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

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

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

THEME :

Extraction de Règles de Priorité pour le Single Machine à base de Recherche Locale et de Fouille de Données

Réalisé par :

- **Mr. MESKI Kaddour Seyf Eddine**

Sous la direction de :

- **Mr. HABIB ZAHMANI Mohamed**

Année Universitaire 2018-2019

Résumé

Dans le cadre de ce projet de fin d'études, nous proposons l'utilisation des Arbres de Décision pour l'extraction de règles de priorité pour la résolution du problème à Machine Unique avec Somme des Retards Pondérés. Pour se faire, il est d'abord nécessaire d'implémenter une des plusieurs méthodes de résolutions proposées dans la littérature et les solutions obtenues par cette dernière seront par la suite analysées à l'aide des arbres de décisions. Ainsi, nous proposons d'utiliser un algorithme de recherche locale pour la résolution du problème d'ordonnancement machine unique avec somme des retards pondérés où son but est de trouver des solutions optimales ou proches de l'optimal. A base de ces solutions, et après une phase de prétraitement, les arbres de décision sont utilisés pour extraire de nouvelles règles de priorité qui peuvent être considérées lors de la résolution en temps-réel de nouveaux de problèmes d'ordonnancement.

Mots-clés : Machine Unique · Somme des Retard Pondérés · Recherche Locale· Fouille de Données · Arbres de Décision · Règles de Priorité

REMERCIEMENTS

Tout d'abord j'exprime ma gratitude à mon encadrant, Dr. Mohamed HABIB ZAHMANI, pour son encadrement continu, son suivi, ses orientations, et ses efforts pour l'aboutissement de ce travail. Je le remercie également pour la confiance qu'il m'a toujours témoigné, mais aussi ses précieux conseils et pas uniquement sur le plan professionnel. Ma gratitude pour sa qualité humaine, son comportement envers moi, son hospitalité, et son soutien tout au long de cette année.

À tous mes précieux amis de l'Université de Mostaganem, À mes vieux amis et compagnons,

À Ma famille en particulier : Mon grand-père, ma grand-mère, ma sœur, et tous la famille MESKI & BENABDELLAH.

Je ne saurais finir sans exprimer mes remerciements aux personnes qui me sont les plus proches et les plus chères. À mes parents pour lesquels je suis éternellement reconnaissant pour m'avoir constamment soutenu, pour avoir cru en moi pour mener à bien ce travail, pour avoir mis à ma disposition tous les moyens dont ils disposaient, mais surtout pour leur amour. Merci infiniment !

Et enfin, merci à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Je remercie ***Dieu***, le Tout Puissant.

Je dédie ce travail à toutes les personnes qui sont chères à mon cœur,

À mes parents,

À ma famille, ma petite sœur, à mon grand-père, et ma grand-mère,

À mes amis.

Listes des figures :

Liste des tableaux :

Liste des abréviations:

SMTWT	Single Machine Total Weighted Tardiness
SPT	Shortest Processing Time
AG	Algorithme Génétique
ELGA	Experienced Learning Genetic Algorithm
m-VNS	multiple Variable Neighbourhood Search
HEA	Hybrid Evolutionary Algorithm
VNS	Variable Neighborhood Search
ASV	Ant Colony-based System
ILS	Iterated Local Search
ILS-RVND	Variable Neighbourhood Descent with Random Neighbourhood
ECD	Extraction de Connaissances à partir des Données
ECBD	Extraction de Connaissances dans les Bases de Données
ADD	Arbre De Décision
LS	Recherche Locale (Local Search en anglais)
TWT	Total Weighted Tardiness
RDD	Range of Due Dates
TF	Tardiness Factor
VP	Vrai Positif
FP	Faux Positif

Sommaire

Introduction Générale

Dans le cadre de la mondialisation, la concurrence devient de plus en plus rude, et les entreprises doivent s'adapter continuellement afin de maintenir ou d'augmenter leur part de marché. Cet objectif peut être atteint en diversifiant leur offre, en répondant favorablement aux exigences du client en termes de qualité, de prix et d'échéance, en respectant les normes environnementales et réglementaires. L'entreprise se trouve aussi dans l'obligation de réduire les coûts et les temps de production. L'amélioration du processus de production est étroitement liée à aux phases de *planification* et *d'ordonnancement* des opérations de productions et de l'utilisation des ressources humaines, techniques, et financières de l'entreprise.

L'ordonnancement est une tâche critique du processus de planification, où la finalité est d'allouer les ressources (humaines ou matérielles) de production disponibles aux tâches ou opérations de fabrication tout en optimisant plusieurs objectifs exprimés en termes de temps, de coût, ou de réactivité, et en respectant différentes contraintes liées à ces ressources et tâches/opérations. Les problèmes d'ordonnancement font partie de la famille des problèmes d'optimisation NP-Difficile [2] et ont attiré l'attention des chercheurs et des praticiens [1], [3] issus de différents domaines tel que l'informatique ou de l'ingénierie de production entre autres. Ces derniers ont proposé de nombreuses méthodes pour une résolution efficace des problèmes d'ordonnancement.

Dans ce projet de fin d'études, nous nous intéressons au processus d'ordonnancement et aux différentes méthodes utilisées pour le résoudre, tel que les algorithmes génétiques, les règles de priorité, ou les méthodes de recherche locale. Le but est d'implémenter l'une de ces méthodes de résolution pour résoudre un problème d'ordonnancement à une machine, plus connu sous le terme de « *single machine total weighted tardiness* ». Les solutions obtenues par la méthode de résolution seront par la suite analysées, et la fouille de données sera utilisée pour extraire de la connaissance sous la forme de règles de priorité *« si-alors »*, qui peuvent par la suite être utilisées à leur tour résoudre les problèmes d'ordonnancement en temps-réel.

Chapitre 1 : Problème d'ordonnancement

1. Introduction

Ce chapitre constitue un rappel de quelques notions de base relatives aux problèmes d'ordonnancement, ses composantes et contraintes. Il donne aussi un aperçu des approches classiques de résolution des problèmes d'ordonnancement.

Les problèmes d'ordonnancement sont présents dans tous les secteurs d'activité de l'économie depuis l'informatique, l'agroalimentaire jusqu' à l'industrie manufacturière. Les différentes données d'un problème d'ordonnancement sont les tâches, les contraintes potentielles, les ressources et la fonction économique.

L'ordonnancement est toujours présent dans les systèmes de production. Il y représente l'une des tâches les plus importantes pour planifier et séquencer les produits. Résoudre un problème d'ordonnancement consiste à déterminer des dates de démarrage de toutes les tâches et prévoir ainsi leurs dates de fin d'exécution. Mais aussi déterminer l'ensemble des ressources, humaines ou matérielles, requises en respectant les contraintes tout en en optimisant un ou plusieurs objectifs.

Cette planification permet aussi à l'entreprise de mettre en place des plans d'approvisionnement en matière première et de définir des calendriers de travail des employés. Il permet de prendre en considération les périodes de maintenance des équipements, les congés des employés ainsi que d'autres évènements pouvant altérer le calendrier de production. L'ordonnancement des ateliers de production doit aussi répondre au mieux aux besoins du client, au meilleur coût et dans les meilleurs délais.

2. Problème d'ordonnancement

Il est possible de rencontrer des problèmes d'ordonnancement dans plusieurs domaines tel que l'informatique avec la gestion des tâches par les processeurs, dans l'industrie avec la gestion de la production et les chaines de distribution, mais aussi dans d'autres domaines tel que la construction, l'administration, les hôpitaux, etc.

Les différentes données d'un problème d'ordonnancement peuvent varier d'un domaine à un autre. Dans les systèmes de production elles sont groupées en tâches, ressources, contraintes et objectifs ou critères d'optimisation.

2.1. Définition

Un problème d'ordonnancement consiste à organiser dans le temps la réalisation de tâches, compte tenu de contraintes temporelles (délais et contraintes de précédence) et de contraintes portant sur la disponibilité des ressources requises.

En production (de biens ou de services), l'ordonnancement peut être présenté comme un problème où il faut déterminer les dates de débuts et de fins de toutes les tâches, ainsi que les ressources humaines, matérielles, ou financières nécessaires pour leur réalisation.

2.2. Formulation d'un problème d'ordonnancement

Les problèmes d'ordonnancement apparaissent dans tous les domaines : informatique, industrie, construction, administration, etc. Les différentes données d'un problème d'ordonnancement sont les tâches, les ressources, les contraintes et les critères. Ainsi, étant donnés un ensemble de tâches et un ensemble de ressources, il s'agit de programmer les tâches et affecter les ressources de façon à optimiser un ou plusieurs objectifs (un objectif correspondant à un critère de performance), en respectant un ensemble de contraintes.

2.2.1. Les tâches

Une tâche (voir Figure 1) est une entité localisée dans le temps, par une date de début et une date de fin, et dont la réalisation nécessite une certaine durée. Le but de l'ordonnancement est l'allocation des ressources aux tâches/opérations de façon à optimiser certain(s) objectif(s). Pour le problème single machine, une tâche est composée d'une opération. Mais pour les autres problèmes, une tâche peut être composée de plusieurs opérations, et chaque opération nécessite une ressource. On distingue deux types de tâches :

- *Les tâches morcelables (préemptibles)* qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes,
- *Les tâches non morcelables (indivisibles)* qui doivent être exécutées en une seule fois et ne peuvent être interrompues.

$$p_i = \sum_{l=0}^{N_i-1} p_{ij}$$

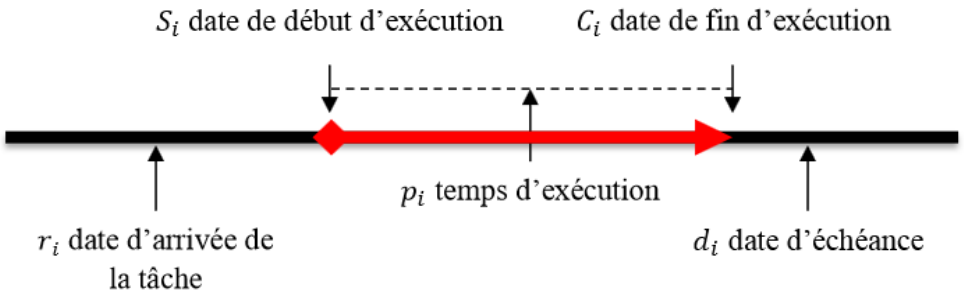

FIGURE 1 : EXEMPLE D'UNE TACHE

2.2.2. Les ressources

Une ressource est un moyen technique ou humain utilisé pour réaliser une tâche. On trouve plusieurs types de ressources :

- *Les ressources renouvelables*, qui, après avoir été allouées à une tâche, redeviennent disponibles (machines, personnel, etc.),
- *Les ressources consommables*, qui, après avoir été allouées à une tâche, ne sont plus disponibles (argent, matières premières, etc.).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours du temps. Par ailleurs, dans le cas des ressources renouvelables, on distingue principalement, les ressources disjonctives qui ne peuvent exécuter qu'une tâche à la fois et les ressources cumulatives qui peuvent être utilisées par plusieurs tâches simultanément mais en nombre limité.

2.2.3. Les contraintes

Suivant la disponibilité des ressources et suivant l'évolution temporelle, deux types de contraintes peuvent être distinguées : contraintes de ressources et contraintes temporelles.

- *Les contraintes de ressources* : plusieurs types de contraintes peuvent être induites par la nature des ressources. A titre d'exemple, la capacité limitée d'une ressource implique un certain nombre, à ne pas dépasser, de tâches à exécuter sur cette ressource.

  Les contraintes relatives aux ressources peuvent être disjonctives, induisant une contrainte de réalisation des tâches sur des intervalles temporels disjoints pour une même ressource, ou cumulatives impliquant la limitation du nombre de tâches à réaliser en parallèle.
- *Les contraintes temporelles* : elles représentent des restrictions sur les valeurs que peuvent prendre certaines variables temporelles d'ordonnancement. Ces contraintes peuvent être :

  - Des contraintes de dates butoirs, certaines tâches doivent être achevées avant une date préalablement fixée,

  - Des contraintes de précédence, une tâche/opération *i* doit précéder la tâche/opération *j*,

  - Des contraintes de dates au plus tôt, liées à l'indisponibilité de certains facteurs nécessaires pour commencer l'exécution des tâches.

2.2.4. Les critères

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi.

Les critères dépendant d'une application donnée sont très nombreux ; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la

plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types, réguliers et irréguliers.

Les différents critères ne sont pas indépendants ; certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement

- *Les critères réguliers* constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelques exemples sont cités ci-dessous :
  - La minimisation des dates de fin des tâches,
  - La minimisation du maximum des dates de fin des tâches,
  - La minimisation de la moyenne des dates de fin des tâches,
  - La minimisation des retards sur les dates de fin des tâches,
  - La minimisation du maximum des retards sur les dates de fin des tâches.
- *Les critères irréguliers* sont des critères non réguliers, c'est-à-dire qui ne sont pas des fonctions monotones des dates de fin d'exécution des opérations, tels que :
  - La minimisation des encours,
  - La minimisation du coût de stockage des matières premières,
  - L'équilibrage des charges des machines,
  - L'optimisation des changements d'outils.

La satisfaction de plusieurs critères à la fois est souvent délicate, car elle conduit souvent à des situations contradictoires et à la recherche de solutions à des problèmes complexes d'optimisation.

2.3. Types de problèmes d'ordonnancement

Une classification des problèmes d'ordonnancement dans un atelier peut s'opérer selon le nombre de machines et leur ordre d'utilisation pour fabriquer un produit, qui dépend de la nature de l'atelier considéré. Un atelier est caractérisé par le nombre de

machines qu'il contient et par son type. On distingue les trois types d'ateliers : flow-shop, job-shop et open-shop, avec des extensions possibles pour chacun d'eux.

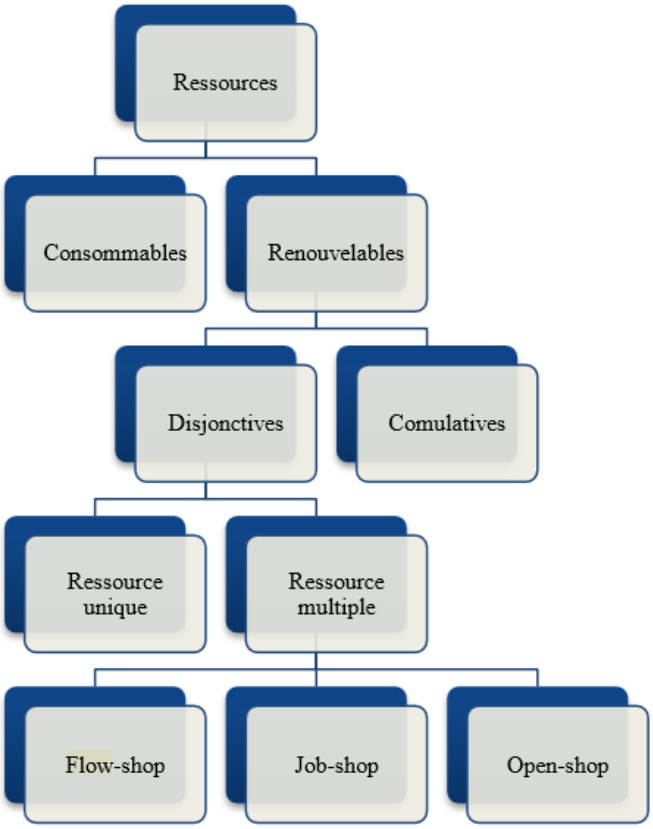

FIGURE 2 : CLASSIFICATION DES TYPES D'ATELIERS

2.3.1. Les ateliers de type flow-shop

Appelés également ateliers à cheminement unique, ce sont des ateliers où une ligne de fabrication est constituée de plusieurs machines en série, toutes les tâches passent par les machines (opérations) dans le même ordre. Dans les ateliers de type *flow-shop hybride,* une machine peut exister en plusieurs exemplaires identiques fonctionnant en parallèle.

