, x_2, x_3, \ldots, x_n$
2. Tant que le critère de terminaison n'est pas atteint faire :
3. Evaluer le degré d'adaptation de chaque individu de la population courante
4. Sélectionner un ensemble d'individus de haute qualité
5. Appliquer à chacune des paires d'individus sélectionnés l'opérateur de croisement qui produira une ou plusieurs individus « enfants »
6. Appliquer un opérateur de mutation aux individus ainsi obtenus
7. Remplacer une partie de la population courante formée par des individus de basse qualité par les individus « enfants » de haute qualité
8. Les individus éventuellement mutés constituent la nouvelle génération de la population

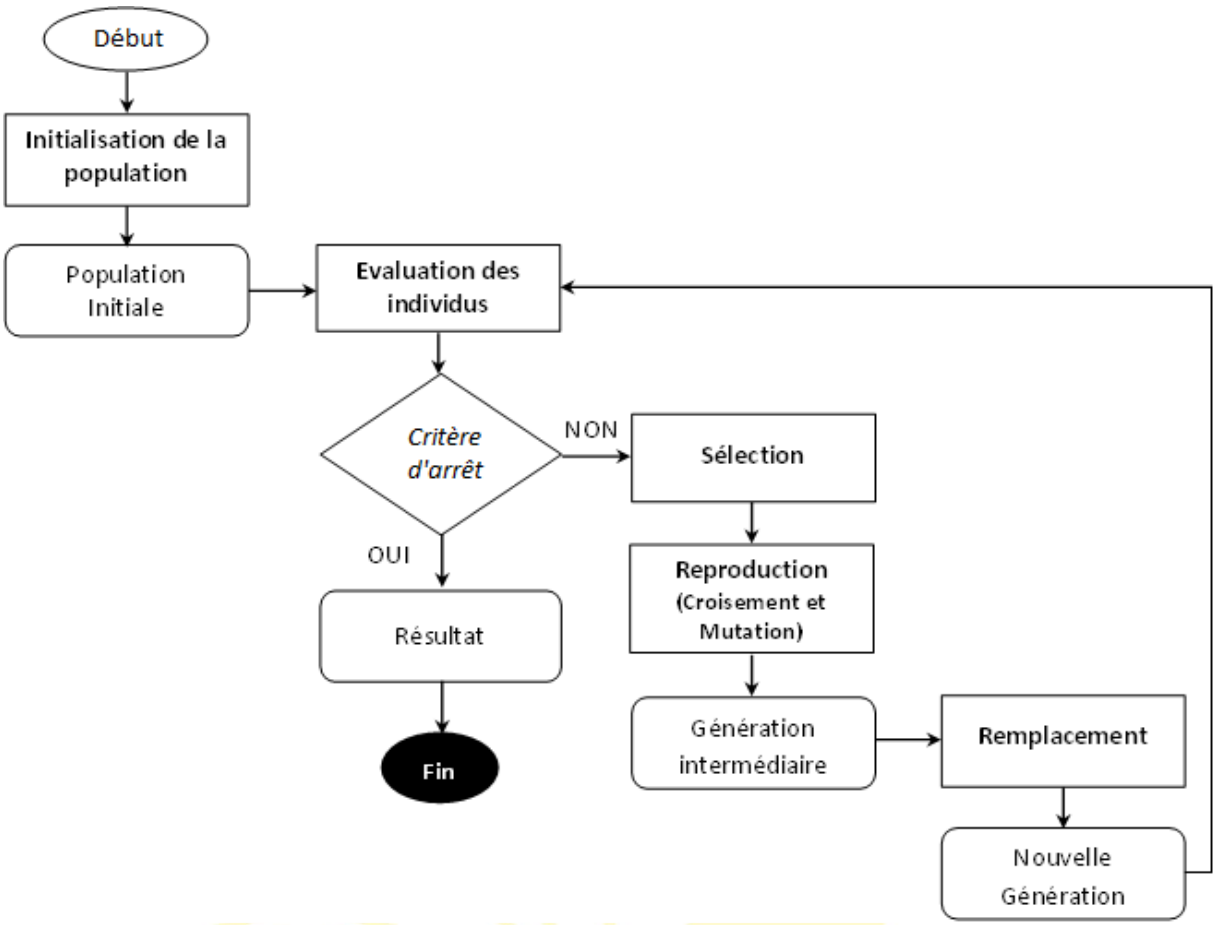

Figure 01 : Principales étapes d'un algorithme génétiques

a. Le Codage des individus

Les individus correspondent aux « solutions » du problème à optimiser. Ces solutions doivent être « codées ». Cette représentation codée d'une solution est appelée chromosome, et est composée de gènes. Plusieurs types de codages sont utilisés : Codage binaire, Codage à caractères multiples, Codage sous forme d'arbre.

b. Génération de la population

La population est l'ensemble des individus d'une même génération. Au départ, il faut générer cette population. La population initiale doit être suffisamment diversifiée et de taille assez importante pour que la recherche puisse parcourir l'espace d'état dans un temps limité.

c. La Fonction d'adaptation

C'est la fonction qui évalue la performance de chaque individu. On cherche alors à optimiser cette fonction. La fonction d'adaptation permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés.

d. La Sélection

La sélection permet de choisir les individus avec lesquels s'appliqueront les opérations de reproduction pour la création de la future génération. Il existe différentes techniques de

sélection comme la Sélection uniforme, la Sélection par roulette, la Sélection par rang, la Sélection par tournoi.

e. Le Croisement

Le croisement s'applique avec deux parents et génère deux enfants. Nous avons plusieurs méthodes pour croiser deux chromosomes, comme le Croisement à un ou à n points, le Croisement uniforme et le Croisement logique.

f. La Mutation

