Université Saâd DAHLAB, Blida

N° D'ordre.....

Faculté des sciences
Département d'informatique

Présenté par :

ALOUACHE Rabah BENLAKEHAL Oussama

En vue d'obtenir le diplôme de master
Domaine : Mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet :

Conception et réalisation d'un logiciel d'ordonnancement industriel

Soutenu le :

M. Cherif Zahar	Président
M. Ferfera	Examinateur
Melle. Ameur	Examinateur
M. SIDOUMOU Redha Mohamed	Promoteur
M. GAHAM Mehdi	Encadreur

Promotion 2012/2013

Remerciement

Tout d'abord, nous tenons rendre grâce à dieu tout puissant pour nous avoir donné le courage et la détermination nécessaire pour finaliser ce travail et le mener à terme.

On ne saurait ne pas remercier encore une fois nos parents respectifs qui, par leur amour et leur affection nous ont permis d'arriver là où nous somme aujourd'hui.

Nous remercions notre promoteur MR SIDOUMOU pour son aide précieuse, ces conseils avisés et ces idées riche.

Nous tenons à remercier MR GAHAM qui a endossé son rôle d'encadreur de la meilleure façon qui soit. Nous retiendrons sa patience, sa disponibilité et sa compréhensibilité.

Nous tenons un grand et un spécial remerciement Les gens de CDTA pour leur sympathie, leur aide et ses encouragements.

Nous remercions le membre de jury pour nous avoir fait l'honneur de juger notre travail.

Nous sommes reconnaissantes à Tous nos enseignants qui nous ont facilité la compréhension et la maitrise. Spécialement "Mme ABED"."Mlle AMEUR Khadidja",

Nous tenons à remercier également et énormément nos amis ainsi toute personne qui nous a aidés de près ou de loin.

Merci.

Mr Rabah et Mr Oussama.

Dédicaces

C'est avec un immense plaisir que je dédie ce travail

A mes très chers parents qui sont toutes ma vie et tout ce que j'ai de plus cher au monde, en

témoignage de ma reconnaissance infinie pour ses nombreux sacrifices.

Qu'ils trouvent en ce travail la preuve de mon éternel amour et ma

reconnaissance envers eux.

Que dieu les gardes et leur procure la santé et le bonheur.

Aussi à mes amis Sans oublier mon ami Rabah qui a été mon acolyte dans cette épreuve et toute sa famille.

Mr BENLAKEHAL Oussama.

Dédicaces

C'est avec un immense plaisir que je dédie ce travail

A mes très chers parents qui sont toutes ma vie et tout ce que j'ai de plus cher au monde, en témoignage de ma reconnaissance infinie pour ses nombreux sacrifices.

Qu'ils trouvent en ce travail la preuve de mon éternel amour et ma reconnaissance envers eux.

Que dieu les gardes et leur procure la santé et le bonheur.

Ainsi qu'à mes soeurs Zohra ,Amina, Khadidja, Fatiha et Yasssmine.

a en témoignage de ma grande affection pour eux.

A mes chers grands parents, mes tantes, mes oncles et toute ma famille.

Aussi à mes amis Sans oublier mon ami Oussama qui a été mon acolyte dans cette épreuve et toute sa famille.

Mr ALOUACHE Rabah.

Sommaire

Liste des figures

Liste des tableaux

Résumé

La théorie de l'ordonnancement est une branche de la recherche opérationnelle. Elle occupe un rôle important dans nombreux secteurs de l'économie, notamment en gestion de production des entreprises.

Les problèmes d'ordonnancement dans les ateliers constituent sûrement pour les entreprises une des difficultés majeures de leur système de gestion et de conduite. En effet, c'est à ce niveau que doivent être prises en compte les caractéristiques réelles multiples et complexes des ateliers.

Le but de notre travail est la réalisation d'un logiciel d'ordonnancement industriel basé sur le modèle job shop flexible qui est une extension de modèle Job Shop Classique et un noyau de calcul méta-heuristique basé sur la recherche Tabou.

L'approche Tabou développée. Elle est basée sur une structure de voisinage modifié qui permet de travaillé directement sur les représentations des solutions, non pas sur l'ordonnancement lui-même, ce qui réduit les temps de calcul du voisinage.

Lorsqu'on compare nos résultats avec les résultats trouvés par d'autres chercheurs, nous sommes arrivés à l'affirmation que notre travail abouti à un résultat satisfaisant.

Mots clés : Ordonnancement, Job Shop Flexible, Méta-heuristiques, Recherche Tabou.

Abstract

The scheduling theory is a branch of operations research. It plays an important role in many sectors of the economy, including production management companies.

The scheduling problems in the workshops are surely one of the major business challenges of their management system and conduct. Indeed, it is at this level that must be taken into account the multiple and complex workshops actual characteristics.

The aim of our work is the realization of industrial scheduling software based on the flexible job shop model which is an extension of Model Job Shop Classic and a core meta-heuristic calculation based on Tabu Search.

The approach developed Tabou. It is based on a modified neighborhood structure that allows working directly on the representations of solutions, not on the schedule itself, which reduces the computation time of the neighborhood

When comparing our results with the results found by other researchers, we arrived at the assertion that our work led to a satisfactory result.

Keywords : Scheduling, Flexible Job Shop, Méta-heuristics, Tabu Search.

ملخص

نظرية الجدولة هي فرع من فروع البحوث ألعملية فهي تلعب دورا هاما في العديد من القطاعات الاقتصادية خاصة بما في ذلك تسيير انتاج الشركات .

تعتبر مشاكل الجدولة في ورشات العمل واحدة من التحديات الرئيسية للشركات في نظام تسييرها و إدارتها في هذا المستوى تجب الاخذ بعين الاعتبار الخصائص المتعددة و المعقدة لورشات العمل.

الهدف من عملنا هذا هو انجاز برنامج جدولة لورشات العمل الصناعية معتمدا على نموذج "محل الوظيفي المرن" الذي هو امتداد لنموذج " محل الوظيفي" و ايضا خوارزمية قائمة على أساس" بحث المحرمات".

الطريقة المحرمة المستبدلة قائمة على اساس الجوار, تسمح بالعمل على تمثيل الحلول و ليس على الجدولة مما يقلل من الوقت اللازم للحساب في المنطقة المجاورة.

عند مقارنة نتائجنا مع النتائج التي عثر عليها من قبل باحثين آخرين، وصلنا للتأكد ان عملنا ادى الى نتيجة مرضية من حيث الفعالية و الكفاءة.

الكلمات المفتاحية : الجدولة، محل الوظيفي المرن، بحث المحرمات.

INTRODUCTION GÉNÉRALE

Introduction Générale

La situation actuelle des entreprises manufacturières a connu de grands bouleversements. L'objectif majeur de toute entreprise actuelle n'est pas seulement la productivité mais aussi et surtout la compétitivité. Cette compétitivité passe forcement par la diversification des produits, un outil de production de plus en plus flexible et une gestion de production fiable. L'amélioration de la gestion de production est alors, à la base de toute tentative de changement. Cette fonction vise en effet à organiser le fonctionnement du système de production, et a mieux gérer ses différentes opérations.

Ainsi, les problèmes d'ordonnancement d'ateliers constituent sûrement pour les entreprises une des difficultés importantes des leurs systèmes de gestion et de pilotage de la production. En effet, c'est à ce niveau que doivent être prises en compte les caractéristiques réelles multiples et complexes des ateliers. Dans ces problèmes, les ressources sont généralement des machines, et chaque travail à ordonnancer concerne un produit ou un lot de produits à fabriquer en respectant les gammes de fabrication.

Le problème d'ordonnancement se défini par un ensemble de ressources, qu'on doit affecter à un ensemble de tâches. Il s'agit d'associer, à chaque tâche une date d'exécution et un sous-ensemble de ressources de façon à satisfaire un ou plusieurs objectifs, et en prenant en compte un certain nombre de contraintes temporelles (délais, contraintes d'enchainement) et de contraintes portant sur la disponibilité des ressources requises.

La plus part des problèmes réels d'ordonnancement sont NP-difficiles et la taille de leurs instances est souvent très grande. Par conséquent, les méthodes approchées constituent une alternative intéressante pour leur résolution. Dans le cadre de ce mémoire, nous nous focalisions sur l'ordonnancement d'atelier de type Job Shop Flexible qui est une généralisation de Job Shop classique. Et pour sa résolution, nous nous intéressons à la méthode Tabou de recherche locale qui est une heuristique générale applicable à de nombreux problèmes. Cette méthode connue pour être un des meilleurs algorithmes pour la résolution de ce type de problème (Job Shop Flexible) et a prouvé son efficacité dans le domaine de la résolution des problèmes combinatoire complexe.

L'objectif de notre travail est la conception et la réalisation d'un logiciel d'ordonnancement industriel basé sur le model job shop flexible et le développement d'un noyau algorithmique de résolution efficace pour ce problème. Pour ce faire nous avons développé une recherche Tabou modifiée basée sur une nouvelle structure de voisinage qui permet la détermination de l'ensemble des solutions voisines à partir de représentations codées des solutions non pas à partir de structures de données complexes. Cette approche permet particulièrement de réduire les temps de calcul ce qui est important pour les logiciel applicatifs d'ordonnancement industriel.

Le mémoire présenté est structuré en quatre chapitres qui permettent un cadrage progressif du sujet.

Le premier chapitre introduit les notions de bases relatives à la gestion des systèmes industriels et à la théorie de l'ordonnancement.

Dans le second chapitre, nous présenterons en détails le problème d'ordonnancement de type Job Shop Flexible, ainsi, nous introduirons différentes méthodes de résolution exactes et approchées pour ce problème.

Dans le troisième chapitre, nous décrivons, d'une manière détaillée, l'approche Tabou implémentée, et les aspects conceptuels du logiciel d'ordonnancement (ORdoRO).

Dans le quatrième chapitre, nous présentons notre logiciel (ORdoRO) et plusieurs expérimentations sont effectuées sur des Benchmarks retenus comme échantillons de tests pour prouver l'efficacité de l'approche Tabou proposée et implémentée.

CHAPITRE 1

PROBLÈME D'ORDONNANCEMENT

Ne me dites pas que ce problème est difficile.
S'il n'était pas difficile, ce ne serait pas un problème.

Ferdinand Foch

I.1. Introduction

La résolution des problèmes d'ordonnancement est une branche de la recherche opérationnelle qui consiste à trouver une séquence optimale pour l'exécution de différentes taches sur différentes ressources afin de minimiser une fonction objective. Il s'agit également de calculer les dates de début et de fin d'exécution des taches.

Dans ce chapitre, quelques notions de base relatives aux problèmes d'ordonnancement sont d'abord introduites. Ensuite nous présenterons les modèles classiques d'ateliers les plus étudiés dans la littérature. Enfin nous terminerons avec un aperçu rapide sur la complexité de ces problèmes.

I.2. Les concepts de base de la production

I.2.1. La production

La production est une activité opérationnelle utilisant les ressources disponibles (humains et matériels) en vue de créer des biens matériels ou d'assurer des services d'utilité suivant une ou plusieurs transformations. Ces transformations sont distinguées par un ensemble d'opérations, qui peuvent être, soit une modification de la forme des pièces, soit la combinaison de plusieurs pièces. L'ensemble de ces opérations constituant la production seront soumises à des critères impératifs de coût, délai, quantité et qualité. [1]

I.2.2. La gestion de production

Le système de gestion de production a pour objet la recherche d'une organisation efficace dans l'espace et dans le temps, en minimisant les coûts, de toutes les opérations relatives à la production, afin d'atteindre les objectifs de l'entreprise.

La gestion de production consiste à produire en temps voulu, les quantités demandées par les clients dans des conditions de coût de revient et de qualité déterminées en optimisant les ressources de l'entreprise de façon à assurer sa continuité, son développement et sa compétitivité. [1]

I.3. Relation de la gestion de production avec les fonctions de l'entreprise[2]

La gestion de production présente des relations avec plusieurs fonctions différentes au sein de l'entreprise. On peut citer : la planification, les approvisionnements, les achats, et la gestion des ressources humaines et techniques. L'une des méthodes de gestion de production permettant de prendre en compte la complexité des produits (nomenclatures) et des organisations des industries manufacturières est la MRP (Manufacturing Resource Planning Planification des Ressources de Production).

Cette méthode présente l'avantage de respecter notamment la forte hiérarchisation des prises de décision dans les entreprises. Les différentes étapes qui composent la MRP sont présentées sur la figure I.1.

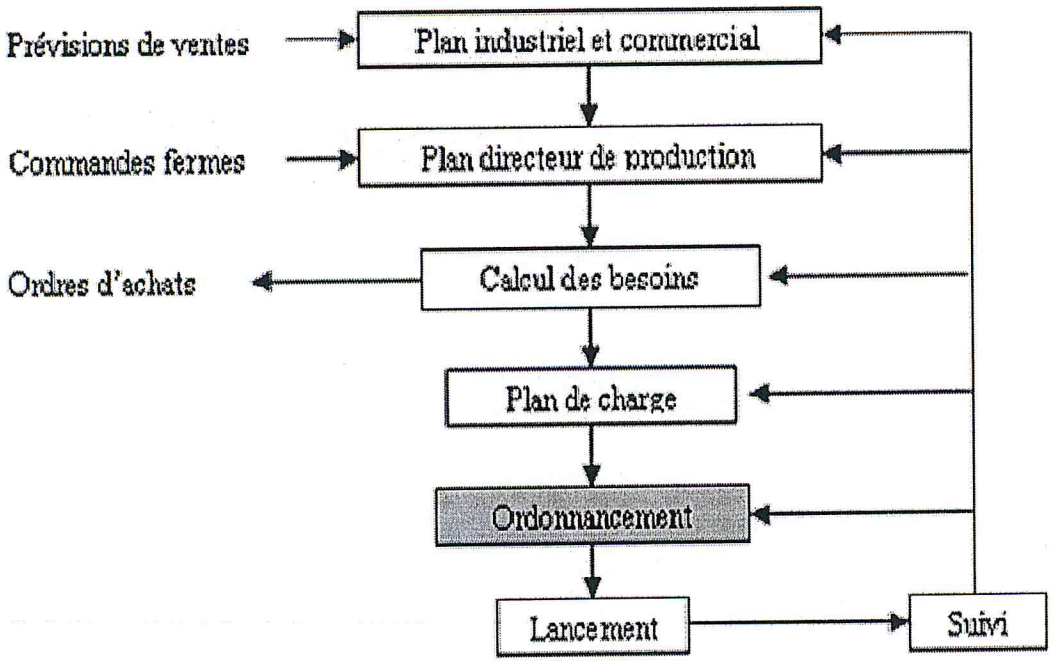

Figure I.1 : Les fonctions de la MRP [2]

Au sommet de la MRP, le plan industriel et commercial (PIC) intègre mois par mois et par famille de produits les prévisions de vente des produits et les objectifs de production, sur un horizon assez long, de l'ordre d'une année. Les prévisions commerciales du PIC sont ensuite détaillées sur chaque produit et réévaluées sur un horizon plus court (de l'ordre de quelques mois) dans le plan directeur de production (PDP). Celui-ci recense toutes les commandes fermes et prévisionnelles et en déduit un échéancier des quantités à fabriquer par produit et

par période. On détermine à ce niveau, si les capacités globales de l'entreprise en hommes et en machines critiques vont être suffisantes ou s'il est nécessaire d'investir dans de nouveaux matériels ou d'embaucher davantage de personnel.

