République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la RechercheScientifique

Université 20 Août 1955SKIKDA

Faculté des Sciences

Département d'informatique

Mémoire de fin d'étude pour l'obtention du diplôme de Master en Informatique.

Thème

Normalisation des Entrées Et Evaluation D'une solution basée Fourmis pour le problème d'Emploi du Temps des Cours Universitaires.

Réalisé par :

FOUGHALI Monia

FOUGHALI Saida

Encadré Par :

Dr.ZEGHIDA Djamel.

Année Universitaire: 2021-2022.

Abstract

Course timetabling is one of the most important activities faced by any educational institution. Furthermore, the course timetabling process is time-consuming and tiresome as it needs to be prepared for each regular semester. This paper aims to apply the Ant Colony Optimization (ACO) method to solve the course timetabling problem. This approach is to optimize the properties of the course requirement and minimize various conflicts for the time slot assignation. This method is based on the life of the ant colony in generating automatic timetabling according to the properties (pheromones) such as time, student, lecturer and room, besides satisfying the constraints. The implementation of this method is to find an effective and better solution for university course timetabling. The result and performance evaluation is used to determine whether it is reliable in providing the feasible timetable.

الملخص

يعد الجدول الزمني للدورة من أهم الأنشطة التي تواجهها أي مؤسسة تعليمية علاوة على ذلك، فإن عملية إعداد الجداول الزمنية للدورة التدريبية تستغرق وقتًا طويلاً ومرهقة حيث يجب إعدادها لكل فصل دراسي عادي. تهدف هذه الورقة إلى تطبيق طريقة تحسين مستعمرة النمل (ACO) لحل مشكلة الجدول الزمني للدورة. هذا النهج هو تحسين خصائص متطلبات الدورة وتقليل التعارضات المختلفة لتخصيص الفترة الزمنية. تعتمد هذه الطريقة على حياة مستعمرة النمل في إنشاء جداول زمنية تلقائية وفقًا للخصائص (الفيرومونات) مثل الوقت والطالب والمحاضر والغرفة بالإضافة إلى تلبية القيود. تنفيذ هذه الطريقة هو إيجاد حل فعال وأفضل للجدول الزمني للدورات الجامعية. يتم استخدام تقييم النتائج والأداء لتحديد ما إذا كان يمكن الاعتماد عليه في توفير الجدول الزمني المناسب.

DÉDICACES

Nous dédions ce modeste travail,

A l'âme de mon père

A celle qui nous soutient toujours par son amour et ses prières

à celle qui mérite tous les remerciements, gratitude et appréciation,

à la plus chère, notre mère

A ma perle, ma petite fille Nour

Au cher petit Mohamed Wassim

A la fille à naître dont nous ne connaissons pas encore le nom

Nos très chères sœurs.

Nos très chers frères

A tous nos amis et nos collègues.

Monia & Saïda

REMERCIEMENTS

Nous tenons tout d'abord à remercier Allah le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.

Nous remercions Dr : ZEGHIDA Djamel, pour ses précieux conseils et son aide durant toute la période du travail.

Nous remercions également tous enseignants durant notre parcours académique.

Ce travail n'aurait pu être réalisé sans le soutien et les encouragements de notre famille que nous remercie tout particulièrement.

Enfin, nous tenons à remercier tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Table Des Matières

CHAPITRE II : Problème Emploi Du Temps Et ACO

CHAPITRE III : Les Mesures De Performance & Benchmarking

CHAPITRE IV : Implémentation & Réalisation

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES EQUATIONS

LISTE DES TABLEAUX

INTRODUCTION

INTRODUCTION GENERALE :

L'emploi du temps dans les établissements d'enseignement est une activité importante pour programmer les cours ou les examens, qui doivent être entièrement répartis dans des plages horaires appropriées pour les étudiants, les enseignants et les salles soumises à des contraintes. L'horaire des cours est un problème commun à toutes les communautés universitaires et est souvent tenté par les chercheurs. Bien que divers travaux scientifiques et commerciaux aient été réalisés dans ce domaine, dans de nombreux établissements d'enseignement, l'horaire est encore programmé manuellement et même si l'ordinateur est fréquemment utilisé, il suffit de présenter des données ou de vérifier la validation des contraintes. Il est extrêmement difficile de résoudre un grand problème d'horaire de cours avec une approche manuelle qui peut nécessiter qu'un groupe de personnel académique travaille pendant plusieurs jours.

Ainsi, le problème d'horaire des cours universitaires est considéré comme un problème difficile polynomial (NP) non déterministe, ce qui signifie que la complexité de la quantité de calcul sur le temps requis augmente de façon exponentielle avec la taille du problème. Les problèmes d'horaires universitaires sont en fait différents les uns des autres selon le type d'établissement d'enseignement, les entités programmées et les contraintes impliquées L'idée principale de ce problème est d'affecter un ensemble d'événements dans un nombre limité d'intervalles de temps sous réserve de satisfaire un certain nombre de contraintes. L'ensemble des contraintes est généralement divisé en deux types particuliers qui sont les contraintes dures et souples. Par conséquent, l'objectif du problème est de satisfaire les contraintes dures et de minimiser les violations des contraintes souples. Il est donc nécessaire d'utiliser une approche efficace pour produire un horaire qui satisfait aux contraintes.

Il y a eu de nombreuses recherches sur la résolution des problèmes d'horaires de cours et d'examens .De nos jours, avec une meilleure technologie informatique, diverses méthodes sont proposées pour produire une meilleure solution du processus d'établissement des horaires. Cet article se concentre sur les problèmes d'horaires de cours.

Les algorithmes d'optimisation méta-heuristiques ou d'approximation deviennent de plus en plus efficaces pour résoudre les problèmes d'horaires de cours tels que l'algorithme génétique (GA), la recherche tabou, le recuit simulé et

l'optimisation des colonies de fourmis (ACO). Ces algorithmes peuvent produire des solutions de haute qualité mais ne garantissent pas l'optimalité. Ces dernières années, ACO a été largement étudié pour résoudre ce problème. Il a été repris par certains scientifiques et mathématiciens et a été exploité et répandu au début des années 90. ACO s'inspire du comportement stigmergique des fourmis qui simule un ensemble d'agents qui travaillent ensemble pour trouver des solutions aux problèmes d'optimisation en s'appuyant sur une communication simple.

À la lumière de ce qui précède, ce travail traite de l'utilisation de l'optimisation des colonies de fourmis dans le traitement du problème d'horaire des cours en fonction des algorithmes appropriés appliqués.

CHAPITRE I

OPTIMISATION COMBINATOIRE ET MÉTHODES DE RÉSOLUTION

1 INTRODUCTION :

La vie courante nous offre un vaste éventail de problèmes où on se trouve en face d'un ensemble important de solutions potentielles dans lequel il s'agit de trouver la meilleure solution qui soit. Le nombre important de ces choix rend impossible leurs parcours exhaustif en vue de recenser le choix le plus adéquat.

L'objectif de ce chapitre est de présenter une étude sur les méthodes appliquées pour la résolution de ces problèmes tout en formulant une analyse critique quant à leur efficacité. Notre attention sera particulièrement portée sur les problèmes d'emploi du temps.

2 PROBLEMES Combinatoires :

Un problème combinatoire est un problème où il s'agit de trouver la meilleure combinaison possible de solutions. Un tel problème peut être soit un problème de décision, un problème de recherche ou un problème d'optimisation, selon la question à la quelle on est censé répondre. [Tro, 06]

2.1 Problème de décision :

Un problème de décision est un problème où la résolution se limite à la réponse par « oui » ou par « non » à la question de savoir s'il existe une solution au problème. Par conséquent, il n'est pas nécessaire de trouver la solution proprement dite. [Tro, 06]

2.2 Problème de recherche :

Dans ce cas précis, la résolution du problème ne s'arrête plus au point de savoir si le problème admet ou non une solution. L'algorithme doit être en mesure de fournir la solution si celle ci existe. Par conséquent, tout problème de décision peut être étendu à un problème de recherche. [Tro, 06]

2.3 Problème d'optimisation :

Un problème d'optimisation est obtenu à partir d'un problème de recherche en associant à chaque solution une valeur. Il consiste à trouver parmi un ensemble de solutions potentielles celle qui répond le mieux à certains critères décrits sous forme d'une fonction objectif à maximiser ou minimiser c'est à dire on cherchera une solution de valeur optimale, minimale, si on minimise la fonction objectif, et maximale, si on la maximise. [Tro, 06]

3 Optimisation combinatoire :

Un problème d'optimisation combinatoire est un problème de recherche qui consiste à explorer un espace contenant l'ensemble de toutes les solutions potentielles réalisables, dans le but de trouver la solution optimale, sinon la plus proche possible de l'optimum permettant de minimiser ou maximiser une fonction dite fonction objective. [Kher, Oub, 17].

C'est-à-dire : domaine qui étudie les problèmes de la forme :

$$\boldsymbol{Max_{x \in X}\, f(x).}$$

Où $\boldsymbol{x}$ est un ensemble fini.

3.1 Définition :

Une instance I d'un problème est l'état où il peut se trouver. Celle-ci est caractérisée par la taille de ce problème ainsi que les relations liant ses éléments. [Kher, Oub, 17]

Remarque : La même définition peut se donner en utilisant le maximum sachant que :

$$\boldsymbol{Max}_{\,s \in S}\;\; \boldsymbol{f(s) = -Min}_{\,s \in S} \boldsymbol{-f(s)}$$

3.2 Formulation mathématique des problèmes d'optimisation

Les problèmes d'optimisation combinatoire peuvent être formulés comme suit:

3.2.1 Problèmes mono-objectifs :

Un problème d'optimisation est généralement formulé comme un problème de minimisation Ou de maximisation, et s'écrit sous la forme suivante:

$$\begin{cases} min\ f\ x\ ,\ telque \\ gi\ (x\ ) \leq 0,\ i = 1, \dots\ , m \\ hj(\ x\ ) = 0,\ j = 1, \dots\ , p \\ x \in S \subset Rn \end{cases} \qquad \text{(I.1)}$$

Equation I.1 : La Formule d'un PO

Où f est la fonction à minimiser, appelée fonction coût ou fonction objectif, x représente le vecteur des variables d'optimisation, g_i sont les contraintes d'inégalité et h_j les contraintes d'égalité, et S est l'espace des variables (appelé aussi espace de

recherche). S indique quel type de variables sont considérées : réelles, entières, mixtes (réelles et entières dans un même problème), discrètes, bornées, etc.

Un point x_A est appelé un point admissible si $x_A \in S$ et si

Les contraintes d'optimisation sont satisfaites :

$$g_i(x_A) \leq 0 \text{ , } i = 1,\dots, m \text{ et } h_j(x_A) = 0 \text{ , } j = 1,\dots, p.$$

S : Espace de recherche, de configurations. [Bech, Boug, 21]

3.2.2 Problèmes multi-objectifs

Un problème d'optimisation combinatoire peut avoir plusieurs fonctions objectif, il s'agit alors d'un problème d'optimisation multicritère ou multi-objectif.
D'un point de vue mathématique, il est représenté, dans le cas où le vecteur F regroupe k fonctions objectif, de la façon suivante:

$$\begin{cases} min\ f\ x\ , telque \\ gi\ (x\ ) \leq 0, i = 1, \dots, m \\ hj(\ x\ ) = 0, j = 1, \dots, p \\ x \in S \subset Rn \end{cases}$$

De ce fait, résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points suivants:

- ✓ La définition de l'ensemble des solutions réalisables.
- ✓ L'expression de l'objectif à optimiser.
- ✓ Le choix de la méthode d'optimisation (exacte ou approchée) à utiliser.

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution [Bech, Boug, 21].

Voisinage

Soit x une solution, on dit que x^* est une solution voisine de x, si on peut obtenir x^*en modifiant légèrement x.

Le voisinage V (x) de x est l'ensemble des solutions voisines de x [Bech , Boug, 21].

Définition : On dit que :

- ➢ x^*est un minimum local si $f(x^*) \leq f(x)\ \forall\ x \in V(x^*)$.
- ➢ x^*est un minimum global si $f\ (x^*) \leq f(x)\ \forall\ x \in D_f$.
- ➢ x^* est un maximum local si $f\ (x^*) \geq f(x)\ \forall\ x \in V\ (x^*)$.
- ➢ x^*est un maximum global si $f(x^*) \geq f(x)\ \forall\ x \in D_f$.

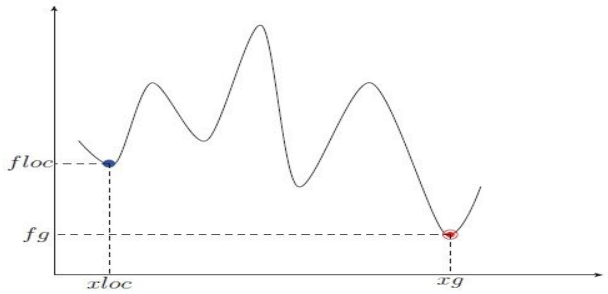


Figure I.1: Optimum global vs optimum local [Hach,13]

4 Différentes classes de complexité :

4.1 La classe P :

La classe P contient tous les problèmes relativement faciles c'est à dire ceux pour lesquels on connaît des algorithmes efficaces. Plus formellement, ce sont les problèmes pour lesquels on peut construire une machine déterministe (e.g. une machine de Turing[1]) dont le temps d'exécution est de complexité polynomiale (le sigle P signifie « Polynomial time ») [Tro, 06].

4.2 La classe NP :

Les problèmes de la classe NP sont ceux pour lesquels on peut construire une machine de Turing non déterministe dont le temps d'exécution est de complexité polynomiale (le sigle NP provient de « Non deterministic Polynomial time ») (et non de « Non Polynomial).

Contrairement aux machines déterministes qui exécutent une séquence d'instructions bien déterminée, les machines non déterministes ont la remarquable capacité de toujours choisir la meilleure séquence d'instructions qui mène à la bonne réponse lorsque celle-ci existe.

Ce concept abstrait est en fait la base de toute la théorie de la NP-complétude. [Tro, 06].

[1] Une machine de Turing est constituée d'un ensemble fini de bandes composées d'un nombre infini de cellules. Chaque cellule, si elle n'est pas vide, contient un symbole représentant une information nécessaire au calcul. Chaque bande ne comporte qu'un nombre fini de cellules non vides. Sur chacune des bandes une tête de lecture-écriture se déplace de cellule en cellule faisant ainsi évoluer les informations contenues dans les bandes et le comportement de la est entièrement décrit par une table finie représentant les actions possibles des têtes de lecture-écriture.

4.3 La classe NP-complets :

Parmi l'ensemble des problèmes appartenant à NP, il en existe un sous ensemble qui contient les problèmes les plus difficiles : on les appelle les problèmes NP complets. Un problème NP-complets possède la priorité que tout problème dans NP peut être transformé (réduit) en celui-ci en temps polynomial. C'est à dire qu'un problème est NP-complets quand tous les problèmes appartenant à NP lui sont réductibles. Si on trouve un algorithme polynomial pour un problème NP-complets, on trouve alors automatiquement une résolution polynomiale de tous les problèmes de la classe NP. [Tro, 06].

4.4 La classe NP-difficiles :

Un problème est NP-difficile s'il est plus difficile qu'un problème NP-complet, c'est à dire s'il existe un problème NP-complet se réduisant à ce problème par la réduction de Turing.

Ceci explique pourquoi, lors de l'étude d'un nouveau problème, on commence par chercher à classer ce problème. Si l'on parvient à montrer qu'il est polynomial, le problème sera résolu. Si par contre, on parvient à montrer qu'il est NP-complet, la recherche d'un algorithme exact pour résoudre un tel problème ne sera pas de première priorité, et il sera approprié de se concentrer sur des méthodes heuristiques que la plupart des spécialistes de l'optimisation combinatoire ont orienté leurs recherches pour les développer.

Une méthode heuristique est souvent définie comme une procédure exploitant au mieux la structure du problème considéré, dans le but de trouver une solution de qualité raisonnable en un temps de calcul aussi faible que possible [Tro, 06].