Une mutation est l'inversion d'un gène aléatoire dans un chromosome. L'opérateur de mutation est un processus où un changement mineur du code génétique est appliqué à un individu.

g. L'Optimisation

Il faut s'assurer que la fonction de création des individus crée toujours des individus valides.
Et que les opérateurs génétiques conservent la validité des individus traités.

h. Le Remplacement

Cet opérateur consiste à réintroduire les descendants obtenus par l'application des opérateurs génétiques dans la population de leurs parents. On trouve essentiellement deux méthodes de remplacement : le Remplacement stationnaire et le Remplacement élitiste.

i. Le Critère d'arrêt

Un algorithme génétique se termine après un certain nombre de générations, ou lorsqu'une certaine condition fixée soit atteinte, ou lorsque la population a convergé.

3.2 Principaux paramètres

Les opérateurs de l'algorithme génétique sont guidés par un certain nombre de paramètres structurels donnés. La valeur de ces paramètres influence la réussite (ou non) et la rapidité de l'algorithme [5]. Présentons brièvement ces principaux paramètres :

1) La taille de la population n et la longueur du codage de chaque individu. Il est préférable de prendre une taille de la population correspondante à la longueur du codage des individus. Si n est trop grand, le temps de calcul s'avère très important. Par contre, si n est trop petit l'algorithme converge trop rapidement ;

2) La probabilité de croisement p_c. Elle dépend de la forme de la fonction de performance.
Les valeurs admises sont généralement comprises entre 0, 5 et 0, 9. Plus p_c est élevée, plus la population subit de changement important. La convergence est très rapide si p_c est proche de 1 ;

3) La probabilité de mutation p_m. Ce taux est généralement faible. Un taux élevé risque de modifier les meilleurs individus et cela entraîne l'éloignement de l'optimalité.

3.3 Modélisation

La génération suivante X_{n+1} est en fonction de la génération courante X_n.

Parmi les n individus de X_n, on crée une sous-population de k individus pour être des parents. Ces k individus seront sélectionnés et on les fait reproduire entre eux en appliquant les opérateurs génétiques jusqu'à ce qu'on obtient k nouveaux individus.