A partir de cet échéancier, le calcul des besoins détermine les quantités de chacun des composants des produits à fabriquer et les dates de fabrication selon la demande et le marché, en se basant sur la nomenclature des produits et les délais de fabrication. A l'issue du calcul des besoins, on dispose des quantités de composants élémentaires à fabriquer (ordres de fabrication) et de matières premières et de composants à acheter (ordres d'achat), accompagnées de leurs dates de besoin. Le jalonnement de ces ordres de fabrication permet de déterminer la charge par machine et par période. On peut donc vérifier si les commandes sont réalisables ou pas, et prendre d'éventuelles mesures correctives telles que des heures supplémentaires pour les opérateurs, la sous-traitance d'une partie du travail ou le lissage de la charge. Lorsque l'adéquation charge/capacité est faite, les opérations sont ordonnancées.

I.4. Généralités sur l'ordonnancement

I.4.1. Définition du problème d'ordonnancement

Plusieurs définitions d'un problème d'ordonnancement sont données dans la littérature, on cite parmi eux ce qui suit :

«Résoudre un problème d'ordonnancement consiste à ordonnancer i.e. programmer ou planifier , dans le temps l'exécution des taches en leur attribuant les ressources matérielles ou humaines nécessaires, de manière à satisfaire un ou plusieurs critères préalablement définis, tout en respectant les contraintes de réalisation».[3], [4]

« Etant donné un ensemble de taches à accomplir, le problème d'ordonnancement consiste à déterminer quelles opérations doivent être exécutées, et à assigner des dates et des ressources à ces opérations de façon à ce que les tachés soient, dans la mesure du possible, accomplies en temps utile, au moindre cout et dans les meilleures conditions». [5]

D'une manière plus simple, un problème d'ordonnancement consiste à affecter des taches à des ressources à des instants données pour répondre au mieux aux besoins exprimer par un client , au meilleur cout et dans les meilleur délais, tout en tenant compte des contraintes.

I.4.2.Éléments d'un problème d'ordonnancement

Les différentes caractéristiques d'un problème d'ordonnancement sont les taches, les ressources, les contraintes et les critères.

I.4.2.1. Les tâches [6]

Une tache (job) qu'on note **i** est une entité élémentaire caractérisée par une date de disponibilité (release date) (date a laquelle elle peut être exécutées au plus tôt r_i), une date d'échéance (due date) (appelée date de fin au plus tard d_i) et une durée opératoire (procession time) (durée d'exécution p_i). Sa localisation dans le temps est définie par une date de débuts $s_i \geq r_i$ et une date de fin $c_i \leq d_i$. En outre, la tache **i** utilise une (ou plusieurs) ressource notée **k** pendant l'exécution. Les ressources concernées par la réalisation de la tache sont généralement supposées connues a priori.

Lorsque les taches ne sont pas liées entre elles par de cohérence technologique (par exemple contraintes d'enchainement liées aux procédés de réalisation), elles sont dites indépendantes.

On distingue deux types de taches :

- **Les taches morcelables (préemptives)** Qui peuvent être exécutées en plusieurs fois, facilitant ainsi la résolution de certains problèmes.
- **Les taches non morcelables (non préemptives)** Qui doivent être exécutées en une seule fois et ne sont interrompues qu'une fois terminées.

I.4.2.2. Les ressources [6]

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

- **Ressources renouvelable** : Il s'agit d'une ressource qui peuvent redevenir disponible en même quantité après avoir été allouées à une tache (machines, personnel,….)
- **Ressources consommable** : Il s'agit d'une ressource dont la disponibilité décroit après avoir été allouées à une tache (argent, matières premières,…).

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut varier au cours de temps, par ailleurs, dans le cas des ressources renouvelables, on distingue principalement deux types de ressources.

- **Ressources disjonctive** : Il s'agit d'une ressource qui ne peut exécuter qu'une tache à la fois.
- **Ressources cumulative** : Il s'agit d'une ressource qui peut être utilisée par plusieurs taches simultanément mais en nombre limité.

I.4.2.3. Les contraintes [6]

Une contrainte exprime des restrictions sur les valeurs que peuvent prendre conjointement les variables représentant les relations reliant les taches et les ressources. Deux types de contraintes peuvent êtres distingués : contraintes des ressources et contraintes temporelles.

- **Les contraintes de ressources :** Traduisent le fait que les ressources sont disponibles en quantité limitée. On distingue deux types de contraintes de ressources. Liées à la nature disjonctive ou cumulative des ressources.
- **Les contraintes temporelles :** Les contraintes temporelles intègrent en général : les contraintes de temps alloué, issues généralement d'impératifs de gestion et relatives aux dates limites des taches (délai de livraison par exemple) ou à la durée total d'un projet.
  - Les contraintes d'antériorité et plus généralement les contraintes de cohérence technologique, qui décrivent le positionnement relatif de certaines taches par rapport à d'autres (par exemple contraintes de gammes dans le cas des problèmes d'atelier).
  - Les contraintes de calendrier liées au respect d'horaires de travail,...etc.

I.4.2.4. Les critères d'optimisation [6]

Pour évaluera la qualité d'un ordonnancement, des mesures connues sous l'appellation de critères, dites aussi objectifs, sont utilisées. On cherche donc à minimiser ou maximiser tels critères. On note par exemple ceux :

- **Lies au temps :**
  - Le temps total d'exécution ou le temps moyen d'achèvement d'un ensemble de taches.
  - Différents retard (maximum, moyen, somme, nombre, etc.,...) ou avances par rapport aux dates limites fixées.
- **Liés aux ressources :**
  - La quantité totale ou pondérée de ressources nécessaires pour réaliser un ensemble de tâches.
  - La charge de chaque ressource.
- **Liés à une énergie ou un débit:**
  - Liés aux couts de lancement, de production, de transport, etc., mais aussi aux revenus, aux retours d'investissements.

I.5. les problèmes d'ordonnancement d'atelier [11]

Les problèmes d'ordonnancement d'atelier sont des problèmes avec contraintes de ressources. Les ressources sont des machines ne pouvant réaliser qu'une opération à la fois (Ressources disjonctives) et les machine sont renouvelables. D'autre part chaque tâche est composer de plusieurs entités appelée opérations. Une tâche ne pouvant se trouver simultanément en deux lieux distincts, donc elle ne peut êtres exécutée qu'à raison d'une opération a la fois sur une seule des machines. Enfin le modèle d'une gamme linéaire de fabrication indique pour chaque tâche un ordre strict des opérations qui le composent.

La littérature à l'habitude de classifier les problèmes d'ordonnancement selon l'organisation de l'atelier.

Les modèles les plus connus sont :

- ✓ Les organisations à une machines.
- ✓ Les organisations à machines parallèles.
- ✓ Les organisations à machines multiples (comme le Flow Shop, le Job Shop et l'Open Shop).

I.5.1. L'organisation a une machine

Dans ce cas, l'ensemble des tâches à réaliser est fait par une seule machine. Les tâches alors sont composées d'une seule opération qui nécessite la même machine. Ce type d'organisation est un cas que l'on peut recentrer rarement dans les entreprises industrielles. [11]

Par contre il constitue un domaine d'étude considérable dans certains systèmes : par exemple dans les systèmes informatiques partagés où l'on dénote souvent la présence d'une seule ressource : imprimante, un seul microprocesseur,... avec plusieurs usagers.

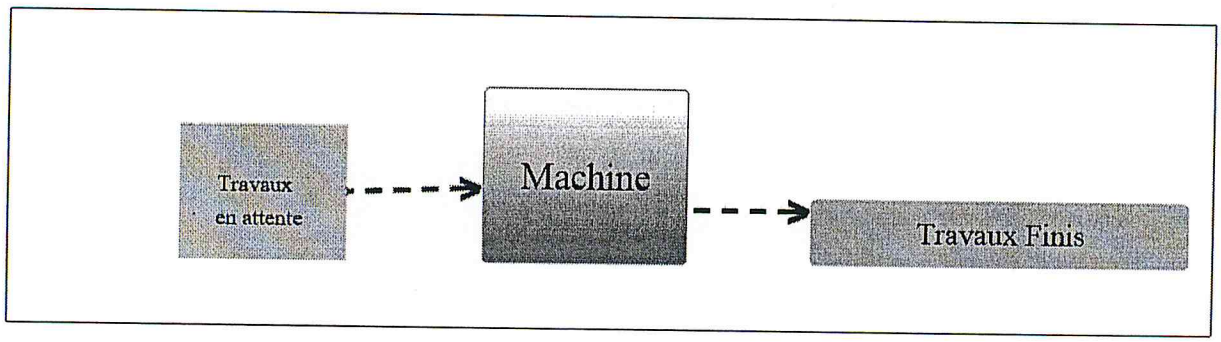

Figure I.2 : Exemple d'organisation d'atelier à machine. [7]

I.5.2. Les organisations à machine parallèles

Dans ces organisations, on dispose d'un ensemble de machines identiques pour réaliser les tâches. Les tâches se composent d'une seule opération et une tâche exige une seule machine. L'ordonnancement s'effectue en deux phases : la première phase consiste à affecter les tâches aux machines et la deuxième phase consiste à établir la séquence de réalisation sur chaque machine. [11]

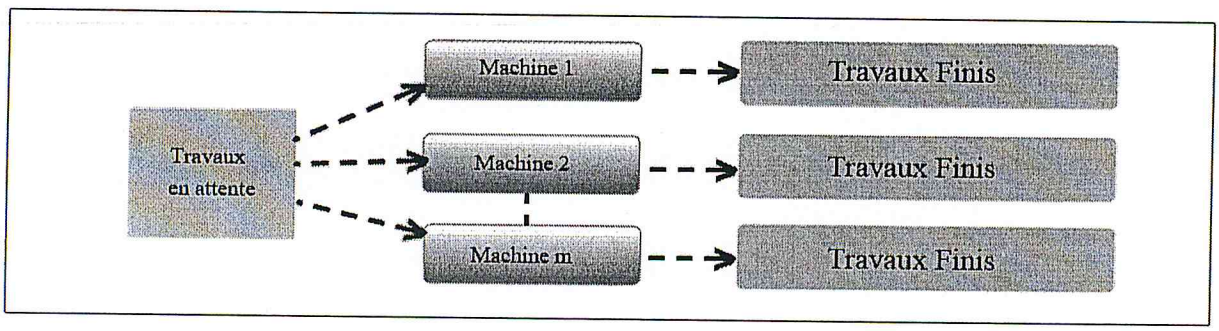

Figure I.3: Exemple d'organisation d'atelier à machines parallèles. [7]

I.5.3. Les organisation à machine multiples

Ces organisations ce caractérisent par l'existence de plusieurs machines, sur lesquelles seront exécutées plusieurs tâches. Ce sont des organisations énormément répondues dans le domaine de la production industrielle, constituant ainsi catégorie différentes d'ateliers .trois types ont une importance particulière dans les problèmes d'ordonnancement : le flow shop, le job shop et l'open shop. [11]

I.5.3.1 Ateliers à cheminement unique : Flow Shop

Ce sont des ateliers ou une ligne de fabrications est constituée de plusieurs machines en série ; toutes les opérations de toutes les taches passent par les machines dans le même ordre. [11]

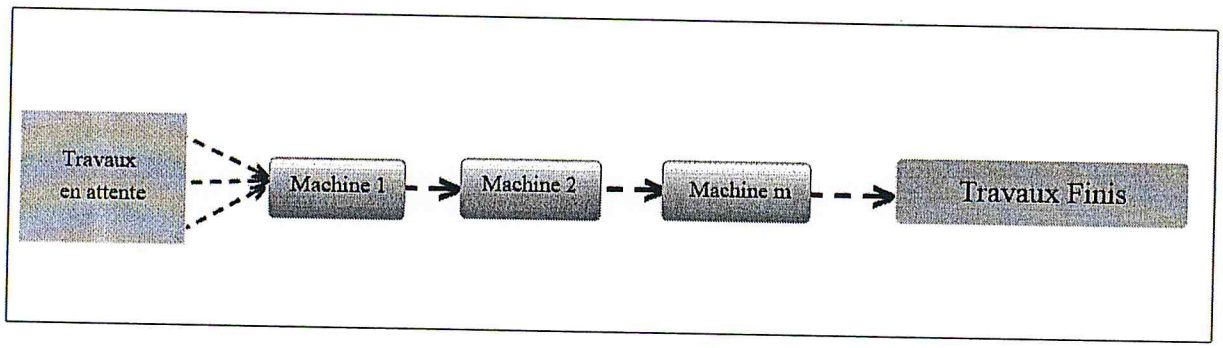

Figure I.4 : Exemple d'organisation d'atelier à cheminement unique (Flow Shop). [7]

Le type flow shop hydride

Les problèmes de type « flow shop hybride » constituent une extension du problème du type « flow shop », ou la première opération de chaque tache est traitée par un premier ensemble de machines parallèles, la deuxième opération par le deuxième ensemble, et ainsi de suite.