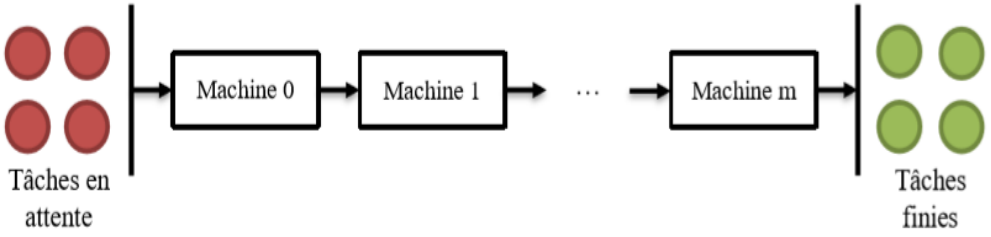

FIGURE 3 : ATELIER FLOW-SHOP

2.3.2. Les ateliers de type job-shop

Appelés également ateliers à cheminement multiple, ce sont des ateliers où les opérations sont réalisées selon un ordre bien déterminé, variant selon la tâche à exécuter. Le *job-shop flexible* est une extension du modèle job-shop classique, sa particularité réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opérations.

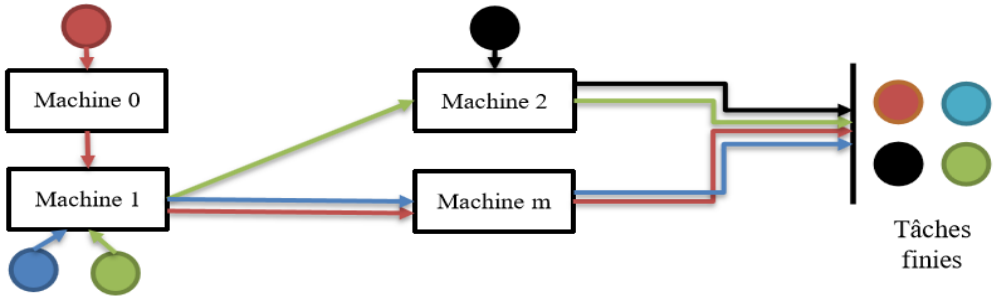

FIGURE 4 : ATELIER JOB-SHOP

2.3.3. Les ateliers de type open-shop

Ce type d'atelier ne comporte pas beaucoup de contraintes. Ainsi, l'ordre des opérations n'est pas fixé a priori, l'ordonnancement consiste, d'une part, à déterminer le cheminement de chaque tâche et, d'autre part, à ordonnancer les produits en tenant compte des gammes trouvées, ces deux problèmes pouvant être

résolus simultanément. Comparé aux autres modèles d'ateliers, l'open-shop n'est pas couramment utilisé dans les entreprises.

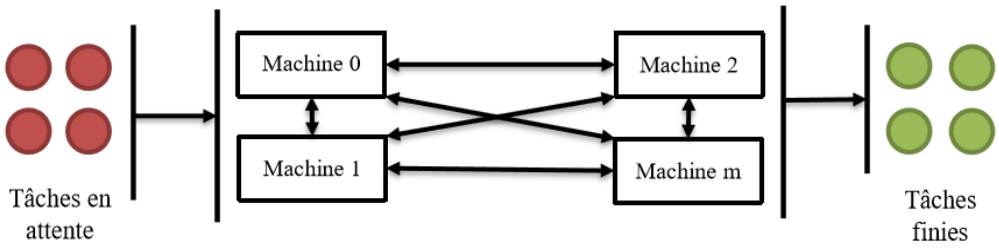

FIGURE 5 : ATELIER OPEN-SHOP

2.4. Single machine

Le problème d'ordonnancement *single machine* concerne le séquencement de n tâches indépendantes sur une seule machine afin d'optimiser un ou plusieurs objectifs. Contrairement aux autres ateliers job-shop, flow-shop ou open-shop, dans le single machine les tâches ne se composent que d'une opération. Parmi les variantes du problème single machine, on trouve le problème de minimisation de la somme des retards pondérés (SMTWT pour *single machine total weighted tardiness*) qui fait l'objet de travail ce projet de fin d'études. Dans le SMTWT, chaque tâche i possède un temps de traitement (exécution) p_i, un poids positif w_i et une date d'échéance d_i. Pour une séquence donnée, la date d'achèvement (fin d'exécution) C_i et déterminée et le retard pondéré est calculé comme suit $T_i = \max\{C_i - d_i, 0\}$. Ainsi, si la tâche i est terminée après la date prévue, une pénalité de retard pondérée $w_i \times T_i$. Le but d'un bon ordonnancement est de réduire la somme des retards pondérés des tâches [4].

3. Méthodes de résolution

Les problèmes d'ordonnancement et les problèmes d'optimisation en général sont en majorité très difficiles à résoudre dû au nombre de solutions possibles. Dans la littérature plusieurs méthodes sont proposées pour la résolution de problèmes d'ordonnancement, comme le job-shop, flow-shop ou encore le single machine.

3.1. Règle de priorité

Les règles de priorité tel que FIFO (First In First Out), LIFO (Last In First Out) ou SPT (Shortest Processing Time) font partie des méthodes les plus sollicitées pour la résolution de problèmes d'ordonnancement. Ceci est notamment dû à leur simplicité et leur rapidité d'exécution car elles reposent sur de simples tests d'attributs des tâches et/ou machines. Chaque machine de l'atelier de production est chargée d'opérer la tâche avec la priorité la plus élevée/faible se trouvant en tête de la file d'attente, évitant ainsi que la machine reste inactive alors qu'il existe au moins une tâche qui la requière. Les règles de priorité sont notamment très utilisées, en raison de leurs points forts, pour la résolution de problèmes d'ordonnancement *NP-Difficile* [5], ou dans un environnement dynamique où l'on ne maitrise pas l'arrivée des tâches, les changements de priorité, ou encore d'autres évènements imprévus telles les pannes machines.

3.2. Algorithme génétique

Parmi les méthodes évolutionnistes, nous citons les algorithmes génétiques. Les Algorithmes Génétiques (A.G.) [6] sont des algorithmes itératifs dont le but est d'optimiser une fonction prédéfinie, appelée fitness. Pour réaliser cet objectif, l'algorithme travaille sur un ensemble de points, appelés population d'individus. Chaque individu ou chromosome (chaîne de valeurs de longueur finie) représente une solution possible du problème donné. Il est constitué d'éléments, appelés gènes, dont les valeurs sont appelées allèles. L'utilisation d'un algorithme génétique nécessite la définition, au préalable, d'un espace de recherche dont les éléments de base sont les chromosomes et d'une fonction définie sur cet espace (fonction fitness) dont la valeur optimale est évaluée en rapport avec les opérateurs de croisement et de mutation choisis. Cinq éléments de base sont nécessaires pour l'utilisation des algorithmes génétiques [7] :

- *Un principe de codage* des éléments de la population, qui consiste à associer à chacun des points de l'espace d'état une structure de données, la qualité de ce codage des données conditionnant le succès des algorithmes génétiques; bien que le codage binaire ait été très utilisé à l'origine, les

codages réels sont désormais largement utilisés, notamment dans les domaines applicatifs pour l'optimisation de problèmes à variables réelles,

- *Un mécanisme de génération* de la population initiale qui doit être capable de produire une population d'individus non homogène servant de base pour les générations futures; le choix de la population initiale est important car il influence la rapidité de la convergence vers l'optimum global; dans le cas où l'on ne dispose que de peu d'informations sur le problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche,
- *Une fonction à optimiser*, appelée fitness ou fonction d'évaluation de l'individu,
- *Des opérateurs* permettant de diversifier la population au cours des générations et d'explorer l'espace d'état; l'opérateur de croisement recompose les gènes d'individus existant dans la population alors que l'opérateur de mutation garantit l'exploration de l'espace d'état,
- *Des paramètres de dimensionnement*, représentés par la taille de la population, le nombre total de générations, ou le critère d'arrêt, ainsi que les probabilités d'application des opérateurs de croisement et de mutation.

L'enchaînement de ces différents éléments est représenté dans la figure 6.

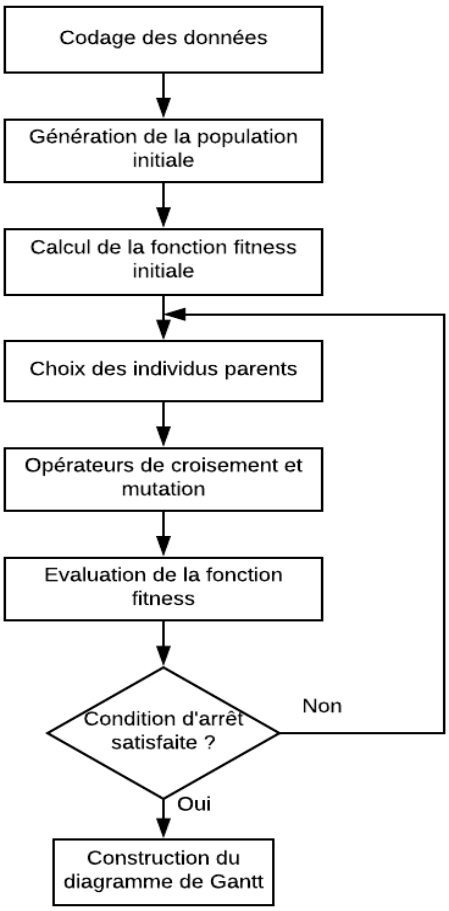

FIGURE 6 : FONCTIONNEMENT DE L'ALGORITHME GENETIQUE

3.3. Recherche tabou

Bien que son origine remonte à 1977, la recherche tabou n'a été proposée qu'au milieu des années 80 par Fred Glover. Cette méthode, développée pour résoudre des problèmes combinatoires, la plupart NP-difficiles, propose de surmonter le problème des optimums locaux par l'utilisation d'une mémoire.

La méthode tabou est une procédure itérative qui, partant d'une solution initiale, tente de converger vers la solution optimale en exécutant, à chaque pas, un mouvement dans l'espace de recherche. Chaque pas consiste d'abord à interpoler un ensemble de solutions voisines de la solution courante pour ensuite en choisir la meilleure, même si ce choix entraîne une augmentation de la fonction objective à minimiser. En acceptant de détériorer

la valeur de la solution courante, le minimum local peut être évité mais, en contrepartie, des parcours répétitifs sont déplorés [7].

Cette mémoire constitue la *liste Tabou* qui va servir à interdire l'accès aux dernières solutions visitées. Lorsqu'un optimum local est atteint, il y a interdiction de revenir sur le même chemin.

Plusieurs stratégies ont été proposées récemment afin d'améliorer l'efficacité de la méthode tabou. L'intensification et la diversification de la recherche constituent deux d'entre elles.

- L'*intensification* consiste à explorer en détails une région de l'espace de recherche jugée prometteuse. Sa mise en œuvre consiste, le plus souvent, en un élargissement temporaire du voisinage de la solution courante dans le but de visiter un ensemble de solutions partageant certaines propriétés.
- La *diversification* a pour objectif de diriger la procédure de recherche vers des régions inexplorées de l'espace de recherche. La stratégie de diversification la plus simple consiste à redémarrer périodiquement le processus de recherche à partir d'une solution, générée aléatoirement ou choisie judicieusement, dans une région non encore visitée de l'ensemble des solutions admissibles.

Les domaines d'application de la recherche tabou sont vastes et variés, ils passent de l'ordonnancement à la robotique, au problème du voyageur de commerce, à l'électronique voire même aux applications médicales, … En tenant compte des notations suivantes, les étapes d'évolution de l'algorithme sont présentées dans la figure 7.

- s_0 → la solution initiale,
- s^* → la meilleure solution actuelle,
- $f(x)$ → la fonction à minimiser,
- $f(s^*)$ → la valeur de la fonction à minimiser pour obtenir la meilleure solution
- T → la liste tabou.

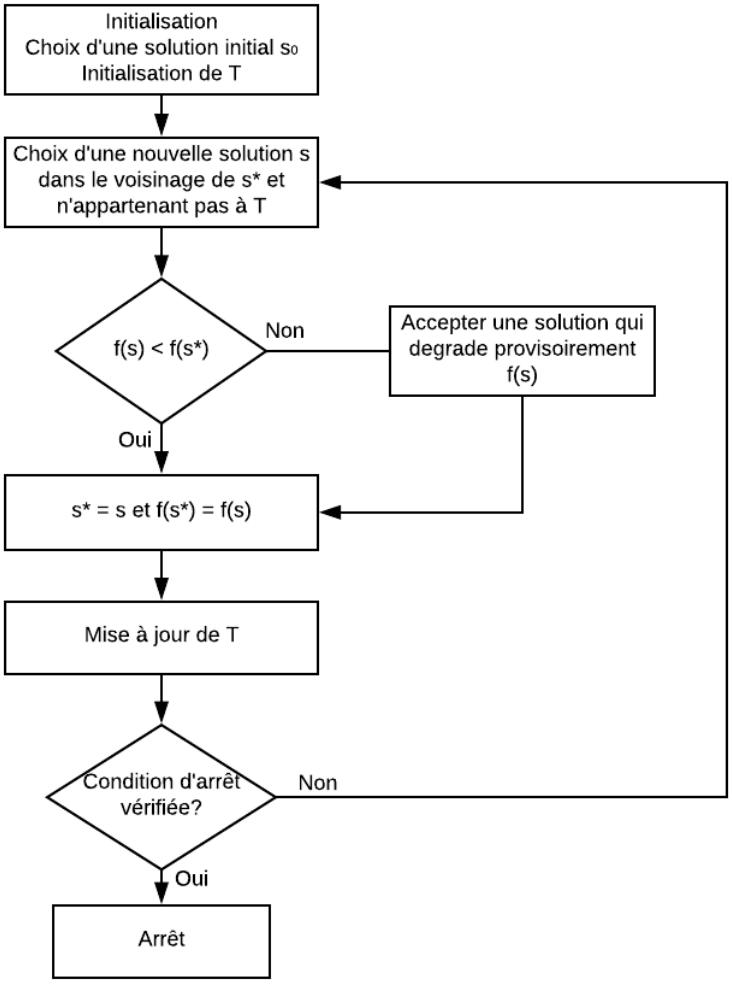

FIGURE 7 : ALGORITHME RELATIF AU FONCTIONNEMENT GENERAL DE LA METHODE DE RECHERCHE TABOU

3.4. Recherche locale

La recherche locale peut être résumée comme étant une procédure de recherche itérative qui, à partir d'une première solution réalisable, l'améliore progressivement en appliquant une série de modifications (ou mouvements) locales, comme montré dans la figure 8. Il faut, pour cela introduire une structure de voisinage qui consiste à spécifier un voisinage pour chaque solution. Ainsi, à chaque itération, la recherche s'oriente vers une nouvelle solution réalisable qui diffère légèrement de la solution courante en remplaçant celle-ci par une meilleure située dans son voisinage. La recherche se termine si un optimum

local est rencontré. L'inconvénient important de cette méthode est qu'à moins d'être extrêmement chanceux, cet optimum local est souvent une solution assez médiocre. Dans la recherche locale, la qualité des solutions obtenues dépend fortement de la richesse de l'ensemble des transformations (mouvements) considérées à chaque itération. Pour faire face à cette limitation, des méthodes de recherche locale plus sophistiquées ont été développées au cours de ces vingt dernières années. Ces méthodes acceptent des solutions voisines moins bonnes que la solution courante afin d'échapper aux minima locaux. En règle générale, seule une portion du voisinage courant est explorée à chaque étape [7]. Les méthodes les plus connues sont le recuit simulé et la recherche tabou.