5 Les grandes classes des problèmes d'optimisation combinatoire :

Les problèmes en optimisation combinatoire sont classés en fonction de leur modélisation. C'est-à-dire de la façon dont ils sont exprimés en termes mathématiques. Donc des problèmes différents dans leur forme peuvent appartenir à une même classe. De ce faite, tout problème d'optimisation combinatoire appartient à une classe des classes suivantes : [Kher, Oub, 17]

- Les problèmes de sac à dos.
- Les problèmes de tournées.
- Les problèmes d'affectation.
- Les problèmes d'ordonnancement.

- Les problèmes de flot.

5.1 Le problème du sac-à-dos :

Considérons n objets, notés $i = 1,\ldots, n$ apportant chacun une utilité u mais possédant un poids p_i . On veut ranger ces objets dans un « sac » de capacité c. Le problème de sac-à-dos (knapsack) consiste à choisir les objets à prendre parmi les n objets de manière à avoir une utilité maximale et respecter la contrainte de la capacité, c, à ne pas dépasser.

La formulation PLNE du problème de sac-à-dos est très simple. On utilise pour chaque objet $i \in 1, \ldots, n$, une variable binaire x_i correspond à 1 si l'objet i est pris 0 sinon. [Kher, Oub, 17]

5.2 Le problème du voyageur de commerce (PVC) :

Un voyageur de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer une et une seule fois dans chaque ville pour finalement revenir à son point de départ, ceci en minimisant la longueur du chemin parcouru. Etant donné un graphe $G=(X, U)$ dans lequel l'ensemble X des sommets représentent les villes à visiter, ainsi que la ville de départ de la tournée et U, l'ensemble des arcs de G, représentent les parcours possibles entre les villes. A tout arc $(i, j) \in U$, on associe la distance de parcours $d_{i,j}$ de la ville i à la ville j. la longueur d'un chemin dans G est la somme de distances associées aux arcs de ce chemin. Le PVC se ramène alors à la recherche d'un circuit hamiltonien de longueur minimale dans G. [Kher, Oub, 17]

5.3 Problème d'ordonnancement :

Le problème d'ordonnancement consiste à ordonnancer un ensemble de tâches indépendantes sur m machines non reliées, le critère à optimiser étant la durée totale, t, de l'ordonnancement.

Dans le contexte non relié, chaque tâche i a une durée d'exécution dépendant de la machine j sur laquelle elle est effectuée. Ainsi la durée de la tâche i sur la machine j est p_{ij} [Bech, Boug, 21].

5.4 Le problème d'affectation :

Nous entendons par problème d'affectation le problème qui consiste à associer chaque élément d'un ensemble de N objets à un seul élément d'un autre ensemble de (avec $N \geq M$) M objets avec un coût minimal. Le problème d'affectation se présente

fréquemment en recherche opérationnelle. Il consiste à réaliser une bijection des éléments *i* d'un ensemble *I* sur des éléments, d'un ensemble *J*, de même cardinalité, de telle manière qu'une certaine fonction de coût, dépendant du choix des couples *(i, j)* soit minimale.

Lorsque cette fonction de coût est linéaire, c'est un problème classique et sa solution est donnée par un algorithme polynomial.) Pourtant, il existe des problèmes appartenant à de nombreux domaines, aussi variés que l'électronique, économie, l'informatique, etc., pour lesquels la fonction de coût est quadratique. Le problème d'affectation s'appelle alors « affectation quadratique ». C'est un problème NP-Complet et donc beaucoup plus difficile à résoudre que l'affectation linéaire. [Kher, Oub, 17]

Considérons l'exemple suivant d'affectation, qui est le sujet de notre mémoire :

Problème d'emploi du temps :

Dans les établissements scolaires, chaque année l'administration est face à un problème, elle essaye de planifier un certain emploi de temps en respectant certaines contraintes telles que :

- Chaque enseignant doit être affecté à une seule salle pour une certaine durée (exemple : une heure) et le même enseignant ne peut être affecté à deux salles en même temps.

- Chaque salle est occupée par un seul enseignant (deux enseignants ne peuvent pas occuper la même salle en même temps).

Le problème à résoudre consiste à concilier toutes ces contraintes pour proposer un emploi du temps sur une certaine durée (un semestre). [Kher, Oub, 17]

En général, on caractérise deux types d'affectation : affectation simple et affectation généralisée.

6 Les méthodes de résolution en optimisation combinatoire :

Elles sont schématisées comme suit :

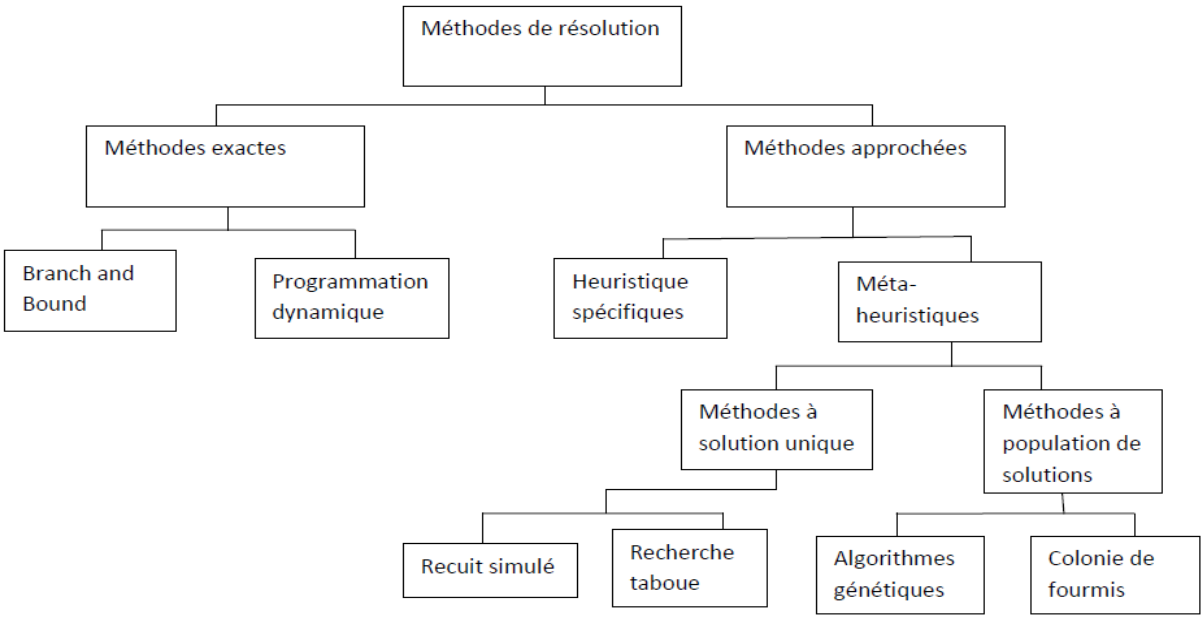


Figure I.2 : Un schéma montrant les différentes méthodes de résolution en optimisation Combinatoire [Kher, Oub, 17]

La majorité des problèmes d'optimisation combinatoire, sont des problèmes NP difficiles et donc ne possèdent pas à ce jour un algorithme efficace, i.e ; de complexité polynomiale, valable pour toutes les données. Ceci a motivé les chercheurs à développer de nombreuses méthodes de résolution en recherche opérationnelle et en intelligence artificielle. Ces méthodes appartiennent à deux grandes familles :les méthodes complètes et les méthodes approchées.

Nous présentons dans ce qui suit les principales méthodes complètes et heuristiques connues dans la littérature.

6.1 Les approches exactes :

Le principe essentiel d'une méthode exacte consiste généralement à énumérer, souvent de manière implicite, l'ensemble des combinaisons de l'espace de recherche. Parmi les méthodes exactes, nous trouvons les méthodes de séparation et évaluation, dites méthodes de Branch and Bound, et la programmation dynamique.

6.1.1 Branch and bound :

Le Branch and Bound est une technique qui effectue un parcours en profondeur de l'arbre de recherche afin de fournir une ou plusieurs solutions optimales à partir

d'un ensemble de solutions potentielles. A chaque étape de la recherche, correspondant à un nœud de l'arbre de recherche, l'algorithme utilise une fonction Bound pour calculer une borne de l'ensemble des solutions du sous-arbre prenant sa racine à ce nœud. En début de résolution, cette borne est initialisée à une valeur maximale (en cas de minimisation). Si cette évaluation est moins bonne que la meilleure solution trouvée jusqu'à ce niveau de recherche, tout le sous-arbre peut être coupé.

Il importe de souligner que l'efficacité de l'algorithme Branch and Bound dépend étroitement du calcul de la borne utilisée. [Alay, 09]

6.1.2 Programmation dynamique :

La programmation dynamique est une méthode ascendante : On commence d'habitude par les sous problèmes les plus petits et on on remonte vers les sous problèmes de plus en plus difficiles. Elle est utilisée pour les problèmes qui satisfont au principe d'optimalité de Bellman : "Dans une séquence optimale (de décisions ou de choix), chaque sous-séquence doit aussi être optimale". Un exemple de ce type de problème est le plus court chemin entre deux sommets d'un graphe.

L'idée de base est d'éviter de calculer deux fois la même chose, généralement en utilisant une table de résultats déjà calculés, remplie au fur et à mesure qu'on résout les sous problèmes.

Il est à noter que la programmation dynamique est utilisée pour résoudre des problèmes polynomiaux (et non NP-difficiles). [Alay, 09]

6.2 Les méthodes approchées :

Contrairement aux méthodes exactes, les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) en un temps de calcul aussi faible que possible.

Une partie importante des méthodes approchées est désignée sous le terme de Méta-heuristiques.

Plusieurs classifications des méta-heuristiques ont été proposées ; la plupart distinguent globalement deux catégories : les méthodes à base de solution courante unique, qui travaille sur un seul point de l'espace de recherche à un instant donné, appelées méthodes à base de voisinage comme les méthodes de recherche locale (méthode de la descente), de recuit simulé et de recherche taboue, et les méthodes à base de population, qui travaillent sur un ensemble de points de l'espace de

recherche, comme les algorithmes évolutionnaires et les algorithmes de colonies de fourmis. [Kher, Oub, 17]

6.2.1 Heuristiques

En optimisation combinatoire, une heuristique est un algorithme approché qui permet d'identifier en temps polynomial, au moins, une solution réalisable rapide, pas nécessairement optimale. L'usage d'une heuristique est efficace pour calculer une solution approchée d'un problème et ainsi accélérer le processus de résolution exacte. Généralement une heuristique est conçue pour un problème particulier, en s'appuyant sur sa structure propre sans offrir aucune garantie quant à la qualité de la solution calculée.

Les heuristiques peuvent être classées en deux catégories :

- Méthodes constructives qui génèrent des solutions à partir d'une solution initiale en essayant d'en ajouter petit à petit des éléments jusqu'à ce qu'une solution complète soit obtenue.
- Méthodes de fouilles locales qui démarrent avec une solution initialement complète (probablement moins intéressante), et de manière répétitive essaie d'améliorer cette solution en explorant son voisinage. [Kher, Oub, 17].

6.2.2 Méta-heuristiques :

6.2.2.1 Les classes de Méta-heuristiques

En général les méta-heuristiques sont subdivisées en deux grandes familles : les méta-heuristiques de recherche locale et les méta-heuristiques évolutives.

a) Les méta-heuristiques de recherche locale :

Le principe des méta-heuristiques de recherche locale est de modifier les résultats d'un ordonnancement admissible en vue d'améliorer la valeur de la fonction objectif, plutôt que de construire un ordonnancement à partir des données initiales du problème.

b) Les méta-heuristiques évolutives :

Tandis que le principe des méta-heuristiques évolutives est de faire évoluer un ensemble de solutions vers l'optimum cherché. Cette évolution se fait à partir de transformations et de coopérations entre les individus qui représentent individuellement une solution de l'espace total du problème. Parmi ces méthodes,

nous distinguons les algorithmes génétiques, la recherche distribuée et les colonies de fourmis.

6.2.2.2 Les méthodes méta-heuristiques :

Face aux difficultés rencontrées par les heuristiques pour avoir une solution réalisable de bonne qualité pour des problèmes d'optimisation difficiles, les méta-heuristiques ont fait leur apparition. Ces algorithmes permettent généralement d'obtenir une solution de très bonne qualité pour des problèmes issus des domaines de la recherche opérationnelle dont on ne connait pas de méthodes efficaces pour les traiter ou bien quand la résolution du problème nécessite un temps élevé ou une grande mémoire de stockage.

Le rapport entre le temps d'exécution et la qualité de la solution trouvée d'une méta-heuristique reste alors, dans la majorité des cas, très intéressant par rapport aux différents types d'approches de résolution.

Une méta-heuristique peut être adaptée pour différents types de problèmes, tandis qu'une heuristique est utilisée à un problème donné (i.e. : une méta-heuristique combine plusieurs approches heuristiques). [Kher, Oub, 17]

Les méthodes approchées utilisent, durant sa recherche de l'optimum, un certain processus appelé le voisinage pour cela on donne quelques définitions le concernant :

a) Méthodes de trajectoires (voisinage)

- Méthode de Descente (Hill Climbing)

Les Méthodes de Descente sont assez anciennes. Leur succès revient à leur simplicité et à leur rapidité [Papadimitriou, 1976]. La principale caractéristique de la méthode de descente est sa grande simplicité de mise en oeuvre. La plupart du temps, elle ne fait que calculer f(s+i)-f(s), où, i correspond à un déplacement élémentaire, et si cette expression peut se simplifier algébriquement, alors on pourra évaluer très rapidement cette différence.

Le principe consiste à choisir une solution s' dans le voisinage d'une solution s en améliorant la recherche tel que f(s')< f(s).

On peut décider soit d'examiner toutes les solutions de voisinage et prendre la meilleure de toutes (ou prendre la meilleur trouvée), soit d'examiner un sous ensemble de voisinage. [Ala,12]

La méthode de Descente peut être décrite comme suit :

Procédure Descente Simple (solution initiale s)

Répéter :

Choisir s' dans N(s)

Si f(s')<f(s) alors s→ s'

Jusqu'à ce que f(s') >= f(s), ∀ s' ∈ S

Algorithme I.1 : Méthode Descente.

L'ensemble S définit l'ensemble des points pouvant être visités durant la recherche. La structure de voisinage N donne les règles de déplacement dans l'espace de recherche.

La fonction f induit une topologie sur l'espace de recherche.

Une variante consiste à parcourir N(s) et à choisir la première solution s' rencontrée telle que f(s') <f(s) (pour autant qu'une telle solution existe).

- Recuit Simulé (Simulated Annealing)

La méthode du Recuit Simulé est issue d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion, qui le conduit à un état solide, de basse énergie. Il faut abaisser lentement la température, en marquant des paliers suffisamment longs pour que le corps atteigne l'équilibre thermodynamique à chaque palier de température pour les matériaux, cette énergie se manifeste par l'obtention d'une structure régulière, comme dans les cristaux et l'acier.

Par analogie avec le processus physique, la fonction à minimiser est l'énergie du système E, on utilise aussi un paramètre « la température du système T ».

A partir de l'espace de solution, s_0 est choisi aléatoirement, on associe avec cette solution une énergie initiale $E=E_0$ et une température initiale $T=T_0$ élevée.

La solution est modifiée à chaque itération de l'algorithme, ce qui entraine une variation ΔE de l'énergie du système. Si cette variation diminue l'énergie du système (ΔE est négative) on l'accepte, sinon on l'accepte avec une probabilité $e^{-\Delta E\backslash T}$, La température est restée constante.