I.5.3.2 Atelier à cheminement multiple : Job Shop

Dans les ateliers de type « job shop », les opérations sont réalisées selon un ordre total bien déterminé, variant selon la tache à exécuter. Ce type d'atelier est nommé aussi atelier à cheminements multiples. [11]

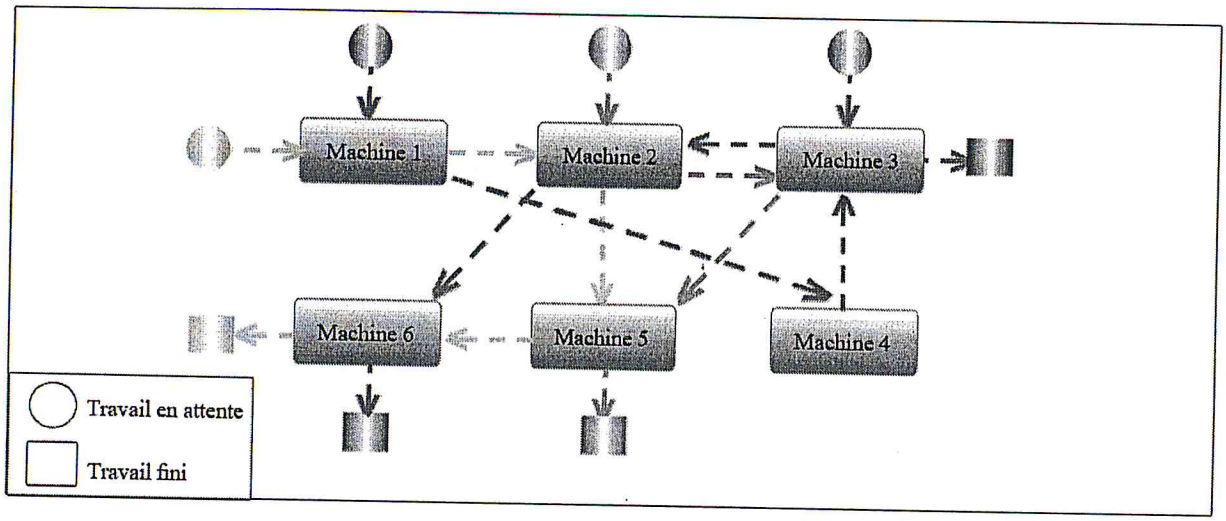

Figure I.5 : Exemple d'organisation d'atelier à cheminements multiples (Job Shop). [7]

Le job shop flexible

Le job shop flexible est une extension du modèle job shop classique. Sa particularité essentielle réside dans le fait que plusieurs machines sont potentiellement capables de réaliser un sous-ensemble d'opération. Plus précisément, une opération est associée à un ensemble contenant toutes les machines pouvant effectuer cette opération.

I.5.3.3. Atelier a cheminement libre : open shop

Ce type d'atelier est moins contraint que celui de type flow shop ou de type job shop. Ainsi, l'ordre des opérations n'est pas fixé a priori, les problèmes d'ordonnancement consiste, d'une part, à déterminer le cheminement de chaque tache et d'autre part à ordonnancer les taches en tenant compte des gammes trouvées, ces deux problèmes pouvant êtres résolus simultanément. Comparé aux autres modèles d'ateliers, open shop n'est pas couramment utilisé dans les entreprises. [11]

I.6. Représentation de la solution par diagramme de Gantt

Le diagramme de Gantt, du nom de son développeur Henry Gantt (1861-1919) est le moyen le plus simple et le plus utilisé pour représenté un ordonnancement, à la fois dans la littérature et dans les entreprises. Dans le cas de problèmes d'atelier, ce diagramme se compose de plusieurs lignes horizontales, chacune d'entre elles désignât une machines. Les opérations

exécutées sur une machine donnée sont représentées sous forme de barres ayant des longueurs proportionnelles à leurs temps opératoires. [6]

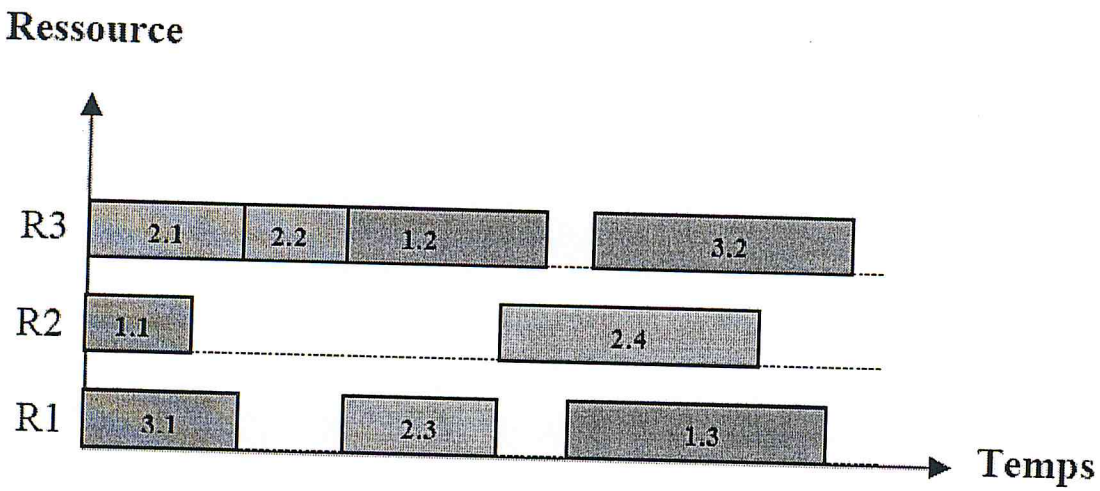

Figure I.6 : Exemple d'un diagramme de Gantt.

I.7. Complexités des problèmes d'ordonnancement

Les problèmes d'ordonnancement sont en général NP-difficile même si l'atelier est simple.

Plusieurs chercheurs montrent que si l'atelier possède un nombre de taches ou de machines supérieur ou égale a trois et pour les problèmes à deux machines avec recirculation le problème devient NP-difficile.

Mais pour certains cas particuliers il existe des algorithmes polynomiaux permettent de les résoudre, ce sont: Les algorithmes de Johnson, les algorithmes de Kubiak, les Algorithmes de Timkovsky à deux machines et n taches. [6]

I.8. Conclusion :

Dans ce chapitre, nous avons tout d'abord présenté les caractéristiques générales d'un problème d'ordonnancement, précisant notamment les différents éléments qui le caractérisent: les taches, les ressources, les contraintes et les critères d'optimisation, ainsi que la classification de ces problèmes et finalement, les modèles d'ateliers les plus importants.

Dans le chapitre suivant, nous présenterons le problème de type job shop flexible qui est l'objet de notre étude et les méthodes méta-heuristique généralement utilisées pour sa résolution.

CHAPITRE 2

JOB SHOP FLEXIBLE & LES MÉTHODES MÉTA-HEURISTIQUE DE RÉSOLUTION

Individuellement, les insectes sont bêtes

Collectivement, ils sont intelligents...

Par analogie, on devrait donc pouvoir enseigner à une colonie de robots simples et stupides la manière de "se mettre ensemble" pour agir avec intelligence

Jean-Louis Deneubourg

II.1. Introduction

Les problèmes d'ordonnancement d'ateliers de type Job Shop Flexible (FJSP –Flexible Job Shop scheduling Problem) sont connus dans la littérature comme étant les problèmes les plus difficiles à résoudre [9]. La difficulté réside dans le choix de la meilleure méta-heuristique pour leur résolution ainsi que pour la détermination des meilleurs ordonnancements en des temps raisonnable, les plus proches possibles de la solution optimale.

Les méthodes de résolution des problèmes d'optimisation combinatoire prennent en considération deux facteurs : la qualité des solutions et le temps de résolution. Ainsi peuvent êtres classées en deux catégories : les méthodes exactes qui garantissent l'optimalité mais avec un temps de calcul très grand, et les méthodes approchées qui peuvent perdre en optimalité pour gagner en temps d'exécution.

Dans ce chapitre, nous présenterons en détails le problème d'ordonnancement de type Job Shop Flexible qui fait l'objet de cette étude, ainsi nous allons, tout d'abord, définir brièvement les méthodes exactes. Ensuite nous introduirons quelques méthodes approchées comme les algorithmes génétiques. Et à la fin, nous nous intéressons à la présentation de la méthode de la recherche Tabou que nous avons utilisé.

II.2. Présentation des problèmes FJSP

II.2.1. Définition

Le problème que nous considérons est le Job Shop Flexible **(FJSP)** qui est une généralisation du problème de Job Shop Classique. Dans ce type de problème, une opération nécessite exactement une machine pour être réalisée et cette machine peut être choisie dans un ensemble défini a priori. Le problème d'ordonnancement consiste alors à affecter une machine à chaque opération et déterminer la séquence des opérations sur les machines obtenues, afin de minimiser un critère donné. [11]

II. 2.2. Formulation des problèmes FJSP

Les problèmes d'ateliers Job Shop Flexibles peuvent être formulés comme suit [8]:

- Soit un ensemble de n produits indépendants à réaliser sur m machine M_k, $k = 1, \ldots, m$.

- Chaque produit J_j est constitué d'une séquence de n_j opérations $O_{i,j}$, $i = 1, 2, \ldots, n_j$, à exécuter selon un ordre bien défini.
- L'exécution de chaque opération i d'un job J_j nécessite une ressource sélectionnée à partir d'un ensemble de machines disponibles.
- Chaque machine ne peut réaliser qu'une seule opération à la fois.
- L'assignation d'une opération $O_{i,j}$ à une machine M_k entraîne l'occupation de cette machine durant tout le temps d'exécution de l'opération, noté $p_{i,j,k}$.
- La préemption n'est pas autorisée.

Le FJSP présente deux difficultés principales :

- La première est relative à l'assignation de chaque opération $O_{i,j}$ à une machine M_k.
- La seconde correspond au calcul des temps de début $t_{i,j}$ et des temps de fin $tf_{i,j}$ de l'opération $O_{i,j}$.

II. 2.3. Les contraintes [6]

Les contraintes du problème représentent les contraintes technologiques auxquelles sont soumis les tâches et les machines. Elles concernent l'utilisation des machines et les liens existant entre les opérations.

- Les machines sont indépendantes les unes des autres (pas d'outils commun par exemple).
- Les tâches sont indépendantes les unes des autres, il n'existe aucun de priorité attaché aux tâches.
- Deux opérations de la même tâche ne peuvent être exécutées simultanément.
- Une machine ne peut exécuter qu'une seule opération à un instant donné.
- Seulement le temps d'exécution proprement dit, est pris en compte. Les temps de transport d'une machine à l'autre, de préparation,... ne sont pas considérés.
- Les machines sont disponible jusqu'à la fin de l'ordonnancement. En particulier les pannes des machines ne sont pas prises en compte.
- Les tâches sont autorisées d'attendre les machines autant qu'il faut. Il n'y à pas de date d'échéance.
- Une opération en cours d'exécution ne peut être interrompue (pas de préemption).

- Une opération peut avoir un ou plusieurs choix de machines avec une durée opératoire donnée.
- Les machines peuvent exister en plusieurs exemplaires non-reliées c.à.d. les durées opératoires dépendent de la ressource choisie.
- Une opération donnée sera exécuté par une seule machine.

II.2.4. Les objectifs

L'objectif du problème est de déterminer à la fois l'affectation des machines et l'ordre de passage de l'ensemble des opérations sur chaque machine, en respectant les contraintes. Le but ensuite est de minimiser la durée totale de l'ordonnancement (Makespan ou Cmax). [6]

II.3. Les méthode de résolutions

II.3.1. Les méthodes exactes

On peut définir une méthode exacte comme une méthode qui garantit l'obtention de la solution optimale pour un problème d'optimisation. L'utilisation de ces méthodes s'avère particulièrement intéressante, mais elles sont souvent limitées au cas des problèmes de petite taille. Parmi ces méthodes on distingue les méthodes de séparation et évaluation progressive « branch & bound », la programmation dynamique, la programmation linéaire,...etc. [10]

II.3.2. Les méthodes approchées

La taille des problèmes influe de façon importante sur les temps de calcul des méthodes exactes. Ce temps de calcul est raisonnable pour les problèmes de petite taille, mais il devient vite prohibitif si la taille des problèmes augmente, et par conséquent il est utile de développer des méthodes approchées. Ces méthodes qui sans garantir l'optimum absolu, peuvent fournir d'excellentes solutions. En plus d'être beaucoup plus génériques et facilement applicables, ces méthodes possèdent l'immense avantage d'être beaucoup moins contraignantes de sorte qu'elles sont pratiquement toujours applicables. Ainsi, dans de nombreux cas, les méthodes approchées deviennent la seule option performante envisageable. Ce sont en quelque sorte des méthodes de dernier recours. [6]

Il existe deux types de méthodes approchées : les méthodes constructives (les heuristiques) et les méthodes d'amélioration (les méta-heuristiques).

II.3.2.1. Les méthodes constructives ou heuristiques

Ce sont des méthodes très utilisées pour obtenir très rapidement une première solution pour les problèmes d'ordonnancement. La recherche d'une solution par ces méthodes commence à partir du zéro et à chaque itération une solution pour le problème est élaborée en utilisant, par exemple, des règles de priorité. Ces méthodes sont généralement très rapides. [6]

II.3.2.2. Les méthodes d'amélioration ou méta-heuristiques [11]

Une méta-heuristique est un algorithme d'optimisation visant à résoudre des problèmes d'optimisation difficile (souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle) pour lesquels on ne connaît pas de méthode classique plus efficace.

Les méta-heuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution (d'une manière proche des algorithmes d'approximation).

Il existe un grand nombre de méta-heuristiques différentes, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents.

II.3.2.2.1. Classification des méta-heuristiques

Les méta-heuristiques n'étant pas, a priori, spécifiques à la résolution de tel ou tel type de problème, leur classification reste assez arbitraire. On peut cependant distinguer deux grandes classes : les méthodes à base de population ou méthodes évolutives, et les méthodes de trajectoire ou méthodes de recherche locale. Cette classification est basée sur le nombre de solutions manipulées à chaque itération. [6]

II.3.2.2.1.1. Les méthodes à base de population

Ces méthodes, comme leur nom l'indique, travaillent sur une population de solution à la fois. Leur principe général consiste à combiner des solutions entre elles pour former de nouvelles, en essayant d'hériter les bonnes caractéristiques des solutions parents. Un tel processus est répété jusqu'à ce qu'un critère d'arrêt soit satisfait. Parmi les méta-heuristiques à base de population, on trouve une grande classe : Les Algorithmes Génétiques. [6]

Les algorithmes génétiques sont les plus populaires des algorithmes évolutionnaires. Un algorithme génétique peut être décrit comme un mécanisme qui limite l'évolution génétique des espèces : croisement, mutation, sélection..., ils appartiennent à la classe des algorithmes évolutionnaires.

Un Algorithme Génétique simple

1- Initialisation: générer une population initiale P de solutions de taille $|P| = \mathbf{n}$.
2- **Répéter.**
3- Sélectionner: choisir deux solutions X et X' par une technique de sélection.
4- Croisement: combiner les deux solutions parents X et X' pour former une solution enfant y.
5- Mutation de y sous conditions.
6- Choisir une solution individuelle y' pour être remplacé dans la population.
7- Remplacer y' par y dans la population.
8- **Jusqu'à** critère d'arrêt satisfait.

Avantages et inconvénients [6]

D'abord, les algorithmes génétiques sont coûteux en temps de calcul, puisqu'ils manipulent plusieurs solutions simultanément. C'est l'évaluation qui est le plus pénalisant, et on optimise généralement l'algorithme de façon à éviter d'évaluer trop souvent cette fonction.