On a alors le modèle mathématique suivant :

$$X_{n+1} = f(X_n) \qquad (1)$$

Telle que :

$$f(X_n) = X_n + G_k - M_k$$

Avec :

- X_n : population courante ;
- G_k : k nouveaux individus issus de la sous-population de k individus sélectionnés après application des opérateurs génétiques successifs croisement et mutation ;
- M_k : k mauvais individus da la population courante.

4. Traitement

4.1 Structure de données

Les traitements commencent par définir les données qui sont l'objet de traitement et de manipulation. Pour le problème de planification, on a cette liste de données : ces données sont :

- Nombre de départements/ateliers : c
- Liste des départements/ateliers : CL_i, $i= \{1, 2, ..., c\}$
- Nombre d'Agents : e
- Liste des Agents : EG_i, $i= \{1, 2, ..., e\}$
- Le nombre de jours de travail par semaine : n_j
- Le nombre d'heures de travail par jours : n_h
- Le nombre de tâches par départements : nc_i, $i= \{1, 2, ..., c\}$
- La liste de tâches par départements : $CR_{i,j}$, $i=\{1, 2, ..., c\}$ et $j=\{1, 2, ..., nc_i\}$
- Nombre d'heures nécessaire pour exécuter chaque tâche : $HC_{i,j}$, $i=\{1, 2, ..., c\}$ et $j=\{1, 2, ..., nc_i\}$
- Agent responsable de chaque tâche : $EC_{i,j}$, $i=\{1, 2, ..., c\}$ et $j=\{1, 2, ..., nc_i\}$
- La charge journalière maximale des agents h_{max}
- Les disponibilités de chaque agent : JDi,j, $i=\{1, 2, ..., e\}$ et $j=\{1, 2, ..., n_j\}$

A partir de ces données, on construit une liste G_k des tâches de tous les départements/ateliers confondus de façon que chaque tâche de cette liste occupe une seule période de temps. *(Par exemple, une tâche de 3 périodes est divisée en 3 tâches d'une période chacune)*

4.2 Codage

Nous proposons le codage suivant :

Un chromosome ou un individu est une matrice de dimension *(e * t)* avec e est le nombre d'agents et t le nombre de périodes tel que : $t=n_j*n_h$ *(le nombre de jours de travail par semaine multiplié par le nombre d'heures de travail par jours)* et dont les gènes sont les éléments de la liste G_k

	T_1	T_2	T_3	...	T_t
Agent 1	G1	G8		...	
Agent 2	G2	G9		...	
Agent 3	G3	Gk		...	
...	...	...	...	...	...
Agent e				...	

Figure 02 : Représentation chromosomique d'un individu

4.3 Population initiale

La population initiale est formée par *n* individus qui sont créés d'une manière aléatoire. Ce nombre reste constant.

Chaque individu est une matrice de dimension *(e * t)* alors, on a :

$$Xn = \{xi(j,k), \forall i \in [1,n], \forall j \in [1,e], \forall k \in [1,t]\} \quad (2)$$

Algorithme pour générer la population initiale :

```
01 Pour chaque individu
02      Répéter
03          Choisir aléatoirement un Agent Ei
04          Si l'Agent Ei est déjà traité
05              Retourner à la ligne 03
06          Sinon
07              Pour chaque Tâche Gj de Ei
08                  Choisir une période Tk
09                  Si Tk est libre
10                      Placer la Tâche Gj dans Tk
11                  Sinon
12                      Retourner à la ligne 08
13                  Fin Si
14              Fin Pour
15          Fin Si
16      Jusqu'à ce que tous les agents soient traités
17 Fin Pour
```

4.4 Fonction d'adaptation

La valeur retournée par cette fonction est calculée selon la satisfaction des différentes contraintes. Ces contraintes peuvent variées d'un problème à l'autre. La fonction d'adaptation est représentée mathématiquement par :

$$Minimiser \sum_{i=0}^{nc} fi \quad (3)$$

Avec :

- *fi* : fonction d'évaluation de la contrainte *i*
- *nc* : nombre de contraintes

4.5 Sélection

Parmi les différentes méthodes existées, nous choisissons la sélection par roulette dans laquelle les individus sont sélectionnés proportionnellement à leur performance (fonction d'adaptation).

