Réublique Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique
Université 20 Aout 1955-Skikda

Faculté des Sciences -Département d'informatique

Mémoire de fin d'études en vue de l'obtention du diplôme de

Master

Option : Réseaux et Systèmes Distribués (RSD)

Thème

Normalisation et évaluation d'une solution basée loups gris pour le problème d'emploi du temps des cours universitaires

Réalisé par :
Boussaid Nassima
Kerche Ghania

Encadreur *Dr.Zeghida Djamel*
Co-Encadreur *Khelifi Sarra*

Année universitaire 2021/2022

Remerciement

*Nous tenons à remercier **DIEU** le tout puissant de nous avoir aidé à mener à terme ce travail.*

*Nous tenons également à remercier nos encadreurs « **Dr. Zeghida Djamel**» et « **Khelifi Sarra**» pour leur aide, leur compréhension,*

Nous remercions les membres du jury, De nous avoir fait l'honneur d'accepter de juger notre modeste travail.

Enfin, que toutes les personnes qui ont contribué de près ou de loin à l'aboutissement de ce travail trouvent ici nos sincères remerciements.

Dédicaces

On dédie ce travail à : Nos très chers parents qui nous ont toujours soutenu et cru en nous.

Nos petites familles

Nos très chers frères. Nos très chères sœurs. Tous Nos amis

Aux : étudiants de la promo de Master 2 informatique 2022 et a l'équipe pédagogique du département de l'informatique de l'université 20 Août 1955 Skikda.

NASSIMA & GHANIA

Résumé

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 exhaustifs en vue de recenser le choix le plus adéquat.

Le problème d'emploi du temps est commun aux institutions académiques telle que les écoles, les collèges ou les universités. C'est un problème d'optimisation combinatoire très difficile qui suscite l'intérêt de beaucoup de chercheurs. Le problème de l'horaire des cours universitaire (PETCU) est difficile à résoudre en raison de la taille du problème et la difficulté des contraintes dures et souples.

Une normalisation des entrés des instances de ce problème selon un Benchmark spécifique au problème PETCU optimisé par l'algorithme GWO (Gray Wolf Optimizer) conçu par [Mirjalili, et al., 2014] et adapté par [K.Sarra et al, 2019] pour PETCU du département d'informatique de l'université 20 Aout 1955 Skikda est proposée dans notre travail.

Mots clés : Emploi du temps, Optimisation Combinatoire, Métaheuristiques, GWO, Benchmark.

Absract

Everyday life offers us wide range of problems where are faced with a large set of potential solutions in which it is a question of finding the best possible solution. The large number of these choices makes it impossible to explore them exhaustively in order to identify the most appropriate choice.

The timetable problem is common to academic institutions such as schools, colleges or universities. This is a very difficult combinatorial optimization problem that arouses the interest of many researchers. The problem of university course timetable (UCTTP) is difficult to address due to the size of the problem and several challenging hard and soft constraints.

A normalization of the inputs of the instances of this problem according to a Benchmark specific to the UCTTP problem optimized by the GWO (Grey Wolf Optimizer) algorithm designed by [Mirjalili and al, 2014] and adapted by [K.Sarra and al,2019] for UCTTP of the computer science department of 20 Aout 1955 Skikda university is proposed in our work.

Key words: Timetabling, Combinatory Optimization, Metaheuristics, GWO, Benchmark.

ملخـــــص

تقدم لنا الحياة اليومية مجموعة واسعة من المشاكل ونواجه في المقابل مجموعة واسعة من الحلول المحتملة التي تتعلق بإيجاد أفضل حل ممكن، بسبب هذا العدد الهائل من الخيرات يصعب بل يستحيل استكشافها جميعا لاختيار الحل الأنسب.

مشكلة الجداول الدراسية شائعة في المؤسسات الأكاديمية من مدارس وكليات وجامعات. إنها مشكلة تحسين توافقي صعبة للغاية نظرا لحجم المشكلة و لتعدد القيود الثابتة الصعبة واللينة غير أنها تثير اهتمام العديد من الباحثين.

توحيد عملية إدخال المعطيات الخاصة بالجدول الزمني تحت معيار (Benchmark) خاص بالجدولة الجامعية، هي مهمتنا في هاته الأطروحة، وعملية اختيار الجدول الزمني الأمثل حسنت باستعمال خوارزمية الذئاب الرمادية لـ [Mirjalili، وآخرون، 2014] بفضل طالبتين في الماستر [.K.Sarra et al، 2019]على مستوى قسم الإعلام الآلي بجامعة 20 أوت 1955 سكيكدة.

الكلمات المفتاحية: التحسين التوافقي، مشكل الجدول الزمني، خوارزمية تقريبية، خوارزمية استمثال الذئاب الرمادية، المعيار.

Table des matières

Chapitre 01 : Optimisation combinatoire et Méta heuristiques

Chapitre 02 : Planification d'emploi du temps des cours universitaires

Chapitre 03 : Optimisation du PETCU par GWO

Chapitre 04 : Benchmarks pour UCTTP

Chapitre 05 : Implémentation

Notions

(OR)	Operational Research
(MOSA)	Multiple Objective Simulated Annealing
(GC)	Graph Colouring
(LSDF)	Least Saturation Degree First
(MILP)	Mixed Integer Linear Programming
(IP)	Integer Programming
(TS)	Tabu search
(RPNS)	Random Partial Neighborhood Search
(SA)	Simulated Annealing
(TSSP)	Tabu Search with Sampling
(SAR)	Simulated Annealing with Reheating
(SAIRL)	Improved Reheating and Learning
(ILS)	Iterated Local Search
(SAR-2P)	SAR with two preliminary runs
(GA)	Genetic Algorithm
(ACO)	Ant Colony Optimization
(SCP)	Soft constraints penalty
(DTF)	Distance to feasibility
(PSO)	Particle Swarm Optimization
(PB-LS)	Population Based Local Search
(ADHH)	Add-Delete Hyper-Heuristic
(MOSA)	Multi-Objective Simulated Annealing
(RR)	Round Robin
(GD)	Great Deluge
(HC)	Hill Climbing
(ATS)	Adaptive Tabu Search
(HGA)	Hybrid Genetic Algorithm
(GRASP)	Greedy Randomized Adaptive Search Procedure
(VND)	Variable Neighborhood Descent
(PE-CTTP)	Post Enrollment based Course Timetabling Problem

Notions (suite)

(VNS)	Variable Neighborhood Search
(MIP)	Mixed Integer Programming
(ILP)	Integer Linear Programming
(HH)	Hyper-Heuristic
UTTP	University Timetabling Problem
(UCTTP)	University Courses Timetabling Problem
(CB-CTTP)	Curriculum Based Course Timetabling Problem
(TTP)	Timetabling Problem
(GWO)	Grey Wolf Optimization
(ITC)	International Timetabling Competition
(PETCU)	Problème d'emploi du temps des cours universitaire

Liste des Algorithmes

Liste des Tableaux

Liste des Tableaux (suite)

Liste des Figures

Introduction Générale

Introduction générale

Lié par ses capacité physiques et intellectuelles limitées, l'homme a toujours cherché à améliorer les conditions de vie. Surtout certaines taches comme soulever des objets lourds, effectuer des calculs avec précision, pour cela être humain créé des machines manuelles et automatiques, des outils et bien d'autres choses pour minimiser son intervention dans le processus de travail.

L'ordonnancement est l'un des domaines fondamentaux de l'optimisation combinatoire. Plus les problèmes d'ordonnancement des multiprocesseurs et des ateliers sont connus pour être difficiles à résoudre de manière optimale. Ainsi, la recherche se concentre sur la fourniture d'algorithmes d'approximation efficaces qui produisent une solution proche de l'optimal.

Entreprise, écoles, universités ou hôpitaux sont régulièrement, les horaires des vols par exemple, sont très crucial pour les compagnies aériennes ainsi que pour les aéroports et les passagers en raison des horaires serrés et les diverses conditions incertaines dans lesquelles les avion opèrent , horaires des trains, à l'intérieur ou à l'extérieur du villes, confrontent différents circonstances telle que les heures de pointe et les heures creuses, sont tous amenés à résoudre un problème d'emploi du temps, construire un bon emploi du temps est souvent une tache chronophage qui nécessite un effort pour répondre aux exigences et aux besoins.

Dans toutes ces catégories de problèmes d'horaires, l'université a suscité un intérêt croissant au cours des deux dernières décennies, en raison de la croissance du nombre d'établissement et d'étudiants à travers le monde, disposent de différents environnements et lieux de travail, implique plusieurs contraintes et cas réels à résoudre, et pour cela la planification d'un emploi du temps universitaire est devenue plus dur que jamais et les chercheurs ont tenté de proposer différentes méthodes et procédures dont l'objectif peut être de minimiser la durée de la période totale sur le quelle les taches doivent être planifiées, dans d'autres cas, trouver une solution réalisable sous réserve d'un délai total fixe avec contraintes, ou encore trouver une solution dans laquelle le plus petit nombre de contraintes sont violées. Généralement à l'emploi du temps dans les établissements d'enseignement, l'importance de ce problème est de réaliser un calendrier satisfasse les préférences de l'administration, les instructeurs et les étudiants.

Dans notre travail nous présentons une étude d'un problème classé comme problème complexe (NP-complet), est de normaliser les entrés selon un benchmark spécifique au ce problème résolut par les graphes de colorage et optimisé par l'algorithme d'optimisation GWO (Grey Wolf Optimizer) est une méthode de méta-heuristique bio-inspiré bien réputée, citée dans plus de 2000 travaux de recherche développé en 2014 par [Mirjalili, et al., 2014], cette méthode-là a été adapté par deux étudiantes [Khelifi & Bouzaouit, 2019] en Master en 2019, pour un emploi du temps universitaire au niveau département d'informatique de l'université 20 Aout 1955 Skikda.

Outre une introduction générale et une conclusion, ce mémoire est organisé en cinq chapitres : Dans le premier chapitre, nous introduisons les notions liées aux problèmes d'optimisation combinatoire avec un aperçu de méthode métaheuristiques existantes de la résolution proposée pour ces derniers. Dans le deuxième chapitre, nous présentons le problème de planification et plus précisément planification d'emploi du temps de cours universitaires, et nous introduisons dans le troisième chapitre l'application de l'algorithme GWO pour l'optimisation du problème UCTTP et en quatrième chapitre nous présentons les différents benchmarks existants spécifique au UTTP avec un détaille sur le la structure de notre jeu de données (dataset).

Dans le dernier chapitre, il sera question de la phase de réalisation de notre emploi du temps en utilisant des données organisées dans le benchmark élaboré.

Le mémoire sera clôturé avec une conclusion générale et des perspectives de continuité et de recherche.

Chapitre 01

Optimisation combinatoire et Métaheuristiques

1. Introduction :

L'optimisation combinatoire est un outil indispensable combinant diverses techniques de la mathématique discrète et de l'informatique afin de résoudre des problèmes d'optimisation combinatoire de la vie réelle. Un problème d'optimisation combinatoire consiste à trouver la meilleure solution dans un ensemble discret de solutions appelé ensemble des solutions réalisables. En général, cet ensemble est fini mais de cardinalité très grande (Abdesslem LAYEB 2010). Il s'agit, en général, de maximiser (problème de maximisation) ou de minimiser (problème de minimisation) une fonction objective sous certaines contraintes. Le but est de trouver une solution optimale dans un temps d'exécution raisonnable. Dans ce chapitre nous présentons certaines méthodes parmi les plus représentatives, développées pour résoudre les problèmes d'optimisation combinatoire.

2. Problème d'optimisation combinatoire

2.1. Définition

L'optimisation combinatoire est minimiser ou maximiser une fonction souvent appelée fonction coût, d'une ou plusieurs variables soumises à des contraintes. L'optimisation combinatoire occupe une place très importante en recherche opérationnelle, en mathématiques discrètes et en informatique. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire (Mostepha, R 2008). Bien que les problèmes d'optimisation combinatoire soient souvent faciles à définir, ils sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas à ce jour de solution algorithmique efficace valable pour toutes les données (Abdesslem LAYEB 2010).

2.2. Notion d'algorithme et de complexité :

La première définition de la notion d'algorithme remonte au IXème siècle, Al-Khawarizmi, un mathématicien perse, publie un ouvrage consacré aux algorithmes : l'étymologie du terme "algorithme" vient du nom de ce mathématicien. Selon Al-Khawarizmi. Un algorithme est composé d'un nombre fini d'étapes, chaque étape est composée d'un ou de plusieurs opérations (LEMOUARI, 2014)

Un algorithme est une méthode précise utilisable par ordinateur pour déterminer la solution d'un problème, à condition que l'algorithme soit correct et qu'il converge (LEMOUARI, 2014).

La théorie de la complexité consiste à estimer la difficulté ou la complexité d'une solution algorithmique d'un problème posé de façon mathématique. Elle se concentre sur les problèmes de décision qui posent la question de l'existence d'une solution comme le problème de satisfiabilité booléenne, c'est-à-dire les problèmes posant une question dont la réponse est « oui » ou « non » pour ces problèmes. La théorie de la complexité repose sur la notion de classes de complexité qui permettent de classer les problèmes de décision en fonction de la complexité des algorithmes utilisés pour les résoudre, Parmi les classes les plus courantes, on définit notamment les deux classes P et N P :

2.2.1. La classe *P* *(Polynomial time)*

Contient l'ensemble des problèmes polynomiaux, i.e., pouvant être résolus par un algorithme de complexité polynomiale. Cette classe caractérise l'ensemble des problèmes que l'on peut résoudre « efficacement ».

2.2.2. La classe NP *(Non deterministic Polynomial time)*

Contient l'ensemble des problèmes polynomiaux non déterministes, i.e., pouvant être résolus par un algorithme de complexité polynomiale pour une machine non déterministe (que l'on peut voir comme une machine capable d'exécuter en parallèle un nombre fini d'alternatives). Intuitivement, cela signifie que la résolution des problèmes de **NP** peut nécessiter l'examen d'un grand nombre (éventuellement exponentiel) de cas, mais que l'examen de chaque cas doit pouvoir être fait en temps polynomial. (Solnon, 2010)

2.2.3. La classe NP-Complet *(Problème NP-complet)*

Les problèmes NP-complets sont des problèmes combinatoires dans le sens où leur résolution implique l'examen d'un nombre exponentiel de cas. Notons que cette classe contient un très grand nombre de problèmes. En effet, pour qu'un problème soit NP-complet, il faut qu'il soit dans la classe NP, i.e., que l'examen de chaque cas puisse être réalisé efficacement, par une procédure polynomiale. Si on enlève cette contrainte d'appartenance à la classe NP, on obtient la classe plus générale des problèmes NP-difficiles, contenant l'ensemble des problèmes qui sont « au moins aussi difficiles » que n'importe quel problème de NP, sans nécessairement appartenir à NP (Solnon, 2010).

2.2.4 La classe NP-difficile

Un problème est NP-difficile (de décision ou non) 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 (Troudi, 2006).

2.3. Les problèmes combinatoires :

2.3.1. Problèmes de décision :

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

2.3.2. Problèmes de recherche :

La résolution du problème ne s'arrête plus an point de savoir si le problème admet ou non une solution. L'algorithme doit être en mesure de fournir la solution si elle existe.

Par conséquent, tout problème de décision peut être étendu à un problème de recherche.

2.3.3 Problèmes 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 objective à maximiser ou minimiser. C'est à dire on cherchera une solution de valeur optimale, minimale, si on minimise la fonction objective, ou maximale, si on la maximise (Troudi, 2006)

2.4 Les caractéristiques des problèmes d'optimisation combinatoire :

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

Les problèmes qui n'ont pas de critère d'optimisation, mais qui possèdent un ensemble de contraintes sont des problèmes de satisfaction de contraintes.

L'ajout au problème précédent un critère d'optimisation, crée un autre type de problème appelé problèmes d'optimisation combinatoire avec contraintes.

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

2.5 Résolution d'un problème d'optimisation combinatoire

Résoudre un problème d'optimisation combinatoire nécessite l'étude de trois points particuliers :

- La définition de l'ensemble des solutions réalisables,
- L'expression de l'objectif à optimiser,
- Le choix de la méthode d'optimisation à utiliser,

Les deux premiers points relèvent de la modélisation du problème, le troisième de sa résolution. Afin de définir l'ensemble des solutions réalisables, il est nécessaire d'exprimer l'ensemble des contraintes du problème. Ceci ne peut être fait qu'avec une bonne connaissance du problème sous étude et de son domaine d'application.

2.6 Les méthodes d'optimisation combinatoire :

Les méthodes d'optimisation peuvent être réparties en deux grandes classes de méthodes pour la résolution des problèmes :

2.6.1 Les méthodes exactes (déterministes) :

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d'un problème. Les algorithmes exacts les plus réussis dans la littérature appartiennent aux paradigmes de quatre grandes classes : La programmation dynamique, La programmation linéaire, Les méthodes de recherche arborescente (Branch & Bound).