Ensuite, l'ajustement d'un algorithme génétique est délicat. L'un des problèmes les plus caractéristiques est celui de la dérive génétique, qui fait qu'un bon individu se met, en l'espace de quelques générations, à envahir toute la population. On parle dans ce cas de convergence prématurée, qui revient à lancer à une recherche locale autour d'un minimum qui n'est pas forcément l'optimum attendu. Les méthodes de sélection proportionnelle

peuvent en particulier favoriser ce genre de dérive. Un autre problème surgit lorsque les différents individus se mettent à avoir des performances similaires : les bons éléments ne sont alors plus sélectionnés, et l'algorithme ne progresse plus.

Le grand avantage des algorithmes génétiques est qu'ils parviennent à trouver de bonnes solutions sur des problèmes très complexes, et trop éloignés des problèmes combinatoires classiques pour qu'on puisse tirer profit de certaines propriétés connues. Ils doivent simplement déterminer entre deux solutions quelle est la meilleure, afin d'opérer leurs sélections. On les emploie dans les domaines où un grand nombre de paramètres entrent en jeu, et où l'on a besoin d'obtenir de bonnes solutions en quelques itérations seulement, par exemple dans les systèmes de régulation de transport en temps réel par exemple.

II.3.2.2.1.2. Les méthodes de recherche locale (méta-heuristiques à trajectoire ou de voisinage)

Une recherche locale nécessite la définition d'un voisinage $N(x)$ pour toute solution x.

En pratique, $N(x)$ contient un petit ensemble de solutions dérivées de x par des transformations simples. La procédure cherche une solution x' meilleure que x dans le voisinage $N(x)$ de la solution actuelle x. Elle stoppe si x' n'est pas trouvée : x est alors localement optimal dans son voisinage. Sinon, x est remplacé par x' et on itère le processus.

Cette classe regroupe plusieurs méthodes, les méthodes de Recuit Simulé et la recherche Tabou sont les plus anciennes et sans doute les plus populaires. [11]

A. Structure de voisinage et minimum local

Soit S un ensemble de solution à un problème d'optimisation, et soit f la fonction objectif.

- Une structure de voisinage (ou tout simplement un voisinage) est une fonction N qui associe un sous-ensemble de S à toute solution $s \in S$. Une solution $s' \in N(s)$ est dite voisine de s. [10]
- Une solution $s \in S$ est un minimum local relativement à la structure de voisinage N si :

$$f(s) \leq f(s') \ \forall\ s' \in N(s).$$

- Une solution $s \in S$ est un minimum global si : $f(s) \leq f(s') \ \forall\ s' \in S$.

Certaines méthodes d'optimisation, qui partent d'une solution initiale et qui l'améliorent en explorant son voisinage immédiat, présentent l'inconvénient de s'arrêter au premier minimum local trouvé, les méta-heuristique contiennent souvent une technique ou une astuce permettent d'éviter de se retrouver piégé dans ces minima locaux, en explorant d'avantage tout l'espace des solutions, de façon à augmenter la probabilité de rencontrer le minimum optimal, c'est-à-dire le minimum global.

B. Les algorithmes de recherche locale

a. Méthode de Descente [11]

C'est l'une des heuristiques de recherche locale les plus simples. Elle consiste à rechercher dans le voisinage de la solution courante, une solution de coût plus faible. Elle procède ainsi jusqu'à arriver à un optimum local.

A partir d'une solution trouvée par une heuristique par exemple, on peut très facilement implémenter des méthodes de descente. Ces méthodes s'articulent toutes autour d'un principe simple. Partir d'une solution existante, chercher une solution dans le voisinage et accepter cette solution si elle améliore la solution courante.

L'algorithme général présente le squelette d'une méthode de descente simple. A partir d'une solution initiale **x**, on choisit une solution **x'** dans le voisinage $N(\mathbf{x})$ de **x**. Si cette solution est meilleure que **x**, $f(\mathbf{x}')$ alors on accepte cette solution comme nouvelle solution **x** et on recommence le processus jusqu'à ce qu'il n'y ait plus aucune solution améliorante dans le voisinage de **x**.

Algorithme générale Descente

1- Initialisation: trouver une solution initiale **x.**
2- **Répéter.**
3- Recherche de voisinage: trouver une solution **x'** dans $N(\mathbf{x})$**.**
4- Si $f(\mathbf{x}') < f(\mathbf{x})$ alors.
5- **x':= x.**
6- Fin si.
7- **Jusqu'à** $f(\mathbf{y}) \geq f(\mathbf{x})$**,** quelque soit $\mathbf{y} \in N(\mathbf{x})$**.**

Avantages et inconvénients

En général, l'efficacité des méthodes de recherche locale simples (descente) est très peu satisfaisante. D'abord, par définition, la recherche s'arrête au premier minimum local rencontré, c'est là leur principal défaut. Pour améliorer les résultats, on peut lancer plusieurs fois l'algorithme en partant d'un jeu de solutions initiales différentes, mais la performance de cette technique décroît rapidement.

En revanche, autoriser de temps à autre une certaine dégradation des solutions trouvées, afin de mieux explorer tout l'espace des configurations, a conduit au développement des deux méthodes que nous explorons aux paragraphes suivants, à savoir le recuit simulé et la méthode Tabou. [11]

b. Le Recuit Simulé [6]

Le recuit simulé est une procédure itérative qui consiste à engendrer, à chaque itération, une nouvelle solution dans le voisinage de la solution courante. Si la nouvelle solution est meilleure, elle est acceptée, sinon elle est acceptée selon une loi de probabilité d'acceptation.

Algorithme général Recuit Simulé

1- Initialisation: trouver une solution initiale **x**, poser une température initiale **t**
2- **Répéter**
3- Recherche de voisinage: trouver une solution **x'** $\in N(\mathbf{x})$.
4- Déterminer $\Delta C = f(\mathbf{x}') - f(\mathbf{x})$.
5- Obtenir $p \sim \mathbf{U(0.\ 1)}$.
6- Si $\Delta C < 0$ ou $\mathbf{e}^{\frac{\Delta C}{t}} > p$ alors.
7- **x' := x.**
8- Fin si.
9- Réduire la température **t** selon un schéma de refroidissement.
10- **Jusqu'à** un critère d'arrêt satisfait.

Avantages et inconvénients [6]

Le recuit simulé présente l'avantage d'offrir des solutions de bonne qualité, tout en restant simple à programmer et à paramétrer. Il offre autant de souplesse d'emploi que l'algorithme de recherche local classique : on peut inclure facilement des contraintes dans le corps du programme.

Sous certaines conditions de décroissance de la température, l'algorithme du recuit simulé converge en probabilité vers un optimum global lorsque le nombre d'itérations tend vers l'infini.

L'un des inconvénients du recuit simulé est qu'une fois l'algorithme piégé à basse température dans un minimum local, il lui est impossible de s'en sortir tout seul. Plusieurs solutions ont été proposées pour tenter de résoudre ce problème, par exemple en acceptant une brusque remontée de la température, de temps en temps, pour relancer la recherche sur d'autres régions plus éloignées. Il est également possible d'empêcher la température de descendre trop bas : on lui donne une valeur minimale au delà de laquelle on ne change plus de palier de température.

c. La Recherche Tabou [13]

La méthode Tabou est une technique de recherche dont les principes ont été proposés pour la première fois par Fred Glover dans les années 80, et elle est devenue très classique en optimisation combinatoire. Elle se distingue des méthodes de recherche locale simples par le recours à un historique des solutions visitées, de façon à rendre la recherche un peu moins « aveugle ». Il devient donc possible de s'extraire d'un minimum local, mais, pour éviter d'y retomber périodiquement, certaines solutions sont bannies, elles sont rendues « taboues ».

A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution voisine s' de $N(s)$ à chaque itération, Tabou examine un échantillonnage de solutions de $N(s)$ et retient la meilleure s' même si $f(s')>f(s)$. La recherche Tabou ne s'arrête donc pas au premier optimum trouvé.

Le danger serait alors de revenir à s immédiatement, puisque s est meilleure que s'. Pour éviter de tourner ainsi en rond, on crée une liste T qui mémorise les dernières solutions

visitées et qui interdit tout déplacement vers une solution de cette liste. Cette liste T est appelée liste Tabou.

Algorithme général Recherche Tabou

1- Choisir une solution $s \in S$, poser $T := 0$ et $s^* := s$.
2- **Tant qu'**aucun critère d'arrêt n'est satisfait faire.
3- Déterminer une solution **s'** qui minimise f**(s')** dans N**(s).**
4- Si f **(s')** $< f$**(s*)** alors poser $s^* := s'$.
5- Poser **s:=s'** et mettre à jour **T.**
6- **Fin du tant que.**

Avantages et inconvénients [13]

La méthode Tabou est une méthode de recherche locale, et la structure de son algorithme de base est finalement assez proche de celle du recuit simulé, avec l'avantage, par rapport au recuit simulé, d'avoir un paramétrage simplifié : dans un première temps, le paramétrage consistera d'abord à trouver une valeur indicative t d'itérations pendant lesquelles les mouvements sont interdits.

En revanche, la méthode Tabou exige une gestion de la mémoire de plus en plus lourde à mesure que l'on voudra raffiner le procédé en mettant en place des stratégies de mémorisation complexe.

L'efficacité de la méthode Tabou fait qu'elle est largement employée dans les problèmes d'optimisation combinatoire : elle a été testée avec succès sur les grands problèmes classiques (voyageur de commerce, ordonnancement d'ateliers) et elle est fréquemment appliquée sur les problèmes de constitution de planning, de routage, d'exploration géologique, etc.

II.4. Approches méta-heuristiques pour la résolution du FJSP :

Les problèmes d'ordonnancement d'ateliers ont été très étudiés dans la littérature depuis plus de 50 ans. Nous ne faisons pas ici un état de l'art exhaustif étant donné qu'il existe de nombreux autres travaux dans la littérature. Cette étude bibliographique permet néanmoins de faire un tour d'horizon assez large des approches de résolution développées et permet de dégager les méthodes actuellement les plus performantes pour la résolution des problèmes

d'ateliers de type Job Shop Flexible. Et donc Plusieurs méthodes ont été développées pour résoudre ce type de problème mono- et/ou multicritères, à savoir des méthodes heuristiques.

Nous avons pour but à travers cette section de donner une brève présentation des différentes approches utilisées pour la résolution approchée des problèmes d'ordonnancement de type Job Shop Flexible que nous avons tiré des références. [6], [11]

- La première étude de ce type de problème est celle proposée dans [12]. Les auteurs se sont développé un algorithme polynomial pour la résolution optimale de ce problème dans le cas de deux jobs.

- Dans [13], l'auteur a utilisé une approche hiérarchique pour la résolution du Job Shop Flexible. Il a commencé par la résolution du sous-problème d'affectation de ressources en utilisant des règles de priorité. Une fois ce problème d'affectation résolu, le problème devient un Job Shop Classique. Pour le résoudre, Brandimarte a utilisé une méthode de recherche Tabou. Les expérimentations ont été menées sur des exemples générés aléatoirement. L'heuristique a de plus été appliquée à des problèmes de Job Shop Flexible avec minimisation des retards pondérés.

- Une approche en deux phases a été proposée dans [14] pour ordonnancer un système de production manufacturier dans lequel le nombre de jobs pouvant être exécutés simultanément est limité. Ainsi, si l'atelier est plein, un nouveau job ne peut être commencé que si l'un des jobs en cours dans l'atelier est terminé. L'objectif est de minimiser le Makespan. Les machines considérées sont identiques. Dans la première phase, les machines sont affectées aux opérations en utilisant des règles de priorité, et le problème devient alors un Job Shop Classique. La deuxième phase consiste à représenter le problème résultant par un graphe disjonctif utilisé pour développer une heuristique de recherche tabou permettant de trouver les meilleures séquences d'entrée. Les deux phases sont répétées et une opération appartenant au chemin critique est à chaque fois réaffectée.

- Une autre méthode de recherche Tabou a été proposée dans [15]. Les auteurs considèrent deux types de voisinages. Le premier consiste à permuter deux opérations critiques adjacentes et le deuxième consiste à réaffecter une opération critique sur toutes les machines pouvant l'exécuter. Ils ont considéré trois exemples de problème de type Job

Shop Classique FT10 de [16], LA24 et LA40 de [17] et ils ont fait une extension de ces derniers afin d'avoir des problèmes flexibles. Les machines considérées sont identiques.

- La méthode proposée dans [18] repose sur une résolution intégrée du problème d'affectation et de séquencement. Elle aussi basée sur une recherche Tabou.

- Dans [19], les auteurs proposent une autre approche intégrée basée sur une recherche Tabou. Les machines considérées sont identiques. Cette approche utilise le graphe disjonctif pour la représentation des solutions, ainsi que deux types de voisinages. Ces voisinages sont basés sur le déplacement d'une opération dans le graphe disjonctif. Tabou est parmi celles fournissant actuellement les meilleurs résultats sur les instances connues de Job Shop Flexible.

- Un algorithme génétique a été également proposé pour résoudre le problème de Job Shop Flexible dans [20]. Cet algorithme a été testé pour la résolution du problème mono et multicritère. L'objectif de cette méthode est d'optimiser conjointement deux critères classiques : le Makespan et la charge des ressources. La première étape de cette approche permet de résoudre le problème d'affectation de ressources et de construire un schéma d'affectation. La deuxième étape est une approche évolutionnaire contrôlée par le modèle des affectations généré dans la première étape. Dans cette approche, les auteurs appliquent des manipulations génétiques avancées pour améliorer la qualité des solutions.

- Dans leur travail, [21] ont proposé un algorithme génétique pour la résolution du Job Shop Flexible multicritère. L'objectif est de minimiser le Makespan, la charge maximale et la somme des charges de ressources. La méthode a été évaluée via les instances utilisées dans [20] et a donné de meilleures solutions.

- Dans [22] l'auteur s'est intéressé au problème du Job Shop Flexible. Pour résoudre ce problème, il a proposé deux recherches Tabou, deux algorithmes génétiques et un algorithme mimétique. Un algorithme mimétique est une hybridation entre un algorithme génétique et une méthode de recherche locale. Ici, l'auteur a proposé de remplacer la mutation de l'algorithme génétique par un algorithme de recherche Tabou. Les expérimentations ont permis de comparer les algorithmes entre eux. Il en résulte que la meilleure méthode est l'algorithme génétique avec une population partiellement

initialisée par la recherche Tabou. Néanmoins, dans certains cas, cette méthode est dominée par l'algorithme mimétique.