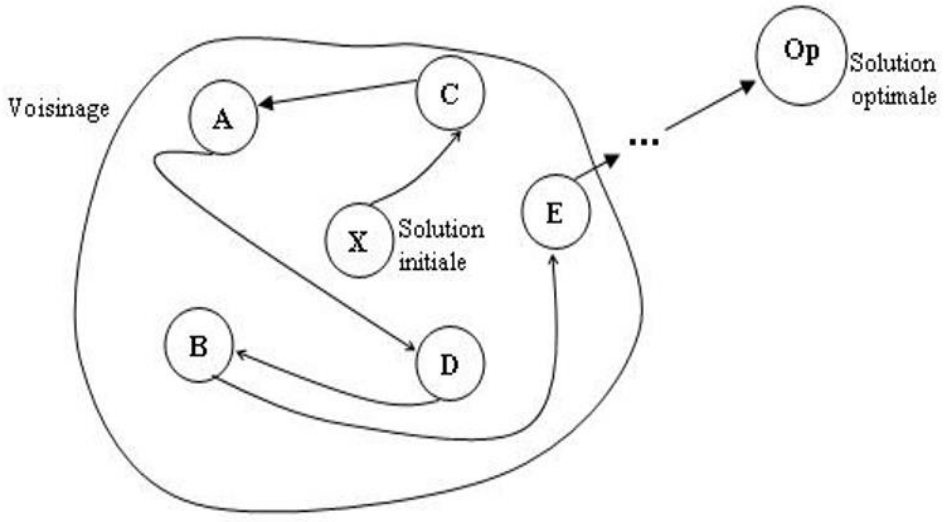

FIGURE 8 : EXPLORATION DE L'ESPACE DE RECHERCHE DANS LA METHODE DE RECHERCHE LOCALE

4. Etat de l'art

4.1 Un algorithme génétique d'apprentissage expérimenté pour résoudre le problème single machine avec somme des retards pondérés

L'auteur ont développé un algorithme génétique appelé ELGA « Experienced Learning Genetic Algorithm » pour la résolution des problèmes SMTWT. La force d'ELGA réside dans le fait qu'il reprend les informations de l'ensemble des chromosomes de la population précédente lors de l'évolution contrairement aux autres algorithmes

génétiques proposés dans la littérature, où uniquement les informations des parents sont prises en considération. Pour se faire, l'auteur utilise deux matrices tâche-tâche et position-tâche pour créer les nouveaux chromosomes à prendre en considération lors de la génération suivante. Les expérimentations sont menées sur des problèmes à Machine Unique issues de l'OR-Library (benchmarks), où ELGA arrive à atteindre quasiment l'ensemble des solutions optimales. [8].

4.2 Recherche de voisinage à variables multiples pour le problème single machine avec somme des retards pondérés

Dans cet article, un algorithme de recherche de voisinage à variables multiples avec des structures de voisinage multiples est proposé pour résoudre le problème SMTWT. Les opérations de mise en correspondance et de renforcement de m-VNS (multiple Variable Neighbourhood Search) peuvent être présentées comme une opération de révision automatique. L'algorithme repose sur des mécanismes d'appariement et de renforcement des opérations (matching and strengthening operations). Il utilise aussi une technique de recherche locale avec auto-revision qui lui permet d'explorer d'une manière efficace l'espace de rechercher et d'éviter de rester coincé dans un optimum local. Les auteurs testent l'algorithme m-VNS résoudre les problèmes benchmark SMTWT issus de la bibliothèque OR-Library. Lors de l'évaluation et la comparaison, le m-VNS est comparé avec d'autres méthodes de recherche locale à savoir PVNS (Population-based Variable Neighbourhood Search), VNS et DEVNS [9].

4.3 Une approche évolutive hybride pour le problème single machine avec somme des retards pondérés

Cet article présente un algorithme évolutif hybride pour résoudre le problème single machine avec somme des retards pondérés, qui intègre plusieurs caractéristiques distinctives telles qu'une recherche de voisinage rapide et une technique de sauvegarde. D'après les auteurs, HEA (Hybrid Evolutionary Algorithm) est le seul algorithme métaheuristique capable d'obtenir les solutions optimales pour des problèmes singles machines composés de 1000 tâches. Les expérimentations démontrent ainsi l'efficacité de

HEA en termes de qualité des solutions et de temps d'exécution (calcul/CPU). Les auteurs proposent aussi une analyse de certaines caractéristiques clés de l'HEA afin d'identifier ses facteurs critiques de succès [10]. Qui peuvent être utilisés dans de future contributions.

4.4 Une nouvelle approche pour résoudre problème single machine avec somme des retards pondérés

Dans cet article, les auteurs examinent le problème single machine avec somme des retards pondérés et proposent un VNS (Variable Neighborhood Search) pour sa résolution. Le VNS considère plusieurs structures de voisinage pour une exploration efficace de l'espace de recherche garantissant ainsi des résultats de haute qualité.

D'après les auteurs, le VNS proposé est la première métaheuristique qui a possède une structure de voisinage auto-révisée pour explorer l'espace de solution. Une autre particularité du VNS proposé réside dans la phase de génération de la solution initiale, où un modèle de probabilité est utilisé pour éviter que le VNS ne reste coincé dans un optimum local [1].

4.5 Systèmes de colonies de fourmis le problème single machine avec somme des retards pondérés

Dans cet article, les auteurs s'intéressent aussi au problème single machine avec somme des retards pondérés. Il est résolu approximativement en utilisant ASV (Ant Colony-based System), un système basé sur une colonie de fourmis avec un nombre réduit de fourmis et de colonies et avec des actions cachées (arrière-plan) qui explorent l'espace de recherche autour des fourmis en utilisant une recherche de voisinage variable (VNS). D'après les expérimentations, les auteurs montrent que l'ASV obtient l'optimum pour la plupart des problèmes SMTWT de petite taille. Alors qu'il donne un écart moyen de 0,2 % par rapport à l'optimum sur l'ensemble des instances du benchmark.

Les auteurs aussi à l'aide d'ASV, que l'utilisation d'un nombre réduit de fourmis et de générations permet d'améliorer l'efficacité de l'algorithme de recherche et de converger vers un optimum global (solution optimale). Ils montrent en outre que les actions cachées

qui consistent en une recherche de voisinage variable produisent une optima globale dans la plupart des cas. Enfin, ils soulignent l'utilité de la diversification via l'optimisation des colonies de fourmis [3].

4.6 Une heuristique de recherche locale itérée pour le problème single machine avec somme des retards pondérés et temps de préparation dépendants de la séquence

Dans cet article, les auteurs proposent un une métaheuristique de recherche locale ILS (Iterated Local Search) comprenant une méthode de recherche de voisinage variable en descendante avec voisinage aléatoire (Variable Neighbourhood Descent with Random Neighbourhood), appelée ILS-RVND, pour la résolution du problème 1 $|s_{ij}|\sum W_j T_j$ (single machine avec somme des retards pondérés et temps de préparation dépendants de la séquence). Malgré sa simplicité, l'algorithme est capable de surpasser les résultats obtenus par d'autres algorithmes heuristiques proposés dans la littérature, avec des temps de calcul similaires. Les auteurs proposent aussi une variante de l'algorithme, ILS-RVND*, qui accepte des solutions ayant la même valeur de fonction objective. Les expérimentations montrent que les solutions obtenues par ILS-RVND* peuvent surpasser celles obtenus avec ILS-RVND [11].

5. Conclusion

Dans ce chapitre nous avons présenté les concepts de base des problèmes d'ordonnancement, les types d'ateliers, ses formules, et les méthodes de résolution utilisées exactes et approchées, ainsi qu'un état de l'art qui porte principalement sur la résolution du problème single machine avec somme des retards pondérés.

Chapitre 2 : Fouille de données

1. Introduction

Ce chapitre constitue un rappel de quelques notions de base sur la Fouille de Données, ses principales tâches et méthodes. Il donne aussi un aperçu sur l'extraction de connaissances à partir de données.

L'extraction de connaissances à partir des données (ECD) est un processus non trivial d'identification de structures inconnues, valides et potentiellement utiles dans les bases de données. Son objectif est d'aider l'être humain à extraire les informations utiles (connaissances) à partir de données dont le volume croît très rapidement. Les étapes de ce processus sont, l'acquisition de données multiformes (textes, images, séquences vidéos, etc.) [12], la préparation de données (prétraitement), la fouille de données, et enfin la validation et mise en forme des connaissances. La Fouille de Données (Data Mining), se situe dans le cadre de l'apprentissage inductif. Cette phase fait appel aux multiples techniques qui permettent de découvrir des connaissances auparavant cachées dans les données et d'aider à la décision.

2. Extraction de connaissances à partir de données

L'extraction de connaissances à partir de données (ECD) est définie comme un processus de découverte d'informations implicites, inconnues auparavant et potentiellement utiles à partir de données. Ce processus s'opère en plusieurs étapes : préparation des données (recherche, nettoyage et encodage des données), fouille des données (recherche d'un modèle de connaissances), validation et interprétation des résultats et enfin intégration des connaissances apprises.

Cependant, il se trouve que dans beaucoup de domaines, les données représentées sont incomplètes et/ou imprécises ce qui rend leur exploitation très difficile et/ou impossible d'autant plus que la dimensionnalité élevée des bases de données complexifie la tâche d'extraction de connaissances à partir de cette masse de données.

2.1. Définition

L'extraction de Connaissances dans les Bases de Données (E.C.B.D.) est une discipline, à l'intersection des domaines des bases de données, de l'intelligence artificielle, de la statistique, des interfaces homme/machine et de la visualisation. A partir de données collectées par des experts, il s'agit de proposer des connaissances nouvelles qui enrichissent les interprétations du champ d'application, tout en fournissant des méthodes automatiques qui exploitent cette information.

2.2. Processus d'ECD

Le processus d'extraction de connaissances dans les bases de données, présenté dans la Figure 9, désigne l'ensemble des opérations qui permettent d'exploiter avec facilité et rapidité des données stockées massivement. Il s'agit d'un processus non trivial, consistant à identifier dans les données des schémas nouveaux, valides, potentiellement utiles et surtout compréhensibles et utilisables. Le processus d'ECD peut avoir deux objectifs, soit vérifier les hypothèses d'un utilisateur, soit découvrir de nouveaux motifs. Un motif, ou schéma, est une expression dans un langage spécifique qui décrit un sous-ensemble de données ou un modèle applicable à ce sous-ensemble [12].

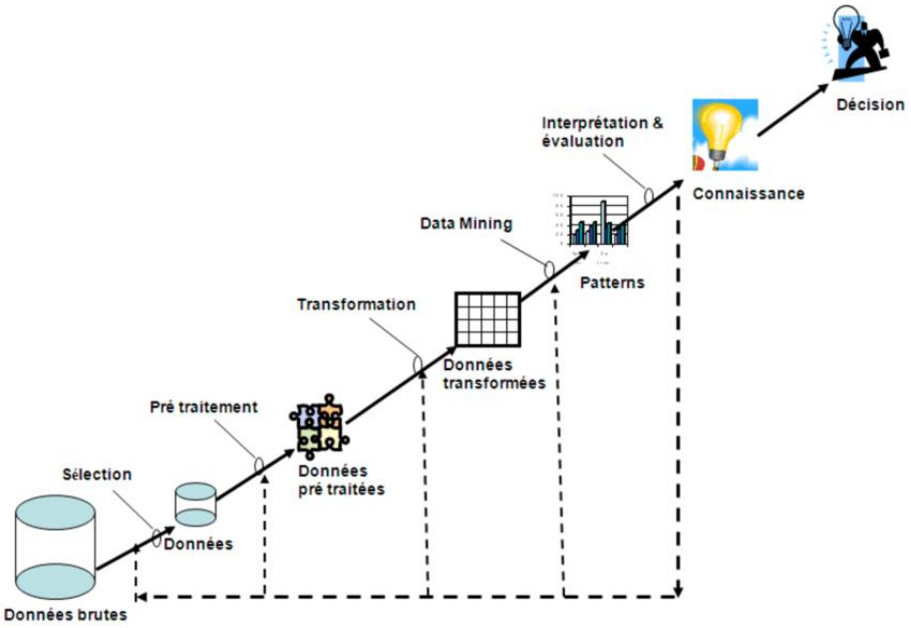

FIGURE 9 : LES DIFFERENTES ETAPES DU PROCESSUS D'ECD.

2.3. Les étapes d'un processus d'ECD

Ce processus comporte quatre étapes principales :

2.3.1. Nettoyage et intégration des données

Le nettoyage des données consiste à traiter ces données bruitées, soit en les supprimant, soit en les modifiant de manière à tirer le meilleur profit. L'intégration est la combinaison des données provenant de plusieurs sources (base de données, sources externes, etc.). Le but de ces deux opérations est de générer des entrepôts de données spécialisés contenant les données traitées pour faciliter leurs exploitations futures.

2.3.2. Prétraitement des données

Il peut arriver parfois que les bases de données contiennent à ce niveau un certain nombre de données incomplètes et/ou bruitées. Ces données erronées, manquantes ou inconsistantes doivent être traitées si cela n'a pas été fait précédemment. Dans le cas contraire, durant l'étape précédente, les données sont stockées dans un entrepôt. Cette étape permet de sélectionner et transformer des données de manière à les rendre exploitables par un outil de fouille de données. Cette seconde étape du processus d'ECD permet d'éditer les données. Si l'entrepôt de données est bien construit, le prétraitement de données peut permettre d'améliorer les résultats lors de l'interrogation dans la phase de fouille de données.

2.3.3. Fouille de données (Data Mining)

La fouille de données, est le cœur du processus d'ECD. Il s'agit à ce niveau de trouver des pépites de connaissances à partir des données. Tout le travail consiste à appliquer des méthodes intelligentes dans le but d'extraire cette connaissance. Il est possible de définir la qualité d'un modèle en fonction de critères comme les performances obtenues, la fiabilité, la compréhensibilité, la rapidité de construction et d'utilisation et enfin l'évolutivité. Tout le problème de la fouille de données réside dans le choix de la méthode adéquate à un problème donné. Il est possible

de combiner plusieurs méthodes pour essayer d'obtenir une solution optimale globale. Nous ne détaillerons pas d'avantage la fouille de données dans ce paragraphe car elle fera l'objet d'une section complète.