Cet algorithme commence par s_0, et continue jusqu'à k_{max} (maximum d'étapes) ou jusqu'à un état ayant une énergie e_{max}, l'appel voisin (s) engendre un état aléatoire voisin d'un état s, l'appel aléatoire renvoie une valeur aléatoire dans l'intervalle [0,1]. L'appel temp(r) renvoie la température à utiliser selon la fraction r du temps total déjà dépensé. [Ala,12]

$s \leftarrow s_0$;
$e \leftarrow E(s)$;
$k \leftarrow 0$;
Tantque $k<k_{max}$ **et** $e>e_{max}$
$s \leftarrow voisin(s)$;
$e_n \leftarrow E(s_n)$;
si $e_n<e$ **ou** aléatoire () < $p(e_n\text{-}e, temp\ (k/k_{max}))$ **alors**
$s \leftarrow s_n$;
$e \leftarrow e_n$;
$k \leftarrow k+1$;
Retourner s

Algorithme I.2 : Méthode Recuit Simulé

- La méthode taboue

La recherche Tabou, développée initialement par F. Glover [Glover et Laguna, 1997], est une amélioration des techniques de recherche locale conçue en vue de surmonter le problème des optimums locaux tout en évitant le phénomène de bouclage (possibilité d'accepter une solution même si elle dégrade la fonction coût). Cette propriété lui permet de sortir des creux ou vallées sous-optimales, mais fait apparaître la possibilité de déplacements cycliques. La parade est alors la création d'une liste de déplacements interdits, appelée « Liste Tabou », correspondant aux solutions déjà visitées par l'algorithme. Cet aspect s'apparente à une mémorisation du cheminement effectué dans l'espace de recherche. Afin de ne pas surcharger la mémoire, la liste Tabou ne contient pas la description complète des configurations Tabou mais les déplacements effectués ou mouvements. Un mouvement élémentaire est le passage d'une configuration vers sa voisine sélectionnée [PITIOT ,09].

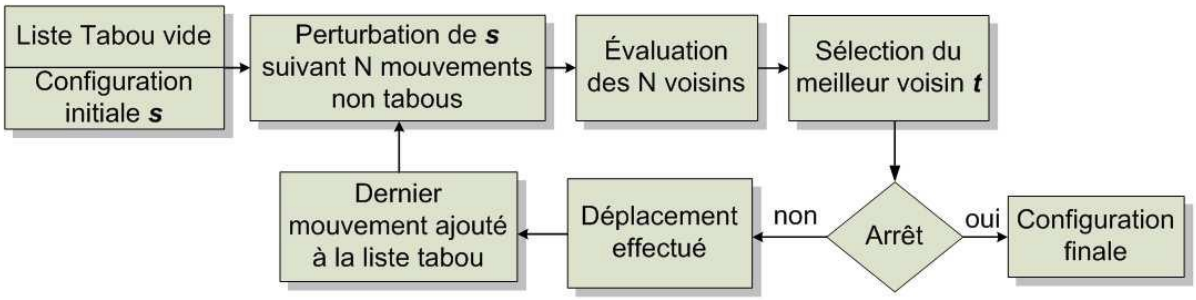


Figure I.3 : Algorithme de recherche tabou

Cette méthode a donné lieu à différentes versions multi-objectif tel que la méthode MOSA et la méthode PASA. Dans la méthode MOSA (*Multiple Objective Simulated Annealing*) [Ulungu et al., 1999], la probabilité d'acceptation d'une solution est évaluée objectif par objectif en utilisant le critère de Metropolis, puis les évaluations sont agrégées en utilisant des coefficients pondérateurs.

La méthode PASA (*Pareto Archived Simulated Annealing*) [Engrand, 1998] utilise un système d'archivage des solutions Pareto-optimales rencontrées qui offre des points de redémarrage intéressants (lancements multiples).

b) Méthodes basées populations :

- Algorithmes Génétiques (Genetic Algorithm) :

Les Algorithmes Génétiques sont des algorithmes d'optimisation qui s'appuient sur des techniques dérivées de la génétique et de l'évolution naturelle : sélection, croisement, mutation. Ils ont été inventés dans les années de 60 [Holland, 1962]. L'auteur Goldberg a étudié les Algorithmes Génétiques et il a enrichit sa théorie par [Goldberg, 1989]: [Ala,12]

- Un individu est lié à un environnement par son code d'ADN.
- Une solution est liée à un problème par son indice de qualité.
- Une bonne solution à un problème donné peut être vue comme un individu Susceptible de survivre dans un environnement donné.

Les éléments suivants sont nécessaires pour utiliser un Algorithme Génétique:

- Le principe du codage d'un élément d'une population : généralement après une phase de modélisation du problème à étudier, on associe à chaque point de l'espace d'états une structure de données. Le succès de l'Algorithme Génétique est lié à la qualité du codage. Il y'a le codage réel (utilisé pour l'optimisation du problème à variables réelles) et le codage binaire qui est le plus utilisé.

➢ La génération d'une population initiale : le choix d'une population initiale est nécessaire pour les générations futures et principalement pour converger rapidement vers un optimum global.

➢ Les opérateurs permettent la diversification de la population au cours des générations et l'exploration d'espace d'états. 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.

➢ Les paramètres de dimensionnement : la taille de la population, le critère d'arrêt, les probabilités d'application des opérateurs de croisement et de mutation.

Un Algorithme Génétique recherche le ou les extrema d'une fonction définie sur un espace de données.

- Générer aléatoirement une population d'individus.
- Pour passer d'une génération k à une génération k+1, on applique les trois opérations pour tous les éléments de la population k :
  - sélectionner les couples de parents p_1, p_2 en fonction de leur adaptation.
  - appliquer l'opérateur de croisement avec une probabilité p_c et générer les couples d'enfants c_1 et c_2, certain éléments sont choisis en fonction de leur adaptation, on applique l'opérateur de mutation avec une probabilité p_m et on génère les individus p'
  -évaluer c_1 et c_2 , p' pour l'insérer dans la nouvelle population.
- Le critère d'arrêt : Soit le nombre de générations est fixé à l'avance ou un temps limité pour trouver une solution où il n'y a pas d'évolution.

Algorithme I.3: (Genetic Algorithm)

- Algorithme de colonie de fourmi (ant colony algorithm) :

Les algorithmes de colonies de fourmis forment une classe du méta heuristique proposée pour des problèmes d'optimisation difficile par Dorigo (M.DORIGO, et al., 1996), Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de piste observés dans les colonies de fourmis.

Une colonie d'agents simples (les fourmis) communiquent indirectement via des modifications dynamiques de leur environnement (les pistes de phéromones) et construisent ainsi une solution à un problème en s'appuyant sur leur expérience collective [Ben, Sou, 19].

c) Les méthodes hybrides :

Le mode d'hybridation qui semble le plus fécond concerne la combinaison entre les méthodes de voisinage et les méthodes évolutives. L'idée essentielle de cette

hybridation consiste à exploiter pleinement la puissance de recherche de méthodes de voisinage et de recombinaison des algorithmes évolutionnaires sur une population de solutions. Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itération ou jusqu'à la découverte d'un ensemble d'optima locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus.

Les algorithmes hybrides sont considérés parmi les méthodes les plus puissantes. Cette puissance réside dans la combinaison des deux principes de recherche fondamentalement différents comme on a vu dans le paragraphe précédent. Le rôle de la méthode de voisinage est d'explorer en profondeur une région donnée de l'espace de recherche alors que la méthode évolutive introduit des règles de conduite générales dans le but de guider la recherche au travers de l'espace de recherche. Dans ce sens, les opérateurs de combinaison ont un effet diversificateur bénéfique à long terme.[Tro, 06].

7 Conclusion :

Dans ce chapitre, nous avons décrit un état de l'art des méthodes d'optimisation basées sur les méta-heuristiques les plus connues. Dans un premier temps nous avons défini les problèmes d'optimisation difficile, l'heuristique et la méta-heuristique et les méthodes de résolution de ces problèmes.

Dans le chapitre suivant, nous allons voir la capacité de résoudre un problème NP-difficile (la recherche d'un sous ensemble d'attributs pertinents) par la méta-heuristique ACO (Ant Colony Optimization).

CHAPITRE II

PROBLÈME EMPLOI DU TEMPS

ET

ACO

1 Introduction :

La génération d'une application qui résoudre le problème d'emploi du temps est parmi les problèmes très complexes, il est classifié dans les cas généraux comme (NP-hard) problèmes, en autre terme, les fonctions nécessaires pour résoudre ce problème doivent augmenter d'une façon exponentielle, ce qui rend difficile a résoudre. Depuis prés de quarante ans, ce type de problème a attiré l'attention de la communauté scientifique de plusieurs disciplines, dont la recherche opérationnelle et l'IA [Sad, Ben, 15].

2 Présentation du problème d'emploi du temps :

2.1 Définition :

L'emploi du temps est un plan représentatif définissant l'affectation d'un ensemble de taches et activités à un ensemble de ressources en se basant sur des périodes de temps bien précises, soumettant à un ensemble de contraintes à respecter [Sad, Ben, 15].

2.2 Problématique de planification horaire (emploi du temps) :

La planification horaire est un processus très complexes qui vise a organisé des activités humaines dans un intervalle du temps (généralement une semaine). La planification horaire est considérée comme un outil à des programmes permettant d'organiser et d'ordonnancer le travail des ressources humaines et affecter pour chaque période du temps sur un horizon donné, de telle manière que les besoins par intervalle soient couverts et que les différentes contraintes soient satisfaites [Tro,06].

Dans le monde pratique, il existe un grand nombre de variantes du problème d'emploi du temps qui diffère les uns des autres selon le type d'établissement impliqué (usine, école, société, … etc.) et selon le type de contraintes. Dans ce cas on va concentrer sur PET universitaire.

3 Problème d'emploi du temps des cours universitaires :

3.1 Introduction :

Dans ce chapitre, nous introduirons les notions liées au problème d'emploi du temps avec un aperçu des méthodes proposées pour la résolution de ce problème.

3.2 Problème d'emploi du temps universitaire :

Dans la vie quotidienne, les universités ont besoin un calendrier de la planification, mais l'élaboration d'un emploi du temps universitaire satisfaisant est considéré comme une taches très difficiles que l'emploi du temps scolaire, car le système des salles est changé périodiquement (Le groupe des étudiants n'a pas eu une salle spécifié) et cela augmente la complexité du ce problème.et avec le progrès réalisé dans les technologies matérielles et logicielles, la communauté scientifique continue à travailler sur le problème [Sad, Ben,15].

Le problème d'emploi du temps universitaire est une instance des problèmes d'ordonnancement cyclique les plus connues dans la littérature, il s'agit d'ordonnancer les tâches (qui possède un caractère cyclique) d'un ensemble d'enseignants en leur allouant un ensemble de salles et en leur fixant leurs dates de début et de fin.

De nos jours, la construction d'un calendrier qui satisfait tous les besoins d'un établissement universitaire est une tâche très importante, mais elle est extrêmement difficile. Avec le progrès réalisé dans les technologies matérielles et logicielles, la communauté scientifique continue à travailler sur ce problème, afin d'élaborer des procédures formelles et automatisées pour élaborer des calendriers efficaces et souhaitables.

Burke et ses collègues notent à cet égard que ce type de problème se divise en deux catégories principales : Les cours et les examens. Différents aspects séparent ces deux catégories. Par exemple, on cherche à regrouper les cours, alors qu'on préfère éloigner les examens les uns des autres le plus possible. Ou encore, un cours peut se tenir à un instant donné dans une salle, alors que plusieurs examens peuvent se tenir en même temps dans la même salle ou un même examen peut être dispatché dans plusieurs salles [Bel et Nou, 15].

Burke et ses collègues notent à cet égard que ce type de problème se divise en deux catégories principales : les cours et les examens. Différents aspects séparent ces deux catégories, Par exemple [Sad, Ben, 15] :

- Emploi du temps des cours :

établir le programme hebdomadaire pour tous les cours d'un ensemble d'université réduisant au minimum que possible les chevauchements des cours ayant les groupes des étudiants communs [Sad et Ben,15].

❖ Emploi du temps des examens :

L'emploi du temps d'un examen est un peu différent par rapport l'emploi du temps des cours car il impose certains contraintes qui ne sont pas prise en compte dans celui des cours. L'emploi du temps des examens possèdent des caractéristiques particulières suivantes :

- Il ya seulement un examen pour chaque module.
- Le nombre des périodes peut changer contrairement au cas du cours.
- plusieurs examens peuvent se tenir en même temps dans la même salle ou un même examen peut être dispatché dans plusieurs salles [Sad, Ben, 15].

3.3 Présentation du problème d'emploi du temps des cours universitaires :

Nous nous sommes concentrés plus précisément sur le problème d'emploi du temps des cours universitaires.

Ce genre de problème est défini comme un ensemble des cours qui se déroulent dans des périodes spécifiés durant cinq ou six jours par la semaine, ces cours ont besoin d'un nombre de salles et des professeurs suffisants pour bien contenir le nombre important des groupes des étudiants.

Et afin d'avoir une meilleure solution à ce problème, il faut prendre en compte l'ensemble des contraintes du problème qui doivent être satisfaites. Ces contraintes sont souvent classées en deux catégories, la première regroupe les contraintes Hard (dures) et la seconde catégorie regroupe des contraintes appelées souvent les contraintes Soft (molles) [Bel, Nou, 15].

a. Les contraintes Hard :

Ce sont des contraintes qui doivent être satisfaites dans n'importe quel environnement, car la violation de ces contraintes peut causer la génération d'une solution inacceptable.

- Deux lectures ne peuvent pas être programmées dans la même salle et dans la même période de temps.
- Les lectures à donner par un même enseignant ne peuvent pas être programmées dans la même période de temps.
- Une salle ne peut être affectée qu'à une seule lecture dans une même période de temps.

- Une séance de lecture en groupe ne peut pas se dérouler dans la même période avec une autre qui n'est pas en groupe pour une même filière ayant un même niveau d'étude.

- Le nombre d'étudiants doit être inférieur ou égale à la capacité de la salle à affecter.

b. Les contraintes Soft :

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

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

- Il faut que l'affectation des salles aux différentes lectures permette de satisfaire au mieux certaines préférences (répartition des séances de façon que tous les jours soient équilibrés en matière de nombre de séances).

3.4 Génération automatique d'un emploi du temps :

La génération automatique d'un emploi du temps est une activité de création, de gestion et de maintenance d'un emploi du temps avec un ordinateur et avec l'intervention minimale de l'être humain en satisfaisant au mieux les ressources humaines, matérielles et temporelles .

La génération automatique d'un emploi du temps nécessite la construction d'un système capable de gérer les ressources temporelles et matérielles, les contraintes imposées et de résoudre les conflits entre ces ressources. Mettre au point un tel système est une tache difficile et factieuse pour l'être humain. Outre la réalisation manuelle d'un emploi du temps est très compliquée, très couteuse [Sad, Ben, 15].

4 Méthodes de résolution du problème d'emploi du temps :

4.1 Problèmes de satisfaction de contraintes :

Le problème d'emploi du temps est un problème défini en termes de contraintes (de temps, d'espaces ou plus généralement de ressources comme d'ailleurs d'autres problèmes tels que :

- Les problèmes de planification et d'ordonnancement : planifier une production, gérer un trafic ferroviaire…

- Les problèmes d'affectation du personnel à des tâches, des entrepôts à des marchandises,… etc.

Ces différents problèmes fortement contraint ayant la particularité commune d'être caractérisés par une très forte combinatoire, peuvent être considérés comme des problèmes de satisfaction de contraintes (**CSP** : **C**onstraint **S**atisfaction **P**roblems) car ces problèmes se formulent aisément en CSP quand ils ne nécessitent que des contraintes fortes.

L'utilisation d'une technique de satisfaction de contraintes est adaptée aux problèmes très contraints où une exploration de l'espace de recherche est envisageable.

Dans ce type de problème, la difficulté est de trouver une solution satisfaisant toutes les contraintes et non de trouver une solution minimisant ou maximisant une fonction, on fait d'ailleurs souvent abstraction de cette fonction en ne tenant compte que des contraintes : on cherche une solution réalisable et non pas la meilleure solution [Tro, 06].

- Qu'est ce qu'un CSP :

Un CSP est défini comme étant un ensemble de contraintes impliquant un ensemble de variables, dont chacune est définie sur son domaine propre. L'objectif consiste à trouver un ensemble de valeurs, choisies dans les domaines susmentionnés, à affecter à ces variables de sorte que toutes les contraintes soient satisfaites [Tro, 06].

- Concepts de base d'un CSP:

Plus formellement, un problème de satisfaction de contraintes est défini par le triplet (X, D, C) tel que :

X = (X1, X2,.....,Xn) est l'ensemble des n variables du problème.