Algorithme de Sélection par roulette

01 Calculer la somme *S1* de toutes les fonctions d'évaluation d'une population
02 Générer un nombre *r* entre 0 et *S1*
03 Calculer ensuite une somme *S2* des évaluations en s'arrêtant dès que *r* est dépassé
04 Sélectionner le dernier individu dont la fonction d'évaluation vient d'être ajoutée.

4.6 Croisement

Le croisement consiste à choisir au hasard un point de croisement pour chaque couple. On échange ensuite les deux sous chaînes terminales de chacun des deux chromosomes parents ce qui nous produit deux nouveaux chromosomes enfants. Nous adoptons le croisement en un point.

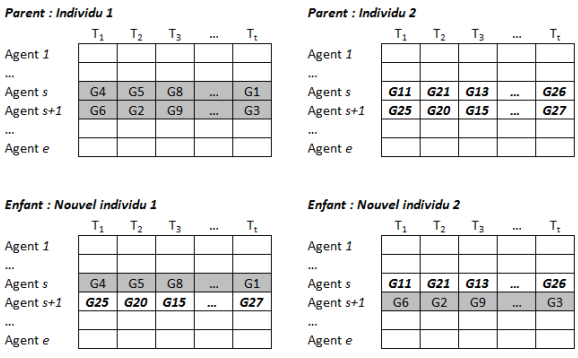

Figure 03 : Illustration de l'opérateur de croisement

4.7 Mutation

Une mutation est l'inversion d'un bit aléatoire dans un chromosome et pour notre cas la mutation consiste à inter-changer deux tâches d'un même agent d'un individu. Nous proposons que l'opérateur de mutation soit applicable seulement sur les individus issus du croisement**.** On choisit au hasard deux éléments d'une ligne et on les permute.

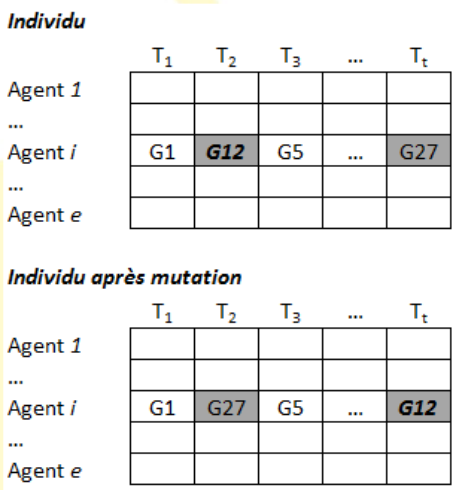

Figure 04 : Illustration de l'opérateur de mutation

Algorithme de mutation :

01 Choisir aléatoirement un nombre *v1* entre 1 et *(e * t)*
02 Choisir aléatoirement un autre nombre *v2* entre 1 et *(e * t)*
03 Si *(v1 = v2)* alors
04 Choisir aléatoirement une ligne *i*
05 Choisir aléatoirement deux tranches d'heure *T1* et *T2* de la ligne *i*
06 Inter-changer *gi(T1)* et *gi(T2)*
07 Fin Si.

4.8 Remplacement

Nous adoptons le remplacement élitiste qui consiste à garder l'individu possédant les meilleures performances d'une génération à la génération suivante.

4.9 Critère d'arrêt

Nous proposons qu'un nombre de générations soit à préciser comme critère d'arrêt. A défaut de cela, c'est la convergence de la population qui détermine l'arrêt de l'algorithme.

5. Application : Domaine pédagogique

Pour les établissements éducatifs, la planification est un travail très important et difficile à réaliser. C'est un problème qui fait assigner quelques évènements dans un nombre limité de périodes. Les entités impliquées sont les enseignants, les cours, les classes et les intervalles de temps.