2.6.2 Les méthodes approchées (stochastiques) :

Sont des méthodes d'optimisation qui ont pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie d'optimalité. Englobent deux classes :

- **Les méthodes constructives**
- **Les métaheuristiques**

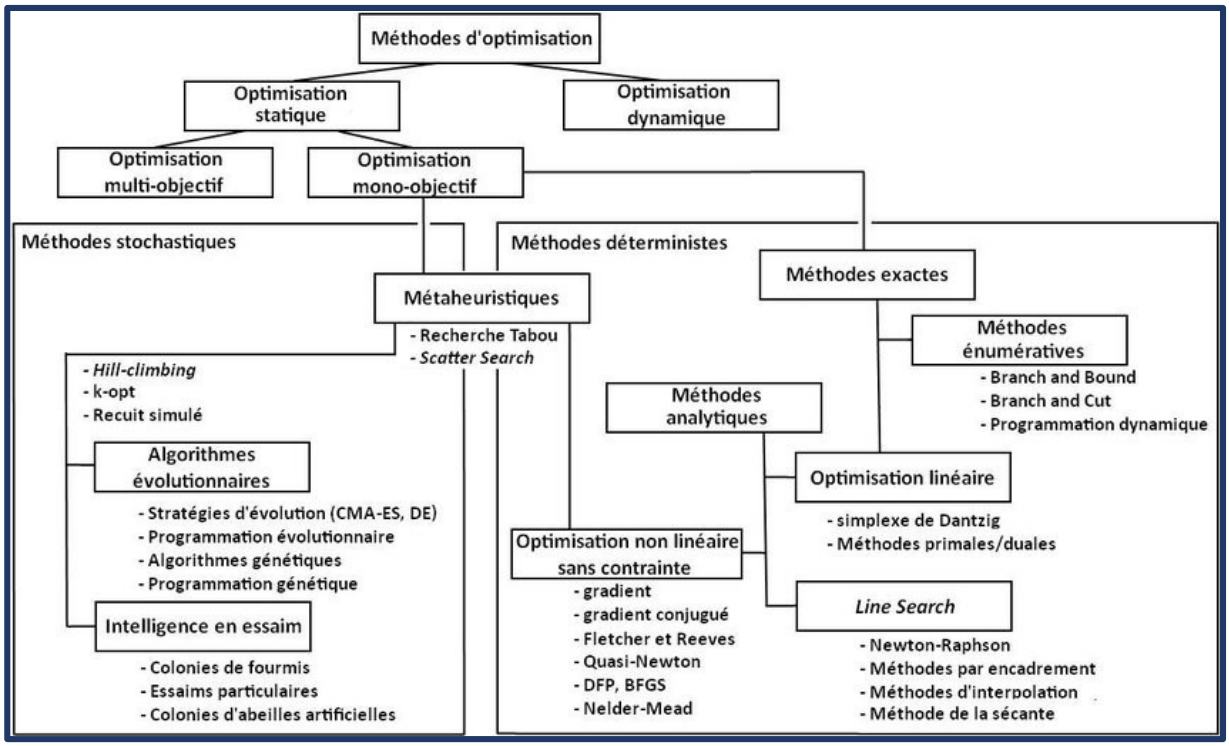


Figure1.1 : Classification des méthodes d'optimisation combinatoire [Vincent GARDEUX,2011]

3. Principe Heuristique vs. Métaheuristique :

Le mot heuristique vient du grec eurisko ϙυρισκω qui signifie « je trouve » d'où le célèbre Eureka d'Archimède.

Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation NP-difficile.

On oppose les méthodes approchées aux méthodes exactes, qui trouvent toujours l'optimum si on leur en laisse le temps (énumération complète, méthodes arborescentes, programmation dynamique).

Les approches heuristiques, contournent le problème de l'explosion combinatoire en faisant délibérément des impasses et n'explorent qu'une partie de l'espace des combinaisons.

Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre. On parle de métaheuristique pour les méthodes approximatives générales, pouvant s'appliquer à différents problèmes.

Le mot métaheuristique est dérivé de la composition de deux mots grecs : méta, du grec μοτα signifiant « au-delà » (ou « à un plus haut niveau ») et heuristique. En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Les métaheuristiques sont en général non-déterministes *(Un algorithme est dit non déterministe lorsque, appliqué sur une instance donnée du problème, donne des résultats qui peuvent varier d'une exécution à l'autre où dont le temps d'exécution varie, contrairement aux algorithmes déterministes, qui se comportent toujours de la même façon et produisent toujours le même résultat)*, elles peuvent ne pas trouver la solution optimale, et encore moins prouver l'optimalité de la solution trouvée.

On peut distinguer les métaheuristiques qui font évoluer une seule solution sur l'espace de recherche à chaque itération et les métaheuristiques à base de population de solutions. En général, les métaheuristiques à base de solution unique sont plutôt axées sur *l'exploitation* de l'espace de recherche, on n'est donc jamais sûr d'obtenir *l'optimum.* Les métaheuristiques à base de population sont plutôt *exploratoires* et permettent une meilleure diversification de l'espace de recherche.

Les métaheuristiques fonctionnent selon un comportement itératif. C'est-à-dire que le même schéma se reproduit un certain nombre de fois au cours de l'optimisation. Généralement, elles s'articulent autour des trois notions suivantes :

- La diversification (exploration)
- L'intensification (exploitation)
- La mémorisation (apprentissage)

3.1 Représentation des solutions :

La représentation de la solution doit être adaptée et pertinente pour le problème d'optimisation considéré, car la qualité d'une représentation a une influence considérable sur l'efficacité des opérateurs de recherche appliqués sur cette représentation. Quatre codages majeurs dans la littérature peuvent être soulignés : le codage binaire (par exemple : problème du sac à dos, problème de satisfiabilité), vecteur de valeurs discrètes (problème de la localisation, problème d'affectation), permutation (par exemple problème du voyageur de commerce, les problèmes d'ordonnancement) et le vecteur de valeurs réelles (par exemple des fonctions continues) [Luong ,2011]

3.2 Diversification et L'intensification :

- **La diversification** est un mécanisme qui vise à générer des solutions diverses dans le but d'explorer plus largement l'espace de recherche. Alors que l'intensification signifie de concentrer la recherche dans une région locale après avoir exploiter des informations obtenues qu'il y a de bonnes solutions dans cette région (Yang, Deb, & Fong, 2014)
- **L'intensification** se fonde sur l'idée d'apprentissage de propriétés favorables : les propriétés communes souvent rencontrées dans les meilleures configurations visitées sont mémorisées au cours de la recherche, puis favorisées pendant la période d'intensification.

Les métaheuristiques progressent de façon itérative, en alternant des phases d'intensification, de diversification et d'apprentissage, ou en mêlant ces notions de façon plus étroite. L'état de départ est souvent choisi aléatoirement, l'algorithme se déroule ensuite jusqu'à ce qu'un critère d'arrêt soit atteint. Les notions d'intensification et de diversification sont prépondérantes dans la conception des métaheuristiques, qui doivent atteindre un équilibre délicat entre ces deux dynamiques de recherche. Les deux notions ne sont donc pas contradictoires, mais complémentaires, et il existe de nombreuses stratégies mêlant à la fois l'un et l'autre des aspects.

3.3 La fonction d'évaluation :

La fonction objectif f formule l'objectif à atteindre. Elle associe à chaque point de l'espace de recherche une valeur qui décrit la qualité de la solution :S → R. Le but de la méthode d'optimisation est alors de minimiser ou de maximiser cette fonction pour un problème donné. La fonction objective représente un élément important dans la conception d'une métaheuristique. Elle va guider la recherche vers les bonnes solutions de l'espace de recherche. Si la fonction objective est mal conçue, elle pourrait conduire à des solutions non acceptables quel que soit la métaheuristique utilisée. (Bouamama et Ayadi 2020).

3.4 Caractéristiques principales des métaheuristiques : [Bulm et Roli 2003]

- Les métaheuristiques sont des stratégies qui « guident » le processus de recherche.
- Le but est d'explorer efficacement l'espace de recherche afin de trouver des solutions quasi-optimales.

- Les techniques utilisées vont de simples procédures de recherche locale aux processus complexes d'apprentissage.
- Les algorithmes sont approximatifs et généralement non déterministes.
- Ils peuvent incorporer des mécanismes pour éviter d'être piégés dans des zones confinées de l'espace de recherche.
- Les concepts de base des métaheuristiques permettent une description du niveau abstrait.
- Les métaheuristiques ne sont spécifique au problème.
- Les métaheuristiques peuvent faire usage du domaine spécifiques des connaissances sous la forme d'heuristiques qui sont contrôlées par la stratégie de niveau spécifique.

3.5 Types des métaheuristiques :

3.5.1 Les métaheuristiques basées solution unique :

3.5.1.1 Méthode de Descente (Hill Climbing) :

Cette méthode procède de manière itérative, en choisissant à chaque itération un point dans le voisinage de la solution courante. S'il est meilleur c'est à dire il améliore la solution courante, il devient la nouvelle solution courante, sinon un autre point est choisi, et ainsi de suite c'est à dire que ce procédé est répété aussi longtemps que la valeur de la fonction objective diminue. La recherche s'interrompe dès lors qu'un minimum local de f est atteint.

Algorithme 1 : Pseudo code de la Méthode de descente générique

Déterminer une configuration aléatoire s_0
Tant que le critère d'arrêt n'est pas atteint **faire**
  Générer *N(s)* ; /* Génération des candidats voisins*/
  Si pour tout $s' \in N(s)$ $f(s') \leq f(s)$ **Alors** Arrêt ;
  s := s' ; /* Sélection d'un meilleur voisin s' $\in$ N(s) */
Fin Tant que.
Résultat : Solution finale (optimum local)

Algorithme 1.1. Méthode de 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).

Plusieurs versions de la méthode de descente ont été proposées :

- descente aléatoire : la nouvelle solution est choisie aléatoirement,
- descente déterministe : le meilleur voisin est choisi,
- descente vers le premier meilleur : le premier voisin meilleur que la solution courante est choisi.

3.5.1.2 Le Recuit Simulé :

La méthode du recuit simulé est une généralisation de la méthode Monte-Carlo ; son but est de trouver une solution optimale pour un problème donné. Elle a été mise au point par trois chercheurs de la société IBM : S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Cerny en 1985 à partir de l'algorithme de Metropolis ; qui permet de décrire l'évolution d'un système thermodynamique [Omessaad, H,2003].

L'idée principale du recuit simulé tel qu'il a été proposé par Metropolis en 1953 est de simuler le comportement de la matière dans le processus du recuit très largement utilisé dans la métallurgie. Le but est d'atteindre un état d'équilibre thermodynamique, cet état d'équilibre (où l'énergie est minimale) représente - dans la méthode du recuit simulé – la solution optimale d'un problème ;

L'énergie du système sera calculée par une fonction coût (ou fonction objectif). La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objective, pour cela, un paramètre fictif de température a été ajouté par Kirkpatrick, Gelatt et Vecchi. En gros le principe consiste à générer successivement des configurations à partir d'une solution initiale S0 et d'une température initiale T0 qui diminuera tout au long du processus jusqu'à atteindre une température finale ou un état d'équilibre.

Algorithme 2 : Pseudo-code de l'algorithme de Metropolis

Initialiser un point de départ x_0 et une température T
Pour i = 1 à n **faire**
  Tant que x_i n'est pas accepté **faire**
    Si $f(x_i) \leq f(x_{i-1})$: accepter x_i
    Si $f(xi) > f(x_{i-1})$: accepter x_i avec la probabilité $e^{-(f(x_i) - f(x_{i-1}))/T}$
  Fin Tant que
Fin Pour

Algorithme 1.2. Metropolis

Algorithme 3 : pseudo-code de l'algorithme de recuit simulé

```
Déterminer une configuration aléatoire s
Choix des mécanismes de perturbation d'une configuration
Initialiser la température T
Tant que la condition d'arrêt n'est pas atteinte faire
    Tant que l'équilibre n'est pas atteint faire
        Tirer une nouvelle configuration S'
        Appliquer la règle de Metropolis
        Si f(s') < f(s)
            s_min = s'
            f_min = f(s')
        Fin Si
    Fin Tant que
Décroître la température
Fin Tant que
```

Algorithme 1.3. Recuit de Simulé

3.5.1.3 Recherche Tabou :

La méthode de recherche avec tabous, ou simplement recherche tabou (TS : Tabu Search) a été formalisée par Fred Glover en 1986 (Glover, 1986). Elle utilise explicitement l'historique de la recherche, à la fois pour échapper aux minima locaux et pour mettre en œuvre une stratégie d'exploration. Sa principale caractéristique est en effet basée sur l'utilisation de mécanismes inspirés de la mémoire humaine. De ce point de vue, la méthode taboue prend un chemin opposé à celui de SA, totalement dépourvu de mémoire, et donc incapable de tirer les leçons du passé Le principe de l'algorithme est le suivant ; à chaque itération, une nouvelle solution de voisinage est sélectionnée. La méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut-être un meilleur voisinage. Des fois, ce principe engendre des phénomènes de cyclage entre deux solutions, tandis que la méthode taboue a l'interdiction de visiter une solution récemment visitée. Pour cela, une liste taboue contenant les attributs des dernières solutions considérées est tenue à jour. Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée. Ainsi, la recherche de la solution suivante se fait dans le voisinage de la solution actuelle sans considérer les solutions appartenant à la liste taboue.

Algorithme 4 : Pseudo-code de l'algorithme de Recherche Tabou

Déterminer une configuration aléatoire s
Initialiser une liste tabou vide
Tant que le critère d'arrêt n'est pas atteint **faire**
  Perturbation de s suivant N mouvements non tabous
  Évaluation des N voisins
  Sélection du meilleur voisin t
  Actualisation de la meilleure position connue s*
  Insertion du mouvement t ⟶ s dans la liste tabou
  s = t
Fin Tant que

Algorithme 1.4. Recherche Tabou

3.5.1.4 La méthode GRASP (Greedy Randomized Adaptive Search Procedure)

Est une autre méthode basée sur la recherche locale répété, dans laquelle chaque recherche locale est initialisée avec une nouvelle solution réalisable construite grâce à une procédure de construction gloutonne et aléatoire [Feo et Resende, 1989,1995]. On retrouve donc 2 phases principales à chaque itération : la construction des solutions de départ, et une recherche locale.

En effet, l'algorithme construit une solution pour l'optimiser localement. Lorsque cela est fait, si la solution est meilleure que celle stockée précédemment, il la remplace et construit une nouvelle solution de départ aléatoirement. Un pseudo-code de l'algorithme de GRASP suit.

Algorithme 5 : Pseudo-code de l'algorithme GRASP

Déterminer une configuration aléatoire s
Tant que le critère d'arrêt n'est pas atteint **faire**
  Construction Aléatoire Gloutonne d'une solution de départ
  Recherche Locale sur le voisinage de cette solution pour l'optimiser localement
    Actualisation de la meilleure solution connue
Fin Tant que

Algorithme 1.5. GRASP

3.5.2 Les métaheuristiques basées solution population :

3.5.2.1 Les Algorithmes génétiques :

Les algorithmes génétiques Les algorithmes génétiques (GA : Genetic Algorithms) sont, sans conteste, la technique la plus populaire et la plus largement utilisée.

Les origines de ces algorithmes remontent au début des années 1970, avec les travaux de John Holland et ses élèves à l'Université du Michigan sur les systèmes adaptatifs (Holland, 1975). L'ouvrage de référence de David E. Goldberg (Goldberg, 1989) a fortement participé à leur essor.

Ces algorithmes se détachent en grande partie par la représentation des données du génotype, initialement sous forme d'un vecteur binaire et plus généralement sous forme d'une chaîne de caractères. Chaque étape de GA est associée à un opérateur décrivant la façon de manipuler les individus :

– **Sélection** : Pour déterminer quels individus sont plus enclins à se reproduire, une sélection est opérée. Il existe plusieurs techniques de sélection, les principales utilisées sont la sélection par tirage à la roulette (roulette-Wheel selection), la sélection par tournoi (tournement selection), la sélection par rang (ranking selection), etc.

– **Croisement** : L'opérateur de croisement combine les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants. Là encore, il existe de nombreux opérateurs de croisement, par exemple le croisement en un point, le croisement en n-points ($n \geq 2$) et le croisement uniforme.

– **Mutation** : Les descendants sont mutés, c'est-à-dire que l'on modifie aléatoirement une partie de leur génotype, selon l'opérateur de mutation.

– **Remplacement** : Le remplacement (ou sélection des survivants), comme son nom l'indique, remplace certains des parents par certains des descendants. Le plus simple est de prendre les meilleurs individus de la population, en fonction de leurs performances respectives, afin de former une nouvelle population (typiquement de la même taille qu'au début de l'itération).

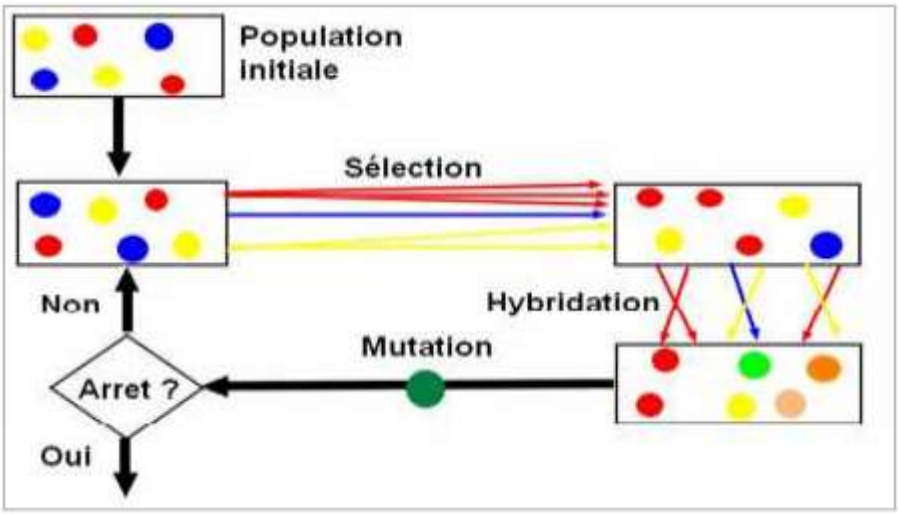


Figure 1.2. Principes Algorithme G

3.5.2.2 L'intelligence par essaim (SI : Swarm Intelligence)

Est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie (E. Bonabeau, M. Dorigo, & G. Theraulaz, 1999). Elle recouvre un ensemble d'algorithmes, à base de population d'agents simples (entités capables d'exécuter certaines opérations), qui interagissent localement les uns avec les autres et avec leur environnement. Ces entités, dont la capacité individuelle est très limitée, peuvent conjointement effectuer de nombreuses tâches complexes nécessaires à leur survie. Bien qu'il n'y ait pas de structure de contrôle centralisée qui dicte la façon dont les agents individuels devraient se comporter, les interactions locales entre les agents conduisent souvent à l'émergence d'un comportement collectif global et auto-organisée.

Deux exemples phares d'algorithmes de l'intelligence en essaim sont les algorithmes de colonies de fourmis et les algorithmes d'optimisation par essaim particulaire.