2.3.4. Evaluation et présentation

Cette phase est constituée de l'évaluation, qui mesure l'intérêt des motifs extraits, et de la présentation des résultats à l'utilisateur grâce à différentes techniques de visualisation. Cette étape est dépendante de la tâche de fouille de données employée. En effet, bien que l'interaction avec l'expert soit importante quelle que soit cette tâche, les techniques ne sont pas les mêmes. Ce n'est qu'à partir de la phase de présentation que l'on peut employer le terme de connaissance à condition que ces motifs soient validés par les experts du domaine.

Il y a principalement deux techniques de validation qui sont la technique de validation statistique et la technique de validation par expertise.

3. Fouille de données

Durant de la production, d'énormes quantités de données sont générées, issues de l'ordonnancement, des ressources, les capteurs, ou autre. Ces données volumineuses et, de par leur nature, complexes sont difficilement analysables et compréhensibles alors que leur potentiel pour les entreprises ou pour le secteur académique est d'une importance capitale. Ceci a donc poussé les acteurs de ces domaines à explorer de nouvelles techniques pouvant analyser et extraire de la connaissance à partir de ces gros volumes de données. La fouille de données offre ainsi une panoplie de techniques qui s'adapte quasiment à tous types d'informations et peuvent être utilisées dans les problèmes d'ordonnancement pour assister les opérateurs humains à trouver une solution. La fouille de données réussie ainsi là où il est impossible de formuler mathématiquement la connaissance des experts humains ou les données issues des systèmes de production [13].

3.1. Définition

Fouille de données, ou le data mining, est l'ensemble des méthodes et techniques destinées à l'exploration et l'analyse de bases de données informatiques (souvent grandes),

de façon automatique ou semi-automatique, en vue de détecter dans ces données des règles, des associations, des tendances inconnues ou cachées, des structures particulières restituant l'essentiel de l'information utile tout en réduisant la quantité de données.

3.2. Principales tâches de fouille de données

Les objets (données) sont représentés par des enregistrements (ou descriptions) qui sont constitués d'un ensemble de champs (ou attributs) prenant leurs valeurs dans un domaine. De nombreuses tâches peuvent être associées au data mining, parmi elles nous pouvons citer :

Classification

Elle consiste à examiner les caractéristiques d'un objet et lui attribuer une classe, la classe est un champ particulier à valeurs discrètes. Des exemples de tâche de classification sont :

- Attribuer ou non un prêt à un client,
- Établir un diagnostic médical,
- Accepter ou refuser un retrait dans un distributeur,
- Attribuer un sujet principal à un article de presse,
- etc.

Estimation

Elle consiste à estimer la valeur d'un champ à partir des caractéristiques d'un objet. Le champ à estimer est un champ à valeurs continues. L'estimation peut être utilisée dans un but de classification. Il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé. Des exemples de tâche d'estimation sont :

- Estimer les revenus d'un client,
- Estimer les risques.

Prédiction

Cela consiste à estimer une valeur future. En général, les valeurs connues sont historisées. On cherche à prédire la valeur future d'un champ. Cette tâche est proche des précédentes.

Les méthodes de classification et d'estimation peuvent être utilisées en prédiction. Des exemples de tâches de prédiction sont :

- Prédire les valeurs futures d'actions,
- Prédire, au vu de leurs actions passées, les départs de clients.

Règles d'association

Elles permettent de trouver des relations entre différentes variables. Un exemple type est celui du panier de la ménagère où un client achetant du café et du lait simultanément est susceptible d'acheter aussi du sucre. Ces règles sont ainsi utilisées afin d'identifier les opportunités de ventes et d'organiser le magasin en groupant les produits qui seront, sûrement, achetés simultanément.

Segmentation

Consiste à former des groupes (clusters) homogènes à l'intérieur d'une population. Pour cette tâche, il n'y a pas de classe à expliquer ou de valeur à prédire définie a priori, il s'agit de créer des groupes homogènes dans la population (l'ensemble des enregistrements). Il appartient ensuite à un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitués. Cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation.

3.3. Les méthodes de fouille de données

Pour tout jeu de données et un problème spécifique, il existe plusieurs méthodes que l'on choisira en fonction de :

- La tâche à résoudre,
- La nature et de la disponibilité des données,
- L'ensemble des connaissances et des compétences disponibles,
- La finalité du modèle construit,
- L'environnement social, technique, philosophique de l'entreprise,
- Etc.

On peut dégager deux grandes catégories de méthodes d'analyse consacrées à la fouille de données. La frontière entre les deux peut être définie par la spécificité des techniques, et marque l'aire proprement dite du "*Data Mining*". On distingue donc :

A. Les méthodes classiques :

On y retrouve des outils généralistes de l'informatique ou des mathématiques :

- Les requêtes dans les bases de données, simples ou multicritères, dont la représentation est une vue,
- Les requêtes d'analyse croisée, représentées par des tableaux croisés,
- Les différents graphes, graphiques et représentations,
- Les statistiques descriptives,
- L'analyse de données : analyse en composantes principales,
- Etc.

B. Les méthodes sophistiquées :

Elles ont été élaborées pour résoudre des tâches bien définies. Ce sont :

- Les algorithmes de segmentation,
- Les règles d'association,
- Les algorithmes de recherche du plus proche voisin,
- Les arbres de décision,
- Les réseaux de neurones,
- Les algorithmes génétiques,
- Etc.

4. Arbre de décision

L'arbre de décision (*ADD*) est une technique de classification supervisée proposée par Quinlan qui est considéré comme l'un des pionniers de ce type de méthodes. L'*ADD* est un outil automatisé, qui analyse un ensemble de données (enregistrements) préalablement classifiés et

génère une structure arborescente, appelé « *ensemble d'apprentissage* ». Cette structure peut ensuite être utilisée pour la classification et/ou la prédiction des nouveaux enregistrements. Le processus consiste à isoler et partitionner le plus efficacement possible les enregistrements d'une même classe. Ces algorithmes sont très prisés en raison de leur simplicité, leur performance, et leur lisibilité due à la nature arborescente des résultats. Lorsqu'un nouvel enregistrement doit être classé, l'arbre de décision est parcouru à partir de la racine et en utilisant les valeurs de l'enregistrement, jusqu'à arriver à une feuille qui représente la classe. Chaque nœud de l'arbre est une décision atomique, et chaque réponse permet de se diriger vers tel ou tel nœud fils ; une feuille représente la réponse donnée par l'arbre à la classification de l'enregistrement [5].

4.1. Types de données

Durant la phase construction de l'arbre de décision, deux types de variables (attributs) peuvent être rencontrées, quantitatives et qualitatives :

- **Quantitative** : un attribut est dit quantitatif (numérique) s'il peut prendre une valeur issue d'un ensemble fini, infini, ou défini dans un intervalle. Un attribut quantitatif peut être aussi continu ou discret :
    - Continu : si la valeur que peut prendre un attribut est réelle, composée d'un ensemble infini d'éléments, comme par exemple la taille ou le poids, ou le temps d'exécution des tâches,
    - Discret : dans le cas où l'attribut prend une valeur d'un ensemble finis d'éléments, comme par exemple les notes des élèves, ou dans le cas de l'ordonnancement, les numéros (ou type) des machines. Un attribut quantitatif discret peut aussi être considéré comme étant qualitatif.

- **Qualitative** : on parle d'attributs qualitatifs s'ils prennent des valeurs non numériques, comme le pays, la ville, ou la priorité d'une tâche (élevée, normale, faible).

On peut aussi parler de données numériques dites « *ordinales* » si les valeurs peuvent être ordonnées, comme {*court, moyen, long*} dit ordinal « *symbolique* », ou peuvent être ordinales « *numériques* » dans le cas où les éléments sont numériques et peuvent être triés

selon un ordre croissant/décroissant {*0, 1, 2, 3*}. D'autres attributs sont susceptibles d'être rencontrés dits « *binaires* », par exemple si une tâche doit être exécutée sur machine quelconque {*oui, non*} ou {*0, 1*}. On parle aussi d'attribut « *nominal* » si l'ordre n'a pas d'importance comme le type de tâches à exécuter (famille), ou la situation familiale d'un individu {*marié, célibataire, …*}.

4.2. Construction

La construction d'un arbre de décision est tributaire des attributs des enregistrements utilisés (ensemble d'apprentissage). L'ensemble d'enregistrements est divisé d'une manière récursive en utilisant différents tests jusqu'à aboutir à des feuilles contenant uniquement, idéalement, des enregistrements appartenant à une seule et unique classe. Pour diviser l'ensemble d'apprentissage il est impératif de définir les attributs qui minimisent l'impureté dans les sous arbres, c'est-à-dire, qui maximisent l'information apportée par les réponses. Une fois un attribut possédant le plus grand gain d'information identifié, il est choisi comme le prochain nœud de l'arbre et le processus est répété sur chaque nœud restant. Le signal d'arrêt est donné si l'arbre contient uniquement des exemples issus d'une même classe, ou s'il n'est plus possible d'améliorer l'arbre.

La construction de l'ADD repose sur le principe de « *diviser pour régner* » (*divide and conquer*) qui est constitué de trois phases :

- Transformer un nœud en une feuille si toutes les instances appartiennent à la même classe, ou si le taux d'erreur est inférieur à un certain seuil toléré,
- Sélectionner un test à associer à un nœud,
- Affecter la classe majoritaire à une feuille.

Pour définir l'attribut test, il est nécessaire de calculer l'apport des attributs en utilisant l'*Équation (1)* de l'entropie :

$$H(C_i) = Entropie(P_i) = -\sum_{i=0}^{i=k} P_i log_2 P_i \qquad (1)$$

Pi est la probabilité de retrouver la classe Ci dans l'ensemble d'apprentissage. L'entropie est égale à un s'il n'existe qu'une seule classe dans l'ensemble d'apprentissage, et est égale à $log_2 k$ si les k classes sont équiprobables. Lors de l'opération de partitionnement des données, on doit calculer l'information gagnante par attribut en se basant sur la formule (voir *Équation (2)*) d'entropie conditionnelle :

$$H(C_i|A) = \sum_{j=1}^{j=n} P(A = a_j) \times H(C_i|a_j) \qquad (2)$$

Où n est le nombre de valeurs possibles aj pour l'attribut A. Le gain d'information est enfin donné par l'*Équation (3)* :

$$Gain(A, C_i) = IM(A, C_i) = H(C_i) - H(C_i|A) \qquad (3)$$

L'attribut qui maximise le gain (minimise en même temps l'entropie), est choisi pour créer le prochain nœud. On crée ensuite un sous-ensemble de données selon les valeurs possibles ai pour l'attribut en question. Le processus est répété récursivement jusqu'à atteindre un critère d'arrêt. Le pseudo-code de l'algorithme peut être donné comme suit :

Algorithme1 : Construction Arbre de Décision (S, R)

Entrée : Ensemble d'apprentissage S, Liste d'attributs R

Sortie : Arbre de décision

```
Si critère d'arrêt alors
    Créer un arbre avec la classe la plus probable
Sinon
    Identifier le meilleur attribut A dans R
    Fractionner S en n sous-ensembles suivant les valeurs de A
    Créer un nœud A
    Pour chaque valeur de A
        Construction-ADD(S_i, L − {A})
    Fin pour
Fin Si
Renvoyer l'arbre de décision
```

Pour construire l'arbre de décision il existe dans la littérature une panoplie de méthodes plus ou moins différentes présentant certains avantages et/ou inconvénients par rapport à d'autres. Nous suggérons la consultation du livre de Quinlan et la thèse de Hawarah pour plus de détails. Dans ce qui suit nous présentons brièvement l'algorithme ID3.

4.3. L'algorithme ID3

L'algorithme ID3 permet de construire un arbre de décision en utilisant un ensemble S d'enregistrements dédiés à l'apprentissage et où chacun est composé d'un ensemble d'attributs R (attribut classe compris).

L'algorithme ID3 est efficace, néanmoins, pose un certain nombre de problèmes. D'abord il nécessite un temps de calcul assez important et qui augmente rapidement en fonction de la taille des données (nombre d'attributs et/ou enregistrements). L'algorithme se fie aux probabilités des attributs qui ont le plus de chance d'aiguiller les résultats, donc n'applique pas une recherche exhaustive. L'utilisation du « OU » ou « XOR » présente un certain nombre de difficultés aussi liées notamment à la redondance des tests durant la création des sous arbres. A ça s'ajoute l'impossibilité de traiter des enregistrements avec valeurs manquantes. En plus, certaines parties de l'arbre ne sont jamais visitées et contribuent par conséquent à la complexité de l'arbre inutilement. Aussi, les attributs doivent être discrétisés ce qui peut être difficile à mettre en place. Par exemple on doit discrétiser les temps d'exécution. Pour répondre à ces inconvénients de l'ID3, d'autres méthodes ont été proposées dans la littérature, parmi lesquels figure C4.5, qui est de loin la méthode la plus utilisée. Cette méthode fera l'objet d'un intérêt particulier dans le mémoire.

5. Conclusion

Dans ce chapitre, nous avons exposé le processus ECD et ses différentes étapes en général, ainsi que les méthodes de fouille de données utilisées et ses principales tâches. Et nous avons aussi exposé l'arbre de décision en général.

La fouille de données, permet d'analyser et d'imiter le comportement de méthodes de résolutions méta-heuristiques tirant ainsi avantage de ces méthodes pour la minimisation/maximisation des objectifs d'ordonnancement, tout en permettant de le faire en temps réel, une fonctionnalité héritée des règles de priorité. Durant la génération de modèles de décision, les chercheurs concentrent leurs efforts sur les arbres de décision ID3 en raison de leur simplicité, facilité de compréhension, d'utilisation, et d'implémentation des résultats qui sont sous la forme « si-alors ».

Chapitre 3 : Approche

1. Introduction

L'ordonnancement est une des tâches cruciales durant la planification, où le but et d'allouer d'une manière efficace les machines disponibles aux tâches ou aux opérations qui composent ces tâches, tout en essayant de trouver un compromis entre plusieurs objectifs exprimés en termes de coût, temps, responsivité, etc. Lors de cette opération, le responsable de production se repose souvent sur son intuition et son expérience, basée sur les connaissances acquises lors de la résolution de ce type de problèmes par le passé. Cependant, lorsque cette connaissance ne peut être explicitement capturée et mise à profit, n'importe quelle approche proposée se révèlera inefficace car ne pouvant prendre en considération les différents aspects du problème, dû en partie à sa complexité [26]. Cet état de fait pousse ainsi l'expert à s'orienter vers l'intelligence artificielle et la fouille de données afin de mettre à profit les informations issues du système de production lui-même.