D = (D1, D2,.....,Dn) est un ensemble de n domaines finis dont chacun est associé à une variable de X. C'est à dire le domaine Di est associé à la variable Xi (leurs valeurs possibles).

C = (C1, C2,.....,Cm) est un ensemble de m contraintes. Chaque contrainte Ci est défini par un couple (vi,ri) tel que :

- vi = {Xi1,.....Xini} est un ensemble de ni variables sur lesquelles porte la Contrainte Ci ; ni est appelé arité de la contrainte.
- ri est une relation définie par un sous-ensemble de produits cartésiens Di1x.....xDni des domaines associés aux variables de vi. Elle représente les n-uplets de valeurs autorisées pour ces variables [Tro, 06].

- Emploi du temps et CSP :

Dans [FRA04], l'auteur utilise une approche énumérative pour résoudre le problème de l'emploi du temps. Ce problème d'optimisation d'emploi du temps se défini par six ensembles [Tro, 06] :

Un ensemble Professeurs = {P1, …, PP} ,

Un ensemble Formations = {F1,…,Ff},

Un ensemble Enseignements = {E1,…, Ee},

Un ensemble Salles = {S1,…,Ss},

Un ensemble Créneaux = {C1,…,Cc}, et un ensemble Contraintes contenant l'ensemble des contraintes entre les variables des cinq ensembles précédents. La connaissance du problème permet pour chaque cours de fixer quelques variables. Ainsi pour chaque professeur Pi de l'ensemble Professeurs, est fixé quel enseignement et à quelle formation il le dispensera.

Il reste donc qu'à fixer pour chaque cours la salle et le créneau qui seront utilisés et qui satisfassent les contraintes.

Le CSP proposé se présente de la façon suivante :

V = {P1… PP, F1, Ff, E1,…, Ee, S1,…,Ss, C1,…,Cc}

D = {D(P1) =…= D(Pp) =[1,p], D(F1) =…= D(Ff) = [1,f], D(E1) =…= D(Ee) = [1,e], D(S1) =…=D(Ss) = [1,s], D(C1) =…= D(Cc) = [1,c]}

C= {les contraintes dures plus les contraintes faibles}.

4.2 Techniques de résolution :

4.2.1 Emploi du temps et méthodes à base de population :

L'application des algorithmes génétiques aux problèmes d'emploi du temps est très abondante dans la littérature :

➢ Dans [ROS95], les auteurs introduisent un mécanisme intéressant permettant de guider la recherche. Ce mécanisme permet de détecter les conflits dans une solution.

Pour cela, un score de violation de contraintes est gardé pour chaque gène de chromosome à l'étape d'évaluation des individus. Cette information est exploitée par l'opérateur de mutation, elle sera utilisée pour favoriser l'altération des gènes à score de violations de contraintes élevé.

Les auteurs proposent aussi une amélioration de l'étape d'évaluation visant à réduire son temps de calcul. La délta-évaluation consiste à utiliser l'évaluation des parents afin de calculer la valeur d'adéquation des enfants. Les individus enfants présentent des similarités avec leurs parents, delà seules les modifications sur la fonction objective due aux parties des chromosomes altérés seront examinées dans l'évaluation de l'individu enfant [Ben , Sou,19].

- Dans [ABR92], l'auteur regroupe dans une même liste les cours ayant lieu à la même période. L'objectif de la résolution se limite à éliminer les cas de conflits. Il définit sa fonction objective comme étant la somme des violations des contraintes. Après le croisement d'Abramson qui définit pour chaque période un site de croisement, la liste des cours associés à la même période sera composée des premiers éléments du parent1 suivie des derniers éléments du parent2 [Tro, 06]

4.2.2 Emploi du temps et méthodes de voisinage :

Les problèmes d'emploi du temps peuvent être traités par les méthodes de voisinage, pour cela de nombreux travaux ont été consacrés à la résolution de ce type de problème [Tro,06] :

- Hertz [HER91] décompose le problème de l'emploi du temps en deux sous problèmes. Le premier consiste à planifier les cours de façon à prévenir les conflits sur les salles et les enseignants. Le deuxième, revient à regrouper les étudiants au sein des sections de cours. L'approche tente de prendre en compte l'ensemble de toutes les contraintes possibles (cours successifs, cours distancés,…).

Dans cette approche les deux sous problèmes sont considérés en tant que problèmes d'assignement. Pour le problème de planification des cours il s'agit d'assigner à chaque cours une date de début. Dans le problème de regroupement des étudiants, il s'agit d'assigner à chaque étudiant une section de cours convenable.

- Dans l'approche de Costa [COS94], les regroupements des étudiants dans les classes est préfixé. Le problème consiste alors à assigner une période de début à chaque cours.

Pour générer un emploi du temps faisable les cours sont triés selon leur recevabilité. La recevabilité d'un cours correspond au nombre de périodes en lesquelles il peut être programmé en considérant les contraintes de pré-assignement, de conflit sur les enseignant et sur les salles. Les cours sont alors programmer l'un

après l'autre à commencer par le cours le moins recevable. Le cours est affecté à la période générant le moins de conflits.

- Le voisinage d'un emploi du temps T englobe tous les emplois du temps faisables qui peuvent être engendrés en changeant la période de début d'un seul cours.
- Pour prévenir les cycles lors de la recherche, le couple (x, t1) est introduit dans la liste tabou lorsque la recherche taboue fait passer la recherche de l'emploi du temps T1 à T2 en modifiant la période du cours x de t1 vers t2.

➢ Schaerf [SCH95] a utilisé une méthode tabou pour résoudre ce problème. Il emploi le codage matriciel $M_{j,k}$ qui contient le nom de la classe du professeur j à la période k. les voisinages proposés sont :

- échanger deux cours pour un même professeur
- déplacer un cours à une autre période pour des instances de taille moyenne, les résultats ont été encourageants.

4.2.3 Emploi du temps et méthode hybride :

Citons les travaux de (TEDDY, et al., 2009)qui ont essayé de combiner les algorithmes génétiques avec les méthodes de résolution des C.S.P pour la résolution du problème d'emploi du temps universitaire. Ils se sont basés sur la création de quatre types de chromosomes11 (R1, R2, R3, R4) dont :

- R1 est un phénotype de type C.L.R.S (Course, Lecture, Room, Slot),
- R2 est un génotype de type L.R.S (Lecture, Room, Slot),
- R3 est un génotype de type L.R.S.R.S.F (Lecture, Room, Slot, Room, Slot, Fonction),
- R4 est un génotype de type L.R.S.R.S (Lecture, Room, Slot, Room, Slot).

Ajoutant aussi, une fonction qui permet de mesurer le degré de satisfaction des contraintes et de pénaliser leurs violations [Ben, Sou, 19].

5 Emploi du Temps des Cours universitaires et ACO

5.1 Introduction :

Les colonies de fourmis dans la nature montrent une fascinante capacité à trouver des chemins entre des sources de nourriture et la fourmilière. A l'échelle d'une fourmi, le problème résolu est complexe tel que le plus court chemin. L'intelligence qui permet la résolution du problème est dite collective et est en fait le

résultat émergeant d'un grand nombre d'interactions élémentaires. Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser. Il est ainsi assez simple de simuler leur comportement par des algorithmes informatiques.

5.2 Pourquoi les fourmis ?

Nous présenterons dans notre travail d'étude, en s'inspirant du comportement social des fourmis qui est un comportement collectif où Chaque fourmi a pour priorité le bien être de la communauté.

Chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre. Ce concept est appelée Hétérarchie (s'opposant à la Hiérarchie) [Dréo ,04], chaque individu est aidé par la communauté dans son évolution et en retour il aide au bon fonctionnement de celle-ci.

La colonie est donc autocontrôlée par le biais de mécanismes relativement simples à étudier [Cos, Lou, Mar, 06].

5.3 Les fourmis réelles et l'inspiration biologique :

Les différentes méta-heuristiques ont été inspirées par les travaux des biologistes qui ont longuement été intrigués par le comportement des insectes sociaux en générale (abeilles, termites…) et les fourmis en particulier (1983 Deneubourg Jean Louis Biologiste, étudie le comportement des fourmis). Ces derniers sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre une source de nourriture et leur nid par des moyens très simples dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée Phéromone.

La fourmi ; Au cours de sa progression ; laisse derrière elle une trace de phéromone (une sorte de marquage du chemin, elle fournit une information sur la qualité des chemins empruntés afin d'attirer et guider les fourmis). Ce procédé basé sur le mécanisme de rétroaction positive, assure que pendant le fourragement pour la nourriture, les fourmis utilisent la voie d'accès la plus courte car elle sera le plus imprégnée par la phéromone.

Ce phénomène est appelé autocatalytique. Le temps nécessaire à la stabilisation du système est imputable au fait qu'aucune phéromone n'est présente sur aucune

route au départ, les fourmis ont alors autant de chances de choisir le plus court que le plus long chemin.

On peut citer plusieurs raisons à cette inspiration:

a) l'influence des fourmis sur leur environnement naturel est extrêmement importante. Il a par exemple été montré (qu'elles déplacent plus de terre en forêt tropicale que les vers de terre, ou encore que le poids total des fourmis sur terre est du même ordre de grandeur que le poids des humains de plus, la domination des fourmis est une preuve de leur adaptation à des environnements très variés) ;

b) l'étude des fourmis se fait assez facilement en laboratoire car elles s'adaptent sans trop de difficultés à des environnements différents de leur habitat d'origine ;

c) les fourmis possèdent une gamme de comportements très variés, collectifs ou individuels. [Bech et Bou, 21]

5.4 L'intelligence collective des fourmis :

Malgré que certaines espèces des fourmis aient des capacités individuelles étonnantes telle que des capacités visuelles inhabituelles et des capacités d'apprentissage, mais la plupart des caractéristiques qui nous intéressent sont cependant collectives.

On parle d'intelligence collective quand un groupe social peut résoudre un problème dans un cas où un agent isolé en serait incapable. Cette intelligence est basée sur les processus d'auto-organisation.

L'auto-organisation se parée bien à l'étude des insectes sociaux montrent des comportements collectifs complexes issus de comportements individuels simples. On peut regrouper les processus d'auto-organisation chez les insectes sociaux en quatre groupes tant leur diversité est importante.

– La division du travail et l'organisation des rôles sociaux : à l'intérieur d'une même société, on peut observer différentes catégories spécialisées dans un certain nombre de tâches (la recherche de nourriture, la défense du nid, ...) ;

– L'organisation de l'environnement : la construction du nid est un symbole de l'organisation distribuée des insectes. Le nid est construit sans que les insectes soient dirigés, ils répondent à un certain nombre de stimuli provenant de leur environnement ;

– La reconnaissance inter-individuelle : chaque fourmi est capable d'identifier ses congénères tout en participant elle-même à l'identité de sa colonie (par exemple l'échange d'aliments entre les individus d'une même colonie '*trophalaxie*') ;

– Le recrutement et l'exploitation collective des sources de nourriture : le fourragement met à jour des stratégies qui permettent aux insectes une grande adaptation à leur milieu.

Les capacités des fourmis en matière de coopération, de communication, de compétition et d'apprentissage, entre autres, peuvent être mises à profit pour la conception d'algorithmes de résolution des problèmes d'optimisation. [Bech et Bou, 21]

5.5 Relation avec l'informatique :

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résoudre des problèmes tels que celui de la recherche du plus court chemin.

Les fourmis résolvent des problèmes complexes par des mécanismes assez simples à modéliser.

Il est ainsi assez simple de simuler leur comportement par des algorithmes [Cos, Lou, Mar, 06].

5.6 Les fourmis artificielles :

Le terme « fourmi » est un mot qui se profile plusieurs domaines : celui de la biologie ou plus précisément de la myrmécologie qui est l'étude du comportement naturel des fourmis, celui de la robotique qui utilise leur comportement pour concevoir de nouvelles machines, et celui de l'informatique où ces créatures sont modélisées pour la simulation ou la création d'algorithme.

Les différentes applications informatiques qui découlent des capacités de communication des fourmis se retrouvent par exemple en optimisation combinatoire où la coopération stigmergétique s'applique parfaitement à la recherche du plus court chemin dans un graphe.

5.6.1 Similarités et différences avec les fourmis réelles :

Les fourmis virtuelles (artificielles) ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin

de les rendre plus efficaces que ces dernières. Nous allons maintenant synthétiser ces ressemblances et différences [Cos, Lou, Mar, 06].

Points communs :

• Colonie d'individus coopérants : Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

• Pistes de phéromones : Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.

• Évaporation des phéromones : La méta-heuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ses anciennes décisions.

• Recherche du plus petit chemin : Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

• Déplacement locaux : Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.

• Choix aléatoire lors des transitions : Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle) [Cos, Lou, Mar, 06].

Différences :

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

• Elles vivent dans un monde non-continu : Leurs déplacements consistent en des transitions d'état.

• Mémoire (état interne) de la fourmi : Les fourmis réelles ont une mémoire très limitée. Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

• Nature des phéromones déposées : Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifier des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

• Qualité de la solution : Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découverte.

• Retard dans le dépôt de phéromone : Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment.

• Capacités supplémentaires : Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

• L'anticipation : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.

• Le retour en arrière : une fourmi peut revenir à un état déjà parcouru car la décision qu'elle avait prise à cet état s'avérait mauvaise [Cos, Lou, Mar, 06].

6 Les colonies de fourmis :

Les fourmis utilisent les pistes de phéromone pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir le plus court chemin vers une source à exploiter, sans que les individus aient une vision globale du trajet.

En effet, les fourmis le plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui empruntent le chemin le plus court car la quantité de phéromone présente sur le plus court trajet est légèrement plus importante que celle présente sur le chemin le plus long. Ainsi, une piste présentant une plus grande concentration en phéromone est plus attirante pour les fourmis, elle a une probabilité plus grande d'être empruntée (J.DENEUBOURG, et al., 1990)

(M.DORIGO, et al., 2004). La piste courte va alors être plus renforcée que la longue, et, à terme, sera choisie par la grande majorité des fourmis. (Figure II.1)

Ce mécanisme de résolution collective de problèmes est à l'origine des algorithmes à base de fourmis artificielles. Ces algorithmes ont été initialement proposés dans (M.DORIGO, et al., 1996) (M.DORIGO, 1992), comme une approche multi-agents pour résoudre des problèmes d'optimisation combinatoire.

Les colonies de fourmis artificielles exploitent cette caractéristique pour construire des solutions à un problème d'optimisation.