3.5.2.3 L'Optimisation par Colonie de Fourmis ACO

L'Optimisation par Colonie de Fourmis est une métaheuristique bio-inspirée proposé par M. Dorigo en 1992 (Dorigo M. , 1992). Elle se base sur le comportement social des fourmis et l'intelligence émergeante de leur communication simple à moyen de phéromone. B. METAPHORE En se basant sur l'analogie avec les colonies de fourmis biologiques, dans la méthode ACO, des agents (fourmis) communiquent entre eux à travers une phéromone représentée par un trace d'information numérique afin de construire des solutions représentées par des chemins à suivre. (Dorigo & Stützle, Ant Colony Optimization: Overview and Recent

Advances, 2010) C. ALGORITHME L'algorithme ACO était initialement conçu pour le problème du voyageur de commerce : Soit l'ensemble S= {s1, s2, …, sn} des sommets (villes). Le but de ce problème d'optimisation combinatoire est de trouver le plus court chemin passant par tous les villes de S. Les agents (fourmis) se déplacent d'une ville à une autre en déposant une quantité calculée de phéromone entre les deux villes et en enregistrant les villes visitées dans une liste taboue. La probabilité de se déplacer d'une ville i à une ville j est calculée comme suit :

$$P_{ij} = \frac{(Qte_{p[i][j]})^{\alpha} . (\eta_{ij})^{\beta}}{\sum_{j=1}^{n} (Qte_{p[i][j]})^{\alpha} . (\eta_{ij})^{\beta}}$$

Entrées :

n	Nombre des villes
d[n][n]	Matrice de distances entre les villes
A	Nombre d'agents
ρ	Taux d'évaporation
τ_{init}	Quantité initiale de phéromone dans chaque ville

Début :

Pour i := 0 à n-1 **faire**
Pour j := 0 à n-1 **faire**
τ[i][j] = τ_{init} ; //initialiser la quantité de phéromone sur tous les arcs à τ_{init}
Fin pour
Fin pour
Pour i := 0 à A-1 **faire**
p[i] := Rand (1, n) ; //placer les agents dans les villes aléatoirement
Fin Pour
Pour c := 1 à NC_max **faire**
Pour a := 0 à A-1 **faire**
Tabou_liste[a] := {∅} ;
Pour s := 0 à n-2 **faire**
Tabou_liste[s] := i ; //i est l'index de la ville courante
$v := argmax_{j<n} \left(\frac{(Qte\_p[i][j])^{\alpha} . (\eta_{ij})^{\beta}}{\sum_{j=1}^{n} (Qte\_p[i][j])^{\alpha} . (\eta_{ij})^{\beta}} \right)$
// Calculer la probabilité de déplacement et choisir une ville pour le déplacement
Déposer_pheromone () ;
$p[a] := v$; //se déplacer vers la ville v
$s := s + 1$;
Fin pour
Fin pour
Fin Pour
Retourner le plus court chemin ;
Fin

Algorithme.1.6 : Algorithme Colonie de Fourmis (ACO)

3.5.2.3 Optimisation par les Loups gris :

L'optimisation par loups gris (Grey Wolf Optimizer) est une métaheuristique de la troisième classe proposée par Mirjalili et al. [MIRJALILI, MIRJALILI, &LEWIS, 2014] en 2014, et comme toutes les métaheuristiques de cette classe, cette méthode est basée population. Elle est unique comme elle suit le comportement de la hiérarchie de chasse des loups gris. Les loups gris appartiennent à la famille des canidés. Ils préfèrent vivre en groupes appelés meutes. Le groupe se compose de 12 à 15 membres en moyenne. Ils suivent une stricte hiérarchie sociale dominante.

Figure 1.3. Les étapes de la chasse des loups [MIRJALILI, MIRJALILI, &LEWIS, 2014]

(A) Poursuite de la proie, (B-C-D) harcèlement et encerclement, (E) Position d'attaque.

A. Hiérarchie sociale des loups gris dans la nature :

- Dans le premier niveau, on a l'alpha (α). Il est chargé de prendre des décisions concernant la chasse, le déplacement…etc. tous les loups de la meute doivent se soumettre à l'alpha.
  - Le deuxième niveau est le beta (β). Il aide l'alpha à prendre les décisions.
  - Le troisième niveau c'est les delta (δ). Ils peuvent être des sentinelles, des aînés, des chasseurs ou des gardiens.
  - Dans le denier niveau c'est les gamma (γ). Ils sont considérés comme des boucs émissaires. Ils doivent se soumettre à tous les autres membres de la meute.

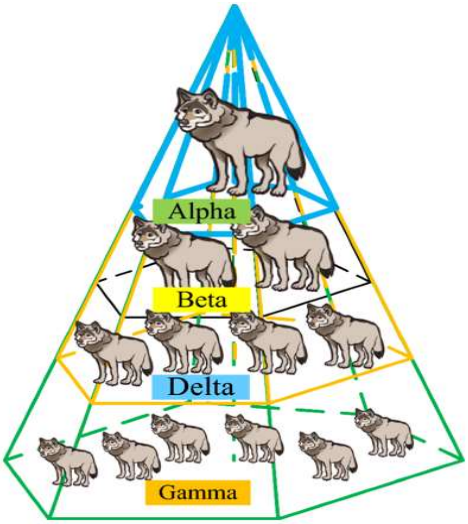


Figure 1.4 : Hiérarchie des Loups gris [Hamouda.Ch, et al ,2019]

Selon Muro et al. (Muro, Escobedo, Spector, & Coppinger, 2011), la chasse passe par trois phases principales :

- Approcher la proie
- Encercler la proie
- Attaquer la proie

B. Mécanisme de la chasse :

a. Encercler la proie :

Dans la nature, les loups encerclent la proie et l'approchent de toutes les directions. Cette étape peut être modélisée à travers les équations mathématiques suivantes :

$$\vec{D} = |\vec{C}.\vec{X}_p(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D}$$

Où $\vec{D}$ est la distance entre la proie et le loup, **t** indique l'itération courante, $\vec{X}_p$ est la position de la proie, $\vec{X}$ est la position du loup et $\vec{A}$ et $\vec{C}$ sont des vecteurs coefficients calculés comme suit :

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a}$$

$$\vec{C} = 2.\vec{r}_2$$

Où les composants de $\vec{a}$ sont diminués linéairement de **2** à **0** à travers les itérations et $\vec{r}_1$, et $\vec{r}_2$ sont des vecteurs aléatoires entre **0** et **1**. La valeur du vecteur $\vec{a}$ dans une itération quelconque t peut être obtenue par l'équation suivante [Panda.M,Das.B, 2019] :

$$\vec{a} = 2 - t\left(\frac{2}{N_{iter}}\right)$$

Les équations de calcul de $\vec{A}$ et $\vec{C}$ permettent au loup gris de se déplaces vers n'importe quel point aléatoire autour de la proie, ce concept peut être étendu à un espace de recherche à n dimensions ou les loups se déplacent dans des hypercubes autour de la proie. Cet effet peut être démontré á travers la figure suivante :

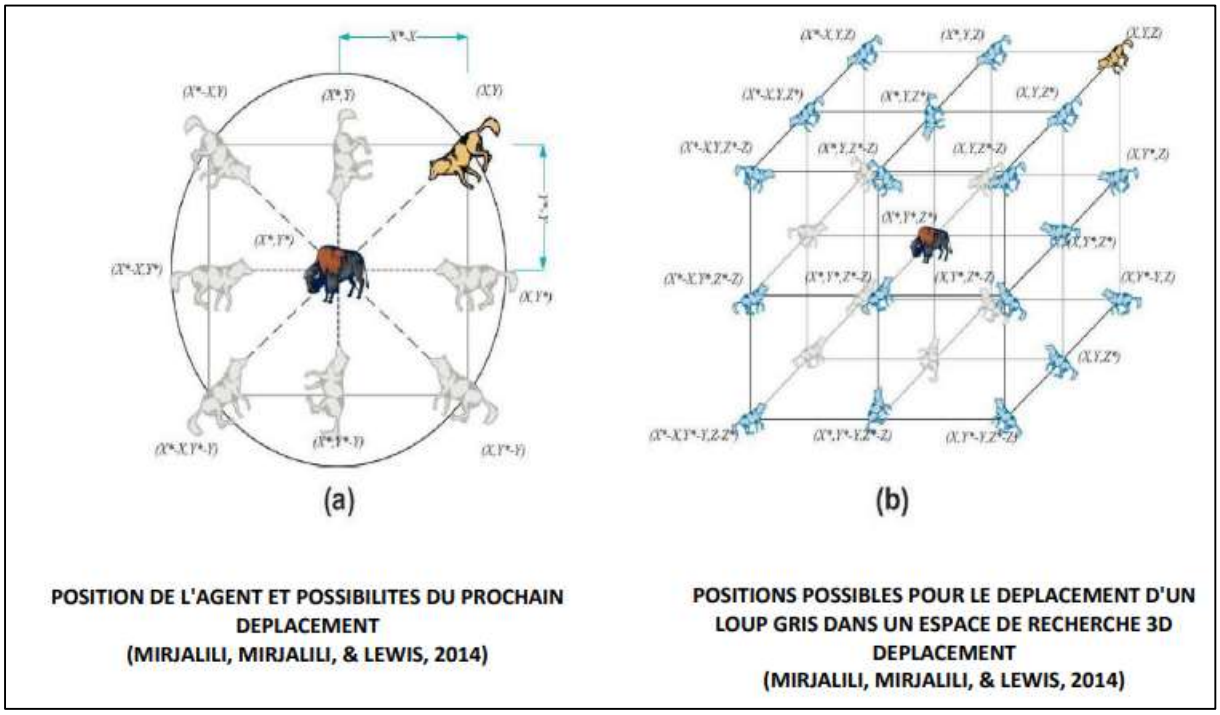


Figure 1.5 : Positions possible de déplacement [MIRJALILI, MIRJALILI, &LEWIS, 2014]

b. La chasse :

Les loups ont la capacité de trouver la position des proies. La chasse est généralement guidée par Alpha. Occasionnellement, Bêtas et Deltas peuvent aussi participer à la chasse. Cependant, dans un espace de recherche abstrait, nous n'avons aucune idée de l'emplacement de l'optimum (la proie). Afin de modéliser ce mécanisme, on suppose que les trois agents Alpha, Bêta, et Delta ont une meilleure connaissance de l'emplacement de la proie. Par conséquent, on sauvegarde les trois meilleures solutions obtenues et on oblige les autres agents loup à mettre à jour leurs positions en fonction de la position des trois meilleurs loups. Les formules suivantes ont été proposées à cet égard [S. Mirjalili, S. M. Mirjalili et A. Lewis, 2014] :

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}(t)|, \vec{D}_\beta = |\vec{C}_2.\vec{X}_\beta - \vec{X}(t)|, \vec{D}_\delta = |\vec{C}_3.\vec{X}_\delta - \vec{X}(t)|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.(\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.(\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.(\vec{D}_\delta)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$

La (Figure1.6) représente le comportement des autres agents dans un espace de deux dimensions. Il est apparu que la position finale des loups serait dans un cercle défini par la position des trois loups Alpha, Bêta et Delta dans l'espace de recherche. Autrement dit Alpha, Bêta, et Delta estiment la position de la proie et les autres loups mettent à jour leurs positions aléatoirement autour de la position estimée de la proie.

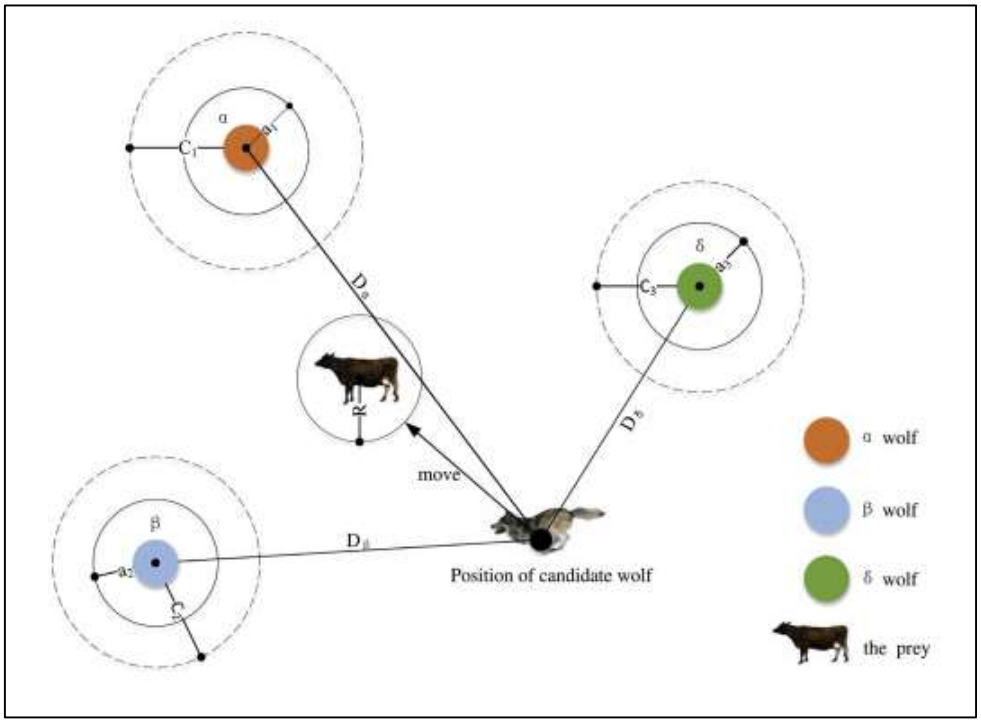


Figure 1.6 : Mise à jour des positions des loups [Peifeng .N, et al,2019]

Wolf / Loup, the prey /La proie, move / deplacer, Position of candidate wolf / position du condidat loup.

c. Attaquer la proie

Ce mécanisme ce manifeste dans l'algorithme à travers la notion d'exploitation. Avec la diminution du paramètre, on pousse les agents à se rapprocher vers la proie puisque $\vec{A}$ est dans l'intervalle $[-2\vec{a}, 2\vec{a}]$. Quand $\vec{A}$ est dans l'intervalle $[-1, 1]$, la prochaine position de l'agent doit être plus proche de la proie.

C. **Exploration et exploitation** :

L'algorithme **GWO** a deux paramètres de contrôle principaux : les variables C et A. Le calcul de ces deux paramètres permet de maintenir l'équilibre entre l'exploration et l'exploitation. D'un côté, la première variable C donne une valeur aléatoire dans l'intervalle [**0, 2**] quel que soit le nombre de l'itération ; ce qui encourage l'exploration de l'espace de recherche et permet de prévenir la stagnation dans un optimum locale. De l'autre côté, la valeur de la deuxième variable est basée sur a qui est diminué linéairement de **2** à **0**. Cette variable a aussi un composant aléatoire qui permet à A de changer sa valeur dans l'intervalle [**-2, 2**] ce qui lui permet de promouvoir l'exploration quand $A > 1$ ou $A < -1$ et l'exploitation quand $-1 < A < 1$. [Faris, Aljarah, Al-Betar, & Mirjalili, 2018] La progression de a augmente les chances que A soit dans l'intervalle [**-1, 1**] vers la fin du processus ce qui favorise l'exploitation dans les dernières itérations pour améliorer les solutions obtenues.

Algorithme GWO
Début
1) Initialiser la population des loups gris $\mathbf{X_i}$ (i=1, 2, …, n) ;
2) Initialiser $\vec{a}$, $\vec{A}$ et $\vec{C}$;
3) Calculer la valeur objectif correspondante pour chaque Loup ;
4) Choisir les trois premiers meilleurs loups et sauvegarde-les sous $\vec{X}\boldsymbol{\alpha}$, $\vec{X}\boldsymbol{\beta}$ et $\vec{X}\boldsymbol{\delta}$;
5) **Tant que (t < Max** nombre d'itérations **)**
   Pour chaque loup de recherche
   Mettre à jour la position du loup de recherche actuel en utilisant l'équation $\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$
   FinPour
   Mettre à jour les paramètres $\vec{a}$, $\vec{A}$ et $\vec{C}$;
   Calculer la valeur objectif correspondante pour chaque Loup ;
   Actualiser $\vec{X}\boldsymbol{\alpha}$, $\vec{X}\boldsymbol{\beta}$ et $\vec{X}\boldsymbol{\delta}$;
   t=t+1 ;
   FinTq
6) Renvoyer $\vec{X}\boldsymbol{\alpha}$.
Fin

Algorithme 1.7 : GWO [S. Mirjalili, S. M. Mirjalili et A. Lewis, 2014]

4. Conclusion :

Dans ce chapitre, nous avons défini les notions d'optimisation combinatoire ainsi un détaille sur les métaheuristiques, et ensuite nous avons présenté GWO (Grey Wolf Optimization inspiré des loups gris) qui est un algorithme efficace et flexible. Ses paramètres de contrôle permettent de maintenir l'équilibre entre l'exploration et

l'exploitation et l'adaptabilité de son mécanisme de déplacement permet de l'appliqué sur une variété étendu des problèmes. Le chapitre suivant sera consacré à exposer la planification d'emploi du temps universitaire.

Chapitre 02

Planification d’emploi du temps des cours universitaires

1. Introduction :

Un emploi du temps est un calendrier de travail, où figure à la fois le temps, l'affectation des enseignants, des salles et des modules enseignés. La gestion des emplois du temps est une tâche très complexe, cette complexité vient du fait qu'on a un ensemble de contraintes à respecter, et des conflits à résoudre. Le problème de l´emploi du temps est un problème ardu dont la réalisation à la main est une tâche draconienne qui peut mobiliser plusieurs personnes plusieurs jours de travail. Sans oublier, que toute modification des données du problème peut complètement remettre en cause la solution trouvée. Pour cela nous définissons les différents types de plannings dans différents domaines de travail et plus particulièrement dans le domaine pédagogique.