Dans ce cadre, la recherche locale a été adaptée avec succès pour différents problèmes d'optimisation et pour la résolution de problèmes d'ordonnancement, y compris le problème à Machine Unique avec somme des retards pondérés, en anglais « Single Machine with Total Weighted Tardiness » (SMTWT). Dans cette approche [27], nous proposons une « *Recherche Locale* » (Local Search en anglais LS). Cette méthode, et grâce à une solution générée aléatoirement, permet d'atteindre de meilleures solutions et de parvenir dans certains cas à trouver les solutions optimales. Par la suite, la fouille de données est utilisée pour générer de la connaissance issue des solutions trouvées à l'aide de la méthode de recherche locale proposée pour générer de nouvelles règles de priorité sous la forme « *si-alors* » qui peuvent être sollicitées par la suite pour résoudre des problèmes d'ordonnancement en temps-réel contrairement à la recherche locale.

2. Single Machine total weighted tardiness

Le problème à Machine Unique avec somme des retards pondérés noté $1//\sum W_i T_i$ [17], se compose de n tâches (jobs) $J = \{J_0, J_1, \dots, J_{n-1}\}$ qui nécessitent d'être exécutées sur une seule machine. Chaque tâche J_i, aussi notée i, est constituée d'une seule opération avec un temps d'exécution $pt_i > 0$ où $\mathrm{i} = 0, \dots, \mathrm{n} - 1$ et est disponible au temps zéro. L'importance d'une tâche est exprimée en fonction d'un poids positif (weight) $w_i > 0, \mathrm{i} = 0, \dots, \mathrm{n} - 1$. Une machine ne peut opérer qu'une seule tâche à la fois, et l'exécution d'une tâche ne peut être interrompue et poursuivie plus tard. Chaque tâche i est supposée être achevée avant une date d'échéance (due date) d_i et est considérée comme « hâtive » (early) si sa date de fin d'exécution effective est plus petite que sa date d'échéance. D'un autre côté, on parle de tâche « tardive » (tardy), si la date de fin d'exécution est supérieure à la date d'échéance, une pénalité est alors assignée à la tâche donnée par $T_i = \max\{C_i - T_i; 0\}$, où C_i est la date de fin d'exécution (completion time) de la tâche dans la séquence actuelle. Le but du SMTWT est de trouver une séquence faisable S qui minimise la somme des retards pondérés, en anglais « Total Weighted Tardiness » (TWT) donnée par l'Équation 4 :

$$\min TWT = \sum_{i=0}^{n-1} w_i \times T_i \qquad (4)$$

Il a été démontré dans la littérature que le problème SMTWT est NP-Difficile [15], [16], [18], [28], par conséquence, l'utilisation de méthodes exactes n'est pas possible en raison du temps de calcul nécessaire qui augmente exponentiellement en fonction de la taille du problème.

3. Approche proposée

L'approche proposée peut être découpée en deux étapes. Durant la première, la recherche locale est utilisée pour résoudre les problèmes d'ordonnancement. Une fois la meilleure solution identifiée, la solution est prétraitée (deuxième étape) et les tâches sont comparées l'une à l'autre et une technique de création d'arbres de décision est utilisée afin de générer de nouvelles règles de priorité sous la forme « *si-alors* ».

3.1. Résolution du problème SMTWT à l'aide de la recherche locale

La technique de recherche locale proposée génère d'abord une première solution (peut aussi dans certains cas, générer un ensemble de solutions initiales (population)) aléatoirement ou à l'aide d'heuristiques ou d'autres métaheuristiques. Elle associe à la solution créée un voisinage de solutions proches de cette dernière. Dans le cas d'un problème d'ordonnancement, la solution générée est une séquence faisable de tâches S, et son voisinage est composé de solutions similaires à S. C'est-à-dire que les solutions du voisinage varient légèrement de S (changement des ordres de passage de quelques tâches dans S). L'identification des solutions du voisinage est réalisée à l'aide d'heuristiques permettant de modifier la solution S en vue, idéalement, d'atteindre de nouvelles solutions meilleures que S. Si cette nouvelle solution est effectivement meilleure, elle remplace (écraser) S et la même procédure est répétée itérativement jusqu'à ce qu'un critère d'arrêt soit atteint.

Dans la méthode de recherche locale proposée dans ce travail, plusieurs techniques sont utilisées pour explorer le voisinage des solutions. Ces techniques sont insertion, échange, insertion de deux blocs (TwoBlockInsert), insertion de trois blocs (ThreeBlockInsert), inversion d'un bloc (BlockReverse).

Insertion :

L'insertion consiste à choisir aléatoirement une tache J_k et à l'insérer dans une position choisie au hasard dans le vecteur (la solution *S*), comme décrit dans l'algorithme 1 et la Figure 10.

Algorithme 2 **: Algorithme de la méthode insertion**

Entrée : S solution
Sortie : S' nouvelle solution
Début
1 : Choisir aléatoirement un job J_k et une position α dans la solution S
2 : Insérer le job J_k à la position α
3 : Décaler à droite les autres jobs
4 : Mettre à jour Solution S'
Fin

Exemple : La technique d'insertion est illustrée dans la Figure 10, où le job J1 est sélectionné et est inséré à la position 1 (position du job J7)

FIGURE 10 : INSERTION

Echange (swap) :

L'échange (swap) consiste à interchanger deux jobs, la procédure de cette méthode est définie par l'algorithme 3 et représentée dans la Figure 11.

Algorithme 3 **: Algorithme de la méthode échange**

Entrée: S Solution

Sortie : S' nouvelle solution

Début

1 : Choisir aléatoirement un job J_β

2 : Choisir aléatoirement un job J_α

3 : Insérer le job J_β à la position α

4 : Insérer le job J_α à la position β

5 : Mettre à jour la solution S'

Fin

Exemple : La Figure 11 démontre un exemple de swap, où les jobs placés aux deuxième et cinquième positions sont échangées (J7 et J5).

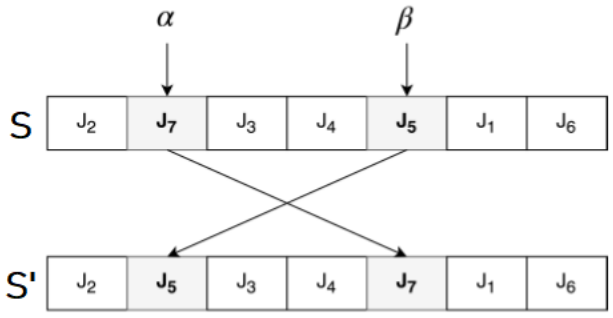

FIGURE 11 : ECHANGE (SWAP)

TwoBlockInsert :

Two Block Insert (l'insertion de deux blocs) consiste à choisir aléatoirement deux tâches consécutives et à les insérer dans une position différente choisie au hasard dans le vecteur (la solution *S*), comme décrit dans l'algorithme 4 et la Figure 12.

Algorithme 4 **: Algorithme de la méthode Two Block Insert**

Entrée : S solution

Sortie : S' nouvelle solution

Début

1 : Choisir aléatoirement un job J_k et une position α dans la solution S

2 : Sélectionner le job J_{k+1}

3 : Insérer le job J_k à la position α

4 : Insérer le job J_{k+1} à la position α+1

5 : Décaler à droite les autres jobs

6 : Mettre à jour la solution S'

Fin

Exemple : L'opérateur de two-bloc insert est illustré dans la Figure 12, où les jobs J2 et J3 sont sélectionnés et sont insérés à la position 1 (position du job J4)

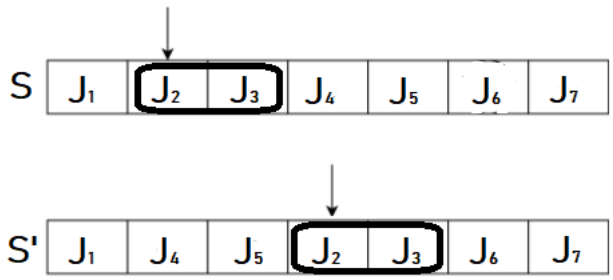

FIGURE 12 : TWO-BLOCK INSERT

ThreeBlockInsert:

Three Block Insert (l'insertion de trois blocs) consiste à choisir aléatoirement trois tâches consécutives et à les insérer dans une position choisie au hasard dans le vecteur (la solution *S*), comme décrit dans l'algorithme 5 et la Figure 13.

Algorithme 5 **: Algorithme de la méthode Three Block Insert**

Entrée : S solution
Sortie : S' nouvelle solution
Début
1 : Choisir aléatoirement un job J_k et une position α dans la solution S
2 : Sélectionner le job J_{k+1}
3 : Sélectionner le job J_{k+2}
4 : Insérer le job J_k à la position α
5 : Insérer le job J_{k+1} à la position α+1
6 : Insérer le job J_{k+2} à la position α+2
5 : Décaler à droite les autres jobs.
6 : Mettre à jour la solution S'
Fin

Exemple : L'opérateur de three-bloc insert est illustré dans la Figure 13, où les jobs J4, J5 et J6 sont sélectionnés et sont insérés à la position 1 (position du job J2)

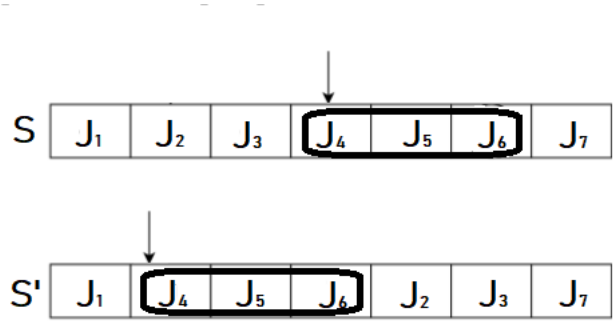

FIGURE 13 : THREE-BLOCK INSERT

Block-Reverse:

Block Reverse (inverser le bloc) consiste à choisir aléatoirement un bloc de tâches et d'inverser leurs positions, comme décrit dans l'algorithme 6 et la Figure 14.

Algorithme 6 **: Algorithme de la méthode Block-Reverse**

Entrée : S solution

Sortie : S' nouvelle solution

Début

1 : Générer aléatoirement une taille n variant entre 4 et 10

2 : Choisir aléatoirement un job J_k et une position α dans la solution S

3 : Sélectionner les jobs { $J_{k+1}, J_{k+2}, J_{k+3}, \ldots, J_{k+n-1}$.}

4 : **For** i =0, n-1 ,

5 : Insérer le job J_{k+n-i} à la position α+i

6 : **End For**

7 : Décaler à droite les autres jobs.

8 : Mettre à jour la solution S'

Fin

Exemple : L'opérateur de block-reverse est illustré dans la Figure 14, où les jobs J2, J3, J4 et J5 sont sélectionnés et sont insérés à l'inverse à la position 1 (position du job J2).

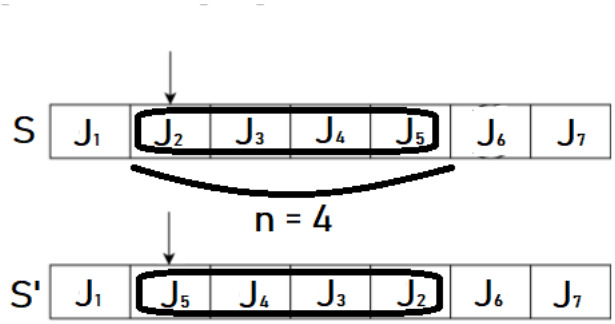

FIGURE 14 : BLOCK-REVERSE

3.2. Prétraitement et apprentissage

Pour passer à l'étape d'extraction de connaissances, il est d'abord nécessaire de préparer et formater les données. Cette procédure est d'une importance capitale si l'on veut obtenir des connaissances utiles. Dans cette approche et en s'inspirant de la littérature [14], [18], [21], [22], [24], les solutions du SMTWT obtenues par recherche locale sont transformées en un fichier de données (tableau) qui peut être utilisé par la suite comme entrée pour la méthode de fouille de données, à savoir, les arbres de décision. Ce fichier sous forme de tableau est composé de lignes et de colonnes. Chacune des colonnes représente l'un des attributs des tâches et une ligne contient une information partielle indépendantes des autres linges [26]. La création de ce fichier est illustrée dans l'exemple suivant. On suppose un problème SMTWT composé de quatre tâches (voir Tableau 1), où chaque tâche possède un temps d'exécution, un poids, et une date d'échéance.

ID tâche	Temps d'exécution $\boldsymbol{pt}$	Poids $\boldsymbol{w}$	Date d'échéance $\boldsymbol{d}$
0	5	2	10
1	4	1	12
2	7	3	15
3	8	4	20

TABLEAU 1 : PROBLEME SMTWT A QUATRE TACHES

La génération du fichier de données et directement dépendante d'une séquence faisable. Cette séquence peut être obtenue en utilisant n'importe quelle méthode de résolution y compris avec les règles de priorité et recherche locale. Pour cet exemple illustratif, on suppose que la séquence obtenue par recherche locale est |3|0|2|1|, où la tâche 3 doit être lancée en premier, suivie par les tâches 0, 2, et 1, dans cet ordre. Une fois cette solution acquise par recherche locale, un tableau (voir Tableau 2) de comparaison tâche-tâche est construit. Chaque tâche représentée par ses attributs est comparée aux autres tâches en se basant sur la séquence de recherche locale et un attribut classe est défini, appelé « CLASSE ». Cette classe peut prendre deux valeurs « oui » si la tâche 1 doit être lancée avec la tâche 2, « non » sinon.

Tâche1	pt_1	$w1$	$d1$	**Tâche2**	$pt2$	$w2$	$d2$	**Tâche1 première**
0	5	2	10	1	4	1	12	Oui
0	5	2	10	2	7	3	15	Oui
0	5	2	10	3	8	4	20	Non
1	4	1	12	2	7	3	15	Non
1	4	1	12	3	8	4	20	Non
2	7	3	15	3	8	4	20	Non

TABLEAU 2 : TABLEAU DE COMPARAISON TACHE-TACHE

Une fois que le fichier est construit, une opération de prétraitement est alors conduite. Le prétraitement peut inclure la construction et la sélection d'attributs, l'agrégation, ou la suppression de données redondantes. Cette étape est très importante dans le processus d'extraction de données. En général, d'énormes efforts doivent être déployés en vue de préparer les données avant de pouvoir utiliser la fouille de données et d'extraire de la connaissance utile. Durant ces expérimentations, seule une suppression des doublons est menée car les données contenues dans ce fichier sont correctes et complètes et ne nécessitent aucune autre opération de prétraitement.

Les attributs pris en considérations dans ces expérimentations, et comme illustré dans le Tableau 1 et le Tableau 2, sont le temps d'exécution, le poids, et la date d'échéance. Ces

attributs sont indépendants et ne peuvent être agrégés. En plus, vu que ces attributs sont disponibles pour chaque tâche et pour chaque problème d'ordonnancement, le fichier obtenu ne contient aucune valeur manquante ou incorrecte. Les identifiants des tâches (colonnes « Tâche 1 » et « Tâche 2 ») affichés dans le Tableau 2 ne sont pas pris en considération lors de l'apprentissage et sont affichés uniquement pour faciliter la compréhension de cet exemple.