Dorigo et al. ont proposé un nouvel algorithme pour la résolution du problème du voyageur de commerce. Depuis ces travaux, la démarche a été étendue à beaucoup d'autres problèmes d'optimisation combinatoires ou mêmes continus. Considérons l'exemple de la figure (II.1) *(Dorigo et al.,* 1996) où une colonie de fourmis se déplace de la source de nourritures vers le nid E en sélectionnant l'un des deux chemins de longueurs différentes (créée par une barrière).[Kam,08]

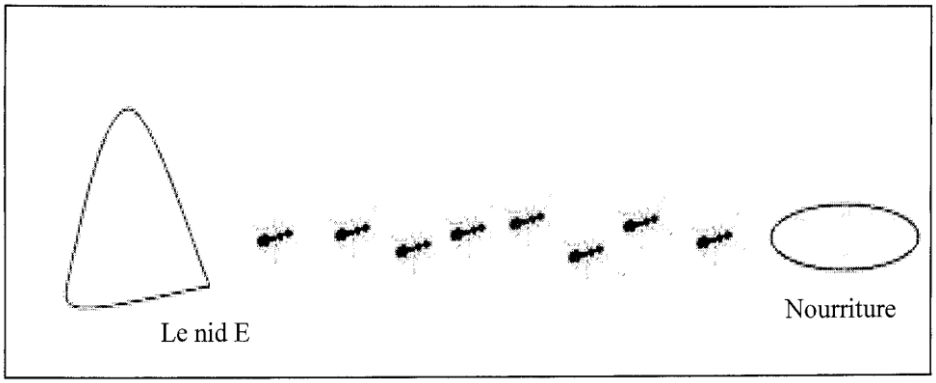


Figure II.1 : Exemple de fourmis réelles

Des fourmis réelles suivent un chemin entre le nid et une source de nourriture :

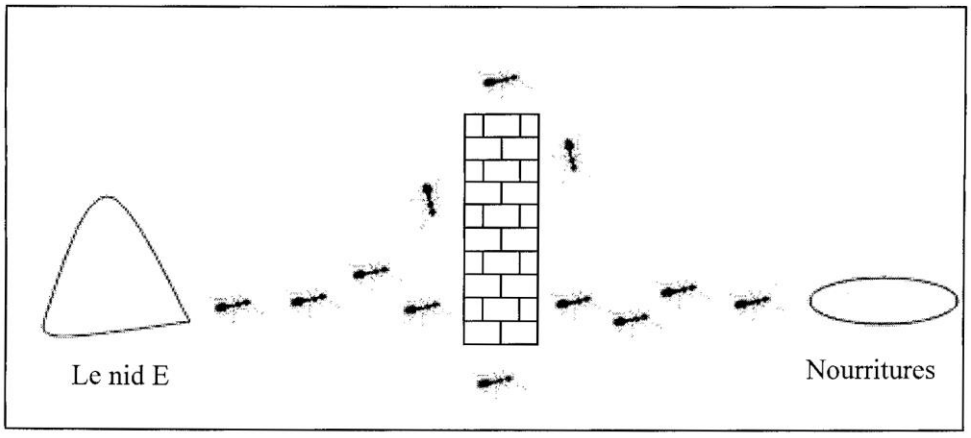


Figure II.2 : Cas d'un obstacle dans le chemin

Un obstacle survient sur le chemin ; les fourmis choisissent de tourner vers la gauche ou vers la droite avec des probabilités égales et la phéromone est déposée plus rapidement sur le chemin le plus court.

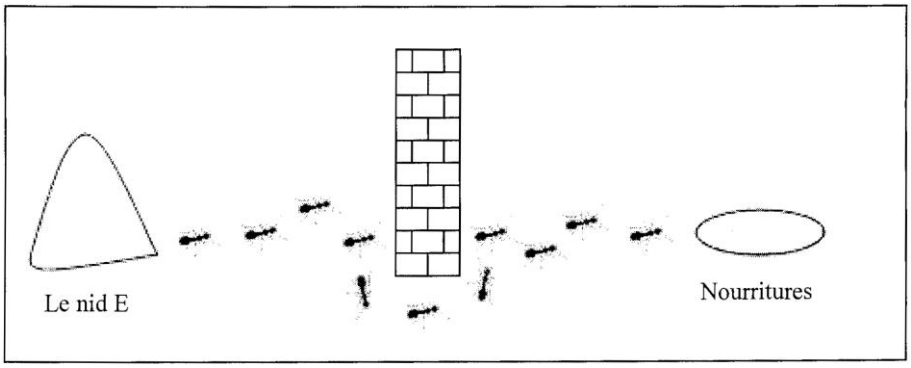


Figure II.3: Les fourmis ont choisi le chemin le plus court

Lorsqu'une colonie de fourmis a le choix entre deux chemins de longueurs différentes pour exploiter une source de nourriture, elle sélectionne le chemin le plus court si la différence entre les longueurs des chemins est suffisamment importante. Les fourmis déposent de la phéromone lors de l'aller vers la source de nourriture et au retour vers le nid.

Au départ, le choix est aléatoires, mais le chemin le plus court devient vite le plus marqué par la phéromone, car les fourmis qui l'empruntent arrivent plus vite au nid et auront statistiquement plus de chance de l'emprunter lorsqu'elles retourneront vers la source de nourriture.

6.1 Les pistes de phéromones :

En marchant du nid à la source de nourriture et vice-versa (ce qui dans un premier temps se fait essentiellement de façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée phéromones. Cette substance permet ainsi donc de créer une piste chimique, sur laquelle les fourmis s'y retrouvent. En effet, d'autres fourmis peuvent détecter les phéromones grâce à des capteurs sur leurs antennes.

Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent être utilisées par les autres fourmis pour retrouver les sources de nourritures trouvées par leurs congénères.

Ce comportement permet de trouver le chemin le plus court vers la nourriture lorsque les pistes de phéromones sont utilisées par la colonie entière. Autrement dit, lorsque plusieurs chemins marqués sont à la disposition d'une fourmi, cette dernière peut connaitre le chemin le plus court vers sa destination. Cette constatation essentielle est la base de toutes les méthodes que l'on va développer plus loin.

On va d'abord étudier le comportement naturel de ces individus afin d'en prouver la validité, avant de l'extraire pour le simuler informatiquement. [Bech , Bou, 21]

6.2 Méthode de colonies de fourmis :

6.2.1 Introduction :

La méthode de colonies de fourmis est apparue d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé stigmergie. En effet, si un obstacle est introduit sur le chemin des fourmis, ces dernières vont, après une phase de recherche, avoir tendance toutes à prendre le plus court chemin entre le nid et l'obstacle. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir une grande probabilité à être attirée par cette zone.

Les fourmis qui sont arrivées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont pris la branche la plus courte du trajet. Il en découle donc que la quantité de phéromones sur ce trajet est plus importante que sur le trajet le plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être pris par les fourmis que les autres chemins, et il sera donc pris par toutes les fourmis.[Hach,13]

Figure II.4 : Schéma de fourmis.

6.2.2 Algorithme de colonies de fourmis :

Ce type d'algorithme a été proposé récemment (1992, par *Marc Dorigo)* et il s'inspire des comportements collectifs de dépôt et de suivie de traces. Dans la nature, les fourmis utilisent des phéromones pour marquer des informations dans leur environnement.

Elles utilisent ensuite leur capteur de phéromones pour déterminer le chemin le plus marqué en phéromones qu'elles suivront. Les études ont démontrées que les fourmis étaient capables de sélectionner le plus court chemin pour aller du nid à une source de nourriture grâce au dépôt et au suivi de pistes de phéromone. Cette conduite est le résultat d'un mode de communication indirecte, via l'environnement: la « stigmergie ». Chaque fourmi dépose, le long de son chemin, une substance chimique, nommée phéromone. Tous les membres de la colonie perçoivent cette substance et orientent préférentiellement leur marche vers les régions les plus odorantes.

- Avantage des algorithmes de colonies de fourmis :

Les algorithmes de colonies de fourmis possèdent plusieurs caractéristiques intéressantes, mentionnons notamment :

- La flexibilité: une colonie de fourmis est capable de s'adapter à des modifications de l'environnement.
- La robustesse: une colonie est apte à maintenir son activité si quelques individus sont défaillants.
- La décentralisation: une colonie n'obéit pas à une autorité centralisée.
- L'auto-organisation: une colonie trouve elle-même une solution, qui n'est pas connue à l'avance. [Kam, 08]

- Inconvénients :
  - Un état bloquant peut arriver.
  - Temps d'exécution parfois long.
  - Ne s'applique pas à tous type de problèmes. [Ben et Sou, 19]

6.2.3 Choix d'implémentation

On précise que chaque fourmi a une mémoire implémentée par une liste de villes déjà visitées.

Cela permet de garantir qu'aucune fourmi ne visitera deux fois une même ville au cour de sa recherche.[Cos, Luo, Mar, 06]

La mémoire de chaque fourmi est vidée lorsqu'elles ont terminé leur cycle.

Les auteurs *M Dorigo, Luca Maria Gambardella (1996)* [Kam,08] , ont proposé l'application de *l'Ant System* sur le problème du voyageur de commerce (TSP). Le problème *TSP* se présente comme suit: étant donné un ensemble *den* villes, l'objectif est de trouver le tour ayant la plus courte distance. La distance séparant une ville *i* à une ville *j* est notée *d iJ* (distance euclidienne). Ce problème peut être représenté par un graphe (N,E), où N est l'ensemble des villes (nœuds) , et E est l'ensemble des arêtes entre les villes.

Soit b_i *(t)* (i= 1 , .. , n) l'ensemble des fourmis dans la ville $_i$ au temps *t* et soit $m = \sum_{i=1}^{n} bi(t)$, le nombre total des fourmis. Chaque fourmi est un agent avec les caractéristiques suivantes :

- La fourmi choisit la prochaine ville de destination avec une probabilité qui est en fonction de la distance de la ville et de la quantité de phéromones présente sur l'arête de connexion.
- Les villes déjà visitées par la fourmi sont retirées de la liste des villes qui restent à visiter. On utilisera pour cela une liste taboue.
- Lorsqu'une fourmi a terminé un tour, une quantité de phéromones est déposée sur chaque arête *(i,j)* visitée.

Soit $\tau_{ij}(t)$ la quantité de phéromones sur l'arête *(i, j)* au temps *t*. chaque fourmi au temps *t* choisit la prochaine ville où elle s'y positionnera au temps *t*+ 1. Une itération comprendra donc l'ensemble des *m* mouvements. Les fourmis compléteront leurs tours après *n* itérations (un cycle) et une mise à jour des phéromones se fera à cet instant [Kam,08].

Pour chaque fourmi, le trajet d'une ville i à une ville j dépend de :

- La liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi k est sur la ville i : J_i^k;

- L'inverse de la distance entre les villes $\eta_{ij} = 1/d_{ij}$, appelée visibilité. Cette information est utilisée pour diriger les fourmis vers des villes proches et ainsi, éviter de trop longs déplacements ;

- La quantité de phéromone déposée sur l'arête reliant deux villes τ_{ij}, appelée intensité de la piste. Cette quantité définit l'attractivité d'une piste, et elle est modifiée après le passage d'une fourmi. C'est la pseudo-mémoire du système.

La règle de déplacement est la suivante :

$$P_{ij}{}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}.\eta_{ij}^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}.\eta_{il}^{\beta}} & \text{si } j \in N_i^k \\ 0 \quad \text{autrement} \end{cases} \qquad \text{(II.1)}$$

Equation II.1: Probabilité de déplacement

Où α et β sont deux paramètres contrôlant l'importance relative de l'intensité et de la visibilité.

- Quantités de phéromones déposées :

Après un tour complet, chaque fourmi dépose une quantité de phéromone $\Delta\tau_{ij}{}^{k}$ (t) sur l'ensemble de son parcours. Cette quantité dépend de la qualité de la solution trouvée, On le définit ainsi :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{si} \quad (i,j) \in T^k(t) \\ 0 & \text{si} \quad (i,j) \notin T^k(t) \end{cases} \qquad \text{(II.2)}$$

Equation II.2: Quantités de phéromones.

Où T^k (t) est le trajet effectué par la fourmi k à l'itération t, L^k (t) est la longueur de T^k (t), et Q est un paramètre fixé et un nombre positif constant. et donc (t)) s'obtient en analysant la mémoire de la fourmi.

On peut définir le $\Delta\tau_{ij}{}^{k}$ (t) de la formule de mise à jour des phéromones ainsi :

$$\Delta\tau_{ij}\ (t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}\ (t) \qquad \text{(II.3)}$$

Equation II.3: Mise à jour de phéromones

Où m est le nombre de fourmis.[Hach,13]

- Mise à jour des phéromones [Dorigo 02] :

A la fin de chaque cycle (chaque fourmi a parcouru les n sommets qui composent le graphe), les variables des phéromones sont mises à jour selon la formule :

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad \text{(II.4)}$$

Equation II.4: Mise à jour des phéromones

Où ρ le taux d'évaporation.

$\rho \in [0, 1[$ est un coefficient qui définira la vitesse d'évaporation des phéromones sur les arcs entre l'instant t et l'instant (t + n), et où $\Delta\tau_{ij}(t)$ représente la quantité de phéromone déposée par les fourmis dans ce même intervalle de temps sur l'arc (i, j).

Le choix de ρ est important, en effet si ρ se rapproche trop de 1, on observe un effet de stagnation des phéromones sur les arcs, ce qui implique des inconvénients tel que le fait de voir les mauvaises solutions persister.

De même, choisir $\rho \approx 0$ implique une évaporation trop rapide des phéromones, donc amène la fourmi à un choix dépendant uniquement de la visibilité des nœuds. [Cos,Luo,Mar,06].

Le principe de la méthode provient analogiquement avec les comportements collectifs des insectes, les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol et le suivi de pistes observés dans les colonies de fourmis et construisent ainsi une solution à un problème en tenant compte de leur expérience collective. Au fait, elles adoptent pour la recherche de la solution la notion du plus court chemin.

D'une manière simplifiée, les fourmis commencent par se déplacer au hasard. Puis, lorsqu'elles trouvent de la nourriture, elles retournent vers leur colonie, en marquant leur chemin à l'aide de phéromone. Si d'autres fourmis rencontrent ce chemin, il y a de fortes chances qu'elles arrêtent leurs déplacements aléatoires et qu'elles rejoignent le chemin marqué, en renforçant le marquage à leur retour, s'il mène bien vers la nourriture. Par conséquent, le chemin le plus court sera davantage parcouru, et donc plus renforcé et plus attractif. Par conséquent, le nombre de fourmis suivant cette trajectoire augmente. Au fil du temps, la quantité de phéromones déposée sur le plus long chemin diminue et finit par disparaître. Toutes les fourmis suivent alors le chemin le plus court.

Étape 1 : Initialisation

- Initialiser les pistes de phéromone.

Étape 2 : Construction de la solution

- Pour chaque fourmi répéter.
- Construction de la solution en utilisant les pistes de phéromone.

Étape 3 : Mise à jour des pistes de phéromone

- Jusqu'à atteindre la condition d'arrêt. [Hach,13]

6.2.4 Le pseudo-code de l'algorithme ACS : est présenté dans l'algorithme suivant [Bech,Bou,21]:

Pour t=1, …, t_{max} **faire**
Pour chaque fourmi k = 1 , … , m **faire**
– Choisir une ville au hasard.
Pour chaque ville non visitée i **faire**
–Choisir une ville j, dans la liste j_{ik} des villes restantes, selon la formule II.1
Fin Pour
– Déposer une piste $\Delta(t)$ sur le trajet $T_k(t)$ conformément à l'équation II.2
Fin Pour
– Evaporer les pistes selon la formule II.3
Fin Pour

Algorithme II.1 : pseudo-code de l'algorithme ACS

7 Emploi du temps et ACO :

Citant quelques méthodes de résolution de ce problème :

Le travail présenté par (KRZYSZTOF, et al., 2002): où les auteurs de ce projet procèdent à une simplification du problème d'emploi du temps d'université en impliquant trois types de contraintes dures et trois types de contraintes souples. Le système de fourmi « Max-Min Ant System » se sert d'une routine de recherche locale séparée, proposée pour aborder ce problème, une construction de graphe approprié et une représentation de matrice de phéromones sont conçus et les résultats de ce système ont démontré qu'il peut construire des horaires meilleurs qu'un algorithme qui réitère le procédé de recherche locale des solutions.

Citons le travail de (Thatchai.Thepphakorn, et al., 2014) : où les auteurs de ce projet présentent de nouvelles variantes pour l'optimisation de colonie de fourmis appelés « best-worst ant system (BWAS) » et « best-worst ant colony system (BWACS) » ces derniers ont été intégrés au problème d'emploi du temps basé sur les colonies de fourmis(ANCOTT). Des stratégies de recherche locale (LS) ont été développées et intégrées dans les systèmes BWAS et BWACS afin d'améliorer leur efficacité et de permettre de trouver le meilleur emploi du temps avec un nombre minimum de violations de contraintes souples [Ben, Sou, 19].

8 CONCLUSION :

L'algorithme des colonies de fourmis forme une classe de méta-heuristique proposée pour les problèmes d'optimisation difficile, dans notre cas l'emploi du temps des cours universitaires, le chapitre suivant sera consacré à la normalisation des entrées « BENCHMARK » et les mesures d'évaluation.

CHAPITRE III :

LES MESURES DE PERFORMANCE ET BENCHMARKING

1 Introduction :

Les mesures d'évaluation ont pour but de mesurer la performance d'un modèle de Machine Learning, des algorithmes. Ces mesures sont des indicateurs qui peuvent être utilisés pour y parvenir. Ils sont adaptés pour évaluer la performance d'un modèle de classification et qui sont calculés à partir de la matrice de confusion.