2. La problématique de la planification d'horaires de travail

Les problèmes de planification d'horaires de travail se retrouvent autant de les entreprises d'industrie que dans les services publics tels que : la santé, l'éducation etc…

La planification d'horaires de travail est un processus très complexe, qui vise à organiser des activités humaines (principalement de travail) dans le temps et à optimiser l'utilisation des ressources, de façon à couvrir un besoin exprimé par une charge de travail prévisionnelle sous diverses contraintes. Elle aboutit à des programmes définissant les horaires de travail et de repos de la force de travail [Chan Y. C. P., 2002].

Pour mieux cerner ce qui est la planification et la complexité à sa réalisation, on s'intéresse à un ensemble de questions :

2.1 Qu'est-ce que la planification

La planification est un instrument de gestion dont l'objectif est d'aboutir à des programmes permettant d'organiser et planifier le travail des salariés afin de rester pérenne dans l'économie globale. Ceci passe par la détermination des capacités de tout un chacun et par le recensement des activités futures et des besoins en personnel.

La planification vise à affecter les ressources humaines pour chaque intervalle de temps sur un horizon donné, de telle manière que les besoins par intervalle soient couverts et que les différentes contraintes soient satisfaites [Chan Y. C. P., 2002].

2.2 Qu'est-ce qu'un planning

Les plannings sont des calendriers de travail, où figurent à la fois le temps, l'affectation du personnel, les jours et les horaires de travail, et les congés et repos [Kennedy.J , Eberhart.R ,1997]. En effet, ils apparaissent dans les cas suivants :

- Si le travail doit être assuré pendant plus d'une journée, il faut prévoir la succession de plusieurs personnes sur le même poste dans la journée. Un outil d'aide est nécessaire lorsque le nombre de postes dépasse la quinzaine, par exemple gérer les absences imprévues des salariés.

- Si le travail doit être assuré pendant plus de 35 heures par semaine, un outil automatique devient indispensable lorsque le nombre de postes dépasse la trentaine pour gérer la succession de plusieurs personnes dans la semaine, ainsi que les absences imprévues.

Les plannings peuvent être utilisés pour planifier les horaires de présences du personnel où les tâches effectuées par le personnel :

- **Planning des horaires de présence** : ce type de planning est utilisé pour prévoir les horaires de présence du personnel sans préciser les tâches journalières à effectuer soit pour des raisons de sécurité, soit pour une meilleure souplesse.

- **Planning des tâches** : ce type de planning est utilisé dans les entreprises à haute technicité, comportant plusieurs métiers et compétences distincts, où il est souhaitable d'affecter le personnel en fonction des tâches. Ce qui exige une décomposition fine des opérations et le repérage des tâches que chaque personne est capable d'accomplir.

Les plannings peuvent être journaliers (spécifiant les pauses et périodes de travail de la journée de chaque employé), hebdomadaires (utilisés pour une paie hebdomadaire), mensuels (utilisés pour le calcul des coûts pour les besoins de la paie mensuelle) ou annuels (permettant de gérer les congés annuels des employés).

Selon leur spécificité et les branches d'activités concernées, les plannings portent différents noms. Un planning spécifiant les programmes de travail de chaque employé nominativement sur un horizon (un intervalle de temps où un planning est élaboré) d'un mois est appelé tableau de service. Lorsque le planning représente les programmes de travail et de repos non nominatifs sur un nombre entier de semaines, on parle de grille de travail.

Certains plannings sont cycliques, s'ils reflètent une certaine périodicité des horaires individuels c'est à dire si au bout d'une durée D (mesurée généralement en semaine), le salarié retrouve son planning de départ. Autrement, ils sont dits acycliques' est à dire ils sont différents chaque semaine.

2.3 A Quoi sert un planning

Depuis le début des années 80, la gestion des ressources humaines a été reconnue comme une activité stratégique pour l'entreprise. Avec cette reconnaissance, l'intérêt d'élaborer des plannings s'est vu accroître de plus en plus car ils permettent :

- aux entreprises exerçant une activité continue ou quasi-continue de répartir convenablement leur personnel (compagnies aérienne, entreprises de transports, hôpitaux, etc…),
- aux entreprises cherchant à se rendre plus accessibles à la clientèle d'étaler les horaires d'ouverture (grands magasins, banques, etc…),
- à toutes les entreprises de surmonter leurs exigences de productivité et de mieux gérer les présences et absences de leur personnel. Les situations où un planning est utile sont nombreuses. Elles justifient même système : plannings à court, moyen et à long terme [Chan Y. C. P., 2002].

2.4 Comment est évalué un planning

Pour que les plannings élaborés soient satisfaisants, ils doivent vérifier un ensemble de contraintes et établir un meilleur compromis entre les différents acteurs (Exemple : le chef d'entreprise, le planificateur, le commercial, le syndicaliste et le salarié).

Lorsque les différentes solutions alternatives sont connues, une négociation se déroule de la manière suivante : chaque acteur donne son opinion. Les points d'accord sont très vite expédiés et les points litigieux sont débattus. Et des solutions de compromis sont dégagées.

Les difficultés de négociation augmentent avec le nombre d'acteurs et le nombre de solutions alternatives. L'aspect combinatoire (pour l'élaboration des plannings) rend d'autant plus difficile la négociation, car les opinions sont plus difficiles à formuler [Remy-R, Alexander-J, 2003]. Les moyens informatiques apportent une aide certaine notamment dans l'acquisition et la confrontation des données individuelles.

2.5 Qui peut se charger de l'élaboration d'un planning

Dans la plupart des entreprises, cette tâche peut être centralisée ou déléguée à des cadres de l'entreprise appelés planificateurs. Le planificateur doit prendre la décision qui correspond le mieux aux préférences des différents acteurs, justifier son choix, car son expérience de la tâche fait de lui un interlocuteur privilégié pour évaluer rapidement et effectuer des jugements de l'orientation à donner à la recherche de solutions de meilleure qualité afin d'aboutir à un choix pertinent. Une collaboration réussie doit permettre au planificateur de participer efficacement à l'élaboration des plannings. La génération automatique des plannings joue un rôle primordial.

3. Différents types de plannings

Dans la construction de plannings d'horaires de travail, si créer un planning optimisé d'une journée est aisé, mais créer un bon planning pour un mois ou une année est beaucoup plus complexe. En plus de la complexité combinatoire du problème, il faut tenir compte de la diversité des contraintes applicables et qui sont souvent contradictoires.

3.1 Notions fondamentales

Parmi les objectifs de la théorie de complexité est de classer les problèmes en fonction des ressources (temps de calcul, espace mémoire, etc…) Nécessaires à leur résolution algorithmique. Ceci a montré qu'il existe des problèmes qui ont une solution calculable mais dont toute réalisation effective sur une machine est pratiquement inutilisable parce que le temps de calcul ou la place mémoire nécessaire sont trop importants. L'enjeu est donc de discerner l'existence de différentes formes de plannings dans un autre les problèmes qui ont une solution réalisable et ceux qui, quelles que soient les améliorations futures des machines, ne peuvent intrinsèquement avoir une telle solution.

Quand on prend pour critère le temps d'exécution, on exprime cette frontière en considérant qu'un problème est réalisable si son temps d'exécution est polynomial en fonction de la taille de l'entrée. Si le degré du polynôme est élevé cette notion devient un peu irréalisable, mais la distinction entre temps polynomial et temps non polynomial donne naissance à une classification des problèmes qui est très utile pratiquement [Troudi-F, 2006].

3.1.1 Notion de complexité

D'une manière générale, pour résoudre un problème, on est appelé à trouver l'algorithme le plus efficace. Cette notion d'efficacité induit normalement toutes les ressources de calcul nécessaires pour exécuter un algorithme. Or, le temps d'exécution est généralement le facteur dominant pour déterminer si un algorithme est assez efficace pour être utilisé dans la pratique, pour cela on se concentre principalement sur cette ressource.

On appelle complexité en temps d'un algorithme le nombre d'instructions élémentaires mises en œuvre dans cet algorithme afin de résoudre un problème donné.

Une instruction élémentaire sera une affectation, une comparaison, une opération algébrique, la lecture et l'écriture etc.… Mais comme le décompte précis de toutes les instructions d'un programme risque d'être assez pénible, et qu'entre deux exécutions du même algorithme avec un jeu de paramètres différent, le nombre d'instructions exécutées peut changer, on se contentera, en général, d'apprécier un ordre de grandeur de ce nombre d'instructions. C'est ce qu'on désigne sous le nom de complexité de l'algorithme. Donc pour mesurer la complexité temporelle d'un algorithme, on s'intéresse plutôt aux opérations les plus coûteuses [Troudi-F, 2006] :

- Racine carrée, Log, Exp, Addition réelle…
- Comparaisons dans le cas des tris…

On dit que $f(n) = O(g(n))(f(n)$ est de complexité $g(n))$, chaque fois qu'il existe

et n_0 tels que : $n > n_0 \Rightarrow f(n) < kg(n)$

3.1.2 Algorithmes polynomiaux et Algorithmes exponentiels

Un algorithme polynomial est un algorithme dont la complexité est O(p(n)) où p est une fonction polynomiale et n dénote la longueur de données. Tout algorithme dont la complexité ne peut être borné par un tel polynôme d'ordre n, est un algorithme exponentiel (bien que cette définition inclue certaines complexités non-polynomiales comme n* log (n), qui n'est pas considéré comme fonction exponentielle) [Troudi-F, 2006].

3.1.3 Problèmes 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 laquelle on est censé répondre [Troudi-F, 2006].

3.1.4 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 [Troudi-F, 2006].

3.1.5 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 [Troudi-F, 2006].

3.1.6 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 objective, et maximale, si on la maximise [Troudi-F, 2006].

3.1.7 La réduction polynomiale

On dit qu'un problème P1 se réduit polynomiale en un problème P2, s'il existe un algorithme polynomial A construisant, à partir d'une donnée D1 de P1, une donnée D2 de P2 telle que la réponse pour D1 soit oui si et seulement si la réponse pour D2 est oui [Troudi-F, 2006].

3.1.8 La réduction de Turing

La réduction polynomiale permet de comparer les problèmes de décision. La réduction de Turing permet de comparer les problèmes de recherche. On dit qu'un problème de recherche P1 se réduit polynomiale à un problème de recherche P2 par la réduction de Turing s'il existe pour résoudre P1 un algorithme A1 utilisant comme sous–programme un algorithme A2

3.2 Types de plannings dans le domaine de la santé

Les plannings dans les domaines de la santé sont des calendriers de travail où figurent à la fois le temps, et l'affectation des personnels (jours et horaires de travail, congés et repos). Ils sont établis au niveau de chaque équipe, ils sont à la fois une tâche, un document

d'organisation du travail, et un élément contribuant à la gestion administrative du personnel. Cette tâche est parmi les plus difficiles et les plus délicates. Difficile parce qu'elle repose sur la recherche de solutions combinatoire, répond à des contraintes multiples.

3.3. Types de plannings dans le domaine de transport

Le transport est une activité complexe qui fait intervenir des investissements lourds, du personnel qualifié et une informatique très coûteuse, En effet, dans le transport routier, il est toujours nécessaire de gérer aux mieux les ressources existantes en optimisant les investissements.

3.4 Types de plannings dans le domaine de la pédagogie

Nous allons d'abord présenter le problème d'emploi du temps en général puis nous le particularisons au cas de celui universitaire.

3.4.1 Problèmes d'emploi du temps

Le problème d'emploi du temps est l'un des problèmes de calcul difficiles dans l'ordonnancement. Dans le processus de génération d'un emploi du temps, le but est de trouver des créneaux horaires adaptés à un nombre de tâches qui nécessitent des ressources limitées. Selon la nature du problème d'emploi du temps, les contraintes peuvent se varier et donc il se pourrait qu'il y ait de nombreux objectifs différents. Dans certains cas l'objectif est de trouver une solution réalisable dans une durée totale fixée au préalable. Dans d'autres cas, le but est de réduire la période durant laquelle les tâches doivent s'exécuter. On peut même tomber dans des cas où l'objectif est de trouver une solution qui viole un nombre minimum de contraintes.

Le problème de génération d'emploi du temps peut survenir dans de nombreux contextes différents. Il peut figurer dans le cadre de gestion des établissements de santé où il faut créer un calendrier (hebdomadaire, mensuel ou annuel) pour les employés (médecins, infirmiers, laborantins, ...) qui gère les jours et heures de travail pour chacun, les jours de permanences, ... etc. Dans le secteur du sport apparaît bien ce problème. L'exemple du football illustre aussi bien l'utilité de la planification d'emploi du temps : un championnat concerne un ensemble d'équipes qui font des compétitions entre elles dans des stades de football précis durant des périodes précises en respectant plusieurs contraintes. On peut en citer les contraintes de disponibilité des stades et des équipes concernées par les matchs, prendre en considération que ces équipes peuvent participer en plusieurs championnats. Enfin, ce problème se réfère évidemment à l'emploi du temps dans le domaine scolaire. L'emploi du temps universitaire, qui est notre cas d'étude, fait partie de ce dernier domaine.

3.4.2 Problème d'emploi du temps universitaire

Certains considèrent le problème de génération d'emploi du temps universitaire comme une instance des problèmes d'ordonnancement cyclique. Sa résolution consiste à ordonnancer les tâches d'un ensemble d'enseignants et de groupes appartenant à des sections en leur allouant un ensemble de locaux et en leur fixant des créneaux horaires, ces tâches ont un caractère cyclique. D'autres le voient comme un problème NP-complet. Ce type de problèmes est réputé complexe et sa résolution optimale exige un temps exponentiel. C'est pour cela qu'on envisage d'utiliser les heuristiques :

Ces deux visions ne sont pas contradictoires vue que la première consiste à chercher une solution réalisable à ce problème (problème d'emploi du temps vu comme un problème de recherche) et la seconde consiste à tenter de spécifier aux séances des périodes ardemment précises (problème d'emploi du temps vu comme un problème d'optimisation).

Le processus de génération d'emploi du temps universitaire pourrait être formellement défini avec une matrice binaire X_{egsjpl} qui prend la valeur **1** si l'enseignant e peut enseigner la séance s avec le groupe g dans la période p du jour j dans le local l.

Contraintes dures (hard) : cette classe représente le type de contraintes dont on n'a pas le droit de les enfreindre.

- Un enseignant ne pouvait pas assister à deux cours en même temps.
- Les enseignants doivent être disponibles aux périodes prévues pour leur cours
- Aucun groupe d'étudiant ne participe à plus d'un cours en même période.
- Plusieurs cours programmés dans le même intervalle ne peuvent pas être attribué dans la même salle.
- La salle doit satisfaire les caractéristiques requises par le cours.
- Le nombre de groupe d'étudiants participant au cours devrait être inferieur ou égale à la capacité de la salle.

Contraintes souples (soft) : c'est l'ensemble de contraintes de préférence. En les relâchant, la sortie reste toujours valide mais pas optimale.

- L'enseignant peut avoir le choix de proposer en priorité certaines périodes pour cours.
- Un enseignant peut demander une salle spéciale pour un cours donné.
- Les cours doivent être planifiés de manière à minimiser les périodes creuses de

l'enseignant et des étudiants.

- Programmer une pose de déjeuner.
- Les étudiants ne devraient pas avoir des séances dans la dernière période de la journée.

4. Les différentes approches de résolution

4.1. Les méthodes séquentielles

Ces méthodes ordonnent les évènements en les assignant séquentiellement dans des périodes de temps valides pour qu'aucun évènement dans une période ne soit en conflit avec un autre dans une période de temps donné. Dans les méthodes séquentielles, les problèmes de confection d'horaires sont généralement représentés par des graphes où les évènements (cours/examens) sont représentés par des nœuds et les conflits entre les évènements sont représentés par les arcs [Wang L, 2013]. Par exemple si quelques étudiants doivent suivre deux évènements, il y a un arc entre les nœuds qui représentent le conflit. La construction d'un emploi du temps peut donc être modelée comme un problème de coloration de graphe.

Chaque fois la période dans l'emploi du temps correspond à une couleur et les nœuds du graphique sont colorés de telle façon qu'aucun des nœuds adjacents ne soit coloré par la même couleur. Une variété d'heuristiques de coloration de graphe pour la construction des conflits d'emploi du temps est disponible dans la littérature. Ces heuristiques ordonnent les évènements basés sur une évaluation de comment il est difficile de les prévoir.

Les heuristiques qui sont souvent employées sont

- le plus grand degré d'abord : Les évènements qui ont un grand nombre de conflits avec d'autres évènements sont prévus tôt. Le raisonnement est que les évènements avec un grand nombre de conflits sont plus difficiles à prévoir et donc doivent être abordé d'abord.

- le plus grand degré pondéré : C'est une modification du plus grand degré d'abord où le poids de chaque conflit est représenté par le nombre d'étudiants impliqués dans le conflit.

- le degré de saturation : Dans chaque pas de la construction de l'emploi du temps, l'événement qui a un nombre petit de périodes valables disponibles pour la planification dans l'emploi du temps est choisi lointainement.

4.2. Les méthodes basées contraintes

Dans ces méthodes, le problème d'emploi du temps est modelé comme un jeu de variables (c.-à-d. les évènements), les valeurs (c.-à-d. les ressources comme les pièces et les périodes) vont être assignées pour satisfaire un certain nombre de contraintes. D'habitude quelques règles sont définies pour l'assignation de ressources aux évènements. Quand aucune règle n'est applicable à la solution partielle actuelle un retour arrière est exécuté jusqu'à une solution est trouvée qui satisfait toutes les contraintes [Troudi-F, 2006].

4.3 Les méthodes méta heuristiques

Pendant les deux dernières décennies une variété d'approches méta heuristiques comme le recuit simulé, la recherche taboue, les algorithmes génétiques, les colonies de fourmis et les approches hybrides ont été étudiées pour la résolution du problème d'emploi du temps. Les métas heuristiques commencent par une ou plusieurs solutions initiales et emploient les stratégies de recherche qui essayent d'éviter des optimums locaux. Tous ces algorithmes de recherche peuvent produiront des solutions de qualité mais ont souvent un coût informatique considérable.

5. Conclusion

Parmi tous les types de plannings cités dans ce chapitre, c'est sur les plannings pédagogiques que nous allons porter notre intérêt, et plus particulièrement sur le planning des cours universitaires. Le chapitre suivant sera consacré à exposer l'adaptation de GWO au ce type de planning.