Lorsque le prétraitement est accompli, un algorithme d'apprentissage machine est utilisé pour générer un modèle prédictif. Plusieurs algorithmes pour l'apprentissage peuvent être considérés comme les réseaux de neurones, les machines à vecteurs de support, ou les arbres de décision. Ces derniers, selon Li & Olafsson [26], sont facilement compréhensibles et peuvent être traités, lors de l'ordonnancement, par un opérateur humain contrairement aux réseaux de neurones considérés comme une boite noire. Dans cette approche, des arbres de décision sont utilisées, car suggérées par Li & Olafsson [26], mais aussi en raison de leur simplicité et leur efficacité. De plus, ces algorithmes ont été largement sollicité dans la littérature pour l'extraction de connaissance notamment dans les travaux de recherche [14], [18], [20], [26]. Le module d'apprentissage repose sur l'algorithme J48 pour l'extraction de connaissance à partir du fichier de données sous la forme « si-alors » (ou sous une forme visuelle arborescente). J48 est une implémentation Java de l'algorithme C4.5 disponible dans le logiciel de fouille de données Weka développé par Hall et al. [25]. L'arbre de décision obtenu est comme suit (voir aussi Figure 15) :

Si $d1 \leq 10$ **et** $pt2 \leq 7$ **alors** la Tâche 1 est lancée en premier (**oui**)

Si $d1 \leq 10$ **et** $pt2 > 7$ **alors** la Tâche 2 est lancée en premier (**non**)

Si $d1 > 10$ **alors** la Tâche 2 est lancée en premier (**non**)

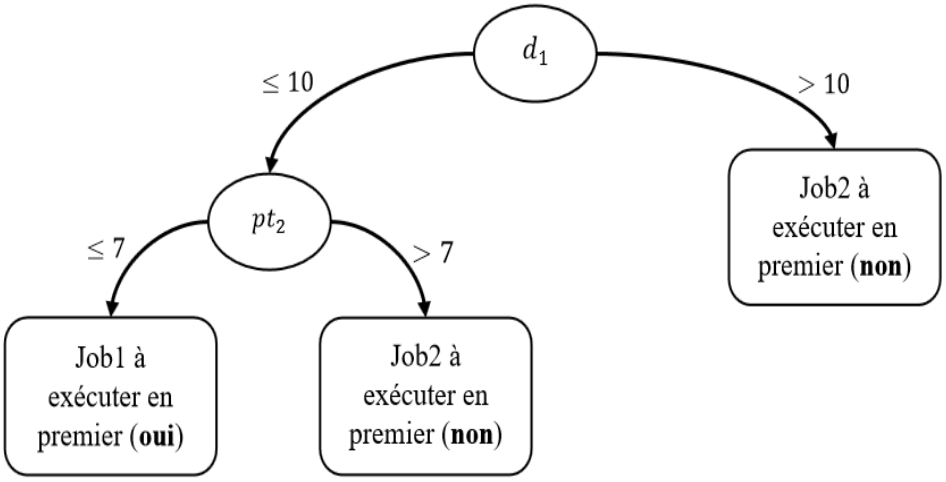

FIGURE 15 : ARBRE DE DECISION

L'arbre de décision obtenu est ensuite testé et évalué en vue d'améliorer son fonctionnement et son efficacité. En ce sens, plusieurs mesures peuvent être calculées, comme le taux d'apprentissage, la précision, ou le rappel. Si les résultats ne sont pas jugés satisfaisants, alors la procédure d'apprentissage et répétée en changeant les paramètres de l'algorithme jusqu'à obtenir les performances désirées. Enfin, ces règles « si-alors » peuvent être utilisées afin de résoudre de nouveaux problèmes d'ordonnancement [22].

4. Expérimentation, Résultats et discussion

Les expérimentations sont menées sur des ateliers SMTWT composés de 40, 50, et 100 tâches. Les résultats de LS sont montrés et discutés en premier, suivis de l'utilisation de l'arbre de décision J48 avec les heuristiques proposées. Les expérimentations sont codées en Java est menées sur un ordinateur avec un processeur Intel Core i7 2.4 GHz et 16 Go de mémoire vive avec un seul thread.

4.1. Description du benchmark (SMTWT)

Les problèmes SMTWT avec leurs meilleures solutions connues de l'ORLibrary [23] sont considérés dans ce projet. Beasley [23] donne la procédure utilisée pour générer

ces problèmes benchmark de taille n = 40, n = 50 et n = 100, où n est le nombre de tâches et où chacune des catégories (taille du problème) est constituée de 125 instances. Chaque tâche arrive au temps zéro et possède un temps d'exécution pt (processing time) oscillant entre 1 et 100, une date d'échéance d (due date), et un poids w (weight) variant entre 1 et 10. Ces deux attributs sont générés suivant une distribution uniforme. En ce qui concerne la date d'échéance, elle est aussi générée aléatoirement suivant une distribution uniforme $[P\ (1 - TF - RDD/2),\ P\ (1 - TF + RDD/2)]$ où RDD désigne « Range of Due Dates » RDD = {0.2,0.4,0.6,0.8,1.0} et TF est « Tardiness Factor » TF = {0.2,0.4,0.6,0.8,1.0}. Lors de la génération de ces problèmes par Beasley [23] pour chaque combinaison de valeurs RDD/TF, c'est-à-dire 25 combinaisons possibles, cinq instances sont générées donnant ainsi un ensemble de 125 instances pour chaque taille de problèmes. Lors des expérimentations, les solutions de la recherche locale proposée sont comparées avec les meilleures TWT connues données dans l'ORLibrary.

4.2. Recherche locale

Pour chaque problème d'ordonnancement à Machine Unique, la meilleure valeur de la somme des retards pondérés est connue ; par conséquent, l'écart entre LS (recherche locale) et les meilleurs résultats est calculé en utilisant l'Équation 5, où *LS* (i) est le plus petit TWT obtenu par LS pour un problème *i* alors que $Meilleur$(i) est la meilleure valeur TWT connue (disponible dans l'ORLibrary) pour ce même problème i.

$$\left(\boldsymbol{LS}(\boldsymbol{i}) - \boldsymbol{Meilleur}(\boldsymbol{i})\right) \Big/ \boldsymbol{Meilleur}(\boldsymbol{i}) \qquad \textbf{(5)}$$

Il est important de noter que pour certains problèmes la meilleure valeur du TWT connue est égale à zéro. En conséquence, si LS trouve une solution avec un objectif égal à zéro,

alors l'écart est automatiquement fixé à zéro. Sinon, si LS donne un TWT supérieur à zéro alors que la meilleure valeur est égale à zéro (division par zéro), le problème est ignoré.

Dans ce qui suit nous présentons les résultats de tous les problèmes SMTWT, c'est-à-dire, 375 problèmes : 125 problèmes à 40 tâches, 125 problèmes à 50 tâches, et 125 problèmes à 100 tâches. Pour rappel, le nombre maximal d'itérations de LS pour les problèmes à 40, 50 et 100 tâches est fixé à 10.000. Une fois ce plafond atteint, la recherche locale est suspendue et la meilleure solution obtenue jusque-là est renvoyée puis sauvegardée pour l'apprentissage.

RDD	TF	Moy LS TWT	Moy Best TWT	Moy Ecart
0.2	0.2	1179	1151.8	3.21%
	0.4	9297.2	9221.2	1.10%
	0.6	21501.6	21464.8	0.19%
	0.8	73129	73120.2	0.01%
	1.0	112514	112514	0.00%
0.4	0.2	67.6	66.4	1.88%
	0.4	4824.6	4815.8	0.17%
	0.6	20223.8	20039.8	0.89%
	0.8	69835.8	69790.8	0.06%
	1.0	91744.4	91736.8	0.01%
0.6	0.2	0	0	0.00%
	0.4	3345	3273.6	2.16%
	0.6	18665	18541.2	0.62%
	0.8	71904.6	71892.4	0.01%
	1.0	90277.4	90276	0.00%
0.8	0.2	0	0	0.00%
	0.4	618.8	609.4	1.21%
	0.6	14610.8	14593.8	0.10%
	0.8	49732.4	49719.8	0.02%
	1.0	121667.6	121667.6	0.00%
1	0.2	0	0	0.00%
	0.4	774	774	0.00%
	0.6	22770.6	22629.2	0.99%
	0.8	51770.6	51664	0.27%
	1.0	91482.4	91482.4	0.00%

TABLEAU 3 : COMPARAISON DES RESULTATS DU LS AVEC LES MEILLEURS RESULTATS CONNUS POUR LES PROBLEMES 40.

Dans le Tableau 3 sont affichés les moyennes des sommes de retards pondérés (TWT) obtenus par la recherche locale, les moyennes TWT des meilleures solutions (best) ainsi que l'écart moyen pour chaque combinaison RDD/TF. Ainsi la recherche locale retrouve les meilleures solutions pour 86 sur 125 problèmes soit un taux de 68,80% avec un écart moyen avec les meilleures valeurs de l'OR-Library (best) de 0,52%. Dans la Figure 16, nous présentons l'écart en pourcentage entre les solutions obtenues par la recherche locale proposée et les meilleures solutions de l'OR-Library pour les 125 instances à 40 tâches.

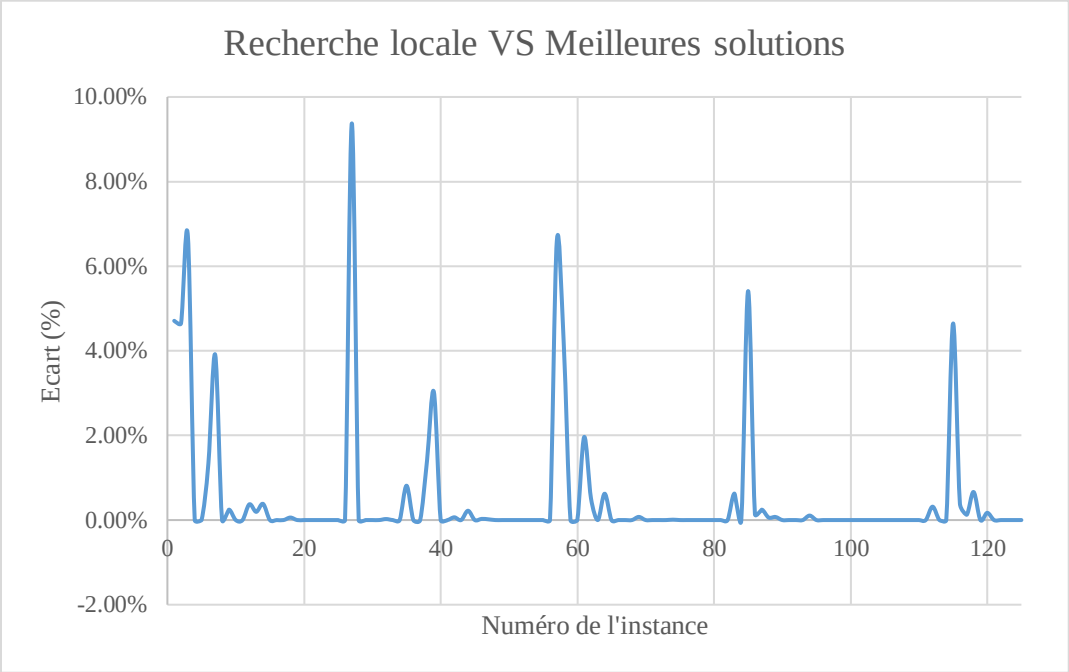

FIGURE 16 : DIFFERENCE EN TERMES DE TWT POUR LES PROBLEMES A 40 TACHES

En ce qui concerne les problèmes à 50 tâches où le nombre maximal d'itérations est aussi fixé à 10.000, la recherche locale retrouve 56,80% des solutions optimales, soit 71 problèmes sur les 125. Pour ces problèmes, LS a un écart moyen de 0,49% en comparaisons avec les valeurs de l'OR-Library. Les résultats de la recherche locale pour les problèmes à 50 tâches sont donnés dans le Tableau 4 et la Figure 17.

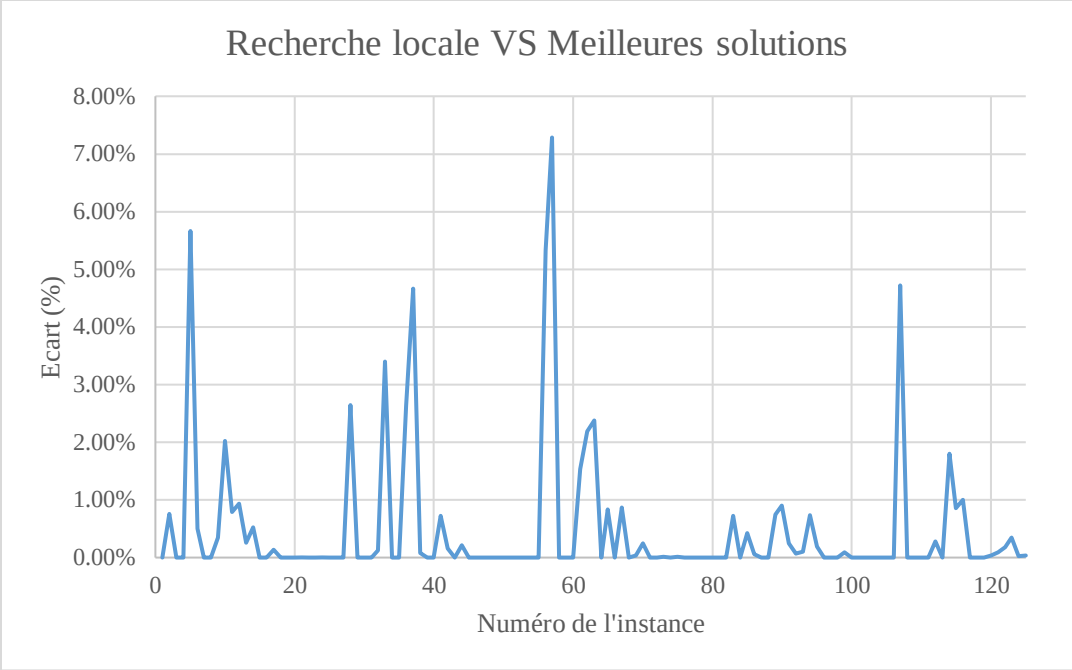

FIGURE 17 : DIFFERENCE EN TERMES DE TWT POUR LES PROBLEMES A 50 TACHES

RDD	TF	Moy LS TWT	Moy Best TWT	Moy Ecart
0.2	0.2	2204.6	2184.4	1.28%
	0.4	13420	13343.4	0.58%
	0.6	43411.4	43196.8	0.50%
	0.8	87737.4	87714.4	0.03%
	1.0	189115.6	189113	0.00%
0.4	0.2	180.4	176.4	0.53%
	0.4	6486	6452.4	0.71%
	0.6	32975.4	32574.6	1.48%
	0.8	90018.4	89835.2	0.22%
	1.0	166049.6	166049.6	0.00%
0.6	0.2	0	0	0.00%
	0.4	3493.6	3426.6	2.52%
	0.6	23618.6	23277.6	1.39%
	0.8	81735.6	81545.4	0.23%
	1.0	130372.8	130365	0.01%
0.8	0.2	0	0	0.00%
	0.4	2195.8	2191.2	0.23%
	0.6	25946.8	25873.8	0.34%
	0.8	63285.8	63134.6	0.27%
	1.0	153187.6	153155.6	0.02%
	0.2	0	0	0.00%

1	0.4	1886.6	1839.4	0.94%
	0.6	20947.6	20864.8	0.59%
	0.8	76236.4	76158	0.21%
	1.0	109997.8	109855.4	0.14%

TABLEAU 4 : COMPARAISON DES RESULTATS DU LS AVEC LES MEILLEURS RESULTATS CONNUS POUR LES PROBLEMES 50.