- Dans [23] les auteurs se sont intéressés au problème de Job Shop Flexible multicritère avec trois objectifs : minimisation du Makespan, minimisation de la charge maximale des machines et minimisation de la charge de travail totale. Les machines considérées sont identiques. Un nouvel algorithme génétique croisé avec une méthode de recherche locale a été proposé pour la résolution du problème. Ils proposent des opérateurs de croisement et de mutation avancés pour s'adapter aux structures de chromosomes et aux caractéristiques du problème. Deux types de voisinage sont utilisés : permutation des opérations et affectation de nouvelles machines pour les opérations critiques. Afin de restreindre le domaine de recherche, la structure de voisinage est ajustée dynamiquement par une recherche locale.

- Dans [24] les auteurs se sont également présenté un algorithme génétique pour le problème de Job Shop Flexible. L'algorithme proposé intègre différentes stratégies pour la création de la population initiale, les choix des individus et la reproduction des nouveaux individus. Les expérimentations ont montré que l'intégration de différentes stratégies pour le croisement et la mutation donne de bonnes solutions. Les résultats obtenus sont assez proches de ceux obtenus par la méthode de recherche Tabou proposée dans [19].

- Dans [25] l'auteur a développé un algorithme de recherche Tabou pour ce type de problème en minimisant le Makespan.

- Dans [26] les auteurs se sont proposé un modèle mathématique pour le problème de Job Shop Flexible et une hybridation de deux méthodes : une méthode de recherche Tabou (TS) et un recuit simulé (SA). Sur la base de ces deux méthodes, les auteurs ont développé six algorithmes différents basés sur la combinaison des deux méthodes.

- Dans [27] les auteurs ont proposé un algorithme génétique pour la résolution du Job Shop Flexible. Leur approche a été testée sur la même série de benchmark que celle de [26] et elle a donné de meilleurs résultats.

En effet, l'algorithme de la recherche Tabou proposé est connu pour être un des meilleurs algorithmes de la recherche locale pour la résolution de ce type de problème (Job Shop Flexible) et à prouvé son efficacité dans le domaine de la résolution des problèmes combinatoire.

II.5. Conclusion

Dans ce chapitre, nous avons présenté le problème de Job Shop Flexible en générale et sa formulation, ainsi, nous avons introduit les différentes méthodes d'optimisation combinatoire, exactes et approchées, qui sont couramment utilisées pour la résolution des problèmes d'ordonnancement d'atelier.

Nous avons défini dans ce chapitre les méthodes exactes. Par la suite, nous avons passé à la représentation des méthodes approchées ou méta-heuristiques les plus connus. A la fin du ce chapitre, un état de l'art sur les approches de résolution du Job Shop Flexible a été donné.

CHAPITRE 3

CONCEPTION ET ARCHITECTURE

DU LOGICIEL (ORdoRO)

Ce n'est pas dans la science qu'est le bonheur, mais dans l'acquisition de la science

Edgar Allan Poe

III.1. Introduction

Le but de notre travail est la réalisation d'un logiciel d'ordonnancement industriel basé sur le modèle Job Shop Flexible et un noyau de calcul méta-heuristique basé sur la recherche Tabou. Nous présentons dans ce chapitre d'une part l'approche Tabou développée. Elle est basée sur une structure de voisinage modifié qui permet de travaillé directement sur les représentations des solutions, non pas sur l'ordonnancement lui-même, ce qui réduit les temps de calcul du voisinage, et d'autre part l'analyse et la conception du logiciel proprement dit. A noté que la structure de voisinage utilisée a été adaptée de travaux concernant l'implémentation d'une méthode de recherche locale pour le Job Shop Classique [27] et qu'elle n'a jamais été utilisée dans le cadre de la recherche Tabou.

III.2. Application de la recherche Tabou

La recherche Tabou est une méta-heuristique d'optimisation présentée par Fred Glover en 1986. L'idée de la recherche tabou consiste, à partir d'une position donnée, à en explorer le voisinage et à choisir la position dans ce voisinage qui minimise/maximise la fonction objectif. Cette méthode a montré son efficacité pour la résolution de plusieurs problèmes difficiles et plus particulièrement les problèmes de Job Shop Flexible [22].

L'approche appliquée ici est détaillé par l'organigramme de la Figure III.1.

Commençant par une solution (ordonnancement) initiale, le processus de la recherche consiste, à chaque itération, à choisir la meilleure solution qui n'est pas tabou dans le voisinage de la solution courante, même si cette solution n'entraine pas une amélioration.
Nous présentons dans cette section la mise en œuvre des éléments suivants nécessaires à l'implémentation de la méthode Tabou.

-Le codage utilisé.

-La génération de la solution initiale (Aléatoire).

-Initialisation la liste tabou à vide.

-Déterminer tous les chemins critiques de S.

- Déterminer tous les opérations critiques de S.

- Déterminer les sous ensemble d'opérations critiques de S.

- Appliquer les fonctions de voisinages : N(S).

- Evaluer tous les voisins N(S).

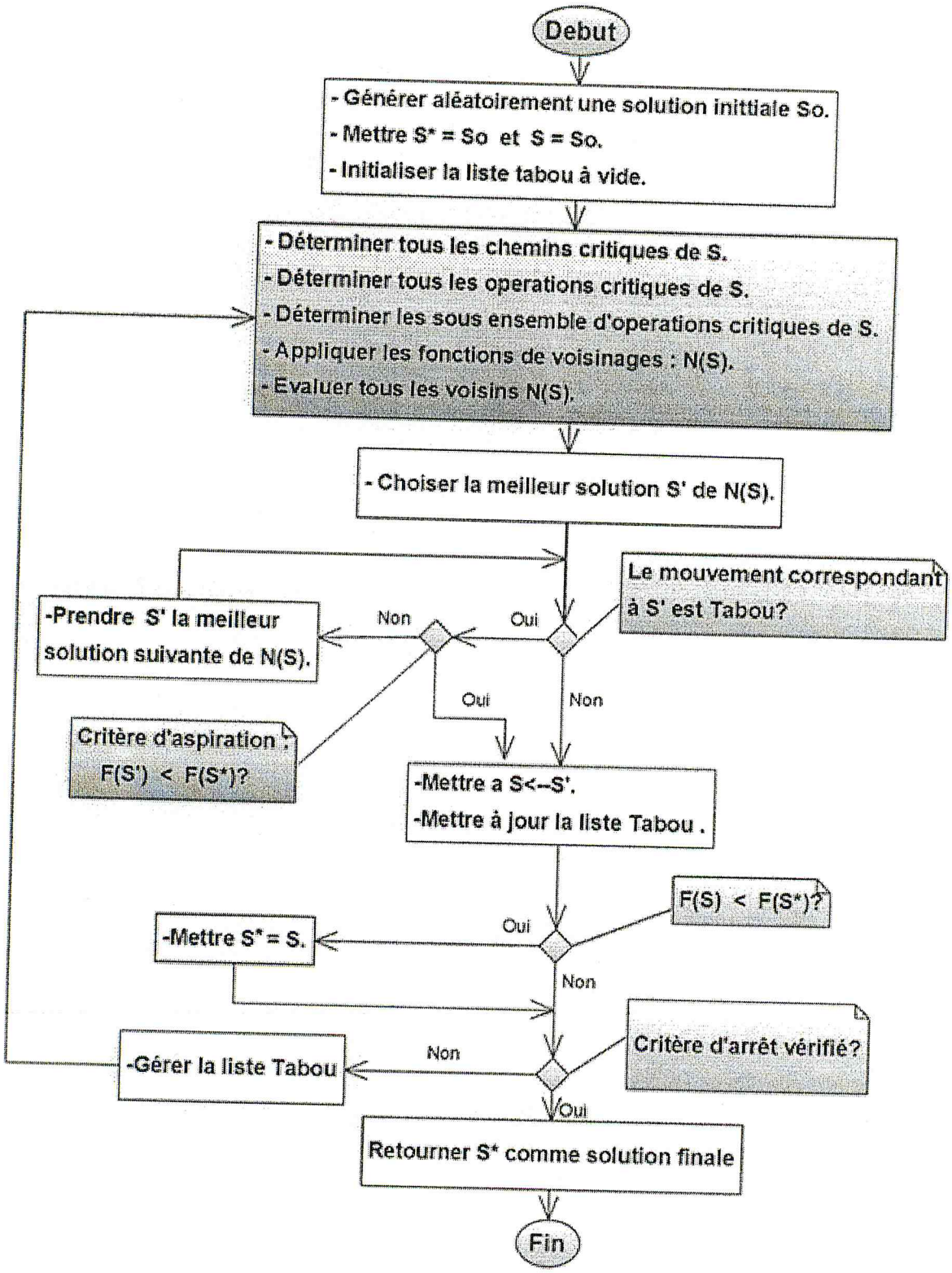

Figure III.1 : Organigramme de l'algorithme de la recherche tabou implémenté.

III.2.1. Codage de la solution [11]

Dans notre travail une solution est codée sur deux parties. Une partie pour coder les opérations : « Opération sélection **OS** », et un autre partie pour le codage des choix de machines pour chaque opération : « Machine sélection **MS** ».

Exemple : le tableau suivant donne un exemple d'un problème de type Job Shop Flexible.

Job	Opération	M1	M2	M4	M5	M6
J1	O11	2	6	5	3	4
	O12	-	8	-	4	-
J2	O21	3	-	6	-	5
	O22	4	6	5	-	-
	O23	-	7	11	5	8

Tableau III.1 : Exemple d'un problème d'ordonnancement de type Job Shop Flexible

- **Partie machine sélection(MS)** elle est constitué d'une série de nombres entiers qui représentent le choix de machine pour les opérations, tel que chaque opération à son choix de machine. La taille de la partie machine sélection (**MS**) est égale à la somme des opérations de toutes les tâches du problème. (Figure III.2).

  La figure suivante représente une solution de l'une des solutions du problème de l'exemple du (Tableau III.1)

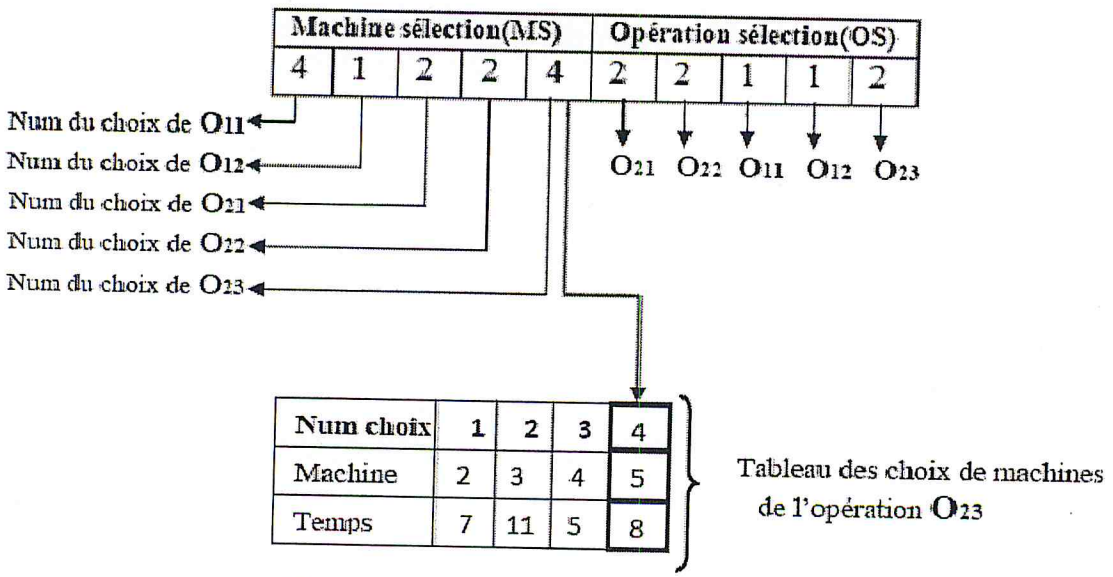

Figure III.2 : Présentation d'une solution.

- **Partie opération sélection(OS)**

Dans cette partie, nous avant utilisé un codage basé sur les opérations (Operation Based).

Codage basé sur les opérations : Un codage basé sur les opérations représente un ordonnancement comme une suite d'opérations. Chaque opération est codée par un gène dans la partie OS de la solution tel que toutes les opérations d'une tâche portent le même symbole (le numéro de la tâche). L'interprétation de ce symbole se fait suivant l'ordre de son occurrence dans la liste (Figure III.2).

Avec ce type de codage, La taille de la partie (**OS**) est égale à la somme des opérations de toutes les tâches du problème. Donc la taille de la solution est égale à la taille de la partie **OS** plus la taille de la partie **MS**.

III.2.2. Détermination du voisinage

Le voisinage d'une solution courante est constitué de toutes les solutions obtenues en effectuant des mouvements élémentaires sur un des chemins critique de cette solution pour obtenir un voisinage. Le voisinage qu'on à utilisé est celui utilisé par les auteurs [28].

III.2.2.1. Définition d'un mouvement

C'est une action qui est appliqué à la solution courante et qui nous permet de passer de cette solution à une autre solution qui est appelé **solution voisine** de cette solution. Dans notre application les mouvements sont soit des décalages soit des réaffectations. Nous allons détailler la procédure de décalage et de réaffectation [11].

III.2.2.1.1. Procédure de décalage

Notre approche est basée sur la notion d'opérations critiques. L'idée principale derrière le mouvement de décalage est de réorganiser les opérations critiques sur leurs machines correspondantes. Cela se fait en enlevant ces opérations puis en les réassignant de nouveau pour trouver une meilleure position sur sa machine correspondante. Ce mouvement est appliqué sur la partie OS de l'individu.

Nous définissions au préalable les notions de chemin critique et d'opération critique.

Avant de définir un chemin critique il faut d'abord définir c'est quoi une opération critique.

Opération critique : Dans un ordonnancement actif, une opération est dite critique, si elle provoque l'augmentation du Makespan lorsqu'elle est retardée [29]

Chemin critique : Un chemin critique est une suite d'opérations critiques liées par des relations de précédence. La longueur d'un chemin critique est égale à la somme des durées des opérations qui le composent. [29]

Pour illustrer la procédure de détermination des opérations critiques, on considère en exemple le problème Job Shop Classique avec quatre tâches et quatre machines (Tableau III.2) et considérant l'ordre d'opération suivant :

	O1	O2	O3	O4
J1	m 3, 2	m 2, 3	m 4, 5	m 1, 3
J2	m 4, 3	m 1, 4	m 2, 4	m 3, 4
J3	m 2, 3	m 4, 4	m 3, 6	m 1, 6
J4	m 3, 6	m 1, 2	m 2, 5	m 4, 3

Tableau III.2 : Exemple d'un problème d'ordonnancement de type Job Shop

OS = [2, 3, 2, 1, 1, 4, 4, 3, 4, 2, 3, 1, 3, 1, 2, 4].