Chapitre 03

Optimisation du PETCU par GWO

1. Introduction :

Comme on a vu dans le chapitre1 le GWO est une métaheuristique bio-inspiré basée population qui est inspirée d'un côté de la hiérarchie sociale des loups gris dans une meute pour définir la solution optimale, et de l'autre côté du mécanisme de la chasse pour le processus d'optimisation qui a pour but de faire des déplacements en utilisant des opérateurs pour approcher une solution qui respecte le plus possible les objectifs, visés durant les itérations. On va présenter dans ce chapitre l'application de la métaheuristique GWO pour le problème d'emploi du temps des cours universitaires en prenant comme cas d'étude le département d'informatique de l'université 20 Août 1955-Skikda pour bien observer l'influence du processus d'optimisation en réalité.

2. L'algorithme GWO pour l'emploi du temps universitaire :

La métaheuristique des loups gris est une méthode basée population c'est-à-dire avant d'entamer le processus d'optimisation il faut fournir un ensemble de solutions admissibles. Dans notre cas une solution admissible est un emploi du temps qui ne comporte aucun type des conflits décrits par les contraintes dures (la méthode utilisée pour la préparation des solutions initiales sera présentée dans la section suivante).

Les problèmes peuvent être classés selon leurs propriétés de l'ensemble des solutions et la complexité de leurs algorithmes : Cet algorithme est une adaptation de l'algorithme GWO (Gray Wolf Optimizer) conçu par [Mirjalili, et al., 2014] pour le problème d'emploi du temps des cours universitaires. Il est conçu par deux étudiantes de master à l'université de 20 Aout 1955 [Khelifi, et al., 2019] qui ont pris département d'informatique de leur université comme cas d'étude pour tester sa performance.

Cet algorithme se déroule selon le diagramme suivant :

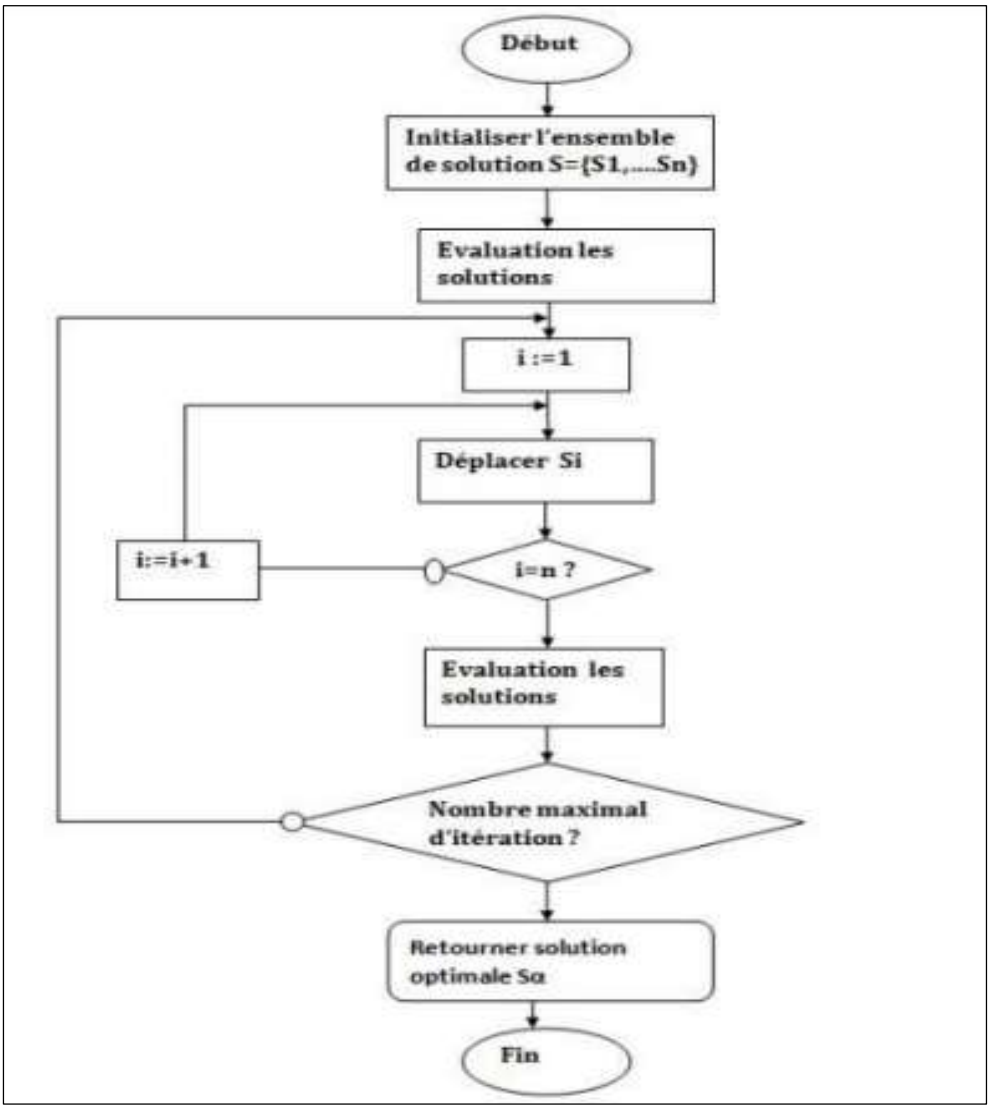


Figure 3.1 : Organigramme de l'algorithme GWO pour PETCU [Bouamama.o,Ayadi.S, 2020]

3. Initialisation de la population des solutions :

L'algorithme GWO nécessite à la démarche, une population de solutions valides et complètes. C'est-à-dire, un ensemble des emplois du temps contenant aucune violation des contraintes durs. Pour cela, la construction des emplois du temps initiaux est traitée comme un problème de coloriage de graphe [Palubeckis, 2008].

De nombreux algorithmes ont été conçus pour la résolution de ce problème :

3.1 L'algorithme RND Random : Dans cet algorithme, les nœuds sont triés aléatoirement puis les couleurs sont attribués comme suit : la 1ière couleur est donnée au sommet s_1. Lorsque les i premiers sommets sont colorés ($1 \leq i \leq n-1$), s_{i+1} est donné le plus petit numéro possible tel qu'aucun sommet adjacent n'ait le même numéro [Leighton*, 1979].

3.2 L'algorithme LF Largest First (ou SL Smallest Last) : Dans l'algorithme LF, au lieu de trier les sommets aléatoirement, ils sont triés de telle façon que les sommets ayant le plus grand nombre des arcs sont mis en premier. L'algorithme SL à presque la même stratégie sauf qu'il commence par mettre les sommets ayant le plus petit nombre des arcs en dernier [Leighton*, 1979].

3.3 Le problème de coloriage de graphe et le problème TTP :

Ce problème dit que pour un graphe donné, on doit trouver un nombre minimum de couleurs avec lesquels on peut colorer tous ses sommets de tel sort qu'aucun pair de sommets adjacent n'aille la même couleur.

Un emploi du temps peut être considéré comme un graphe où les sommets sont les activités d'enseignement (Séances). Chaque deux activités ayant une ou plusieurs ressources sont considérés liées par un arc. Alors, pour construire un emploi du temps sans conflits, on doit « colorer » ses sommets (Séances) d'une manière qu'aucun pair d'activités liés par un arc (ressource commune) auront la même couleur.

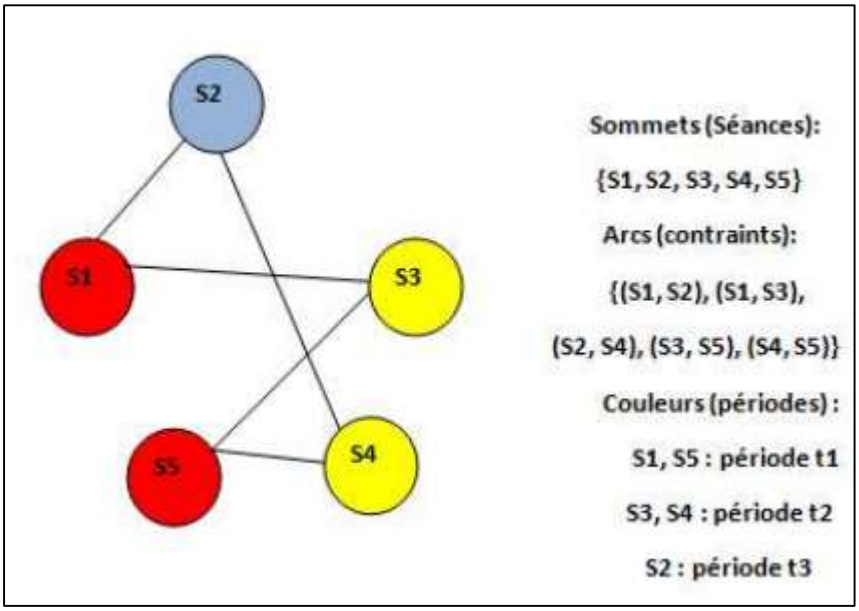


Figure 3.2 : Présentation d'emploi du temps par un Graphe [Khelifi.S et al, 2019]

4. Evaluation des solutions :

Un bon emploi du temps des cours universitaire est celui qui, en plus d'être sans aucun conflit dans les contraintes durs, satisfait les enseignants et les étudiants au maximum (contraintes faibles). Comme toutes les solutions initiales dans cette méthode satisfont totalement les contraintes dures, on travaille sur la satisfaction des contraintes faibles. Pour cela, une fonction objective est conçue qui permet d'estimer le taux de satisfaction de ces contraintes.

4.1 Evaluation de taux de satisfaction pour les étudiants :

Cette valeur exprime le taux des heures creuses effectives d'un groupe d'étudiants par rapport aux nombre total des heures vides dans la semaine.

G : Nombre de groupe.

h_s : Nombre des heures creuse.

h_v: Nombre total des heures vide.

$$F_g = \frac{\sum_{g=1}^{G} \frac{h_s}{h_v}}{G}$$

4.2 Evaluation de la satisfaction des vœux pour les enseignants :

4.2.1 Satisfaction du nombre de jours :

Parmi les buts d'optimisation est de réduire le nombre de jours de travail pour chaque enseignant. Le taux de satisfaction de ce critère est exprimé par la formule qui suit :

$$Fe1 = \begin{cases} 1, & Si\ le\ nombre\ des\ jours\ enseignés\ par\ l'enseignant\ e \leq 3 \\ 0, & Si\ non \end{cases}$$

4.2.2 Satisfaction des jours demandés :

Cette satisfaction est exprimée par le respect des vœux des enseignants concernant les jours de travail demandés selon la formule :

$$F_{e2} = \frac{Sj}{nbj}$$

Sj : Nombre de jours enseignés dans des jours demandés.

nbj : Nombre de jours total enseignés.

4.2.3 Satisfaction des périodes (matin/soir)

La satisfaction des jours demandé n'est pas suffisante pour cela nous pousse de prendre en considération le critère de la période d'enseignement.

$$F_{e3} = \frac{hs}{H}$$

h_s : Nombre des heures satisfaisant les périodes demandées.

H : Nombre total des heures enseignées dans les jours demandés.

4.2.4 Satisfaction des enseignants :

Le degré de satisfaction de chaque enseignant sera exprimé par l'assemblage des évaluations cité précédemment pondérées selon l'importance du critère :

$$F_{enseignant} = F_{e1} + F_{e2} + F_{e3}$$

Enfin, en calcule la moyenne de satisfaction de tous les enseignants :

$$Fe = \frac{\sum_{e=1}^{E} \sum_{i=1}^{3} Pi * Fe_i}{E}$$

E : Nombre d'enseignant.

Pi : Le vecteur P= {p_1, p_2, ..., p_n} est un poids qui est accorder à chaque fonction objective

4.3 Evaluation totale :

La fonction objective **F** qu'on cherche à maximiser sa valeur durant le processus d'optimisation est la suivante :

$$F = max P_e * F_e + P_g * F_g$$

Pe : c'est le pourcentage qui exprime l'influence de la satisfaction des enseignants **Fe**

Pg : c'est le pourcentage qui mesure l'influence la satisfaction des étudiants **Fg**

5. Processus d'optimisation et opérateurs utilisés :

Dans le processus d'optimisation, l'objectif est d'approcher la solution optimale, qui est la proie. Dans la méthode, pour un problème numérique, la proie est définie par la moyenne des trois premières solutions (α, β et δ). Dans notre cas, une solution est un emploi du temps qui est loin d'être une valeur numérique, la signification de la moyenne des trois meilleures solutions est loin d'être vue comme une « moyenne » arithmétique ou autre. La signification adoptée pour exprimer « la moyenne » est d'exploiter les trois solutions individuellement et [2]

éviter qu'une solution α, β ou δ n'exploite soi-même. Tous les loups de la meute approchent (se déplacent vers) une des trois meilleures solutions.

5.1 Exprimer la distance entre une solution et la solution ciblée :

On exprime la distance entre la solution et la solution ciblée par le nombre de différences dans les horaires des séances entre ces solutions.

$$D = |C.N - Ns|$$

N : est le nombre de séance dans l'emploi du temps

Ns : nombre de séance compatible.

C = $2.r_2$.

5.2 Déterminer le nombre de déplacement :

L'objectif de cette opération est d'exprimer le taux de convergence d'une solution à la solution ciblée (α, β ou δ). A partir de cette opération on détermine le nombre de changements appliqué à une solution dans une itération. On applique dans cette opération le mécanisme d'entourer la proie da la méthode :

$$X = |N - A.D| \quad \text{Où : } A = 2.a.r_1 - a$$

5.2.1 Les opérateurs utilisés :

Approcher une solution à une solution optimale se manifeste ici par le changement effectué à cette solution par le déplacement des séances en respectant les conflits pour ne pas détruire le travail de l'étape d'initialisation, cela nous oblige d'utiliser trois types d'opérateurs (déplacer une séance, échanger deux séances, échanger les séances de deux périodes, chaine de Kempe) afin d'assurer le plus possible l'exploration de l'espace de recherche.

5.2.1.1 Opérateur déplacer une séance (light mutation) :

Cet opérateur s'effectue en déplaçant une séance à une autre période disponible. Ce type de mutation ne permet pas une convergence rapide puisque le déplacement dans l'espace de solution est faible. De plus, la disponibilité des salles ainsi que les conflits d'horaire peuvent aussi rendre le déplacement d'une seule séance impossible. [Abdullah, 2006]

5.2.1.2 Opérateur d'échanger deux périodes (heavy mutation) :

Cet opérateur consiste à échanger les périodes de deux séances. Cette mutation permet une plus grande perturbation que la méthode par déplacement d'une séance et en échangeant deux périodes, l'utilisation de cette opération ne nécessite aucune satisfaction.

5.2.1.3 Chaines de Kempe :

Un autre opérateur intéressant est le déplacement par chaine de Kempe. Une chaîne de Kempe consiste à permuter un sous-ensemble de séances en deux périodes distinctes ayant des conflits entre eux. Supposons que nous avons deux périodes ti et tj. Une séance s1 dans la période ti va être déplacée à la période tj. Les séances dans la période ti et tj forment un graphe biparti (les arcs présentent les conflits entre ces séances). La chaîne de Kempe est définie à partir du choix initial d'une séance en utilisant les composantes connexes d'un graphe biparti [Abdullah, 2006].

5.2.2 Mécanisme d'application des opérateurs choisis :

Face à la faiblesse d'utiliser ces opérateurs avec la préservation de solution contre les conflits et la nécessité d'éviter la stagnation de la recherche. La figure 3.7explique comment les opérateurs sont appliqués.

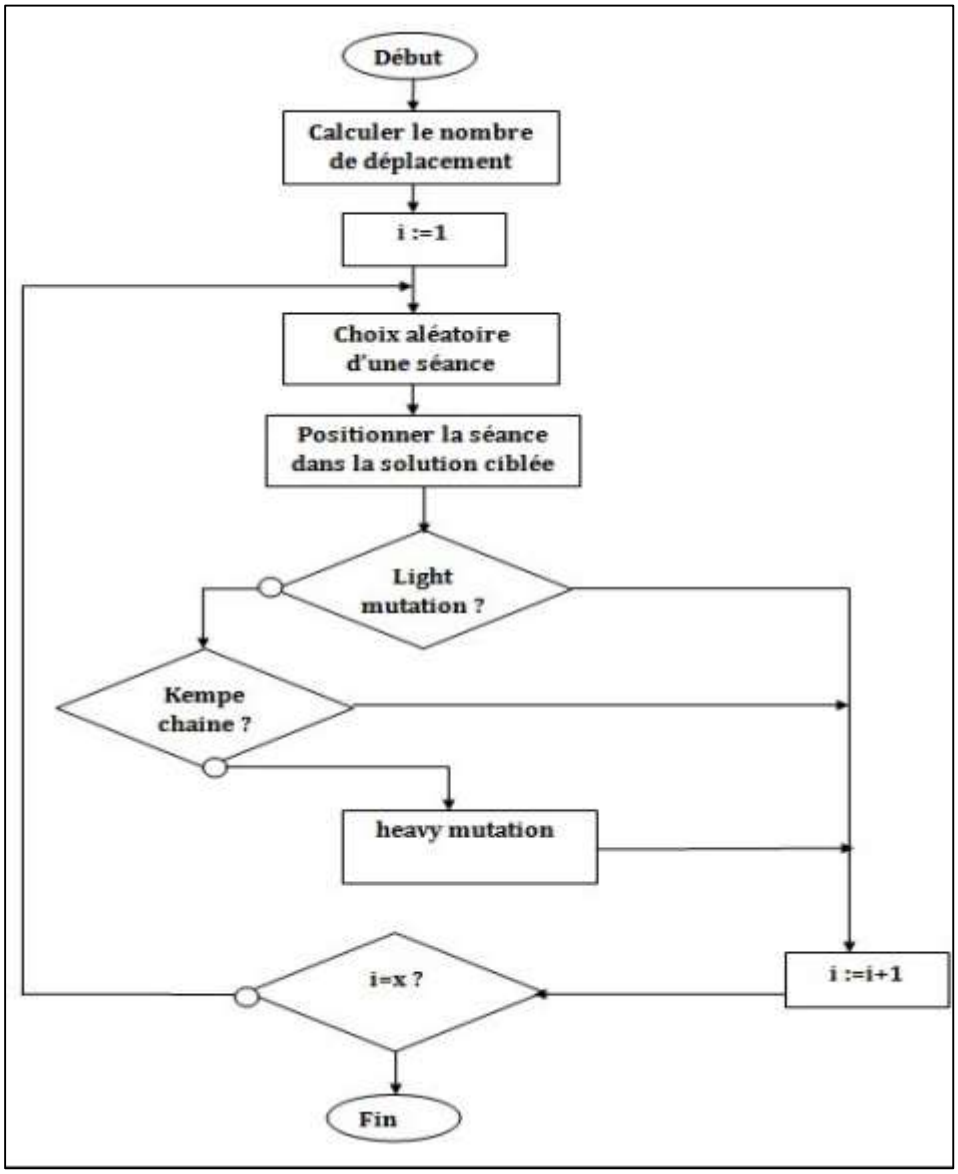


Figure 3.3 : Enchainement des tests d'application d'opérateurs pour un déplacement [Khelifi.S et al, 2019]

Cet enchainement de test d'opérateurs peut s'appliquer en trois situations comme suit :

1- La première situation est d'appliquer les opérateurs pour faire le changement à la position de la séance trouvée exacte dans la solution ciblée.