Quant aux problèmes composés de 100 tâches, la recherche locale retrouve les meilleures solutions pour 54 sur 125 problèmes soit un taux de 43,20% des solutions optimales, Pour ces problèmes, LS a un écart moyen de 0,31% en comparaisons avec les valeurs de l'OR-Library. Dans la Figure 18 et le Tableau 5 sont présentés les écarts en TWT entre LS et les meilleurs résultats connus pour chaque problème.

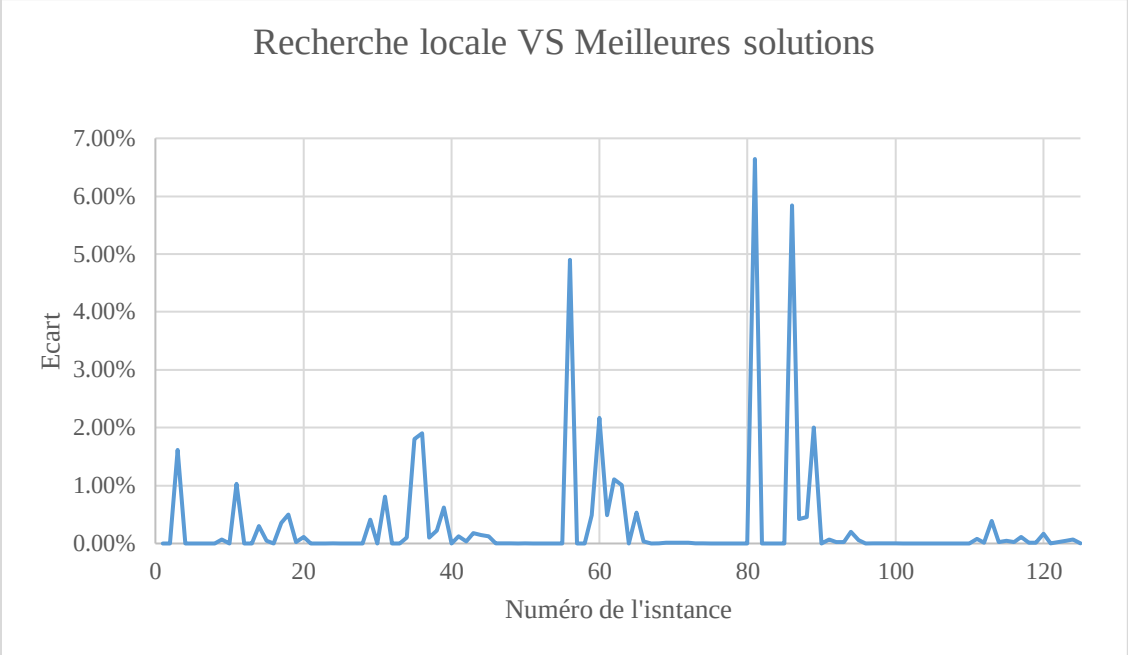

FIGURE 18 : DIFFERENCE EN TERMES DE TWT POUR LES PROBLEMES A 100 TACHES

RDD	TF	Moy LS TWT	Moy Best TWT	Moy Ecart
0.2	0.2	5357.6	5343.8	0.32%
	0.4	52575.4	52570	0.01%
	0.6	185513.8	185027.8	0.28%
	0.8	434714.2	433824.6	0.20%
	1.0	665022.2	665021.4	0.00%
0.4	0.2	257	256.6	0.08%
	0.4	24905.4	24792.8	0.54%
	0.6	133082.6	132402.4	0.57%
	0.8	375431.6	374993.8	0.12%
	1.0	691641.2	691626.8	0.00%
0.6	0.2	0	0	0.00%
	0.4	13137.8	12955	1.51%
	0.6	86076	85544.2	0.63%
	0.8	315212.2	315179.2	0.01%
	1.0	607146.2	607101.8	0.01%
0.8	0.2	0	0	0.00%
	0.4	675.2	656.6	1.33%
	0.6	68380.4	67259.2	1.74%
	0.8	295588.4	295368.4	0.08%
	1.0	576924.4	576902	0.00%
1	0.2	0	0	0.00%
	0.4	285	285	0.00%
	0.6	132739.8	132623	0.11%
	0.8	300628.8	300435	0.07%
	1.0	486236.2	486114.2	0.03%

TABLEAU 5 : COMPARAISON DES RESULTATS DU LS AVEC LES MEILLEURS RESULTATS CONNUS POUR LES PROBLEMES 100

4.3. Prétraitement et construction de l'arbre de décision

La partie de fouille de données et basée sur le classifieur J48 pour l'extraction de connaissances à partir de la seconde base de données après un processus de prétraitement. J48 est une implémentation de l'algorithme C4.5 disponible dans le logiciel de fouille de données « Weka » développé par Hall et al. [25]. Après la création de l'arbre de décision plusieurs mesures sont calculées qui permettent d'apprécier et d'analyser les performances de ce dernier ainsi que de s'assurer de la bonne classification des tâches en utilisant l'ensemble d'apprentissage. Ces mesures sont :

- Vrai Positif (VP) : le nombre de résultats positifs correctement classés sur le nombre total de positifs pour chaque classe (règle de priorité),
- Faux Positif (FP) : le nombre de résultats négatifs incorrectement classés sur le nombre total de négatifs pour chaque classe (règle de priorité),
- Précision : nombre d'instances correctement classées sur le nombre total d'instances prédites par classe, Rappel : nombre d'instances correctement classées sur le nombre total d'instances positives par classe,
- F-Mesure : est une mesure qui permet d'évaluer l'arbre de décision et est données par l'Équation 6 :

$$\boldsymbol{F - Mesure = \frac{2 \times Rappel \times Précision}{Rappel + Précision}} \quad \textbf{(6)}$$

Les valeurs de VP, FP, précision, rappel, et F-Mesure sont affichés dans le Tableau 6 pour les problèmes de 40 jobs.

RDD	TF	Moy. VP (%)	Moy. FP (%)	Moy. Précision (%)	Moy. Rappel (%)	Moy. F-Measure (%)
0.2	0.2	91.94%	8.06%	91.90%	91.94%	91.91%
	0.4	91.33%	8.67%	91.23%	91.33%	91.24%
	0.6	88.67%	11.33%	88.63%	88.67%	88.61%
	0.8	90.64%	9.36%	90.67%	90.64%	90.64%
	1.0	86.06%	13.94%	86.39%	86.06%	86.12%
0.4	0.2	92.34%	7.66%	92.60%	92.34%	92.44%
	0.4	90.98%	9.02%	91.48%	90.98%	91.16%
	0.6	89.37%	10.63%	89.58%	89.37%	89.40%
	0.8	91.80%	8.20%	91.82%	91.80%	91.78%
	1.0	86.99%	13.01%	87.51%	86.99%	87.15%
0.6	0.2	90.32%	9.68%	90.20%	90.32%	90.17%
	0.4	90.99%	9.01%	90.99%	90.99%	90.94%
	0.6	89.49%	10.51%	89.58%	89.49%	89.37%
	0.8	89.87%	10.13%	90.00%	89.87%	89.89%
	1.0	88.98%	11.02%	88.84%	88.98%	88.84%
0.8	0.2	90.99%	9.01%	91.27%	90.99%	91.05%
	0.4	88.79%	11.21%	88.88%	88.79%	88.80%
	0.6	88.46%	11.54%	88.50%	88.46%	88.46%
	0.8	90.25%	9.75%	90.17%	90.25%	90.19%

	1.0	88.80%	11.20%	89.04%	88.80%	88.88%
1	0.2	90.84%	9.16%	90.77%	90.84%	90.74%
	0.4	90.70%	9.30%	91.66%	90.70%	91.02%
	0.6	88.53%	11.47%	88.50%	88.53%	88.50%
	0.8	91.64%	8.36%	91.81%	91.64%	91.66%
	1.0	88.32%	11.68%	88.25%	88.32%	88.10%
Moyenne		89.88%	10.12%	90.01%	89.88%	89.88%

TABLEAU 6 : MESURE DE PERFORMANCE DE L'ADD POUR LES PROBLEMES DE 40 JOB.

Les valeurs de VP, FP, précision, rappel, et F-Mesure sont affichés dans le Tableau 7 pour les problèmes de 5 jobs.

RDD	TF	Moy. VP (%)	Moy. FP (%)	Moy. Précision (%)	Moy. Rappel (%)	Moy. F-Measure (%)
0.2	0.2	88.50%	11.50%	89.28%	88.50%	88.84%
	0.4	88.56%	11.44%	88.82%	88.56%	88.64%
	0.6	87.36%	12.64%	87.48%	87.36%	87.39%
	0.8	86.50%	13.50%	86.91%	86.50%	86.67%
	1.0	86.76%	13.24%	87.34%	86.76%	86.97%
0.4	0.2	87.86%	12.14%	88.19%	87.86%	88.01%
	0.4	88.71%	11.29%	89.07%	88.71%	88.78%
	0.6	88.02%	11.98%	88.38%	88.02%	88.13%
	0.8	88.75%	11.25%	89.05%	88.75%	88.82%
	1.0	88.68%	11.32%	89.07%	88.68%	88.84%
0.6	0.2	89.58%	10.42%	90.04%	89.58%	89.76%
	0.4	85.72%	14.28%	87.76%	85.72%	86.29%
	0.6	87.10%	12.90%	87.41%	87.10%	87.22%
	0.8	88.18%	11.82%	88.21%	88.18%	88.05%
	1.0	87.34%	12.66%	88.13%	87.34%	87.63%
0.8	0.2	90.16%	9.84%	90.36%	90.16%	90.20%
	0.4	88.06%	11.94%	88.78%	88.06%	88.29%
	0.6	90.85%	9.15%	91.67%	90.85%	91.17%
	0.8	88.91%	11.09%	89.34%	88.91%	89.06%
	1.0	87.16%	12.84%	87.36%	87.16%	87.23%
1	0.2	89.85%	10.15%	90.39%	89.85%	90.06%
	0.4	87.25%	12.75%	87.66%	87.25%	87.37%
	0.6	89.01%	10.99%	89.11%	89.01%	89.05%
	0.8	86.69%	13.31%	86.60%	86.69%	86.52%
	1.0	86.19%	13.81%	86.71%	86.19%	86.36%
Moyenne		88.07%	11.93%	88.53%	88.07%	88.21%

TABLEAU 7 : MESURE DE PERFORMANCE DE L'ADD POUR LES PROBLEMES DE 50 JOB.

Les valeurs de VP, FP, précision, rappel, et F-Mesure sont affichés dans le Tableau 8 pour les problèmes de 100 jobs.

RDD	TF	Moy. VP (%)	Moy. FP (%)	Moy. Précision (%)	Moy. Rappel (%)	Moy. F-Measure (%)
0.2	0.2	87.89%	12.11%	87.88%	87.89%	87.85%
	0.4	86.32%	13.68%	86.39%	86.32%	86.35%
	0.6	85.45%	14.55%	85.39%	85.45%	85.40%
	0.8	87.77%	12.23%	87.56%	87.77%	87.65%
	1.0	85.85%	14.15%	86.02%	85.85%	85.92%
0.4	0.2	86.46%	13.54%	86.46%	86.46%	86.44%
	0.4	86.26%	13.74%	86.44%	86.26%	86.28%
	0.6	89.69%	10.31%	89.50%	89.69%	89.53%
	0.8	85.46%	14.54%	85.55%	85.46%	85.45%
	1.0	87.04%	12.96%	87.18%	87.04%	87.09%
0.6	0.2	86.34%	13.66%	86.94%	86.34%	86.55%
	0.4	87.48%	12.52%	87.63%	87.48%	87.55%
	0.6	87.62%	12.38%	87.82%	87.62%	87.70%
	0.8	86.42%	13.58%	86.56%	86.42%	86.47%
	1.0	85.82%	14.18%	86.10%	85.82%	85.94%
0.8	0.2	86.54%	13.46%	86.65%	86.54%	86.57%
	0.4	87.12%	12.88%	86.96%	87.12%	87.02%
	0.6	87.39%	12.61%	87.15%	87.39%	87.25%
	0.8	86.76%	13.24%	87.16%	86.76%	86.86%
	1.0	82.18%	17.82%	81.85%	82.18%	81.96%
1	0.2	86.18%	13.82%	86.00%	86.18%	86.07%
	0.4	88.43%	11.57%	88.46%	88.43%	88.44%
	0.6	84.73%	15.27%	85.09%	84.73%	84.85%
	0.8	86.98%	13.02%	86.76%	86.98%	86.83%
	1.0	86.33%	13.67%	86.36%	86.33%	86.29%
Moyenne		86.58%	13.42%	86.64%	86.58%	86.57%

TABLEAU 8 : MESURE DE PERFORMANCE DE L'ADD POUR LES PROBLEMES DE 100 JOB.

5. interface de l'application

Dans cette partie nous allons nous pencher sur l'interface de l'application, et expliquer comment l'ont fait pour la faire fonctionner. Lors de la conception de l'interface, nous avons tenté de créer quelque chose de très simple et intuitive pour les utilisateurs.

Tout d'abord, l'utilisateur doit choisir le type de problème (taille) et sélectionner l'instance à résoudre (numéro du problème) et le nombre maximal d'itération (voir Figure 19).

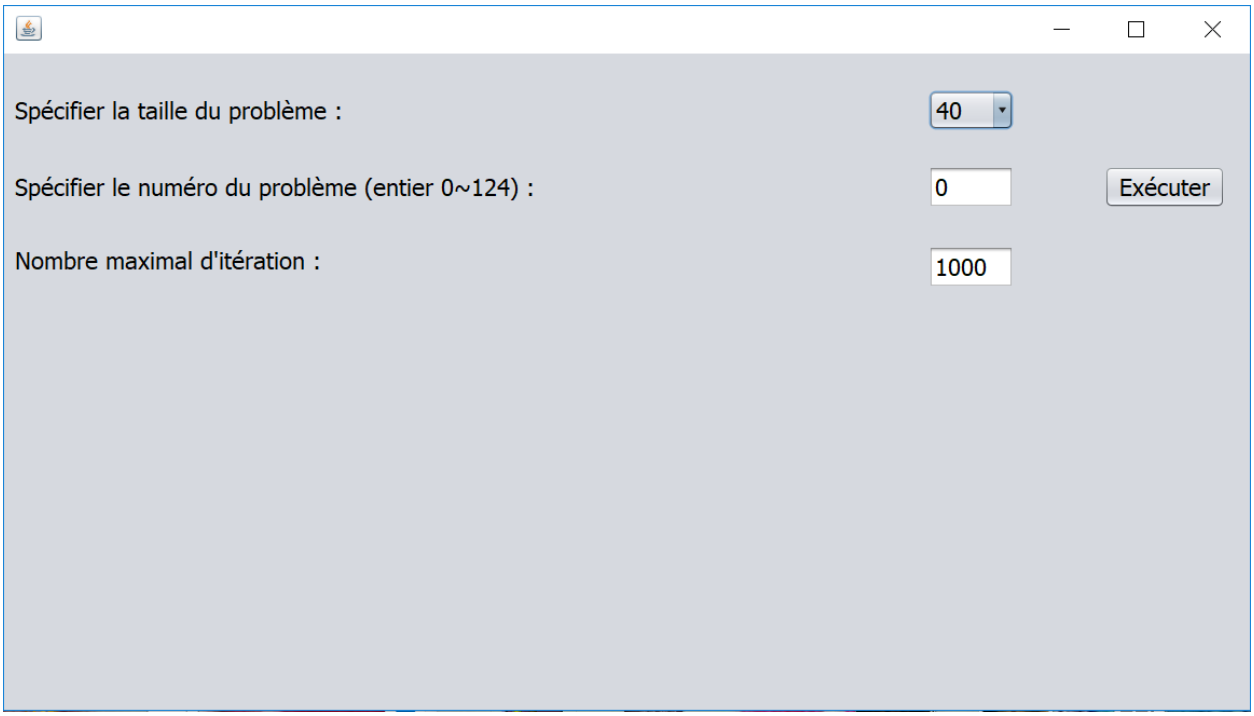

FIGURE 19 : SAISIE DES VALEURS

Une fois la recherche locale est lancée (bouton Exécuter), l'utilisateur obtient ceci.