L'ordre d'opérations pour cette partie OS est :

$O_{21} \rightarrow O_{31} \rightarrow O_{22} \rightarrow O_{11} \rightarrow O_{12} \rightarrow O_{41} \rightarrow O_{42} \rightarrow O_{32} \rightarrow O_{43} \rightarrow O_{23} \rightarrow O_{33} \rightarrow O_{13} \rightarrow O_{34} \rightarrow O_{14} \rightarrow O_{24} \rightarrow O_{44}$.

On considérant les affectations pour ces opérations nous pouvons obtenir le digramme de Gantt suivant :

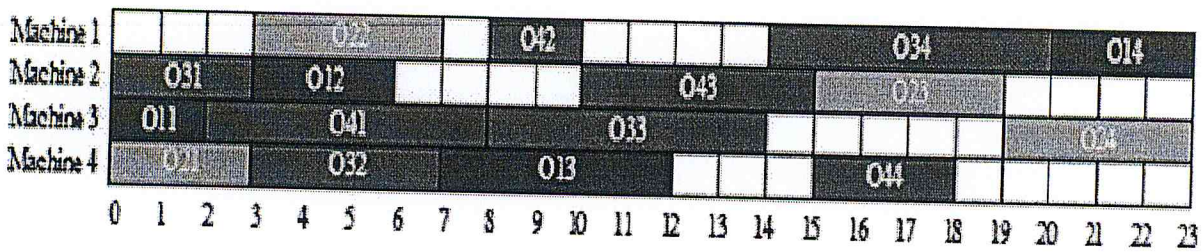

Figure III.3: Diagramme de Gantt qui représente Job Shop Avec deux Chemin Critique

Les chemins et opérations critiques pour cette solution sont donc :

1. $O_{11} \rightarrow O_{41} \rightarrow O_{33} \rightarrow O_{34} \rightarrow O_{14}$

2. $O_{11} \rightarrow O_{41} \rightarrow O_{42} \rightarrow O_{43} \rightarrow O_{23} \rightarrow O_{24}$

Pour améliorer la performance de la fonction de voisinage, seulement un sous ensemble d'opérations critiques est sélectionné. La détermination de ce sous ensemble est basé sur la règle suivante :

Seulement les opérations qui satisfont au deux critères suivants sont sélectionnés :

1- L'opération critique se trouve au milieu du chemin critique, non pas à son début ni à sa fin.
2- Les opérations qui la précèdent et qui la succèdent sur le chemin critique sont ou bien respectivement du même job et de la même machine, ou alors de la même machine et du même job.

Ainsi, parmi les 9 opérations critiques seulement 5 opérations satisfont ces critères :

O_{41}, O_{33}, O_{34}, O_{43}, O_{23}. Nous appelons ces opérations critiques : le Sous Ensemble d'Opération Critiques (SEOP).

Les opérations du SEOP seront réinsérées dans d'autres positions selon leur position dans le vecteur OS, et cela entre leur tâches antécédente et précédente. Sinon avant la fin de l'ordonnancement si c'est les dernières opérations.

Par exemple l'opération O_{33} peut être repositionnée entre les positions de O_{32} et O_{34} tandis que l'opération O_{34} peut être réaffectée à partir de la position de O_{33} à la fin de l'OS. Donc, les OS possibles pour les nouvelles positions de O_{34} sont comme suite :

$\{O_{21}, O_{31}, \ldots, O_{33}, \mathbf{O_{34}}, O_{13}, O_{14}, O_{24}, O_{44}\}$

$\{O_{21}, O_{31}, \ldots, O_{33}, O_{13}, \mathbf{O_{34}}, O_{14}, O_{24}, O_{44}\}$

$\{O_{21}, O_{31}, \ldots, O_{33}, O_{13}, O_{14}, \mathbf{O_{34}}, O_{24}, O_{44}\}$

$\{O_{21}, O_{31}, \ldots, O_{33}, O_{13}, O_{14}, O_{24}, \mathbf{O_{34}}, O_{44}\}$

$\{O_{21}, O_{31}, \ldots, O_{33}, O_{13}, O_{14}, O_{24}, O_{44}, \mathbf{O_{34}}\}$

Puisque les opérations O_{34} et O_{13} doivent être traitée sur des machines différentes, leurs ordres sur l'OS est indifférent est ne change pas l'ordonnancement. Dans ce cas, les deux premiers

OS sont équivalents et leur ordonnancement est représenté dans la figure 3.3. Puisqu'il y a une situation semblable pour les opérations O_{24}, O_{34} et O_{44}, leurs ordres ne sont pas importants. Les trois derniers OSs sont aussi équivalents à et leur ordonnancement est représenté dans la figure III.3.

Ainsi il ya seulement deux ordonnancement différents : celui qui considère O_{34} avant O_{14} et un autre qui assigne 0_{34} après O_{14}. Depuis l'ancien est l'ordonnancement actuel, il y a seulement un nouvel OS qui assigne O_{34} exactement après O_{14}.(figure III.4)

$\{O_{21}, O_{31}, \ldots, O_{33}, O_{13}, \mathbf{O_{14}}, \mathbf{O_{34}}, O_{24}, O_{44}\}$

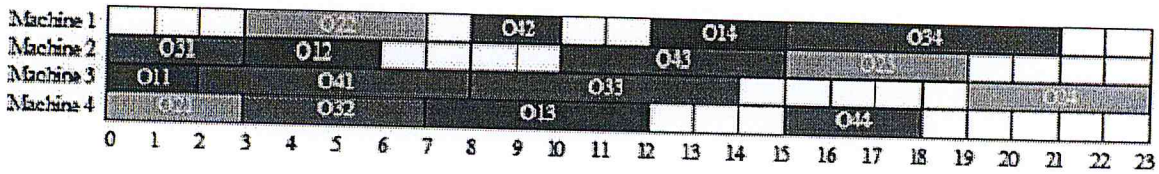

Figure 3.4: Diagramme de Gantt qui représenté Job Shop après repositionnement de O_{34}.

Puisque les ordonnancements sont indépendants de l'ordre des opérations qui ne sont pas du même job et ne sont pas traité sur la même machine, le nombre de nouveaux OS pour le mouvement de décalage du SEOP est limité est facilite le calcul.

III.2.2.1.2. Procédure réaffectation :

La procédure de réaffectation est simplement basée sur le changement de machine pour chaque opération de l'ensemble globale des opérations critiques du système.

III.2.3. Mémoire Tabou

Afin d'éviter le piège des optima locaux dans lequel le processus de recherche peut être facilement attrapé, la recherche Tabou utilise la mémoire tabou pour enregistrer des mouvements déjà effectués.

L'un des aspects critique d'une recherche Tabou est la nécessité d'ajuster la longueur de la liste tabou pour parcourir efficacement l'espace des solutions.

Si la liste est trop courte, la recherche finit par explorer un optimum local de rayon légèrement supérieur, les différentes solutions explorées forment un cycle qui va se répéter indéfiniment.

A l'inverse, si la liste est trop longue, tous les mouvements peuvent devenir tabous et sans nouveau voisins la recherche s'arrête, c'est un blocage [11].

II.2.3.1. Gestion de la mémoire tabou

Les éléments de la mémoire tabou doivent comporter suffisamment d'informations pour mémoriser fidèlement la solution visitée. La mémorisation de configurations entières serait trop couteuse en temps de calcule et en espace mémoire et ne serait sans doute pas la plus efficace. Par conséquent, elle n'est pas applicable pour la majorité des problèmes. Couramment, ce sont des caractéristiques de solutions, ou des mouvements, qui se gardent dans la mémoire, au lieu de solution complète. [28]

Dans notre application les éléments de la mémoire Tabou sont les mouvements de décalage et réaffectation et la taille de la mémoire Tabou est fixée à 40. La gestion de la mémoire se fait selon la règle FIFO.

II.2.4.Critère d'aspiration

Le critère d'aspiration est « Une condition nécessaire et satisfaisante pour qu'un mouvement tabou soit accepté malgré son statu tabou ». [30]

Le critère d'aspiration appliqué ici est celui adopté dans plusieurs applications de la recherche Tabou au problème de Job Shop Flexible et qui semble performant. Ce critère consiste à révoqué le statu tabou d'un mouvement si ce dernier conduit à une meilleur solution obtenue jusqu'à lors.

Une fois les algorithmes sont défini nous allons passer a la présentation de la démarche utilisée.

III.3. Présentation de la démarche utilisée

Nous expliquons dans cette étape du chapitre, les besoins de notre application afin de pouvoir passer à l'étape de conception et d'architecture. Nous présentons le cycle de vie que nous avons suivi pour la réalisation de ce projet. Nous illustrerons les solutions apportées par notre outil face aux problèmes posés, en se basant sur le langage UML (Unified Modeling Language) en utilisant le processus UP (Unified Process).

UP est une méthode de prise en charge du cycle de vie d'un logiciel développé en orienté objet. Il représente les étapes du cycle de vie sous formes de diagrammes UML.

III.3.1. Le cycle de vie

Le cycle de vie d'un logiciel est un ensemble séquentiel de phases, dont le nom et le nombre sont déterminés en fonction des besoins du projet, permettant généralement le développement d'un service ou d'un produit, en ce qui concerne notre projet nous avons suivi le modèle en cascade. [31]

III.4. Modèle en cascade :

Ce modèle est constitué d'une suite d'étapes qui ont pour but de réaliser un produit logiciel fini et testé. Le résultat de chaque étape est testé et on ne passe à l'étape suivante que lorsque l'étape actuelle est satisfaisante. Ce modèle est un cycle de vie linéaire, séquentiel, défini dans les années 70.

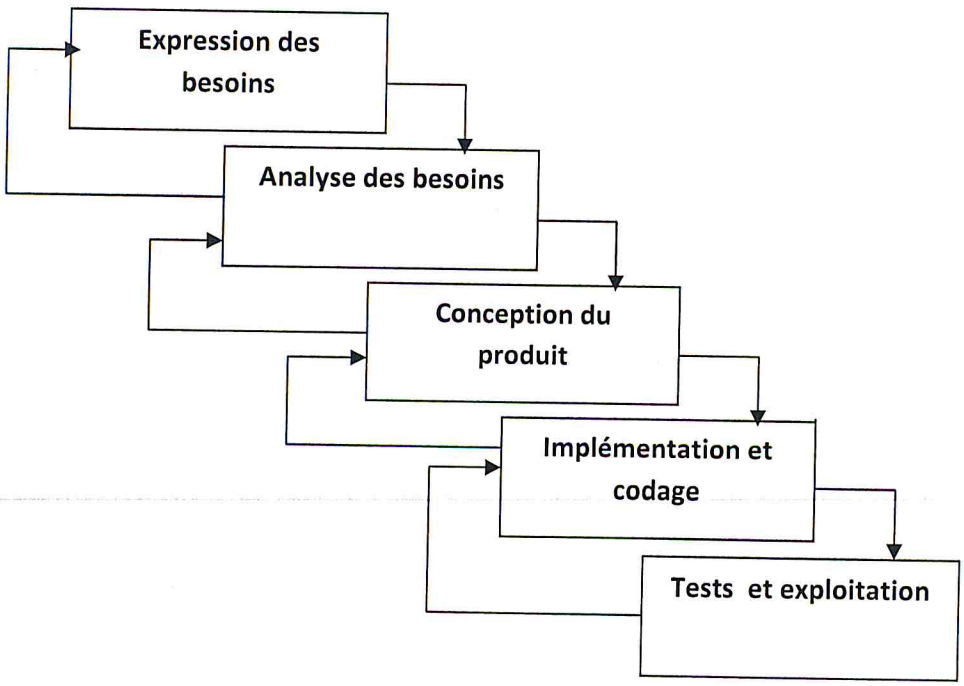

Figure III.5. Cycle de vie selon le modèle en cascade.

III.4.1. Expression des besoins

La spécification des besoins est une étape essentielle au début de processus de développement, elle consiste généralement à déterminer précisément les besoins des utilisateurs du système afin d'éviter de développer un logiciel non adéquat.

Cette étape ne préoccupe pas des solutions mais des questions : elle identifie le « quoi faire ?»Et identifie les entités de l'environnement du système. Pour modéliser ces besoins on utilise le diagramme des cas d'utilisation d'UML. [31]

III.4.2. Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client, il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution.

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structures sous une forme qui facilite la compréhension(Scenario) en utilisant le diagramme de séquence d'UML pour représenter les interactions entres les objets. [32]

III.4.3. Conception

C'est la phase la plus importante du processus de développement d'un logiciel. Elle s'intéresse d'abord au «comment ? », à savoir la solution du problème énoncé.

La conception a pour but de décomposer le logiciel en module, de préciser les interfaces et les fonctions de chaque module. A l'issue de cette étape, on obtient une description de l'architecture du logiciel et un ensemble de spécifications de ces divers composants en utilisant le diagramme de classe d'UML. [32]

III.4.4. Implémentation

L'implémentation est le résultat de la conception pour implémenter le système sous forme de composants, c'est-à-dire, de code source, de scripts exécutables et d'autres éléments du même types.

Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous forme de code source. [33]

III.4.5. Tests

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun. [31]

III.5.Expression des besoins

Cette phase consiste à définir les besoins fonctionnels de notre futur système, nous allons parler des fonctionnalités que peut offrir ce dernier afin de bien connaître les acteurs qui interagissent avec lui par la suite nous allons les modéliser en utilisant le diagramme des cas d'utilisation d'UML. [31]

III.5.1. Identification des acteurs et de cas d'utilisation

- *Les* acteurs

  L'acteur est, par définition le rôle joué par une personne qui interagit avec le système. Il est en principe extérieur au système, délimité par ses bornes.
  L'acteur a un nom, qui le définit, ou qui précise son rôle dans la transaction décrite. Notre système est composé d'un seul type d'acteur :
  Utilisateur : possède de simples connaissances dans le domaine d'ordonnancement. Il peut tout de même interagir avec notre outil graphique et l'exploiter au niveau qu'il souhaite.

- Les cas d'utilisations

  Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service.

Nous allons présenter notre diagramme de cas d'utilisation global, suivi par chaque cas d'utilisation détaillé.