2- En cas qu'aucun déplacement n'est réalisé, on teste le même enchainement des opérateurs avec la séance précédente.

3- Lorsqu'aucun déplacement ne s'effectue encore, on applique l'enchainement des opérateurs avec la séance suivante.

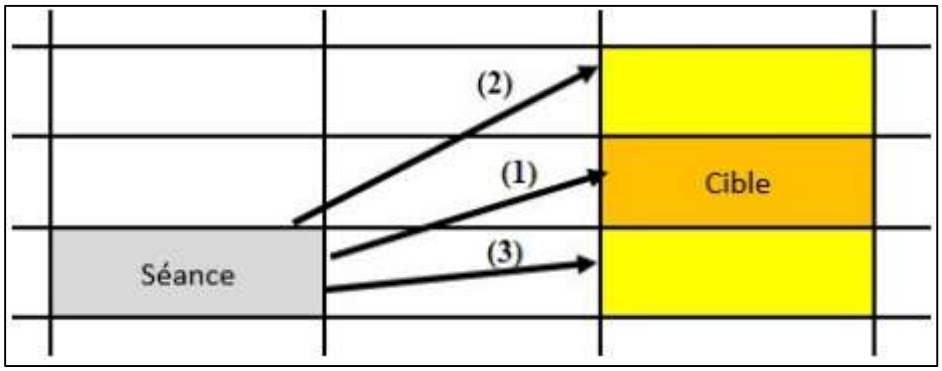


Figure 3.4 : Possibilité de déplacement [Khelifi.S et al, 2019]

5.3 Le principe d'exploration

L'effet de l'exploration dans la méthode GWO (recherche la proie), s'exprime par la valeur du paramètre A, lorsque |A|>1. Dans notre cas quand |A|>1, cela nous donne la signification d'exploration comme une augmentation du nombre des séances différentes (augmenter la distance entre les solutions).

5.4 Le principe d'exploitation

Contrairement au concept d'exploration, l'exploitation vise à approcher la proie (attaquer la proie). Ce type de déplacement s'effectue quand |A|<1. Cela est exprimé par la diminution du nombre des différences dans les horaires des séances. Alors dans l'exploitation on doit préserver la qualité de la solution c'est-à-dire on doit s'assurer que le déplacement réalisé ne doit pas diminuer la qualité de la solution, pour cette raison on n'applique aucun opérateur avant de tester leur influence sur la qualité de la solution.

6. Conclusion

Dans ce chapitre nous avons présenté la modélisation de notre application d'une optimisation loup gris adaptée pour le problème d'emploi du temps des cours universitaires, en expliquant comment exploiter les caractéristiques de la méthode pour rependre aux exigences et aux objectifs du problème et dans le chapitre suivant est consacré de la conception de notre benchmark relativement au département d'informatique de l'université de 20 Aout 1955 Skikda.

Chapitre 04

Benchmarks du PETCU

1. Introduction :

Dans ce chapitre, nous concentrerons nos attentions sur les versions de Benchmark du problème d'emploi du temps des cours universitaires. Cette version problématique était définie à l'origine en 2001 afin qu'il puisse être utilisé à divers fins de recherche par le réseau de métaheuristiques **[**Site Web réseaux Métaheuristiques**]** et visait à surmonter certaines des ambiguïtés et incohérences qui existent actuellement dans l'étude des emplois du temps automatisés. Cependant, en 2002, il a également été utilisé pour les compétions international des plannings des horaires en 2007 ITC-2007 et en 2019 ITC-2019 **(** International Timetabling Competition) et le résultat c'été des benchmarks avec plusieurs instances (petites et mediums et larges)**.** Le premier est formulé par Ben Paechter, le problème actuel est étroitement basé sur de nombreux problèmes d'emploi du temps typiques du monde réel, il y avait un certain nombre de raisons pour lesquelles cela a été fait. Premièrement, notamment en ce qui concerne l'analyse de ce qui se passe réellement dans les algorithmes conçus pour résoudre ce genre de problèmes. (Dans de nombreux cas, les vrais problèmes sont souvent trop compliqués et désordonnés pour permettre aux chercheurs d'étudier ces processus avec suffisamment de détails.) Deuxièmement, le grand nombre des contraintes dures et souples que l'on trouve souvent dans les problèmes du monde réel feront généralement le processus d'écriture de code (ou de mise à jour de programme existant pour qu'ils soient adaptés) est très long et processus ardu pour l'emploi du temps des chercheurs. Troisièmement, généralement bon nombre des contraintes rencontrées dans les problèmes du monde réel sont idiosyncratiques et ne feront que concernent une ou deux institutions. Ainsi, leur inclusion dans un ensemble de problèmes particulier ne sera pas nous permettent généralement d'en apprendre beaucoup sur les problèmes d'emploi du temps en termes généraux.

2. Définition Benchmark :

2.1 Sens 1 :

Terme marketing qui désigne l'ensemble des techniques permettant d'analyser les modes de gestion, d'organisation des autres entreprises dans le but de s'en inspirer.

2.2 Sens 2 :

Benchmark est un anglicisme informatique qui désigne un banc d'essai grâce auquel on peut prendre la mesure d'un système à des fins de comparaison avec d'autres systèmes. Ces jeux de tests, sont utilisés pour évaluer les performances et les capacités de convergence des algorithmes d'optimisation.

2.3 Benchmark d'emploi du temps :

Référence d'ensembles des données (Datasets), des données statiques d'établissement, a compris les ressources humaines et matérielles et temporelle toute en satisfaisons plusieurs contraintes organisées selon certaines format (Textuel, XML, Excel…).

Les benchmarks et leurs méthodologies de pointe respectives sont abordés dans cette section. Comme évident d'après le tableau 4.1, les ensembles de données de benchmarks utilisés dans les compétions internationales d'horaires sont les bancs d'essai les plus populaires entre les chercheurs pour comparer les algorithmes.

Dataset	Year	Methods
Socha	2010	Fish Swarm Intelligent
		RR scheduling algorithm (HC, GD, SA)
	2012	Honey-bee mating
		SA
		GC(LSDF)
	2014	PB-LS
	2016	ACO
	2017	TSSP and SAR
	2018	TS(RPNS)
	2019	SAIRL
	2020	SAR, ILS and SAR-2P
ITC-02	2012	SA
	2017	TSSP and SAR
	2018	TS(RPNS)
	2019	SAIRL
	2020	SAR, ILS and SAR-2P
ITC-07 (Track 2)	2010	Time-dependent meta-heuristic
	2012	SA
		ACO
	2017	TSSP and SAR
	2018	TS(RPNS)
	2019	SAIRL
	2020	SAR, ILS and SAR-2P
ITC-07 (Track 3)	2010	ATS
	2018	HGA, SA, HC
		MILP
		ADHH
	2019	ILP
		Network flow, GRASP and SA
	2020	MOSA
Hard	2011	Clique-based
	2012	SA
	2018	ILS
	2020	SAR, ILS and SAR-2P

Tableau 4.1 : benchmarks et leur état de l'art des méthodologies respectives

3. Benchmarks existants :

3.1 Benchmark (Dataset) Socha :

Est développé en utilisant un algorithme créé par Ben Paechter. Il se compose de 11 instances. Les caractéristiques de cet ensemble de données sont présentées dans le tableau 4.2. Au cours des 10 dernières années, 11 approches différentes ont été proposées pour cet ensemble de données. Les variantes de l'algorithme Recuit simulé proposées par [Say Leng Goh et al, 2020] sont supérieurs aux autres en termes de performances. Autre méthode de pointe pour ce benchmark est l'approche basée sur TS appelée partie aléatoire recherche de voisinage (Random Partial Neighbourhood Search (RPNS)), par [Yuichi Nagata,2018].

Instances	Students	Events	Rooms	Features
S01	80	100	5	5
S02	80	100	5	5
S03	80	100	5	5
S04	80	100	5	5
S05	80	100	5	5
M01	200	400	10	5
M02	200	400	10	5
M03	200	400	10	5
M04	200	400	10	5
M05	200	400	10	5
L	400	400	10	10

Instances/ S Small, M Medium, L Large

Student /nbr Etudiants

Events /nbr Activités

Rooms/nbr Salles

Features/nbr Caracteristiques

*Tableau 4.2 : Les caractéristiques du Benchmark **Socha***

3.2 Benchmark ITC-02 :

Compétions international 2002 est organisé par le Metaheuristic Network et parrainé par la pratique et la théorie des horaires automatisés (Practice and Theory in Automated Timetabling « PATAT »). Le benchmark (20 instances) est produit à l'aide d'un Algorithme de Ben Paechter, il y a une exigence de délai pour cet ensemble de données qui est dicté par l'exécution d'un programme sur l'ordinateur hôte. Au cours des 10 dernières années, cinq différents des approches ont été proposées TSSP, ILS et SAR-2P proposé par [Say Leng Goh et al, 2020] ont mieux performé que les autres quatre. Les caractéristiques de cet ensemble de sonnées sont présentées dans le tableau 4.3.

stances	Students	Events	Rooms	Features
01	200	400	10	10
02	200	400	10	10
03	200	400	10	10
04	300	400	10	5
05	300	350	10	10
06	300	350	10	5
07	350	350	10	5
08	250	400	10	5
09	220	440	11	6
10	200	400	10	5
11	220	400	10	6
12	200	400	10	5
13	250	400	10	6
14	350	350	10	5
15	300	350	10	10
16	220	440	11	6
17	300	350	10	10
18	200	400	10	10
19	300	400	10	5
20	300	350	10	5

*Tableau 4.3 : Les caractéristiques du Benchmark **ITC-02***

3.3 Benchmark ITC-07 Track02 (Voie 2)

Benchmark de variantes PE-CTT (24 instances) de concours international d'horaires 2007 (ITC-07), peut être téléchargé [1]. Au cours des 10 dernières années, sept différentes approches ont été proposées pour cet ensemble de données. L'état actuel des méthodes de l'art les plus connues sont RPNS, SAR,-2P et SA. Les caractéristiques du jeu de données sont présentées dans le

Instances	Students	Events	Rooms	Features
01	500	400	10	10
02	500	400	10	10
03	1000	200	20	10
04	1000	200	20	10
05	300	400	20	20
06	300	400	20	20
07	500	200	20	20
08	500	200	20	20
09	500	400	10	20
10	500	400	10	20
11	1000	200	10	10
12	1000	200	10	10
13	300	400	20	10
14	300	400	20	10
15	500	200	10	20
16	500	200	10	20
17	500	100	10	10
18	500	200	10	10
19	1000	300	10	10
20	1000	400	10	10
21	300	500	20	20
22	500	600	20	20
23	1000	400	20	30
24	1000	400	20	30

*Tableau 4.4 : Les caractéristiques du Benchmark **ITC-07 Track02***

[1] : https://www.unitime.org/uct_datasets.php

3.4 Benchmark ITC-07 Track03 (Voie 3)

L'ensemble de données de variantes CB-CTT (21 instances) pour International Timetabling Competition peut être téléchargé [2]. Au cours des 10 dernières années, sept approches différentes ont été proposées pour ce jeu de données. Une méthodologie de Network flow (GRASP+SA) est supérieur à l'ATS. Assouplissement IP proposé par [N.-C.-F. Bagger, G. Desaulniers, and J. Desrosiers, 2019] a surpassé les six autres méthodologies en améliorant la borne inférieure pour trois des instances problématiques. Les caractéristiques du benchmark sont présentées dans le tableau 4.5.

Instances	Rooms	Courses	Curricula	Constraints
01	6	30	14	53
02	16	82	70	513
03	16	72	68	382
04	18	79	57	396
05	9	54	139	771
06	18	108	70	632
07	20	131	77	667
08	18	86	61	478
09	18	76	75	405
10	18	115	67	694
11	5	30	13	94
12	11	88	150	1368
13	19	82	66	468
14	17	85	60	486
15	16	72	68	382
16	20	108	71	518
17	17	99	70	548
18	9	47	52	594
19	16	74	66	475
20	19	121	78	691
21	18	94	78	463

*Tableau 4.5 : Les caractéristiques du Benchmark **ITC-07 Track03***

3.5 Benchmark Dure (Hard)

Les 60 instances (20 petites (Smalls), 20 moyennes (mediums), grandes (Larges)) proposé par [R. Lewis and B. Paechter,2007] peut être téléchargé depuis le site web[3] « **C**entre for Emergent **C**omputing », l'état de l'art de la méthode pour ce jeu de données est ILS et TSSP-ILS. Ils ont réussi à trouver des solutions réalisables pour 58 et 57 cas respectivement. Le tableau 4.6 montre les caractéristiques de ce benchmark.

[2] : https://www.unitime.org/uct_datasets.php

[3] : http://www.rhydlewis.eu/hardTT/

Instances	Students	Events	Rooms	Features
01	1000	1000	28	20
02	1000	1000	25	20
03	900	1000	25	20
04	800	1050	25	20
05	1000	1075	25	20
06	1000	1075	25	20
07	1100	1050	25	20
08	1000	1025	25	20
09	800	1050	25	20
10	1000	1075	25	20
11	1000	1075	25	20
12	1000	1000	26	25
13	1000	1000	25	25
14	1000	1000	25	25
15	1000	1000	25	25
16	1000	1000	25	10
17	1200	1000	25	10
18	1000	1000	25	10
19	1000	1000	25	10
20	1000	1000	25	10
21	400	400	10	10
22	400	390	10	10
23	400	390	10	10
24	400	410	10	9
25	450	410	10	9
26	450	410	11	10
27	450	410	11	10
28	400	400	10	10
29	400	400	10	10
30	500	400	10	8
31	800	400	10	8
32	800	400	10	8
33	800	400	10	8
34	1000	400	10	8
35	500	425	10	8
36	1000	400	10	8
37	800	400	10	8
38	1000	400	10	8
39	1000	410	10	8
40	1000	410	10	8
41	200	200	5	5
42	400	210	6	5
43	400	200	6	5
44	500	200	5	8
45	500	200	5	8
46	1000	200	5	3
47	800	200	5	3
48	1000	225	5	10
49	900	225	5	10
50	1000	220	5	10
51	1000	200	5	4
52	1000	225	5	10
53	1000	225	5	10
54	1000	225	5	3
55	900	200	5	3
56	900	200	5	3
57	900	200	5	3
58	1000	225	5	3
59	1000	225	5	3
60	1000	225	5	3

*Tableau 4.6 : Les caractéristiques du Benchmark **Dure (Hard)***

3.6 Benchmark ITC-2019

Le concours international d'horaires (ITC-2019) est le dernier concours d'horaires. Son jeu de données de référence (30 instances) peut être téléchargées[4]. Le sectionnement des étudiants est pris en compte dans ces cas problématiques. Le tableau 4.7 montre ces caractéristiques.