La construction d'un emploi du temps signifie attribuer pour les triplets enseignant-cours-classe les intervalles de temps que le processus d'enseignement aura lieu. Ceci doit être fait d'une manière que le plus grand nombre de contraintes possibles soient satisfaites.

Ces contraintes peuvent inclure à la fois des contraintes dures qui doivent être respectées, et des contraintes souples utilisées pour évaluer la qualité de l'emploi du temps.

5.1 Données et contraintes

Avant d'exécuter le processus d'élaboration d'un emploi du temps, il faut récolter les données et les différentes contraintes. Ces données concernent : les enseignants, les classes, les cours et le temps.

Quant aux contraintes, nous en avons considéré huit dont 5 sont des contraintes dures et 3 contraintes souples [6].

a. Contraintes dures :

1) Tous les cours sont figurés dans chaque solution.

$$f_1 = \sum_{i=1}^{ng} CHi \qquad (4)$$

$$CHi = \begin{cases} \frac{1}{ng}, & si\ gi\ est\ absent \\ 0, & sinon \end{cases}$$

2. Aucun enseignant ne peut être affecté à deux séances de cours différentes à la même période.

$$f_2 = \sum_{i=1}^{e} \sum_{j=1}^{t} PRij \qquad (5)$$

$$PRij = \begin{cases} \frac{1}{e * t}, & si\ le\ prof\ i\ a\ deux \\ & classes\ au\ temps\ j \\ 0, & sinon \end{cases}$$

3. Les heures de cours de tous les professeurs sont en correspondance avec leurs disponibilités.

$$f_3 = \sum_{i=1}^{e} DPi \qquad (6)$$

$$DPi = \begin{cases} \frac{1}{e}, & si\ le\ prof\ i\ n'est\ pas \\ & disponoble \\ 0, & sinon \end{cases}$$

4. Chaque enseignant ne doit pas dépasser la charge journalière maximale autorisée.

$$f_4 = \sum_{i=1}^{e}\sum_{j=1}^{nj} CHij \quad (7)$$

$$CHij = \begin{cases} \frac{1}{e*nj}, & si\ le\ prof\ i\ enseigne\ plus \\ 0, & sinon \end{cases}$$

5. Aucune classe ne peut rencontrer deux professeurs à la même période.

$$f_5 = \sum_{i=1}^{c}\sum_{j=1}^{t} LCij \quad (8)$$

$$LCij = \begin{cases} \frac{1}{c*t}, & si\ la\ classe\ i\ a\ deux\ classes\ au\ temps\ j \\ 0, & sinon \end{cases}$$

b. Les contraintes souples :

1. Les heures d'enseignement de toutes les classes doivent commencer à partir de la première période de la journée.

$$f_6 = \sum_{i=1}^{c}\sum_{j=1}^{nj} HRij \quad (9)$$

$$HRij = \begin{cases} \frac{1}{c*nj}, & si\ la\ classe\ i\ ne\ commence\ pas\ à\ l'heure\ au\ jours\ j \\ 0, & sinon \end{cases}$$

2. La succession des cours qui nécessitent plus d'une période ne doit dépasser le nombre maximal de périodes successives autorisées

$$f_7 = \sum_{i=1}^{ng} SCi \quad (10)$$

$$SCi = \begin{cases} \frac{1}{ng}, & si\ TCxi > mp \\ 0, & sinon \end{cases}$$

3. Eviter, pour toutes les classes, des pertes de temps par de trop longs espacements entre deux séances de cours d'une même journée.

$$f_8 = \sum_{i=1}^{c}\sum_{j=1}^{nj} TRij \quad (11)$$

$$TRij = \begin{cases} \frac{1}{c*nj}, & si\ la\ classe\ i\ a\ trop\ d'heures\ creuses\ au\ jours\ j \\ 0, & sinon \end{cases}$$