III.5.2. Diagrammes de cas d'utilisation

- **Diagramme de cas d'utilisation global**

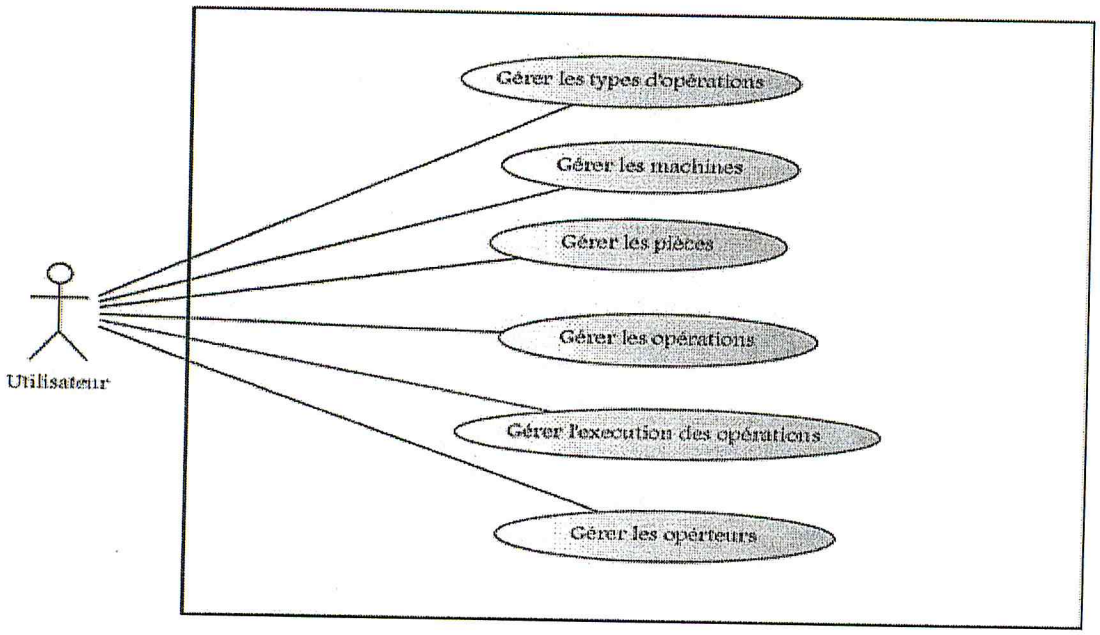

Figure III.6. Diagramme de cas d'utilisation global.

- **Description du diagramme de cas d'utilisation global**

Cas d'utilisation	Description
Gérer les types d'opérations.	L'utilisateur peut gérer les différentes fonctions de type d'opérations.
Gérer les machines.	L'utilisateur peut gérer les différentes fonctions de machines.
Gérer les pièces.	L'utilisateur peut gérer les différentes fonctions de pièces.
Gérer les opérations.	L'utilisateur peut gérer les différentes fonctions d'opérations.
Gérer l'exécution des opérations	L'utilisateur peut gérer les différentes fonctions d'exécution de chaque opération.
Gérer les opérateurs.	L'utilisateur peut gérer les différentes fonctions des opérateurs.

Tableau III.3. Description détaillée du diagramme de cas d'utilisation global

- **Diagramme de cas d'utilisation** « Gérer les types d'opérations»

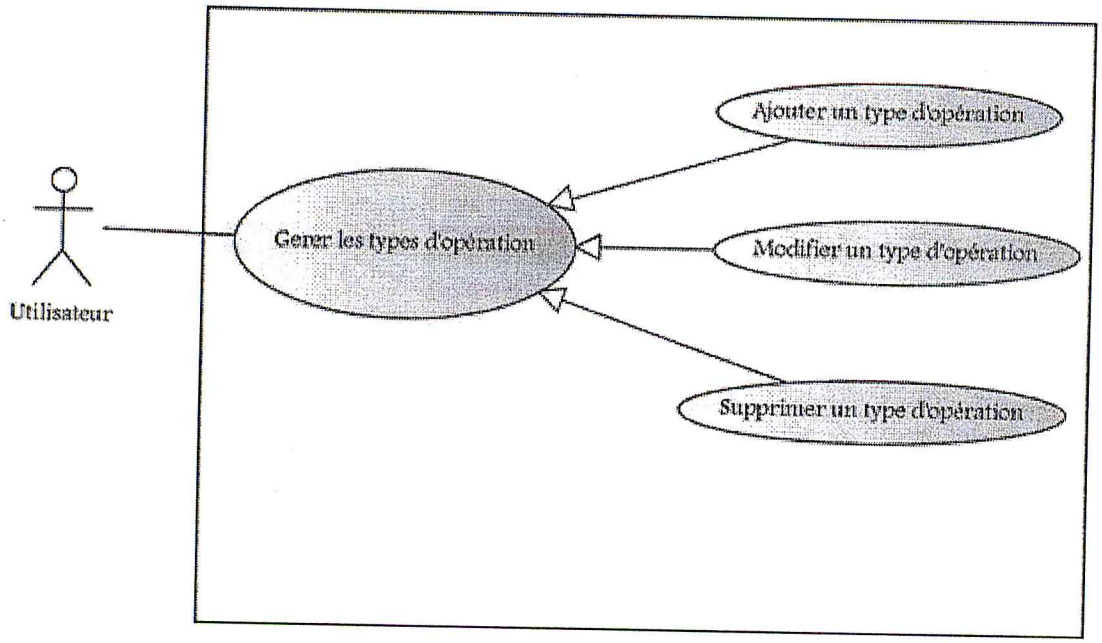

Figure III.7. Diagramme de cas d'utilisation gérer les types d'opérations.

- **Description du diagramme cas d'utilisation de Gérer les types d'opérations**

cas d'utilisation	**Description**
Ajouter un type d'opération.	L'utilisateur peut ajouter un type d'opération après avoir rempli les informations.
Modifier un type d'opération.	L'utilisateur peut modifier un type d'opération après avoir changer les informations.
Supprimer un type d'opération.	Ici l'utilisateur sélectionne le type d'opération qu'il veut supprimer.

Tableau III.4. Description du diagramme de cas d'utilisation gérer les types d'opérations.

- **Diagramme de cas d'utilisation** « Gérer les machines»

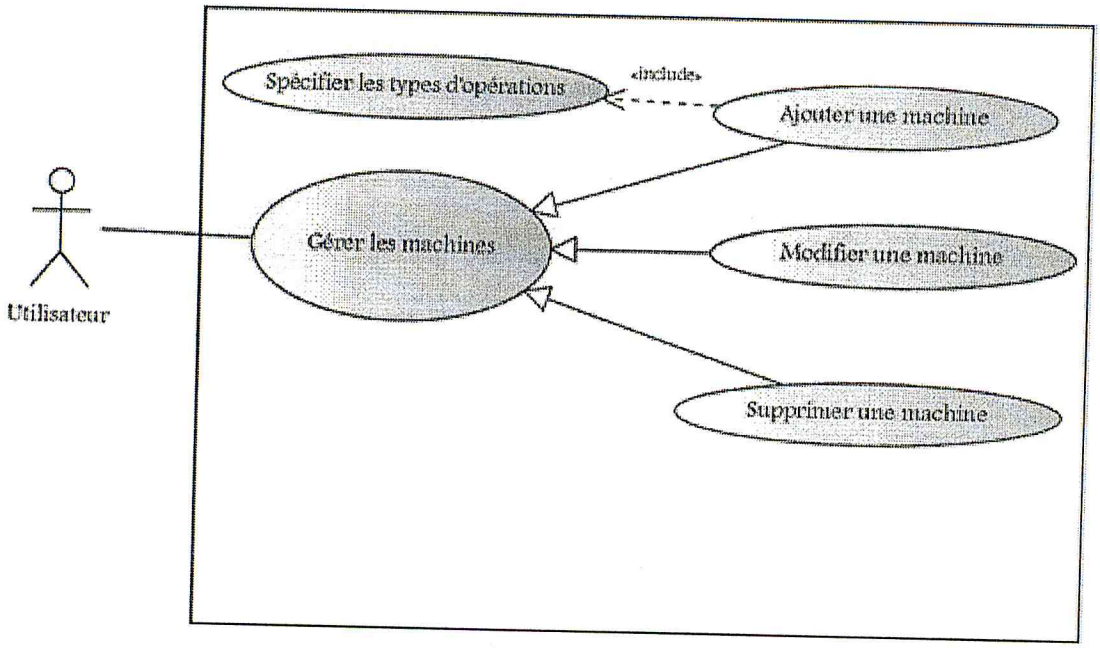

Figure III.8. Diagramme de cas d'utilisation gérer les machines.

- **Description du diagramme cas d'utilisation gérer les machines**

cas d'utilisation	**Description**
Ajouter un type d'opération.	L'utilisateur peut ajouter une machine après avoir rempli les informations.
Spécifier les types d'opérations.	l'utilisateur doit Spécifier les types d'opérations pour chaque machine.
Modifier un type d'opération.	L'utilisateur peut modifier une machine après avoir changer les informations.
Supprimer un type d'opération.	Ici l'utilisateur sélectionne la machine qu'il veut supprimer.

Tableau III.5. Description du diagramme de cas d'utilisation gérer les machines.

- **Diagramme de cas d'utilisation** « Gérer les pièces»

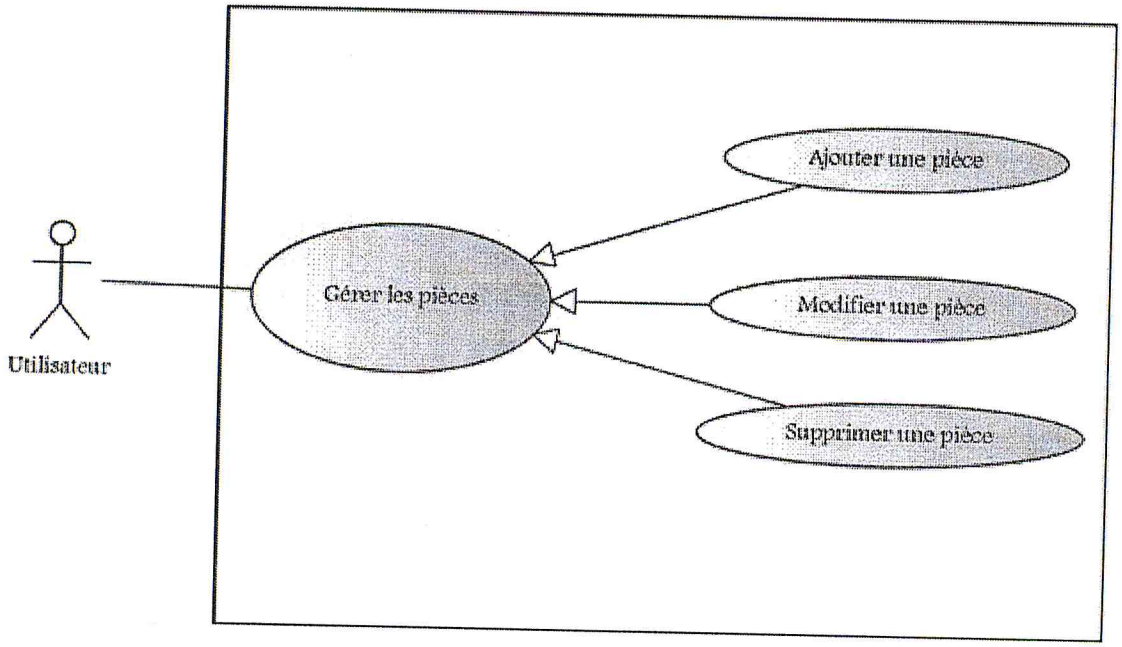

Figure III.9. Diagramme de cas d'utilisation gérer les pièces.

- **Description du diagramme de cas d'utilisation gérer les pièces:**

cas d'utilisation	**Description**
Ajouter un type d'opération.	L'utilisateur peut ajouter une pièce après avoir rempli les informations.
Modifier un type d'opération.	L'utilisateur peut modifier une pièce après avoir changer les informations.
Supprimer un type d'opération.	Ici l'utilisateur sélectionne la pièce qu'il veut supprimer.

Tableau III.6. Description du diagramme de cas d'utilisation gérer les pièces.

- **Diagramme de cas d'utilisation** « Gérer les opérations»

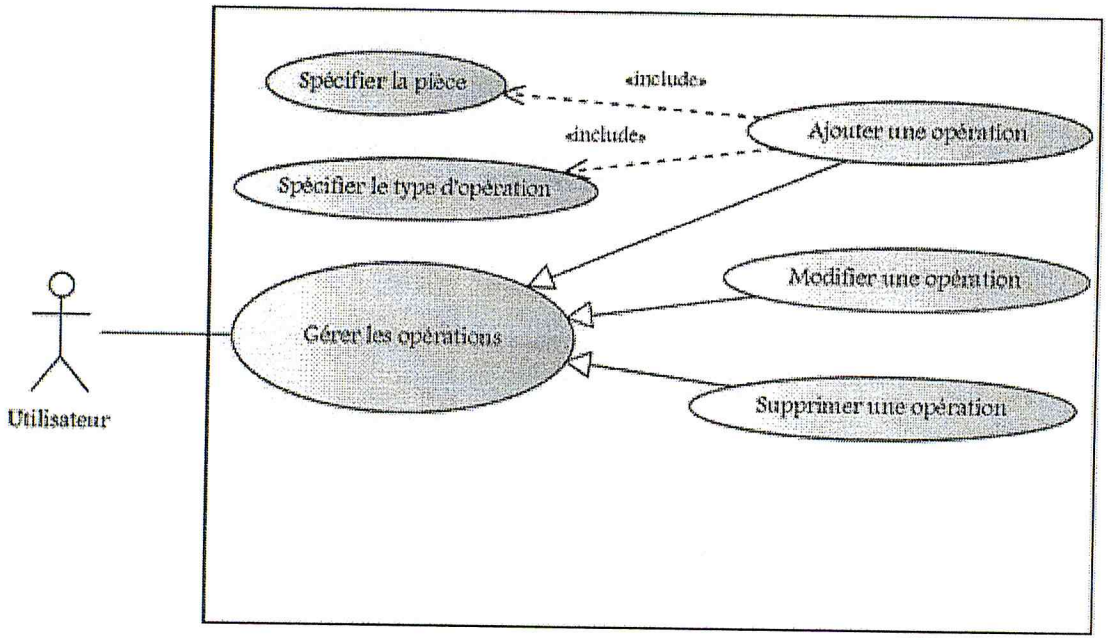

Figure III.10. Diagramme de cas gérer les opérations.

- **Description du diagramme de cas d'utilisation gérer les opérations:**

cas d'utilisation	**Description**
Ajouter un type d'opération.	L'utilisateur peut ajouter une opération après avoir rempli les informations.
Spécifier la pièce.	L'utilisateur doit spécifier la pièce.
Spécifier le type d'opération.	L'utilisateur doit spécifier le type d'opération.
Modifier un type d'opération.	L'utilisateur peut modifier une opération après avoir changer les informations.
Supprimer un type d'opération.	Ici l'utilisateur sélectionne l'opération qu'il veut supprimer.