2 Evaluateurs de performance du classificateur :

Avant de présenter les métriques, il est pertinent de souligner qu'elles ont été définies pour des problèmes à deux classes et qu'elles sont basées sur la matrice de confusion, un outil qui informe sur les types de résultats et d'erreurs commis par un classificateur. Les classes sont nommées positives et négatives et la matrice de confusion a quatre valeurs calculées en termes de classes réelles et prédites, à savoir :

D TP (vrais positifs) : la quantité d'éléments positifs prédits comme positifs.

D FP (faux positifs) : la quantité d'éléments négatifs prédits comme positif.

D FN (faux négatifs) : la quantité d'éléments positifs prédit comme négatif.

D TN (vrais négatifs) : la quantité d'éléments négatifs prédit comme négatif.

Les évaluateurs de performance les plus courants sont :

2.1 Sensibilité (recall):

Également appelée taux de réussite ou rappel, elle mesure à quel point un classificateur peut reconnaître des exemples positifs.

Sous forme mathématique, on a :

$$\text{sens} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{(III.1)}$$

Equation III.1 : Sensibilité (recall)

Le recall permet de savoir le pourcentage de positifs bien prédit par notre modèle.

En d'autres termes c'est le nombre de positifs bien prédit (Vrai Positif) divisé par l'ensemble des positifs (Vrai Positif + Faux Négatif).

Plus il est élevé, plus le modèle de Machine Learning maximise le nombre de Vrai Positif.

Mais, cela ne veut pas dire que le modèle ne se trompe pas.

Quand le recall est haut, cela veut plutôt dire qu'il ne ratera aucun positif Néanmoins cela ne donne aucune information sur sa qualité de prédiction sur les négatifs.

2.2 Précision :

C'est le rapport des exemples positifs prédits qui sont réellement positif.

Cela nous donne sous forme mathématique :

$$prec = \frac{TP}{TP + FP} \qquad \text{(III.2)}$$

Equation III.2 : Précision

La précision est assez similaire au recall, il faut donc bien comprendre leur différence.

Elle permet de connaître le nombre de prédictions positifs bien effectuées.

En d'autres termes c'est le nombre de positifs bien prédit (Vrai Positif) divisé par l'ensemble des positifs prédit (Vrai Positif + Faux Positif).

- Qu'est ce que nous apporte la précision ?

Plus elle est élevé, plus le modèle de Machine Learning minimise le nombre de Faux Positif.

Quand la précision est haute, cela veut dire que la majorité des prédictions positives du modèle sont des positifs bien prédit.

- Exploration de données :

Vrais Positifs, Faux Négatifs – La méthode simple :

Ces termes définissent le type de résultat qu'on peut avoir après une prédiction à deux possibilités (en Machine Learning on appelle ça une classification binaire)
Ils permettent d'évaluer la performance de notre modèle, à quel point ses prédictions sont fiables.

Souvent ces termes sont confondus entre eux et c'est pourquoi on vous donne ici la technique pour s'en souvenir !

À savoir :

En Machine Learning, pour afficher le résultat d'un modèle de prédiction à deux possibilités on utilise une matrice de confusion.

La matrice de confusion est en fait un tableau dans lequel on affiche le nombre de prédictions selon chaque possibilité :

Nombre de Vrais Positifs	Nombre de Faux Négatifs
Nbr de Faux Positifs	Nbr de Vrai Négatifs

Tableau III.1: Matrice de confusion**.**

La matrice de confusion est un tableau permettant d'afficher les résultats du modèle.

En une phrase :

- Plus le recall est haut, plus le modèle repère de positif.
- Plus la précision est haute, moins le modèle se trompe sur les positifs.

Bien qu'ils soient utiles, ni la précision ni le recall ne permettent d'évaluer entièrement un modèle de Machine Learning.

La precision et le recall sont deux métriques essentielles en classification, du fait de leur robustesse et de leur interprétabilité.

La précision répond à la question : combien d'individus sélectionnés sont pertinents? Le recall répond à la question : combien d'individus pertinents sont sélectionnés ?

Un compromis doit être fait entre l'optimisation de la précision et l'optimisation du recall**.** C'est pour cette raison que ces deux métriques sont étudiées conjointement dans l'évaluation d'un modèle. Pour en faire la synthèse, d'autres métriques sont fréquemment utilisées, comme l'AUC Precision-Recall ou le F1-score.

La figure ci-dessous permet de bien visualiser ces indicateurs :

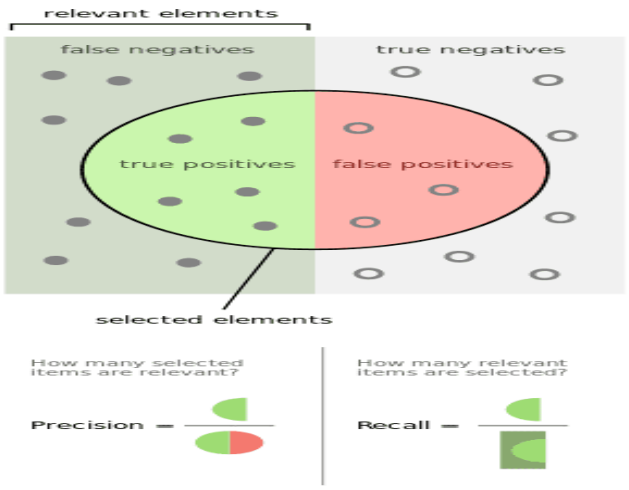


Figure III.1: Precision et Recall

Séparément c'est deux métrique sont inutiles :

- Si le modèle prédit tout le temps « positif », le recall sera élevé.
- Au contraire, si le modèle ne prédit jamais « positif », la précision sera élevée.

On aura donc des métriques qui nous indiquent que notre modèle est efficace alors qu'il sera au contraire plus naïf qu'intelligent.

Heureusement pour nous, une métrique permettant de combiner la précision et le recall existe : le F1 Score.

2.3 Le F1 Score :

Permet d'effectuer une bonne évaluation de la performance de notre modèle**.**

Il se calcule ainsi :

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision} \qquad \text{(III.3)}$$

Equation III.3 : F1 Score

Alors pourquoi calculer le F1 Score et pas simplement la moyenne des deux métriques ?

En fait, en statistique, le calcul sur les pourcentages n'est pas exactement le même que sur les nombres entiers.

Ici le F1 Score est ce qu'on appelle la moyenne harmonique. C'est un autre type de moyenne que celle habituelle et c'est un excellent moyen de calculer la moyenne de taux ou de pourcentage (ici le recall et la précision).

Cela fait du F1 Score une des métriques les plus utilisés chez les Data Scientist !

Le plus F1 Score est élevé, le plus modèle est performant.

2.4 F-mesure :

C'est la moyenne harmonique de la sensibilité et de la précision.

$$\text{F.mea} = \frac{(\beta^2 + 1) \times \text{sens} \times \text{prec}}{\text{sens} + \beta \times \text{prec}}, \ \beta \geq 0 \qquad \text{(III.4)}$$

Equation III.4 : F-mesure

2.5 G-mean1 :

C'est la moyenne géométrique de la sensibilité et de la précision.

$$GSP = \sqrt{sens \times prec} \qquad \text{(III.5)}$$

Equation III.5 : G-mean1

2.6 G-mean2 :

C'est la moyenne géométrique de la sensibilité et de la spécificité.

$$GSS = \sqrt{sens \times spec} \qquad \text{(III.6)}$$

Equation III.6: G-mean2

2.7 L'inertie intra-classes :

L'inertie intra-classes permet de mesurer le degré d'homogénéité entre les objets appartenant à la même classe. Elle calcule leurs distances par rapport au point représentant le profil de la classe.

$$Intra = \frac{1}{n}\sum_{C \in P}\frac{1}{2n_c}\sum_{i \in C}\sum_{j \in C} d(i,j)^2 \qquad \text{(III.7)}$$

Equation III.7 : L'inertie intra-classes

2.8 L'inertie interclasse :

L'inertie interclasses mesure le degré d'hétérogénéité entre les classes. Elle calcule les distances entre les points représentant les profils des différentes classes de la partition.

$$Inter = \frac{1}{n}\sum_{C \in P} n_C d^2(c, c_G) \qquad \text{(III.8)}$$

Equation III.8 : L'inertie interclasse

Avec *c* le centre de la classe *C* et *G c* est le centre du nuage de points.

- Plus les données à l'intérieur des classes sont homogènes, plus leurs distances par rapport au point représentant la classe sont faibles. Par conséquent, une valeur

faible de l'inertie intra-classes décrit une homogénéité des données à l'intérieur des classes.

- Plus les classes sont hétérogènes entre elles, plus les distances entres les points représentants les profils des classes sont élevées. Donc, une valeur élevée de l'inertie interclasses traduit une hétérogénéité entre les classes. Cet indice a le défaut d'augmenter quand on augmente le nombre de classes. [Ghr, Cux, Lam, Lel]

3 BENCHMARKING:

3.1 Définition d'un BENCHMARK 1 :

Un benchmark, en anglais, est un point de référence servant à effectuer une mesure. Le terme vient du vocabulaire professionnel des géomètres, et désigne à l'origine un repère de nivellement. En français, il désigne plusieurs choses.

En gestion, et en économie en général, où l'usage du terme tend à se répandre, le benchmarking est la confection d'un étalonnage pour mesurer diverses performances. Voir aussi : évaluation d'entreprise.

En finance, sur les marchés de taux d'intérêt, un benchmark est un emprunt d'État particulièrement liquide qui sert de base de comparaison à l'ensemble du marché obligataire. Dans la zone Euro, cet emploi est rempli par certains Bunds.

En anglais, un benchmark (français : étalon ou repère) est un point de référence servant à effectuer une mesure. Le terme vient du vocabulaire professionnel des géomètres, et désigne à l'origine un repère de nivellement. En français, il désigne plusieurs choses.

En physique, un *benchmark* est un banc d'essai permettant de tester une méthode par comparaison à une série de cas connus et validés.

En informatique, un *benchmark* est un banc d'essai permettant de mesurer les performances d'un système pour le comparer à d'autres.

3.2 La définition d'un benchmark 2 :

Benchmark, aussi appelé benchmarking, est un terme anglais qui peut être traduit par les mots référence, étalon ou repère.

Le benchmark est en effet un technique marketing basé sur l'analyse comparative.

Développée dans les années 1980, elle vise à étudier et analyser d'autres entreprises pour optimiser ses propres techniques de gestion et modes d'organisation.

En d'autres termes, benchmarker consiste à étudier ce que fait la concurrence. Ce peut-être un produit ou un service qui est à l'étude. L'objectif pour l'entreprise qui benchmark est de s'améliorer et de rivaliser avec les meilleurs sur le march

3.3 Pourquoi faire un benchmark ?

Deux motifs essentiels peuvent motiver la réalisation d'un benchmark pour une entreprise :

-Améliorer la compétitivité et la productivité de l'entreprise avec une optimisation des pratiques.

-Préparer le lancement d'un nouveau produit.

L'idée du benchmark est en effet de s'inspirer d'autres pratiques pour rendre celles de l'entreprise plus efficaces. C'est également un outil très utilisé dans les domaines à forte innovation, qui nécessitent une veille concurrentielle pointue.

La réalisation d'un benchmark permet ainsi :

- De définir les critères qui favorisent la performance et la réussite.
- D'identifier les meilleures entreprises dans votre domaine d'activité.
- D'avoir une vision claire du marché et de ses tendances.
- De maîtriser l'univers concurrentiel.
- De connaître le positionnement, les points forts et les points faibles de chacun de vos concurrents.
- D'effectuer une synthèse des bonnes pratiques.
- Et de mettre en place un plan d'action au niveau de l'entreprise, pour s'approprier.

Ces *Best practices* et optimiser la performance de l'entreprise (ou réussir son lancement produit).

3.4 Quels sont les différents types de benchmark ?

On distingue 4 différents types de benchmark :

-Le Benchmark compétitif ou concurrentiel, qui consiste à analyser ses concurrents directs, c'est-à-dire les entreprises qui se positionnent dans le même métier et sur le même secteur d'activité.

- Le Benchmark interne, qui effectue un comparatif de la performance de différents services internes sur une même activité. L'objectif étant d'identifier les potentiels et perspectives d'optimisation sur cette activité.

- Le Benchmark fonctionnel, qui analyse une fonction spécifique du processus - le service après-vente par exemple - en prenant comme élément de comparaison les entreprises les plus performantes, tous secteurs d'activité confondus.

- Le Benchmark générique ou horizontal, qui repose sur une analyse comparative des processus et méthodes de travail. Comme pour le benchmark fonctionnel, l'objectif du benchmark horizontal est de recueillir les *best practices* des entreprises de référence, qu'elles soient positionnées dans le même secteur d'activité ou non.

- Une autre différenciation peut se faire entre les benchmarks globaux - avec une analyse à 360° - et les benchmarks précis, centrés sur un métier, une fonction…

3.5 Les avantages du benchmark :

Le benchmarking comporte des avantages non-négligeables pour l'entreprise, mais aussi pour celui qui l'opère :

- Il ne nécessite que très peu de matériel. Le plus souvent, un ordinateur, une connexion internet, un papier et un crayon.
- Il est peu coûteux.
- Il permet d'aller piocher les meilleures idées pratiquées par les leaders du marché visé.
- Il permet de se surpasser et d'aller chercher le meilleur de l'entreprise.
- Il permet de mieux maîtriser son environnement concurrentiel.

3.6 Les inconvénients du benchmark :

Les avantages du benchmark sont nombreux. Mais ses inconvénients existent aussi, notamment s'il est mal cadré. Les voici :

- Il mobilise du temps.
- Pour être utile et profitable à l'entreprise, le benchmark doit être exhaustif et balayer les pratiques de tous les concurrents sérieux.
- Il peut mobilier une équipe entière de salariés.
- Pour être suivi d'effets, il faut que l'équipe dirigeante de l'entreprise soit dans une dynamique de changements.[https://fr.wikipedia.org/wiki/Benchmark.]

4 Un dataset :

En machine learning regroupe un ensemble de données. Celles-ci dépendent d'une variable associée aux valeurs. Leur accès peut se produire de manière individuelle ou

collective. Il existe différents modèles, comme le dataset d'entraînement, le dataset de test et le dataset de validation.

4.1 Un dataset en machine learning, c'est quoi ?

Le dataset est un outil numérique qui intègre plusieurs données. Il peut s'agir de fichiers vidéo, d'images, de textes, de sons ou même de statistiques. Leur regroupement forme un ensemble. Dans le domaine du machine learning (ou apprentissage automatique), le dataset demeure indispensable pour la création de modèles qui, eux-mêmes, permettent l'expression d'algorithmes à travers différents usages et résultats :

- ✓ Les fonctions prédictives et les tendances prévisionnelles d'un secteur.
- ✓ Les besoins d'une cible ou d'un consommateur.
- ✓ L'analyse d'un fichier image.
- ✓ La gestion des anomalies de l'équipement.
- ✓ La traduction automatique…

Le dataset constitue donc une mécanique essentielle en intelligence artificielle (IA). En apprentissage supervisé ou non supervisé, les champs d'application s'étendent de la cyber-sécurité à l'économétrie, en passant par la bio-informatique ou même la segmentation d'images, pour ne citer que quelques exemples. On distingue différentes catégories d'outils. Les plus connues demeurent les datasets d'entraînement, de test et de validation.

4.2 Qu'est-ce qu'un dataset d'entraînement ?

Le dataset d'entraînement est le premier outil employé par les spécialistes de la business intelligence et des technologies de l'information (TI). Il se sert de paramètres initiaux pour générer un réseau. Si nécessaire, il peut les modifier en vue de les optimiser. Il demande une exploitation conséquente des volumes de données. Le dataset d'entraînement est considéré comme l'étape d'apprentissage automatique pour l'IA, avant la mise en production du modèle concerné.