5.2 La fonction d'adaptation

Nous avons huit contraintes, alors la fonction d'adaptation s'écrit :

$$Minimiser \sum_{i=1}^{ng} CHi + \sum_{i=1}^{e}\sum_{j=1}^{t} PRij + \sum_{i=1}^{e} DPi + \sum_{i=1}^{e}\sum_{j=1}^{nj} CHij + \sum_{i=1}^{c}\sum_{j=1}^{t} LCij + \sum_{i=1}^{c}\sum_{j=1}^{nj} HRij + \sum_{i=1}^{ng} SCi + \sum_{i=1}^{c}\sum_{j=1}^{nj} TRij \quad (12)$$

5.3 Résultats

a. Complexité Algorithmique

Puisque le problème de planification est un problème de classe NP difficile, alors regardons la complexité de notre algorithme. On a l'outil mathématique O(n) qui donne une information sur le temps d'exécution d'un algorithme.

La complexité temporelle de notre algorithme est de la forme $O(n^p)$ **(Figure 05)**. Nous avons donc un algorithme polynomial : efficace et exécutable en temps raisonnable.

Etape	Action	Complexité Algorithmique
1	Entrée des données	$O(n^2)$
2	Génération de la population initiale	$O(n^2)$
3	Fonction d'adaptation	$O(n^3)$
4	Sélection	$O(n^2)$
5	Croisement et mutation	$O(n^3)$
6	Remplacement	$O(n^2)$

Alors :

$$O(AG) = O(n^2) + O(n^2) + O(n^3) + O(n^2) + O(n^3) + O(n^2)$$
$$O(AG) = O(n^3)$$

Figure 05 : Complexité temporelle de notre algorithme

b. Tendance vers la solution optimale

Le deuxième point à prendre en compte c'est la convergence vers une solution optimale. Pour cela, nous allons présenter une simulation basée sur la fonction d'adaptation.

Car nous retenons huit contraintes, la valeur maximale que la fonction d'adaptation peut retourner est égale à 8 ; la minimale vaut 0. L'objectif est de minimiser cette valeur, c'est-à-dire de la faire tendre vers 0.

Nous avons tracé trois courbes qui représentent la fonction d'adaptation au fur et à mesure de la progression de la génération. Ces courbes sont en fonction du nombre de gènes d'un individu *t* et la taille de la population *n*. (**Figure 06**)

Nous avons fixé *t* à *35* et les valeurs de *n* sont respectivement *100, 200, 300*.

Si nous supposons qu'un individu est acceptable d'être la solution si sa fonction d'adaptation est inférieure ou égale à 1, alors cette figure montre que :

- L'algorithme aboutira à une solution quelle que soit la taille de la population (car toutes les courbes tendent vers zéro).
- Cette solution est vite atteinte, par rapport au nombre de générations, si la taille de la population est assez élevée *(car C < B < A).*
- Plus grande est la taille de la population, plus meilleur est le résultat.

Nous concluons donc que pour avoir un résultat satisfaisant, la taille de la population doit être autour de 250. Si le critère d'arrêt est basé sur le nombre de la génération, ce nombre doit être aux environs de 700.