[4] https://www.itc2019.org/

Instance	Size (MB)	Courses	Classes	Rooms	Students
1	14.55	340	1239(543 fixed)	80	1641
2	5.82	272	1852(332 fixed)	44	2116
3	3.88	353	983(79 fixed)	62	3018
4	12.60	1206	2641(530 fixed)	214	0
5	2.94	544	882(63 fixed)	90	3666
6	1.41	228	575(128 fixed)	35	1543
7	1.48	226	561(191 fixed)	44	865
8	15.92	1089	2526(1132 fixed)	70	2938
9	4.69	687	1001(318 fixed)	75	27018
10	1.94	36	711(74 fixed)	15	0
11	11.18	406	1144(97 fixed)	84	2254
12	10.74	234	460(2 fixed)	39	1988
13	3.03	313	487(3 fixed)	73	0
14	1.39	186	516(63 fixed)	35	1469
15	7.70	116	650(32 fixed)	29	395
16	6.71	881	1515(159 fixed)	83	3443
17	3.53	404	782(41 fixed)	67	2293
18	2.82	212	1061(115 fixed)	84	13497
19	32.11	2839	8813(2809 fixed)	768	38437
20	2.05	91	417(14 fixed)	28	821
21	42.84	1363	5081(341 fixed)	327	6925
22	4.16	357	1083(97 fixed)	63	2921
23	12.31	1290	2782(838 fixed)	208	0
24	3.10	328	502(8 fixed)	73	0
25	2.49	540	951(186 fixed)	93	5051
26	1.36	188	535(60 fixed)	36	1685
27	11.52	515	1623(443 fixed)	33	1152
28	27.25	1635	3717(312 fixed)	86	5651
29	10.83	1154	2798(449 fixed)	224	35213
30	1.74	44	676(47 fixed)	18	0

*Tableau 4.7 : Les caractéristiques du Benchmark **ITC-2019***

4. Conception du Benchmark Skikda-Dep-Inf-2022 :

Les données d'entrée en format XML, avec la structure suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<TimeTable version="1.0" initiative="département informatique université
Skikda" annee="2022" cree="Tue June 10 19:06:19 EDT 2022" nbrJour="6"
nbrhoraires="5">
```

Attribut	Description
`version`	Format données et version
`creation`	Date de génération du fichier
`nbrJour`	Nombre des jours des cours par semaine
`nbrhoraires`	Nombre des périodes par jour

Tableau 4.8 Attributs de l' Entête (Header)

```
<salles>
      <salle type="amphi" qte="1" capacite="600"/>
      <salle type="td" qte="10" capacite="60"/>
      <salle type="tp" qte="5" capacite="30"/>
  </salles >
```

Attribut	Description
type	Type des salle
Qte	Nombre des salles
capacite	Capacite des salles

Tableau 4.9 Attributs de salle

```
<promo>
   <promo code="L2-INF" nom="Licence 2" nbr="200"/>
   <groupe niveau="L2-INF" id="1" nbr="40"/>
   <groupe niveau="L2-INF" id="2" nbr="40"/>
           .. . . . .
   <promo code="M2-GL" nom="Master 2 Génie Logiciel Avancé et
   Applications "
            nbr="80"/>
   <groupe niveau="M2-GL" id="1" nbr="40"/>
   <groupe niveau="M2-GL" id="2" nbr="40"/>
 </promo>
```

Attributs	Description
code	Codification promo L2-INF…
Nom	Nom du promo (Licence, Master, Doctorat)
Nbr	Nombre des étudiants par promo

Tableau 4.10 Attributs de promo

Attribut	Description
Id	Identifiant du groupe
niveau	Code promo L2-INF…etc
Nbr	Nombre des étudiants par groupe

Tableau 4.11 Attributs de groupe

```
<matiere-promos>
      < matiere-promo s= "s1" promo="L2-INF" titre="Archit-Ordinat"
      coeff="1" credits="2" evaluation-mode="40%/60%" C="1" TD="1" TP="1"/>
      < matiere-promo s= "s1" promo="L2-INF" titre="Algo-struc-D3"
      coeff="3" credits="5" evaluation-mode="40%/60%" C="1" TD="1" TP="1"/>
      ..........
      < matiere-promo s= "s2" promo="M2-GLAA" titre="Ethiq-GL-Quest-Juridi"
      coeff="1" credits="1" evaluation-mode="50%/50%"C="1" TD="0" TP="0"/>
</matiere-promos>
```

Attribut	Description
promo	Identifiant du groupe
S	Semestre s1 ou s2
titre	Code promo L2-INF…etc
coeff	Coefficient de chaque module
credits	Les crédits de module
evaluation-mode	Le mode d'évaluation de chaque module
C	Cours =1 s'il y a des cours ou 0 si non

TD	TD=1 s'il y a des td ou 0 si non
TP	Tp=1 =1 s'il y a des tp ou 0 si non

Tableau 4.12 Attributs du matière-promo

```
<enseignant>
        <enseignant id="1" nom=" enseignant1" />
        <enseignant id="2" nom=" enseignant2"/>
          <enseignant id="3" nom=" enseignant3"/>
            . . . . .
          <enseignant id="55" nom=" enseignant55"/>
<enseignant>
```

Attribut	Description
Id	Identifiant de l'enseignant
Nom	En général libellé enseignant1 …etc

Tableau 4.13 Attributs de l'enseignant

```
<activite/>
      <activite id="1" s="s1"enseignant="36" module="Sys-Emb-Mob"
      promo="M2_RSD" type="C"/>
      <activite id="2" s="s1"enseignant="14" module="Res-ad-hoc"
      promo="M2_RSD" type="C"/>
      ..................
      <activite id="529" s="s2"enseignant="2" module="Seminaire" promo="M2-
      SI" type="C"  />
      <activite id="530" s="s2"enseignant="2" module="TravPers" promo="M2-
      SI" type="C"/>
</activite>
```

Attribut	Description
Id	Identifiant de l'activité
S	Semestre s1 ou s2
enseignant	Identifiant d'enseignant
module	Module affecté à l'enseignant
promo	Code promo
type	Type du module C / TD / TP

Tableau 4.14 Attributs de l'activité

```
<period>
   <period id="1" day_name="samedi" duree="90" horaires="08:00 - 09:30"/>
   <period id="2" day_name="samedi" duree="90" horaires="09:00 - 11:00"/>
   <period id="3" day_name="samedi" duree="90" horaires="11:00 - 12:30"/>
   <period id="4" day_name="samedi" duree="90" horaires="12:30 - 14:00"/>
   <period id="5" day_name="samedi" duree="90" horaires="14:00 - 15:30"/>
   <period id="6" day_name="samedi" duree="90" horaires="15:30 - 17:00"/>
 . . . . . .
   <period id="31" day_name="jeudi" duree="90" horaires="08:00 - 09:30"/>
```

```
   <period id="32" day_name="jeudi" duree="90" horaires="09:00 - 11:00"/>
   <period id="33" day_name="jeudi" duree="90" horaires="11:00 - 12:30"/>
   <period id="34" day_name="jeudi" duree="90" horaires="12:30 - 14:00"/>
   <period id="35" day_name="jeudi" duree="90" horaires="14:00 - 15:30"/>
   <period id="36" day_name="jeudi" duree="90" horaires="15:30 - 17:00"/>
</period>
```

Attribut	Description
Id	Identifiant de de la période (5 horaires * 6 jours = 36 périodes)
day_name	Les jours d'enseignement (6 jours)
duree	La durée de chaque séance 1 h 30 min = 90 min
horaiares	Commence de 8 :00 jusqu'à 17 :00 en ignorant la période déjeuné

Tableau 4.15 Attributs du période

```
<veux>
  <veux id="1" enseignant="1"period ="1">1</veux>
  <veux id="2" enseignant="1"period ="1">0</veux>
. . . . . .
  <veux id="3663" enseignant="55"period ="35">0</veux>
  <veux id="3664" enseignant="55"period ="36">1</veux>
  <veux id="3665" enseignant="55"period ="36">0</veux>
</veux>
```

- Pour chaque enseignant (nbr globla1 d'enseignants = 55) il y a la probabilité d'enseigner ou non pendent 36 périodes pendent la semaine donc pour chaque enseignant on 72 veux 36 oui je veux enseigner et 36 non je ne veux pas.

Attribut	Description
Id	Identifiant de la fiche de veux
enseignant	Identifiant de l'enseignant
period	Identifiant de la période

Tableau 4.16 Attributs de la fiche de veux