4.3 Qu'est-ce qu'un dataset de validation ?

Le dataset de validation intervient au terme du dataset d'entraînement. Il vérifie que ce dernier outil a correctement effectué les ajustements de paramètres. Si ce n'est pas le cas, il entreprend les modifications nécessaires afin d'obtenir la meilleure configuration possible. A l'issue du contrôle, ce dataset valide le travail réalisé.

L'exploitation des données ou des valeurs est moindre par rapport à la précédente étape. Le dataset de validation est alors assimilé à la phase d'ajustage.

4.4 Un dataset de test, qu'est-ce que c'est ?

Comme sa dénomination l'indique, le dataset de test effectue les vérifications adéquates pour contrôler les performances du système avant son déploiement. Contrairement à ses prédécesseurs, il ne modifie pas les paramètres, mais lance un ou plusieurs tests en vue d'estimer la puissance réelle du réseau. On s'assure ainsi que l'exploitation des données dispose des ressources nécessaires. Afin d'éviter des erreurs ou des anomalies, de nouvelles valeurs lui sont associées pour tester ses résultats et ses fonctions prédictives.

[https://www.cadremploi.fr/editorial/conseils/conseils-carriere/quest-ce-quun-benchmark]

- Exemple :

Dans notre projet on a utilisé le xml marker pour modéliser les données d'entrées comme illustré dans la figure : III.2

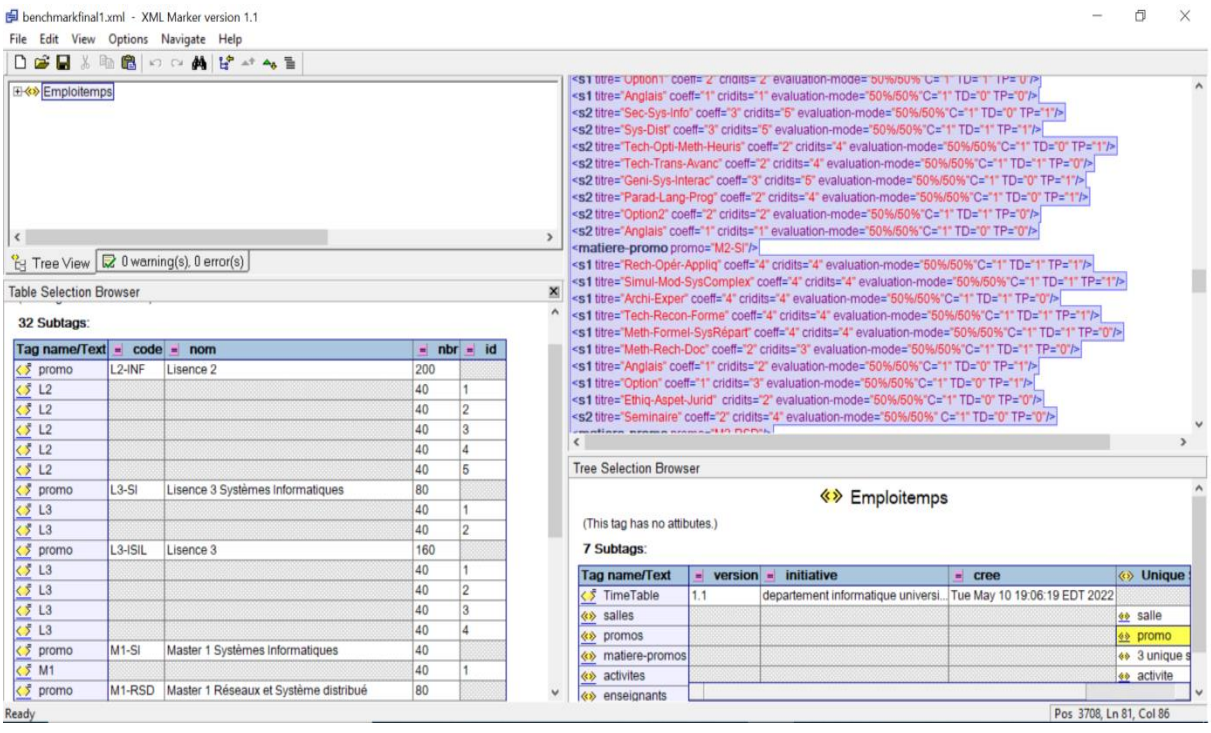


Figure III.2 : Partie de Benchmark.

CHAPITRE IV

IMPLÉMENTATION ET RÉALISATION

1 Introduction

Ce chapitre est consacré à la réalisation de notre système qui concrétise l'application d'une solution à la problématique présentées au cours des chapitres précédents. Il s'agit dans le cas ce mémoire l'application d'une approche basée ACO pour le problème d'emploi du temps des cours universitaires du Département d'Informatique de l'université du 20 août 1955-Skikda.

2 Conception avec UML :

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

Il s'articule autour de treize types de diagrammes, Ces types de diagrammes sont répartis en deux grands groupes : diagrammes structurels et diagrammes comportementaux. On va présenter une modélisation avec un diagramme de classe.

Le diagramme de classes donnera une vision assez claire des informations et les données qui seront utilisées, mais également des fonctions (ou opérations) qui devront s'appuyer sur ces informations. L'approche objet mélange donc habilement l'analyse des informations et des actions au lieu de les analyser séparément. [Khe, 19]

2.1 Diagramme de classe :

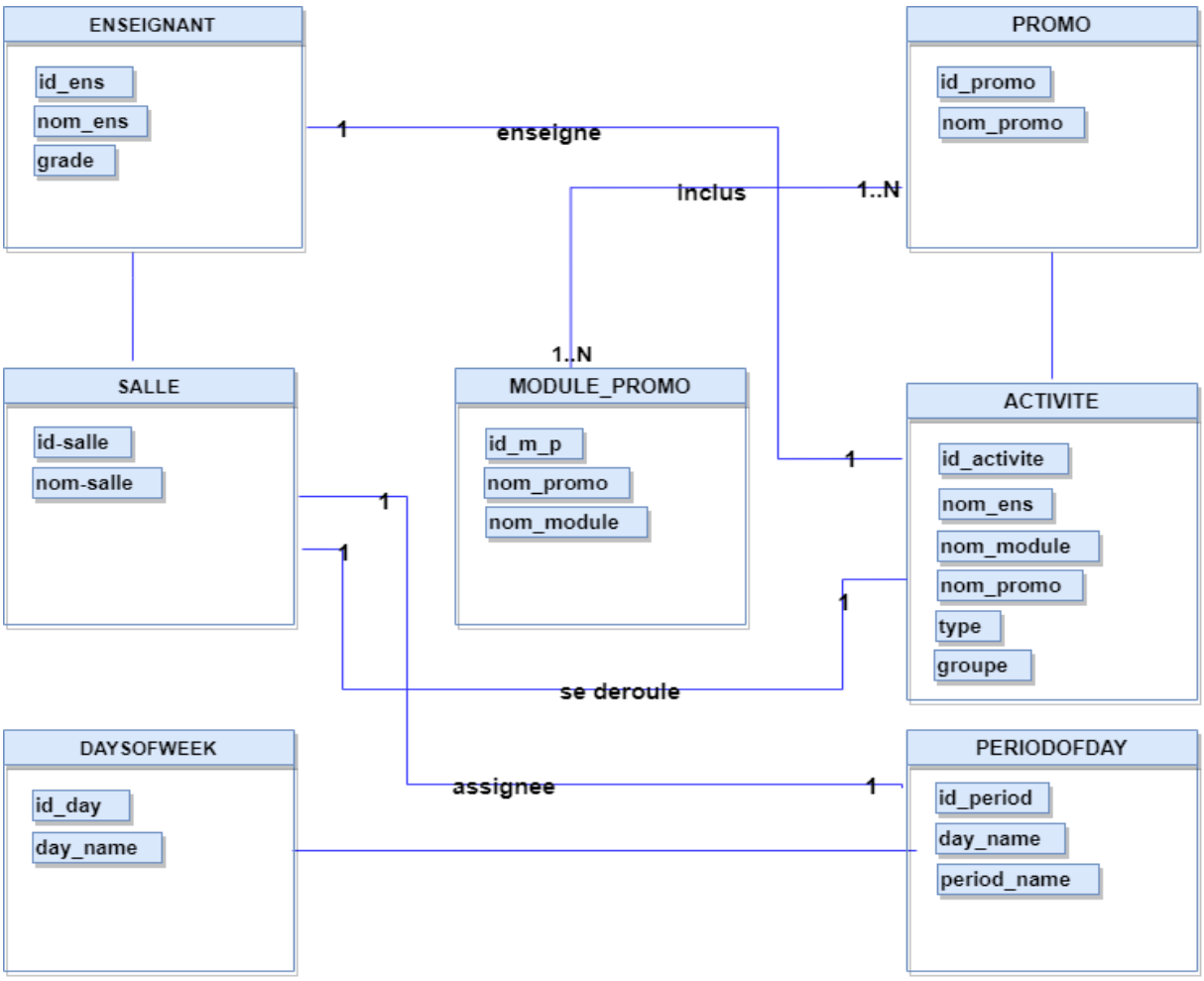


Figure IV.1 : Diagramme de classe.

3 Environnement de travail :

3.1 Environnement matériel :

L'application a été développée et exécutée sur un ordinateur « DELL » ayant les caractéristiques suivantes :

- Spécifications de l'appareil :

  - Nom de l'appareil : DESKTOP-3UK2FD0 DELL.

  - Processeur : Intel(R) Core(TM) i7-10610U CPU @ 1.80GHz 2.30 GHz.

  - Mémoire RAM installée : 32,0 Go (31,7 Go utilisable).

3.2 Environnement logiciel :

3.2.1 Système d'exploitation :

Le système d'exploitation utilisé pour la réalisation et l'exécution de notre application est comme suit :

- Spécifications de Windows :

-Type de système : Système d'exploitation 64 bits, processeur x64.

-Edition : Windows 10 Professionnel.

-Version : 21H1.

3.2.2 Language de Développement :

3.2.2.1 Historique du langage Matlab :

Le langage MATLAB a été conçu par Cleve Moler à la fin des années 1970 à partir de deux bibliothèques écrites en Fortran : LINPACK et EISPACK

Alors professeur de mathématiques à l'université du Nouveau-Mexique, il souhaitait permettre à ses étudiants de pouvoir utiliser ces deux bibliothèques sans connaitre le Fortran. Cleve Moler l'utilisa ensuite pour des cours donnés à l'université Stanford où il reçut un accueil mitigé de la part des étudiants en mathématiques, habitués au Fortran.

3.2.2.2 Langage de programmation Matlab :

MATLAB est une abréviation de *Matrix LABoratory*. Écrit à l'origine, en Fortran, par *C. Moler*, MATLAB était destiné à faciliter l'accès au logiciel matriciel développé dans les projets LINPACK et EISPACK. La version actuelle, écrite en C par the MathWorks Inc., existe en version professionnelle et en version étudiant. Sa disponibilité est assurée sur plusieurs platesformes : Sun, Bull, HP, IBM, compatibles PC (DOS, Unix ou Windows), Macintoch, iMac et plusieurs machines parallèles.

MATLAB est un environnement puissant, complet et facile à utiliser destiné au calcul scientifique. Il apporte aux ingénieurs, chercheurs et à tout scientifique un système interactif intégrant calcul numérique et visualisation. C'est un environnement performant, ouvert et programmable qui permet de remarquables gains de productivité et de créativité.

MATLAB est un environnement complet, ouvert et extensible pour le calcul et la visualisation. Il dispose de plusieurs centaines (voire milliers, selon les versions et les modules optionnels autour du noyeau Matlab) de fonctions mathématiques, scientifiques et techniques. L'approche matricielle de MATLAB permet de traiter les données sans aucune limitation de taille et de réaliser des calculs numériques et symboliques de façon fiable et rapide. Grâce aux fonctions graphiques de MATLAB, il devient très facile de modifier interactivement les différents paramètres des graphiques pour les adapter selon nos souhaits.

3.2.2.3 Démarrage de MATLAB :

Pour lancer l'exécution de MATLAB :

• sous Windows, il faut cliquer sur Démarrage, ensuite Programme, ensuite MATLAB,

• sous d'autres systèmes, se référer au manuel d'installation.

L'invite '>>' de MATLAB doit alors apparaître, à la suite duquel on entrera les commandes.

La fonction "***quit***" permet de quitter MATLAB : >>quit

La commande "***help***" permet de donner l'aide sur un problème donné.

4 Quelques fenêtres de l'application :

4.1 Fenêtre « Gestion Enseignants » :

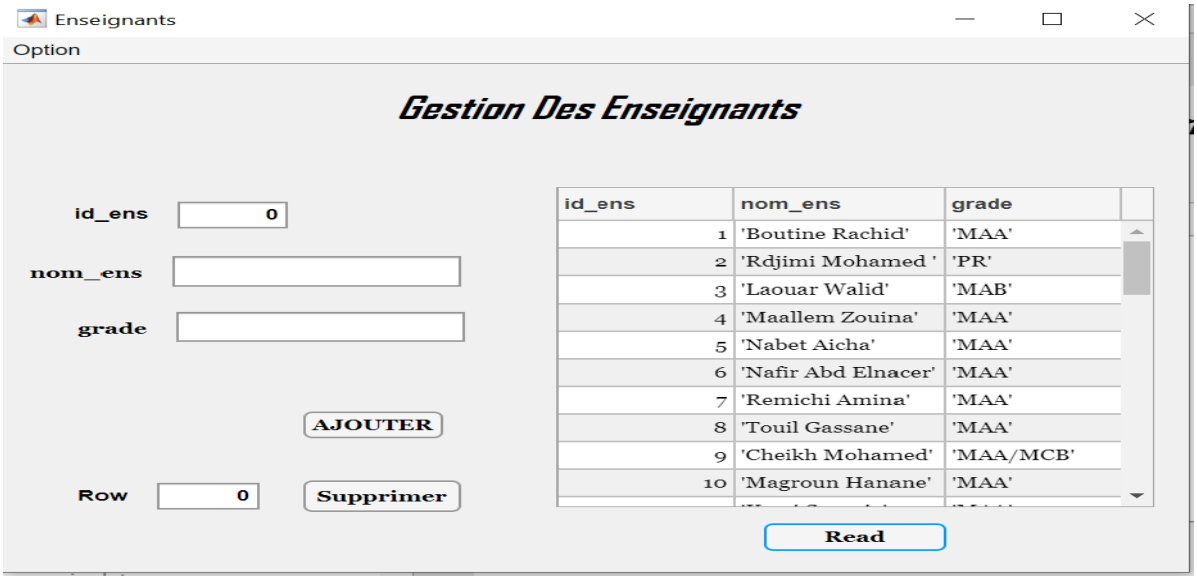


Figure IV.2 : Interface gestion Enseignant

4.2 Fenêtre « Gestion DES modules» :

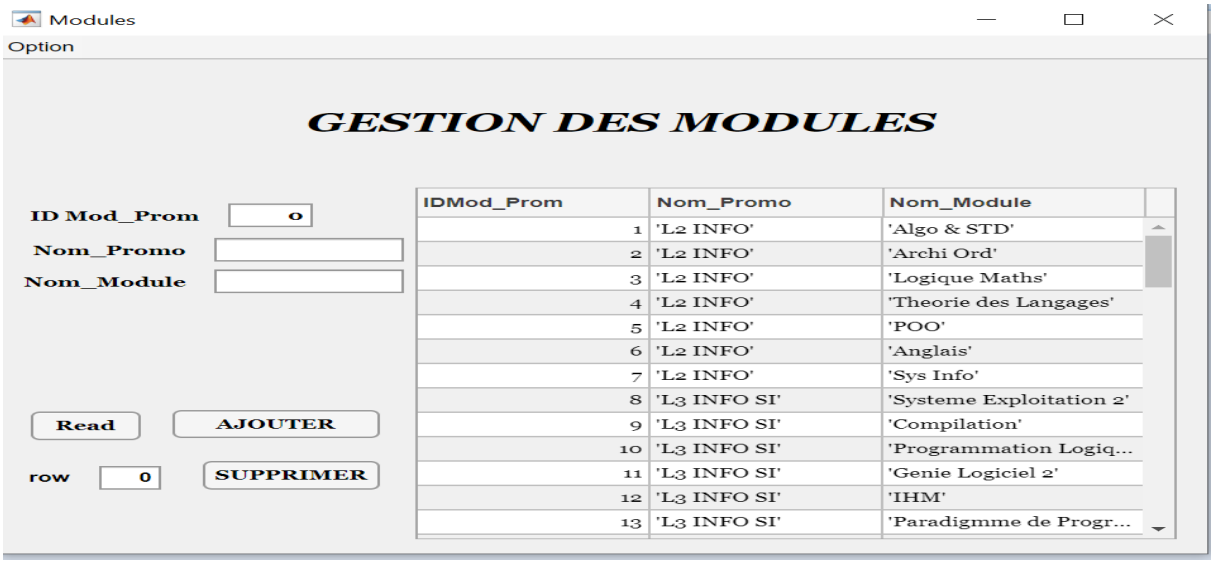


Figure IV.3 : Interface gestion Module.