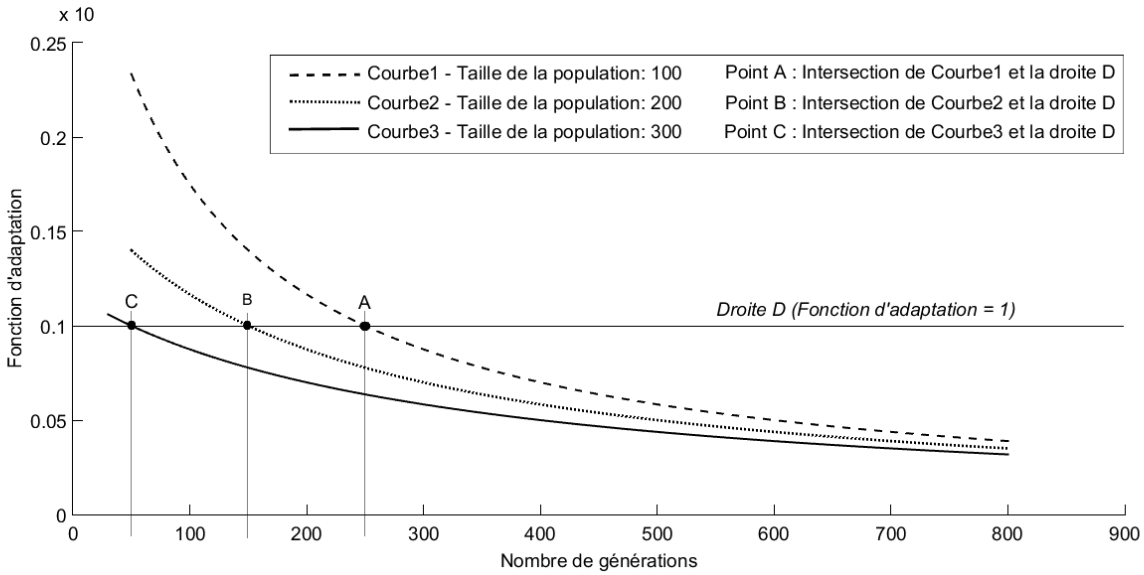

Figure 06 : Courbes simulatrices de la convergence vers la solution optimale

6. Conclusion

Nous avons présenté un modèle mathématique pour le problème de planification d'horaire qui appartient à la classe de NP difficile. La méthode utilisée est celle des algorithmes génétiques. L'efficacité de notre approche réside sur le fait que sa complexité algorithmique est polynomiale. Le processus s'exécute en temps raisonnable. La fonction d'adaptation correspondante tend vers zéro ; cela confirme qu'à une certaine génération, la solution optimale sera obtenue.

Bien que nous avons atteint notre objectif, des nombreuses situations restent à traiter. Nous avons-nous limité dans le cas traitant seulement les données et contraintes des enseignants. On peut élargir le champ d'étude en considérant les autres cas. Les choix des étudiants, par exemple, ou encore les contraintes liées aux salles de classes : nombre, type, capacité et disponibilité. On peut aussi améliorer la formulation de la fonction d'adaptation en y ajoutant des coefficients. De cette façon, chaque contrainte est évaluée selon son degré d'importance. Les questions qui restent à répondre sont alors : Est-ce qu'on peut trouver une approche algorithmique à complexité polynomiale traitant tous ces cas additionnels tout en considérant les cas précédents ? Est-ce que les algorithmes génétiques pourront encore donner des solutions satisfaisantes ?

Références

[1] Liebchen Christian. The first optimized railway timetable in practice. In: Transportation Science, Vol. 42, No. 4, Focused issue on rail transportation, pages 420-435, 2008.

[2] Berger J and Barkaoui M. A new hybrid genetic algorithm for the capacitated vehicle routing problem. In: The Journal of the Operational Research Society, Vol. 54, No. 12, pages 1254_1262, 2003.

[3] P. Fouilhoux. Méthodes heuristiques en Optimisation Combinatoire. Université Paris 6, 2010.

[4] P. Lara-Velázquez, R. López-Bracho, J. Ramírez-Rodríguez, and J. Yáñez. A model for timetabling problems with period spread constraints. The Journal of the Operational Research Society, Vol. 62, No. 1, pages 217-222, 2011.

[5] R Santiago-Mozos, S Salcedo-Sanz, M De Prado-Cumplido, and C Bousono-Calzon. A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a spanish university. Comput Opns Res 32, pages 1161-1116, 2005.

[6] E.K. Burke, B.L. MacCarthy, S. Petrovic, and R. Qu. Knowledge discovery in a hyperheuristic for course timetabling using case-based reasoning. Proc. 4th Internat. Conf. Practice Theory Automated Timetabling (PATAT 2002), (E. Burke, P. De Causmaecker). Lecture Notes in Computer Science, Vol. 2740. Springer, Heidelberg, Germany, pages 276-287, 2003.