```
</TimeTable>   Fin du fichier XML
```

5. Conclusion

Dans ce chapitre on a vu les différents Benchmark et leurs caractéristiques spécifique au problème d'emploi du temps universitaire ainsi la conception de notre benchmark relativement au département d'informatique de l'université de 20 Aout 1955 Skikda. Le chapitre suivant sera consacré à l'implémentation de notre application et à la discussion des résultats obtenus.

Chapitre 05

Implémentation

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 GWO au TTP des cours universitaires du Département d'Informatique de l'université du 20 août 1955-Skikda. Après définition et présentation des outils employés dans son développement, on va présenter quelques interfaces de notre système. Les résultats obtenus par cette réalisation reflètent la performance de la métaheuristiques GWO par rapport au problème étudié seront aussi présentés et discutés.

2. Présentation du département Informatique

Une université est composée des facultés qui à leur tour sont composées de épartements. Le département Informatique fait partie de la Faculté des Sciences.

Depuis sa création en 1991 au sein de l'école normale supérieure de l'enseignement technique de Skikda, le département d'informatique a formé des cadres en informatique de différents niveaux : D.E.U.A, licence, ingénieurs d'état, master, magisters et doctorat.

Dans le cadre du nouveau système L.M.D ; Le département d'informatique offre les deux parcours de formation en licence et trois en Master décrits ci-dessous :

a. Licence systèmes informatiques (SI).

b. Licence ingénierie des systèmes d'information et du logiciel. (ISIL).

c. Master académique systèmes informatiques (SI).

d. Master académique génie logiciel avancé et applications (GLAA).

e. Master académique réseau et systèmes distribués (RSD).

f. Master académique systèmes informatiques avances et applications (SIAA)

2.1 Organisation du département Informatique

Le département de l'informatique de l'université 20 Août 1955 Skikda est organiséselon la structure suivante :

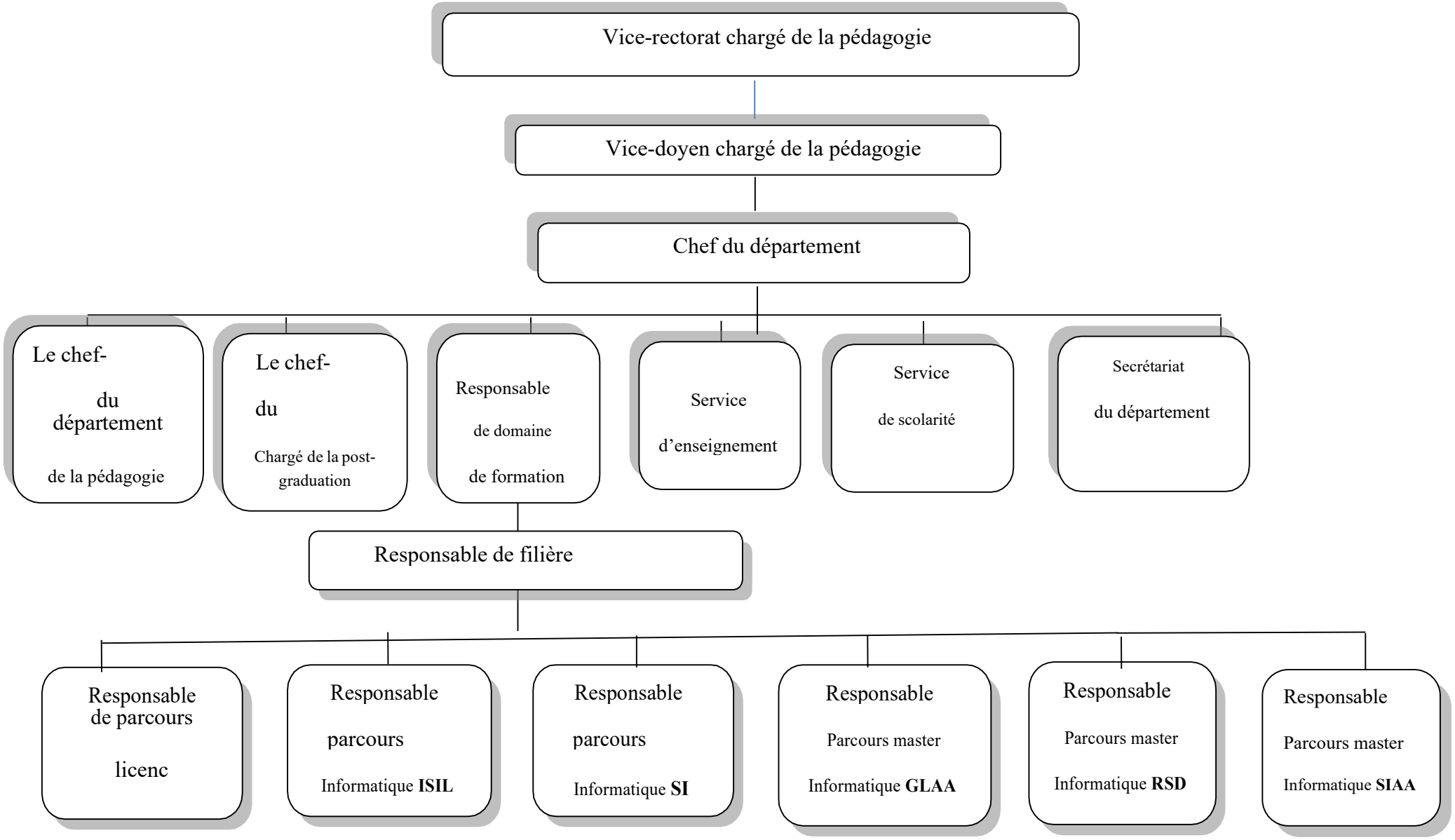


Figure 5. 1: Structure organisationnelle du département

2.2 Missions du département

Le département d'informatique assure un suivi pédagogique des cycles de graduation et de post-graduation. Il s'agit là d'une gestion quotidienne et soutenue dans divers domaines. La gestion de la scolarité est l'une des tâches les plus lourdes et les plus importantes. Il s'agit principalement de : gérer la scolarité des étudiants (inscription, évaluation, présence aux enseignements, absences et sanctions) et des enseignants (matières enseignées, affectation des modules, volume horaire, planning des examens, absences, saisie des notes, délibérations …).

2.3 Les données du département de l'informatique de l'université 20 Août 1955

Les données du département de l'informatique de l'université 20 Août 1955 Skikda de l'année universitaire 2021-2021 qu'on a pris comme cas d'étude ont présentés comme suit :

		Désignation									
Enseignants	**Total**	**51**									
Promos	Niveau	**L2**	**L3**		**M1**				**M2**		
			SI	**ISIL**	**SI**	**SIA A**	**RSD**	**GLAA**	**SI**	**RSD**	**GLAA**
	Nombre de groupes	**5**	**2**	**4**	**2**	**1**	**2**	**1**	**2**	**2**	**2**
	Total	**23**									
	Nombre Des étudiants	**125**	**63**	**144**	**31**	**24**	**51**	**36**	**45**	**60**	**43**
	Total	**622**									
Salles	Type	**TD**			**TP**				**Amphi**		
	Nombre	10			5				1		
	Capacité	60			30				200		
	Total	**16**									

Tableau 5.1: Données utilisées

3. Etude technique

L'étude technique est une phase d'adaptation de conception à l'architecture technique. Elle a pour objectif de décrire au plan fonctionnel la solution à réaliser d'une manière détaillée ainsi que la description des traitements. Cette étude, qui suit l'étude détaillée, constitue le complément de spécification informatique nécessaire pour assurer la réalisation du système.

3.1 Matériel de base

Pour la réalisation de notre application, nous avons eu recours à plusieurs moyens matériels et logiciels. Le développement de l'application est réalisé via un ordinateur portable ayant les caractéristiques suivantes :

Caractéristique	
Marque	HP
Processus	AMD A6-7310 APU with AMD Radeon R4 Graphics 2.00 GHz
RAM	08 Go
Système d'exploitation	Windows-10 64-bits

Tableau 5.2: Matériel de base

4. Outils et langage de développement

Les principaux outils et langages qui ont contribué à la qualité du développement sont

4.1 Le langage de programmation Java

C'est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels.

4.2 La Plateformes Java FX

Java FX est une plateforme qui permet de développer des interfaces riches et simplifier leur développement grâce au langage FXML et l'outil Scene Builder d'Oracle. La richesse de la plateforme autorise des effets visuels comme la manipulation de contenumultimédia.

Java FX est le successeur officiel de Swing. La version Java FX 8, sortie avec Java 8succède à Java FX 2.

4.3 EDI (Environnent Développent Intégré) Eclipse

Eclipse est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

4.4 XML

XML est un langage de balisage extensible et est considéré comme une spécification pour les documents “lisible par les machines”. Il est naturellement utilisé pour encoder les langages du Web sémantique. L’extensibilité du XML indique la différence importante avec d’autres langagesprécédents qui est aussi la caractéristique essentielle du XML. XML est un métalangage (une description de type de document, DTD, permet de décrire la grammaire des documents admissibles) : en effet XML fournit une structure pour représenter d’autres langages (appelé parfois dialectes XML) d’une manière normalisée. La structure d’un document XML est définie par une DTD (Document Type Définition). La DTD peut être écrite dans un document à part puis référencée dans le document XML ou peut être directement intégrée dans ce dernier.

XML Schéma2, une nouvelle recommandation du W3C, est un langage XML qui peut remplacer la DTD. XML schéma présente des types de données XML. Il est possible de définir des types de données complexes en utilisant des éléments imbriqués. XML schéma a plusieurs avantages par rapport à la DTD : il offre tout d’abord une grammaire plus riche pour décrire la structure des éléments. Ensuite, il fournit un mécanisme d’inclusion et de dérivation qui permet de réutiliser les définitions des éléments communs ou bien d’adapter une définition existante à une nouvelle. Enfin, XML schéma utilise l’espace de nommage (name space) XML qui permet d’identifier une définition du document spécifique avec un nom unique et de préfixer toutes les balises avec ce nom unique.

4.5 XML Marker

Propose aux développeurs un éditeur XML et Json. L'application affiche à la fois la vue hiérarchique, la vue tabulaire ainsi que le code de vos données XML. **XML Marker** produit automatiquement l'affichage en tableau d'une étiquette choisie en récupérant les attributs et les noms des tags redondants pour ensuite les ranger en colonnes. Enfin, le logiciel est très rapide et offre une navigation au sein de grandes bases XML.

5. Quelques fenêtres de l'application

5.1 Fenêtre "Accueil"

C'est la première fenêtre de l'application qui permet d'accéder à ses fonctionnalitésprincipales

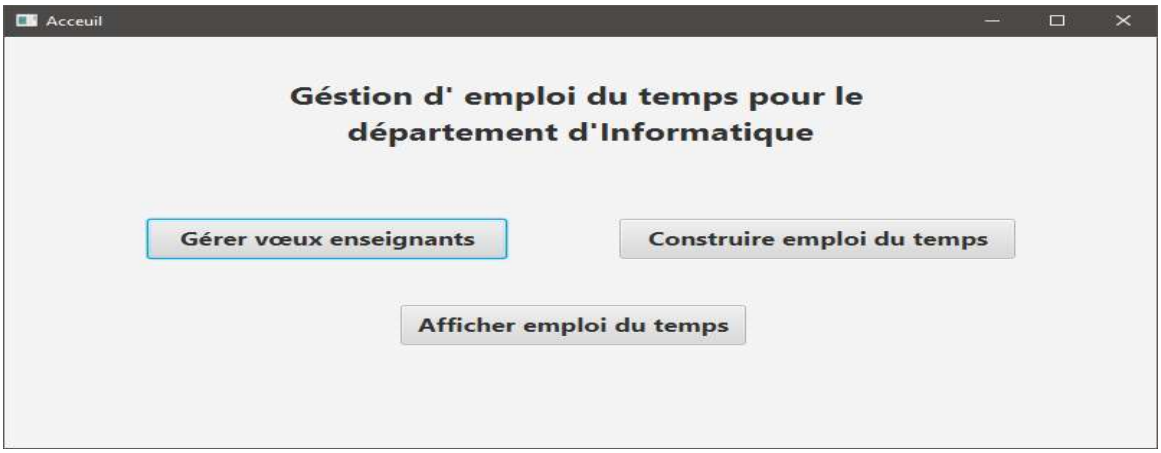


Figure 5. 2 Fenêtre "Accueil "

5.2 Fenêtre "Fiche de veux"

C'est la fenêtre qui permet de saisir les veux des enseignants en termes des horaires d'enseignement.

Figure 5. 3 Fenêtre "Fiche de veux"

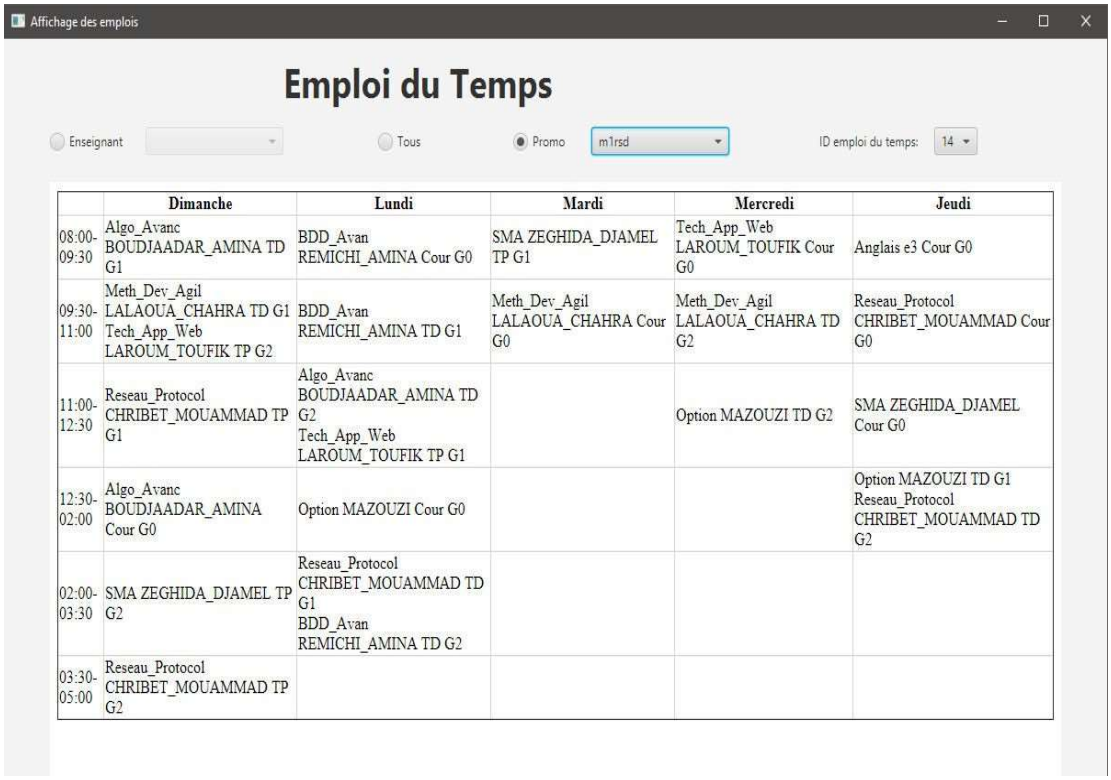
5.3 Fenêtre "Construction de l'emploi du temps"

C'est la fenêtre utilisée pour construire un nouvel emploi du temps. Elle permet de saisir les paramètres de l'algorithme ainsi qu'observer l'avancement du processus d'optimisation.

Figure 5. 4 Fenêtre "Construction de l'emploi du temps"

5.4 Fenêtre "Affichage des emplois" :

Cette fenêtre permet de visualiser les emplois du temps construits et enregistrés dans la base de données. On peut choisir d'afficher l'emploi du temps de tout le département ou on peut afficher seulement l'emploi d'une section ou d'un enseignant uniquement.



	Dimanche	Lundi	Mardi	Mercredi	Jeudi
08:00-09:30	Algo_Avanc BOUDJAADAR_AMINA TD G1	BDD_Avan REMICHI_AMINA Cour G0	SMA ZEGHIDA_DJAMEL TP G1	Tech_App_Web LAROUM_TOUFIK Cour G0	Anglais e3 Cour G0
09:30-11:00	Meth_Dev_Agil LALAOUA_CHAHRA TD G1 Tech_App_Web LAROUM_TOUFIK TP G2	BDD_Avan REMICHI_AMINA TD G1	Meth_Dev_Agil LALAOUA_CHAHRA Cour G0	Meth_Dev_Agil LALAOUA_CHAHRA TD G2	Reseau_Protocol CHRIBET_MOUAMMAD Cour G0
11:00-12:30	Reseau_Protocol CHRIBET_MOUAMMAD TP G1	Algo_Avanc BOUDJAADAR_AMINA TD G2 Tech_App_Web LAROUM_TOUFIK TP G1		Option MAZOUZI TD G2	SMA ZEGHIDA_DJAMEL Cour G0
12:30-02:00	Algo_Avanc BOUDJAADAR_AMINA Cour G0	Option MAZOUZI Cour G0			Option MAZOUZI TD G1 Reseau_Protocol CHRIBET_MOUAMMAD TD G2
02:00-03:30	SMA ZEGHIDA_DJAMEL TP G2	Reseau_Protocol CHRIBET_MOUAMMAD TD G1 BDD_Avan REMICHI_AMINA TD G2			
03:30-05:00	Reseau_Protocol CHRIBET_MOUAMMAD TP G2				

Figure 5. 5 Fenêtre "Affichage des emplois

6 Résultats et discussion

Toutes réutilisations ou améliorations de techniques et méthodes existantes ou applications de nouvelles doivent être sujettes à une analyse de performance ou une étude comparative des résultats obtenus.

Cette analyse n'est pas toujours aisée, et peut, selon cas et contexte, s'avérer à la fois très cruciale que délicate.

En effet, le problème d'emploi du temps se doit le respect de plusieurs critères à satisfaire qui diffèrent d'un établissement à un autre, donc pas de benchmarks de référence. Par conséquence, il devient impossible d'introduire une fonction objective standard qui permet une évaluation "intègre" répondant, à la fois, à tous les critères de satisfaction de n'importe quel établissement. Pour cela, nous nous somme restreint aux données réelles couvrant les objectifs visés et les contraintes imposées par le département d'Informatique de l'université du 20 août 1955-Skikda.

Le premier objectif, ou prérequis de rigueur, lors de la construction d'un emploi du temps fiable est d'éviter les conflits entre les séances programmées en même temps : cours enseignés par le même enseignant / étudiés par le même groupe d'étudiants ou le dépassement du nombre de salles disponibles. D'autre objectifs, moins prioritaires (contraintes légères), doivent être observés pour l'amélioration d'un emploi du temps (l'optimisation proprement dite). Ces objectifs peuvent concerner, par exemple :

- La maximisation des vœux respectés des enseignants concernant les jours et les périodes de travail.
- La minimisation des jours de travail des enseignants.
- La minimisation des heures d'inactivités (séances creuses) pour les étudiants.

En premier temps, nous avons construit une matrice d'adjacence représentant un graphe où chaque nœud représente une activité. Un lien est présent entre une paire de nœuds lorsque les activités correspondantes ne doivent pas se dérouler en même temps. Cela a pour but d'assurer définitivement l'absence de conflits à tout moment.

Ensuite, nous avons utilisé un algorithme de coloriage de graphe qui prend comme entrées un ensemble d'activités à ordonnancer et une matrice de conflit spécifiant les activités qui ne doivent pas être programmées en même temps. Cet algorithme retourne comme sortie un ensemble de groupes d'activités dont les activités de chaque groupe n'ont aucun conflit entre eux. Il prend en compte, aussi, le nombre de salles disponible dans le département. Pour chaque type d'activité, le nombre d'activités dans un groupe ne doit pas dépasser le nombre de salles disponibles de ce type. Ce qui assure le respect de l'objectif de rigueur (contraintes dures).

Après avoir groupé les activités, nous avons généré la population initiale des solutions en distribuant aléatoirement ces groupes sur les créneaux horaires de la semaine comme une étape finale de la construction des emplois du temps (à améliorer). Pour assurer plus de hasard et de dispersion de regroupement dans la population initiale, nous avons, pour chaque individu, réarrangé les activités aléatoirement avant de les grouper. Enfin, nous avons appliqué notre variante de l'algorithme GWO pour améliorer la

qualité des emplois du temps par le respect des contraintes légères. Pour guider la recherche, nous utilisons une fonction objective qui exprime la qualité de la solution : une valeur numérique reflétant le taux de satisfaction des exigences prescrites. Nous avons exécuté notre programme en utilisant les données fournis par l'administration du département d'Informatique en appliquant les paramètres suivants :

- Nombre d'itérations : 300 itérations
- Poids de satisfaction pour un enseignant :
  - ♣ Poids de satisfaction du nombre de jours de travail : 0.4.
  - ♣ Poids de satisfaction des jours : 0.4.
  - ♣ Poids de satisfaction des périodes de travail : 0.2.
- Poids de satisfaction globale (pour les enseignants et les étudiants) :
  - ♣ Poids de satisfaction globale des enseignants : 0.9.
  - ♣ Poids de satisfaction globale des groupes d'étudiants : 0.1.

Le tableau suivant démontre les résultats obtenus de dix exécutions du programme :

	Satisfaction Enseignants	Satisfaction des groupes	Fonction objective
1	98.16%	98.61%	98.21%
2	98.21%	97.63%	98.15%
3	98.22%	97.77%	98.18%
4	98.22%	98.34%	98.23%
5	98.07%	99.70%	98.23%
6	98.22%	99.38%	98.34%
7	98.14%	98.21%	98.15%
8	98.21%	98.05%	98.19%
9	98.15%	97.83%	98.12%
10	98.22%	97.36%	98.13%
Valeur max.	98.22%	99.70%	98.34%
Valeur min.	98.07%	97.36%	98.12%
Valeur moyenne	98.18%	98.28%	98.19%

Tableau 5.3: Tableau de taux de satisfaction

6. Conclusion

Dans ce chapitre, nous avons présenté les éléments essentiels à la mise en œuvre et le test de notre algorithme GWO adapté pour le problème d'emploi du temps des cours universitaires. Les résultats obtenus sont très encourageants face aux critères (objectifs et contraintes) de rigueur employée dans l'évaluation des emplois du temps construits par notre système ce qui prouve encore une fois l'efficacité de cette métaheuristique.

Conclusion générale

Le problème d'emploi du temps est classé NP-difficile. Sa difficulté se manifeste par l'explosion combinatoire de son espace de recherche cette explosion est justifiée par la considération d'une grosse masse d'informations soumise à diverses contraintes (TROUDI, 2006), particulièrement dans l'université.

Le travail présenté au niveau de ce mémoire a pour but, la normalisation et évaluation d'une solution basée sur loups gris pour le problème d'emploi du temps (TimeTabling Problem:TTP) des cours universitaires au sein du département d'Informatique de l'Université 20 Août1955-Skikda.

La normalisation des entrées était réalisée avec élaboration d'un benchmark en utilisant un format XML. Compte tenu des objectifs visés et des contraintes imposées, les résultats obtenus sont encourageants.

Perspectives

En se basant sur notre benchmark acquis, ce dernier laisse plusieurs perspectives à développer pour d'éventuels projets similaires :

- Exploration d'autres jeux de données ayant pour but la généralisation concernant toutes les facultés de l'université 20 Aout 1955 de Skikda pour offrir des emplois du temps optimisés en utilisant GWO, satisfaisant les contraintes enseignants / étudiants.
- Les perspective de compétitivité au ITC afin d'évaluer et comparer notre approche.

[Abdesslem .L, 2010]	Abdesslem LAYEB, Utilisation des Approches d'Optimisation Combinatoire pour la Vérification des Applications Temps Réel. Thèse de Doctorat, Université Mentouri de Constantine 2010.
[Abdullah, 2006]	Abdullah, S. (2006, Juin). Heuristic Approches For University Timetabling Problems. Thesis submitted to The University of Nottingham.
[Bouamama.o,Ayadi.S, 2020]	L'Altruisme pour l'amélioration de l'Optimisation des loups gris adaptée pourle problème d'emploi du temps des cours universitaires. Université 20 Aout 1955 Skikda
[Boussaid I, 2013]	BOUSSAÏD, Ilhem. (29 juin 2013). PERFECTIONNEMENT DE MÉTAHEURISTIQUES. Perfectionnement de métaheuristiques pour l'optimisation continue. Paris-Est: THÈSE DE DOCTORAT.
[Blum et Roli 2003]	Blum.C., and Roli.A., (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM computing Surveys, 35:268-308,2003.
[C. Muroa, R. Escobedoa, b. L. Spectorc et R. P,2011]	C. Muroa, R. Escobedoa, b. L. Spectorc et R. P. Coppingerc, «Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations» 2011.
[Chan Y. C. P., 2002]	La planification du personnel : acteurs, actions et termes multiples pour une planification opérationnelle des personnes, Thèse de doctorat, Institut IMAG, Université Joseph Fourier-Grenoble, 1 octobre 2002.
[FOE, T. A., & RESEND, M. G. 1989]	A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. Operations Research Letters, 8(2), 67-71.
[Faris, Aljarah, Al-Betar, & Mirjalili, 2018]	Faris, H., Aljarah, I., Al-Betar, M., & Mirjalili, S. (2018). Grey wolf optimizer: a review of recent variants and applications. *Neural Computing and Applications, 30*(2), 413-435.
[Glover, F,1986]	Future Paths for Integer Programming and Links to Artificial Intelligence. Computers & Operations Research, 13(5), 533-549.
[Goldberg, 1989]	D. E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine.
[Hamidani.H, 2013]	Hamidani Hichem, C. (2013). Conception et Développement d'une application d'optimisation basée sur les systèmes multi-agents cas d'étude "le problème d'emploi du temps". Mémoire Master. Université KASDI MERBAH Ouargla, Algérie.
[Hamouda.Ch et al,2019]	Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification.
[Holland, J. 1975]	Adaptation in Natural and Artificial Systems. Michigan: The University of Michigan Press.
[Khelifi.S, et al, 2019]	Optimisation par loup gris adaptée pour le problème d'emploi du temps des cours universitaires. Université 20 Aout 1955 Skikda

[Leighton*, 1979]	A Graph Coloring Algorithm for Large Scheduling. JOURNAL OF RESEARCH of the Notional Bureau of Standards, 84(6), 489-505
[Luong, 2011]	Luong,T.V. (2011) Métaheuristiques, parallèles sur GPU, thèse de doctorat, Université Lille Nord- de France-Science et Technologies.
[R. Lewis and B. Paechter,2007]	‘‘Finding feasible timetables using group-based operators,’’ IEEE Trans. Evol. Comput., vol. 11, no. 3, pp. 397–413, Jun. 2007.
[Mirjalili. S, S. M. Mirjalili et A. Lewis, 2014]	S. Mirjalili, S. M. Mirjalili et A. Lewis, « Grey Wolf Optimizer» 2014.
[Mostapha .R, 2008]	Résolution de problèmes d’optimisation combinatoire par systèmes artificiels auto-organisés. Thèse de magister, Université Mentouri de Constantine ,2008.
[N.-C.-F. Bagger, G. Desaulniers, and J. Desrosiers, 2019]	Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem, J. Scheduling, vol. 22, no. 2, pp. 155–172, Apr. 2019
[Omessaad, H,2003]	Contribution au développement de méthodes d’optimisation Stochastiques application à la conception des dispositifs électrotechniques, Thèse de Doctorat, Université De Lille France 2003.
[Palpant. M,2005]	Palpant.M, Recherche exacte et approchée en optimisation combinatoire : schémas d’intégration et applications. Thèse de Doctorat, Université d’Avignon, 2005.
[Panda.M,Das.B, 2019]	«Grey Wolf Optimizer and Its Applications: A Survey» 2019.
[Peifeng Niu, et al 2019]	Peifeng Niu,Songpeng Niu, Nan liu, Lingfang Chang The defect of the Grey Wolf optimization algorithm and its verification method 01.May.2019
[Palubeckis, G. 2008]	On the recursive largest first algorithm for graph colouring. 85 (2), pp. 191-200.
[Remy-R, Alexander-J, ,07 novembre 2003]	« Systèmes interactifs d’aide à l’élaboration de plannings de travail de personnel », Thèse de doctorat, Laboratoire TIMC ,Institut IMAG , Université Joseph Fourier- Grenoble.
[Say Leng Goh, Graham Kendall, Nasser R. Sabar, and Salwani Abdullah, jun 2020.]	An effective hybrid local search approach for the post enrolment course timetabling problem. OPSEARCH
[Solone 2010]	Christine Solnon, C. (2010). Résolution de problèmes combinatoires et optimisation par colonies de fourmis.
[Troudi-F, 2006]	« Résolution de problème de l’emploi du temps : propose un algorithme multi objectif » Thèse de magister en informatique, Université Mentouri, Constantine 2006
[Vincent GARDEUX, 2011]	THÈSE DE DOCTORAT Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à

	ADN. UNIVERSITÉ DE PARIS-EST CRÉTEIL 30 novembre 2011
[Wang L, 2013]	An Improved Cooperative Particle Swarm Optimizer, Telecommunication System, vol.53, issue 1, pp. 147–154, 2013.
(Website of the Metaheuristics Network)	http://www.metaheuristics.org
[Yuichi Nagata,2018]	Random partial neighborhood search for the post enrollment course timetabling problem. Computers and Operations Research, 2018