4.3 Fenêtre « Gestion Des Ressources» :

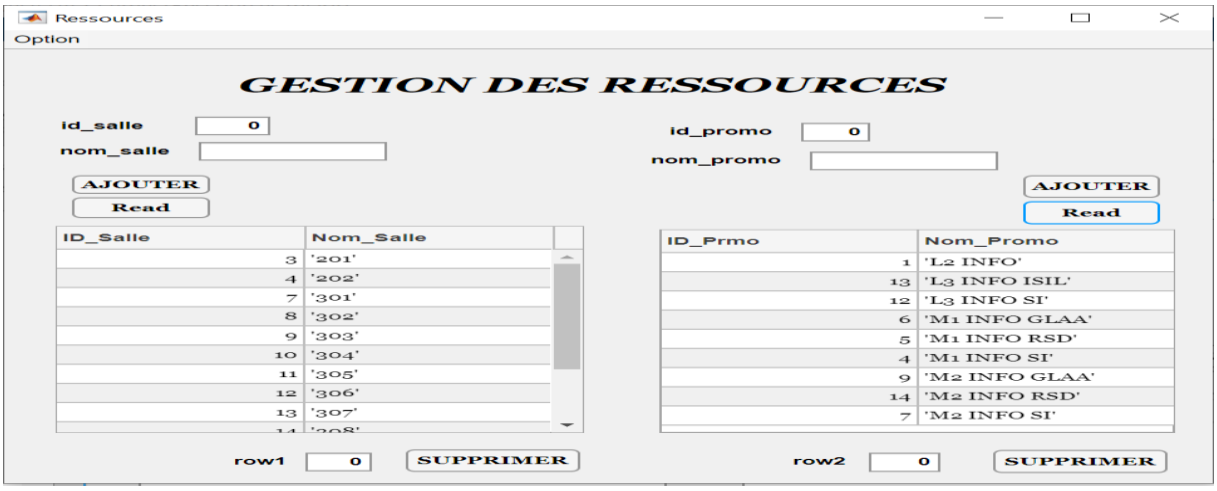


Figure IV.4 : Interface gestion Ressources

5 Résultats :

Après notre étude appliquée à un cas générale, les résultats obtenus sont comme le suivant :

- La promotion choisit est la 2éme année Licence informatique.

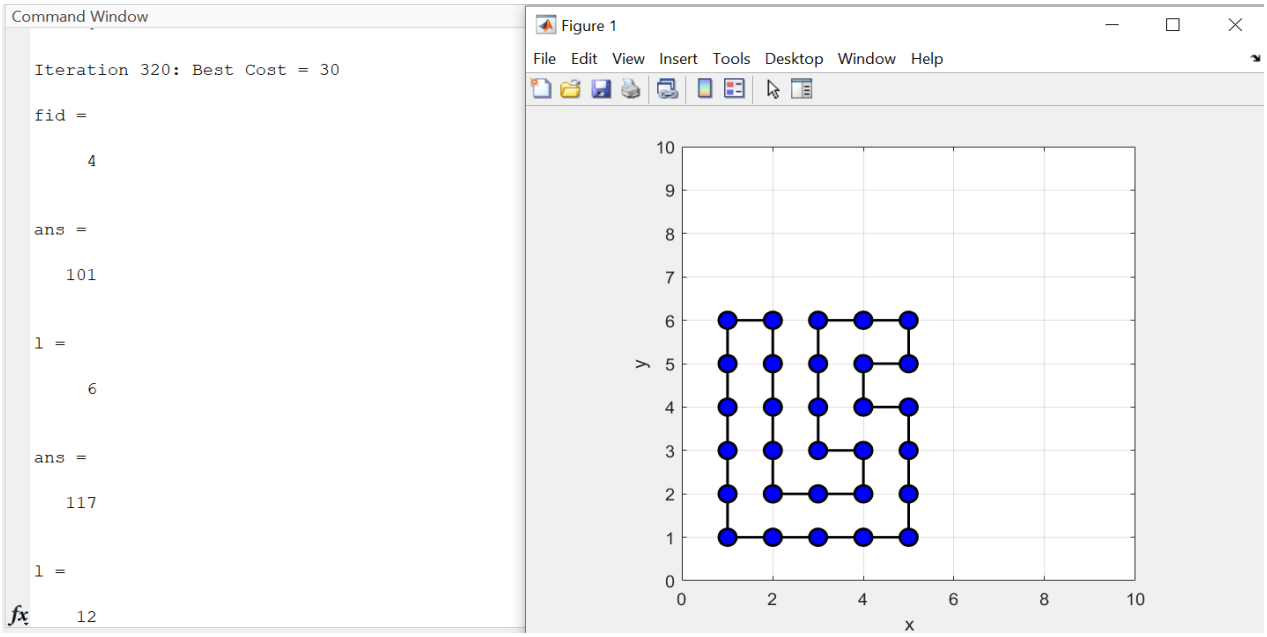


Figure IV.5: Affectation des séances/jour

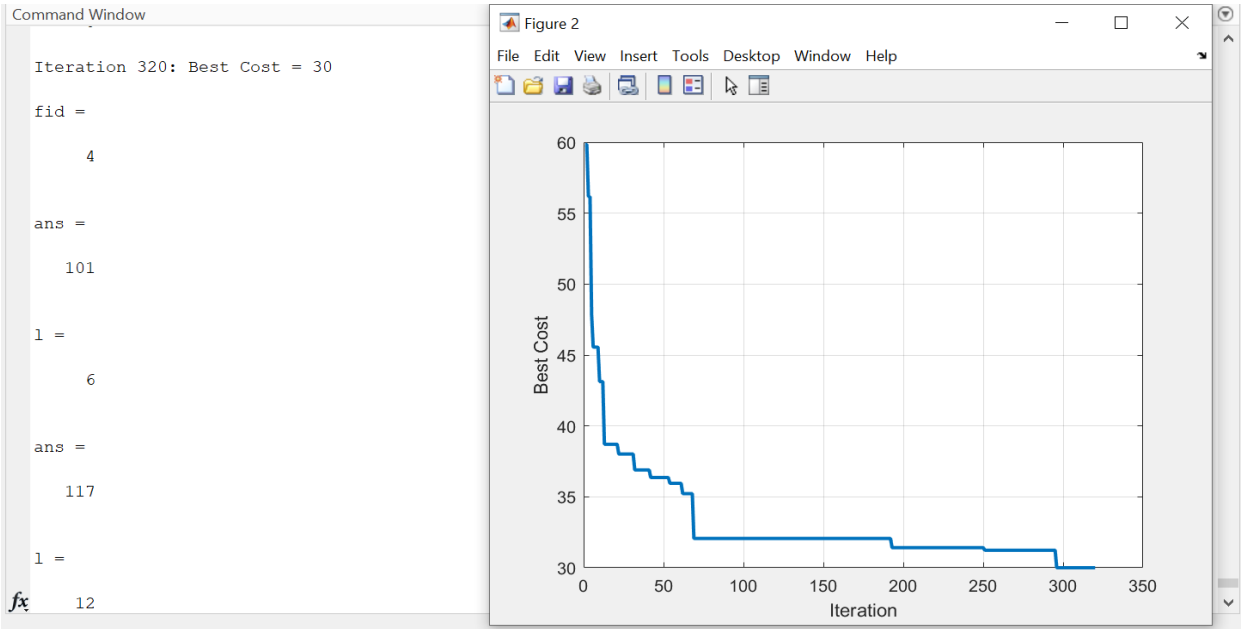
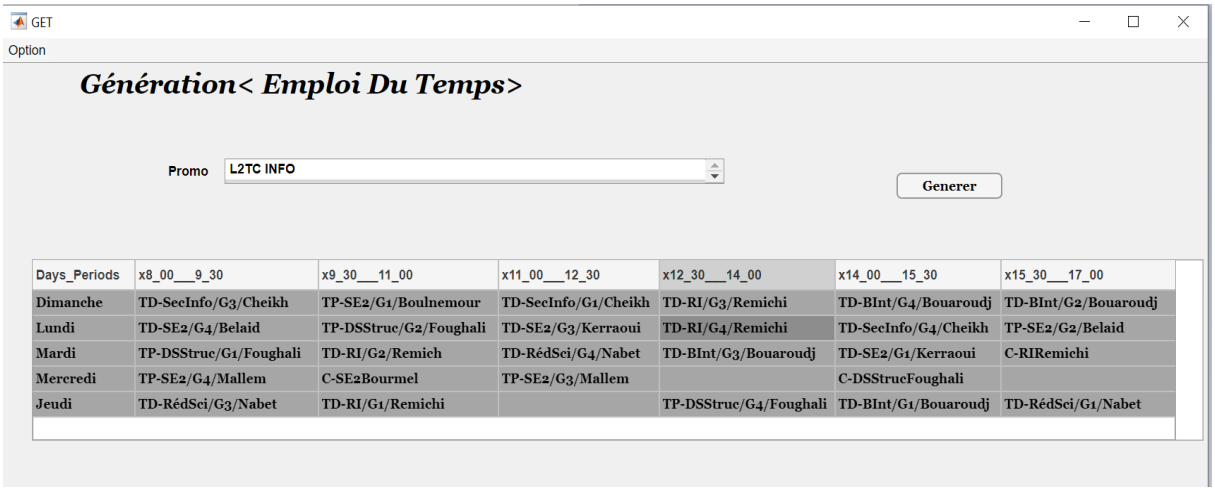


Figure IV.6: Courbe Best Cost / Itération



Days_Periods	x8_00___9_30	x9_30___11_00	x11_00___12_30	x12_30___14_00	x14_00___15_30	x15_30___17_00
Dimanche	TD-SecInfo/G3/Cheikh	TP-SE2/G1/Boulnemour	TD-SecInfo/G1/Cheikh	TD-RI/G3/Remichi	TD-BInt/G4/Bouaroudj	TD-BInt/G2/Bouaroudj
Lundi	TD-SE2/G4/Belaid	TP-DSStruc/G2/Foughali	TD-SE2/G3/Kerraoui	TD-RI/G4/Remichi	TD-SecInfo/G4/Cheikh	TP-SE2/G2/Belaid
Mardi	TP-DSStruc/G1/Foughali	TD-RI/G2/Remich	TD-RédSci/G4/Nabet	TD-BInt/G3/Bouaroudj	TD-SE2/G1/Kerraoui	C-RIRemichi
Mercredi	TP-SE2/G4/Mallem	C-SE2Bourmel	TP-SE2/G3/Mallem		C-DSStrucFoughali	
Jeudi	TD-RédSci/G3/Nabet	TD-RI/G1/Remichi		TP-DSStruc/G4/Foughali	TD-BInt/G1/Bouaroudj	TD-RédSci/G1/Nabet

Figure IV.7 : Résultat et affichage de l'emploi du temps

6 Conclusion :

Dans ce chapitre, nous avons présenté l'environnement et le processus de développement. Nous avons exposé ainsi le résultat de développement à l'aide des aperçus écran de notre application ACO adaptée pour le problème d'emploi du temps des cours universitaires.

CONCLUSION

Conclusion Générale :

Les problèmes de planification d'horaires de travail notamment le problème de l'emploi du temps des cours universitaires ont reçu une grande attention. Dans ce mémoire, nous nous sommes intéressés à l'étude de ce problème dans sa version multi objectif en utilisant l'approche d'optimisation basée colonie de fourmis.

Un algorithme ACO est proposé pour la résolution du problème de l'emploi du temps des cours universitaires au sein du département d'Informatique de l'Université 20 Août 1955-Skikda. (TimeTabling Problem : TTP)

Notre approche était réalisée avec une modélisation par un diagramme de classe. Cette spécification a été implémentée sous un langage de programmation « Matlab » Compte tenu des objectifs visés et des contraintes imposées, les résultats obtenus sont encourageants.

BIBLIOGRAPHIES

[Ala,12] : Présenté par Alaoui Abdiya 2011/2012 Application des techniques des métaheuristiques pour l'optimisation de la tâche de la classification de la fouille de données page :42-46

[Alay, 09] : author: Ines Alaya january 2009 Ecole Nationale des Sciences de l'Informatique Optimisation multi-objectif par colonies de fourmis. Cas des problèmes de sac à dos.Page :21-22-23

[Bech ,Bou,21] : Réalisé par Kenza Bechtoula & Hadjer Bouguerra. 27/09/2021Algorithme hybride (aco-ape) pour La Résolution du problème de Voyageur de commerce page :6-7 -8-22-23-24.

[Bel et Nou, 15]: Olfa Belkahla Driss, Houssem Eddine Nouri, Book · January 2015

[Ben, 18] : Présenté par : Melle. BENYETTOU Assia,2018 , Intitulé Contribution en apprentissage semi-supervisé sous contexte multi-label.

[Ben, Sou, 19] : Réalisé par : BENDJAMA Besma & SOUAMES Imen Session : Juillet 2019 Optimisation par colonie de fourmis basée agent pour le problème d'emploi du temps des cours universitaires. (Implémentation Madkit) page :11-16-17-24-25

[Cos, Lou, Mar, 06] : COSTANZO Andrea, LUONG Thé Van ,MARILL Guillaume le : 19 mai 2006 :Optimisation par colonies de fourmis page :4-5-6-7-12

[Ghr,Cux,Lam,Lel] : Article, M. GHRIBI, P. CUXAC, J.C. LAMIREL, A. LELU,

[Hach,13] : le 29 Juin 2013 présentée par Hanaâ Hachimi Hybridations D'algorithmes Métaheuristiques En Optimisation Globale Et Leurs Applications Page :31—35-36-59-60

[Kam,08] : Par Adilkamil, Fevrier 2008, Application D'un Algorithme Hybride A Colonies De Fourmis Au Probleme D'affectation Quadratique , page :59-60-61

[Kher, Oub, 17] : Présenté par :Kherbouche Lynda & Oubahri Zohra Quelques méthodes de résolutions en optimisation combinatoire Octobre 2017 page :15-16-19-23-27

[PITIOT ,09] : Présentée et soutenue par PITIOT Paul Le 25 Mai 2009 Titre : Amélioration des techniques d'optimisation combinatoire par retour d'expérience dans le cadre de la sélection de scénarios de Produit/Projet page : 44

[Sad et Ben,15] : Préparé par: Melle Meriem SADKI et Melle. Selma BENALLA, Année Universitaire 2014 / 2015, Approche multi-agents pour résoudre le problème d'emploi du temps, page : 21- 22.

[Tro,06] : Présenté par Mme Troudi Fatiha Résolution du problème de l'emploi du temps : Proposition d'un algorithme évolutionnaire multi objectif 2005-2006 page :2-19-20-21-37-52-56

https://fr.wikipedia.org/wiki/Benchmark.

https://kobia.fr/classification-metrics-precision-recall

https://www.cadremploi.fr/editorial/conseils/conseils-carriere/quest-ce-quun-benchmark

https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501339-dataset-en-machine-learning-definition-et-techniques,Mesures de qualité de clustering de documents, **Recall, Precision, F1 Score - Explication Simple Métrique en ML (inside-machinelearning.com),** Résolution multi-agents du problème d'emploi du temps universitaire, page :12-17

https://www.iro.umontreal.ca/~mignotte/IFT2425/Matlab.pdf