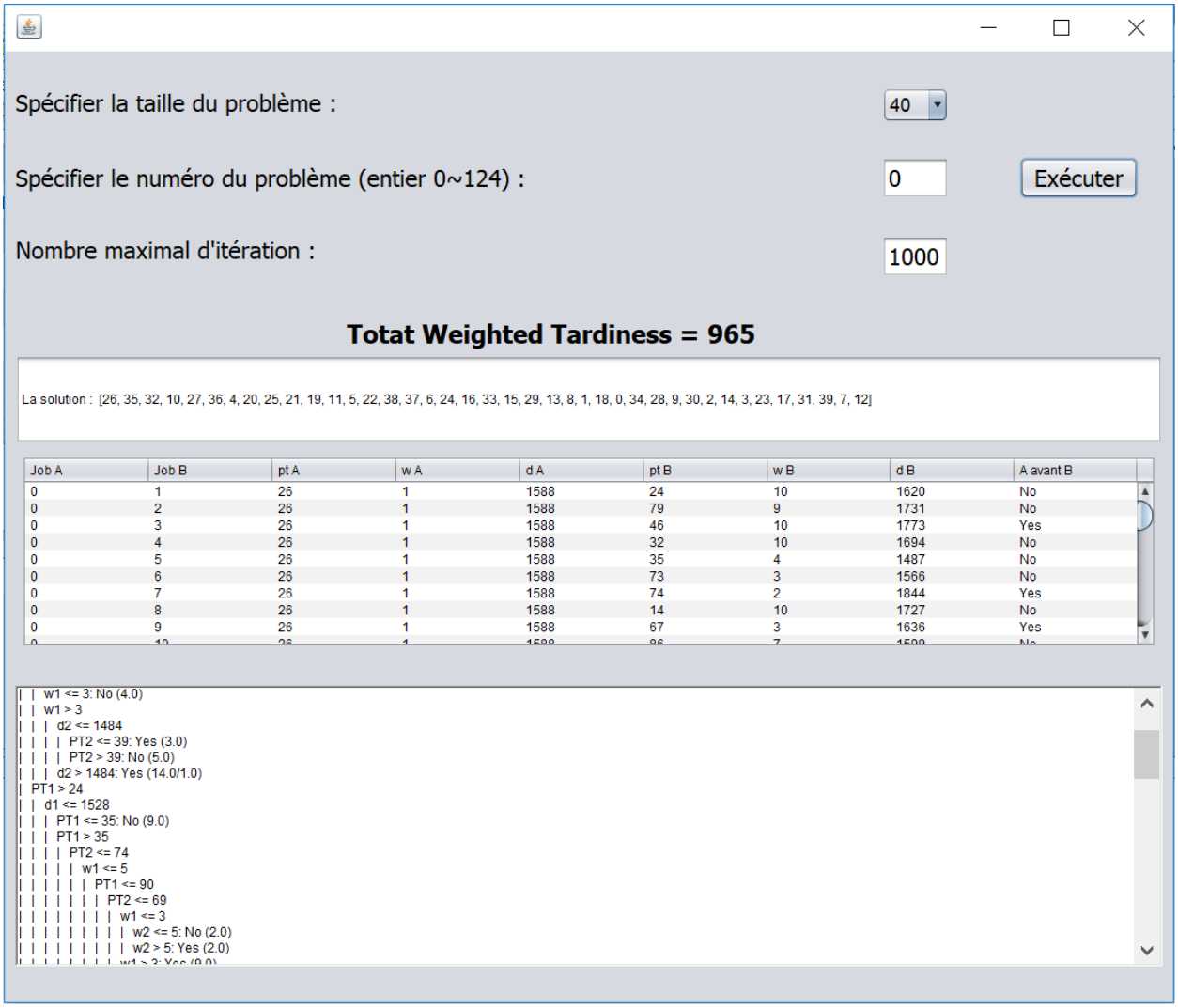

FIGURE 20 : AFFICHAGE

À travers cette fenêtre l'utilisateur peut consulter différents résultats comme suit :

- Le premier affichage représente la somme minimale des retards pondérés (TWT) (voir Figure 21).

FIGURE 21 : LA SOMME DES RETARDS PONDERES (TWT).

- Le deuxième affichage représente la solution trouvée pour le problème choisi par l'utilisateur (voir la Figure 22).

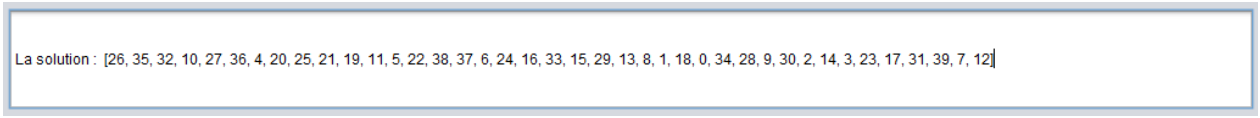

FIGURE 22 : LA SOLUTION OPTIMALE.

- Le troisième affichage représente la comparaison Job vs Job (voir la Figure 23).

Job A	Job B	pt A	w A	d A	pt B	w B	d B	A avant B
0	1	26	1	1588	24	10	1620	No
0	2	26	1	1588	79	9	1731	No
0	3	26	1	1588	46	10	1773	Yes
0	4	26	1	1588	32	10	1694	No
0	5	26	1	1588	35	4	1487	No
0	6	26	1	1588	73	3	1566	No
0	7	26	1	1588	74	2	1844	Yes
0	8	26	1	1588	14	10	1727	No
0	9	26	1	1588	67	3	1636	Yes
0	10	26	1	1588	86	7	1599	No

FIGURE 23 : LA COMPARAISON JOB VS JOB.

- Et le dernier affichage représente l'arbre de décision obtenu à base du tableau de comparaison Job vs Job (voir la Figure 24).

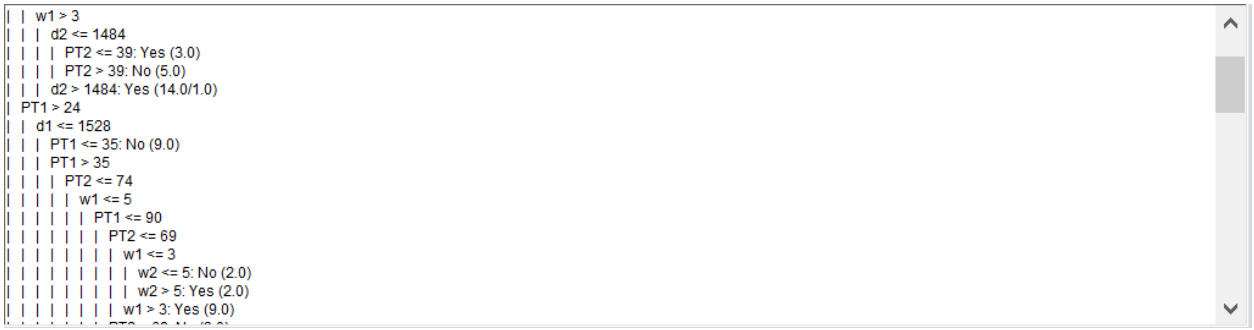

FIGURE 24 : ARBRE DE DECISION.

6. Conclusion & perspectives

Dans cette contribution, une approche innovante est développée montrant que la fouille de données peut être considérée comme une bonne alternative pour la résolution du problème à Machine Unique avec minimisation de la somme des retards pondérés. La fouille de données peut imiter jusqu'à un certain point les solutions retournées par n'importe quelle méthode de résolution y compris la recherche locale.

Dans cette approche, un algorithme recherche locale est proposé pour la résolution du SMTWT. LS (local search) est efficace pour résoudre les problèmes SMTWT. Il donne de bons résultats tout en trouvant des solutions optimales pour plusieurs problèmes avec un taux de 56,26% pour les 375 problèmes considérés avec un écart moyen de 0,44% en comparaison avec les meilleurs résultats connus. Ensuite, un arbre de décision est appliqué avec succès pour extraire de la connaissance.

Aussi nous avons parlé de notre application et leur affichage avec un exemple.

Conclusion Générale

Le but de ce mémoire de fin d'études est de développer une approche qui, à l'aide de la fouille de données et plus particulièrement les arbres de décision, analyse et reproduit le comportement des méthodes de résolutions métaheuristiques.

Nous nous sommes ainsi intéressés aux problèmes d'ordonnancement dans les ateliers de production à machine unique qui font l'objet de nombreuses contributions de la part et des chercheurs et des praticiens afin de d'améliorer le processus de planification et d'ordonnancement de la production. Ceci permet entre autres, d'améliorer la compétitivité de l'entreprise et de lui permet d'accaparer de nouvelles parts de marché et de garantir sa survie.

L'approche proposée dans ce mémoire consiste à utiliser la recherche locale pour résoudre des problèmes d'ordonnancement de type single machine avec somme des retards pondérés. Les solutions obtenues sont par la suite prétraitées et un tableau job vs job est créé. Ensuite, à l'aide de la fouille de donnée, et plus précisément les arbres de décisions, de nouvelles règles de priorité sous la forme « *si-alors* » sont extraites. Ces nouvelles règles peuvent par la suite être utilisées pour résoudre de nouveaux problèmes d'ordonnancement SMTWT, sans recourir aux métaheuristiques, notamment la recherche locale, permettant ainsi un gain en termes de temps de calcul et de réactivité du système de production.

Références

[1] *Fu-Chung - A New Approach for Solving Single Machine Total Weighted Tardiness (SMTWT) Problem - 2016*

[2] *Tangour Toumi – Ordonnancement Dynamique dans les Industries Agroalimentaires - 2007*

[3] *MHallah-Alhajraf - Ant colony systems for the single-machine total weighted earliness tardiness scheduling problem - 2015*

[4] *Ourari - De l'ordonnancement déterministe à l'ordonnancement distribué sous incertitudes - 2011*

[5] *HABIB ZAHMANI Mohamed - Contribution à l'ordonnancement dynamique : proposition d'une approche guidée par effet de Simulation/Datamining - 2018*

[6] *Chaari - Un algorithme génétique pour l'ordonnancement robuste application au problème du flow shop hybride - 2011*

[7] *Boukef Ben Othman - Sur l'ordonnancement d'ateliers job-shop flexibles et flow-shop en industries pharmaceutiques Optimisation par algorithmes génétiques et essaims particulaires - 2009*

[8] *Chou. - An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem - 2009*

[9] *Chung-et-al. - Multiple-variable neighbourhood search for the single-machine total weighted tardiness problem - 2017*

[10] *Ding-et-al. - A hybrid evolutionary approach for the single-machine total weighted tardiness problem - 2017*

[11] *Subramanian-Battarra-Potts - An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times - 2014*

[12] *Brahimi - Extraction de connaissances à partir de données incomplètes et imprécises - 2011*

[13] *Benamar - Etude explorative d'outils pour le data mining – 2007*

[14] *A. Shahzad and N. Mebarki, "Learning Dispatching Rules for Scheduling: A Synergistic View Comprising Decision Trees, Tabu Search and Simulation," Computers, vol. 5, no. 1, p. 3, 2016.*

[15] *A. Ferrolho and M. Crisóstomo, "Single machine total weighted tardiness problem with genetic algorithms," 2007 IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA 2007, pp. 1–8, 2007.*

[16] *R. Maheswaran, S. G. Ponnambalam, and N. Jawahar, "Hybrid heuristic algorithms for single machine total weighted tardiness scheduling problems," Int. J. Intell. Syst. Technol. Appl., vol. 4, no. 1–2, pp. 34–56, 2008.*

[17] *F. D. Chou, "An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem," Expert Syst. Appl., vol. 36, no. 2 PART 2, pp. 3857–3865, 2009.*

[18] *S. Olafsson and X. Li, "Learning effective new single machine dispatching rules from optimal scheduling data," Int. J. Prod. Econ., vol. 128, no. 1, pp. 118–126, 2010.*

[19] *C. Koulamas, "The single-machine total tardiness scheduling problem: Review and extensions," Eur. J. Oper. Res., vol. 202, no. 1, pp. 1–7, 2010.*

[20] *R. Balasundaram, N. Baskar, and R. S. Sankar, "A New Approach to Generate Dispatching Rules for Two Machine Flow Shop Scheduling Using Data Mining," Procedia Eng., vol. 38, pp. 238–245, 2012.*

[21] *D. Koonce and S. Tsai, "Using data mining to find patterns in genetic algorithm solutions to a job shop schedule," Comput. Ind. Eng., vol. 38, no. 3, pp. 361–374, Oct. 2000.*

[22] *A. Shahzad and N. Mebarki, "Discovering dispatching rules for job shop scheduling problem through data mining," in 8th International Conference of Modeling and Simulation - MOSIM'10, 2010.*

[23] *J. E. Beasley, "OR-Library: Distributing Test Problems by Electronic Mail," J. Oper. Res. Soc., vol. 41, no. 11, p. 1069, Nov. 1990.*

[24] *N. Aissani, B. Atmani, D. Trentesaux, and B. Beldjilali, "Extraction of Priority Rules for Boolean Induction in Distributed Manufacturing Control," Serv. Orientat.*

Holonic Multi-Agent Manuf. Robot. Stud. Comput. Intell., vol. 544, pp. 127–143, 2014.

[25] *M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," ACM SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10–18, 2009.*

[26] *X. Li and S. Olafsson, "Discovering Dispatching Rules Using Data Mining," J. Sched., vol. 8, no. 6, pp. 515–527, Dec. 2005.*

[27] *M. Habib Zahmani and B. Atmani, "Extraction of Dispatching Rules for Single Machine Total Weighted Tardiness using a Modified Genetic Algorithm and Data Mining," Int. J. Manuf. Res., vol. 13, no. 1, pp. 1–25, 2018.*

[28] *H. M. Soroush, "Stochastic bicriteria single machine scheduling with sequence–dependent job attributes and job–dependent learning effects," Eur. J. Ind. Eng., vol. 8, no. 4, pp. 421–456, 2014.*