Tableau III.7. Description du cas d'utilisation gérer les opérations.

- **Diagramme de cas d'utilisation** « Gérer l'exécution des opérations»

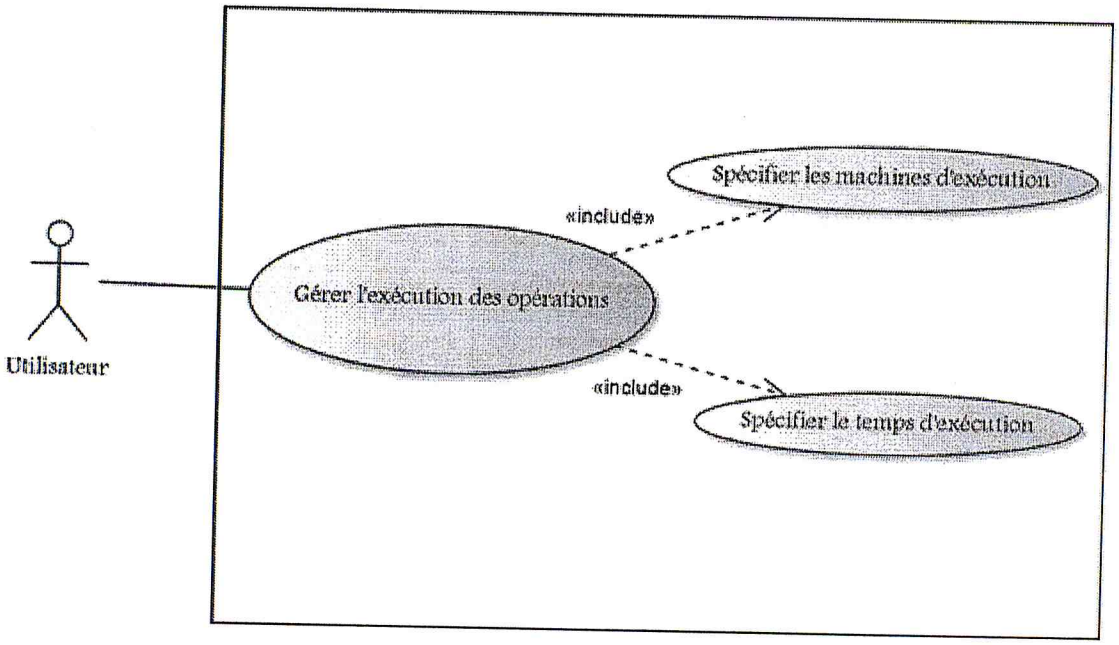

Figure III.11. Diagramme de cas d'utilisation gérer l'exécution des opérations.

- **Description du diagramme de cas d'utilisation gérer l'exécution des opérations:**

cas d'utilisation	**Description**
Spécifier les machines d'exécution.	L'utilisateur doit spécifier les machines d'exécution.
Spécifier le temps d'exécution.	L'utilisateur doit spécifier le temps d'exécution.

Tableau III.8. Description du cas d'utilisation gérer l'exécution des opérations.

- **Diagramme de cas d'utilisation** « Gérer les opérateurs»

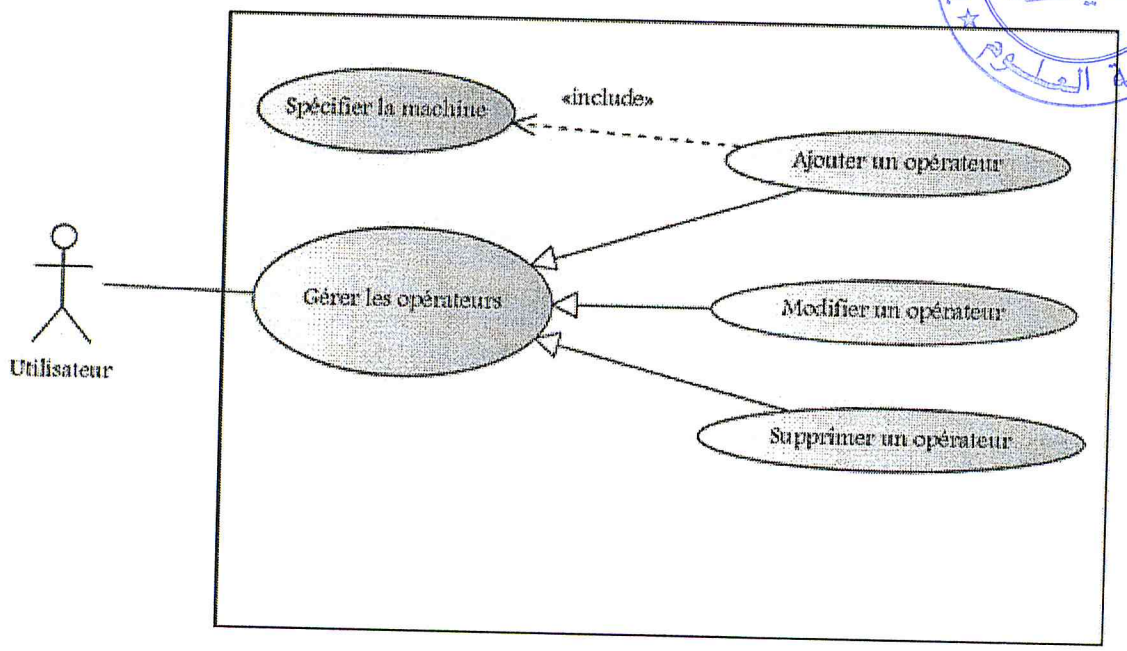

Figure III.12. Diagramme de cas d'utilisation gérer les opérateurs.

- **Description du diagramme cas d'utilisation Gérer les opérateurs:**

cas d'utilisation	**Description**
Ajouter un type d'opération.	L'utilisateur peut ajouter un operateur après avoir rempli les informations.
Spécifier la machine.	L'utilisateur doit spécifier la machine affecter au operateur.
Modifier un type d'opération.	L'utilisateur peut modifier un operateur après avoir changer les informations.
Supprimer un type d'opération.	Ici l'utilisateur sélectionne l'operateur qu'il veut supprimer.

Tableau III.9. Description du diagramme de cas diagramme d'utilisation gérer les opérateurs.

- **Diagramme de cas d'utilisation** « Configurer l'exécution d'algorithme »

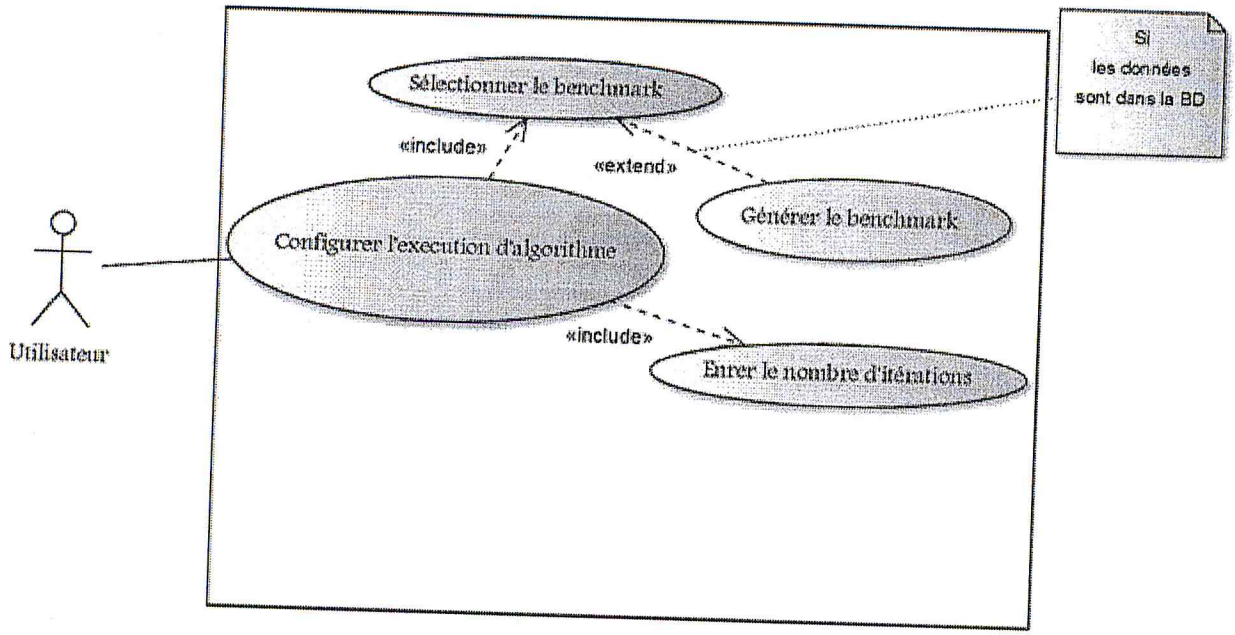

Figure III.13. Diagramme de cas d'utilisation configurer l'exécution d'algorithme.

- **Description du diagramme de cas d'utilisation configurer l'exécution d'algorithme :**

cas d'utilisation	Description
Sélectionner le benchmark.	L'utilisateur doit choisir le benchmark.
Générer le benchmark.	Si le benchmark n'est pas encore crée , l'utilisateur doit le créer à partir d'une base de données déjà crée.
Entrer le nombre d'itérations.	Ici l'utilisateur doit entrer le nombre d'itérations

Tableau III.10. Description du diagramme de cas configurer l'exécution d'algorithme.

III.6. Analyse

L'analyse permet de lister les résultats attendus, en terme de fonctionnalités, de performance, de robustesse, de maintenance,… etc.

L'analyse répond donc à la question « *que faut-il faire ?* » et a pour but de se doter d'une vision claire et rigoureuse du problème posé et du système à réaliser en déterminant ses éléments et leurs interactions.

L'analyse livre une spécification plus précise des besoins grâce à l'utilisation du diagramme de séquence. Elle peut être envisagée comme une première ébauche du modèle de conception. [34]

III.6.1. Scenarios et diagramme de séquences

Définition d'un scénario [35]

Un scénario représente un ensemble ordonné de messages échangés par des objets. On parle ici d'objet au sens large : instance de classe d'analyse ou instance d'acteur.

Les échanges de messages entre objet peuvent être représentés en UML dans une sorte de diagramme complémentaire appelé *diagramme de séquence*.

Définition du diagramme de séquence [36]

Un diagramme de séquence est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et / ou de ses acteurs.

Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages.
Dans ce qui suit nous allons présenter les diagrammes de séquence afin de formaliser les scénarios des cas d'utilisation vus précédemment. Nous allons voir le système comme un ensemble d'objet en interaction.

III.6.1.a. Diagramme de séquence « Gérer les types d'opérations»

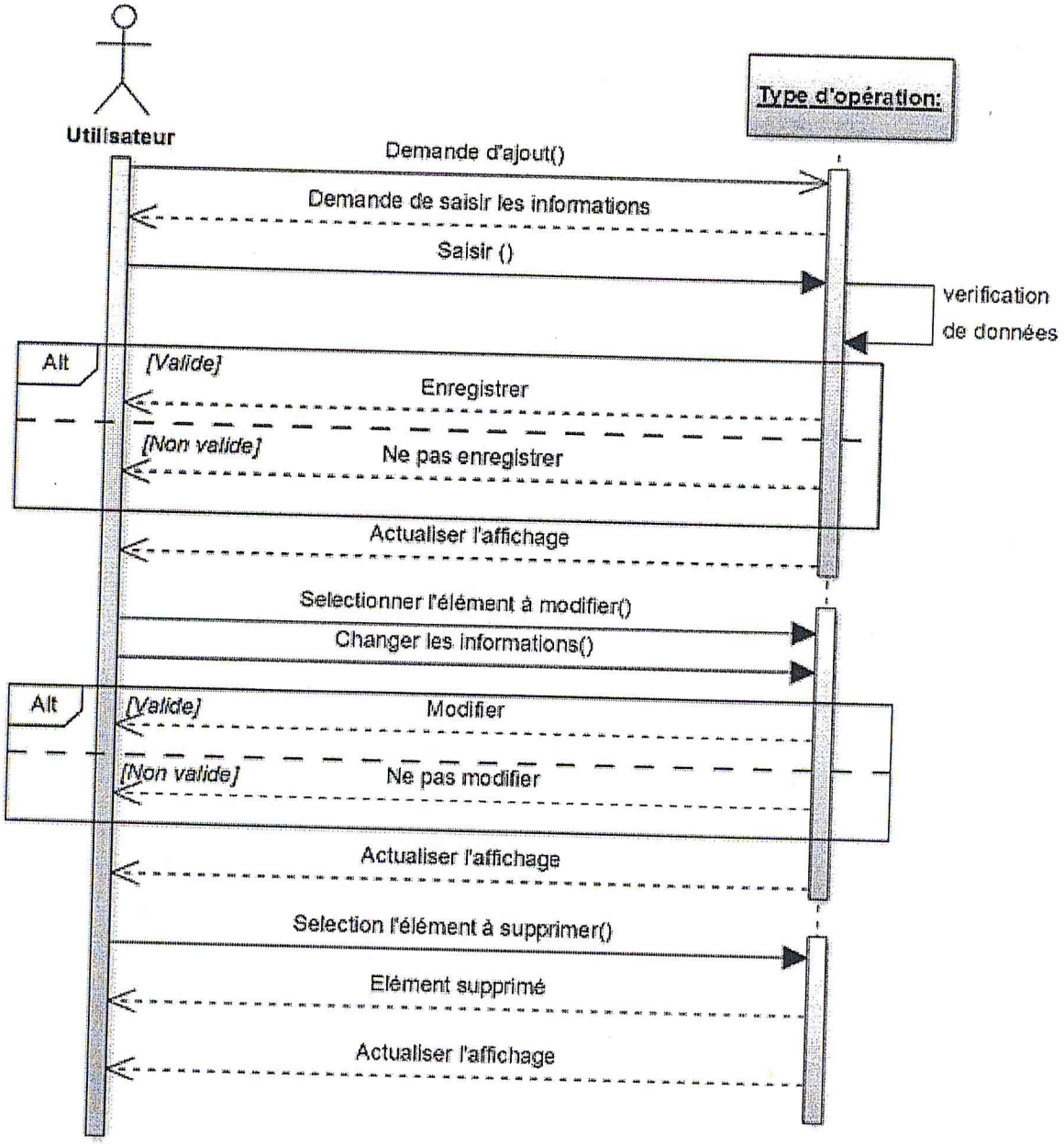

Figure III.14. Diagramme de séquence gérer les types d'opérations.

III.6.1.b. Diagramme de séquence« Gérer les machines»

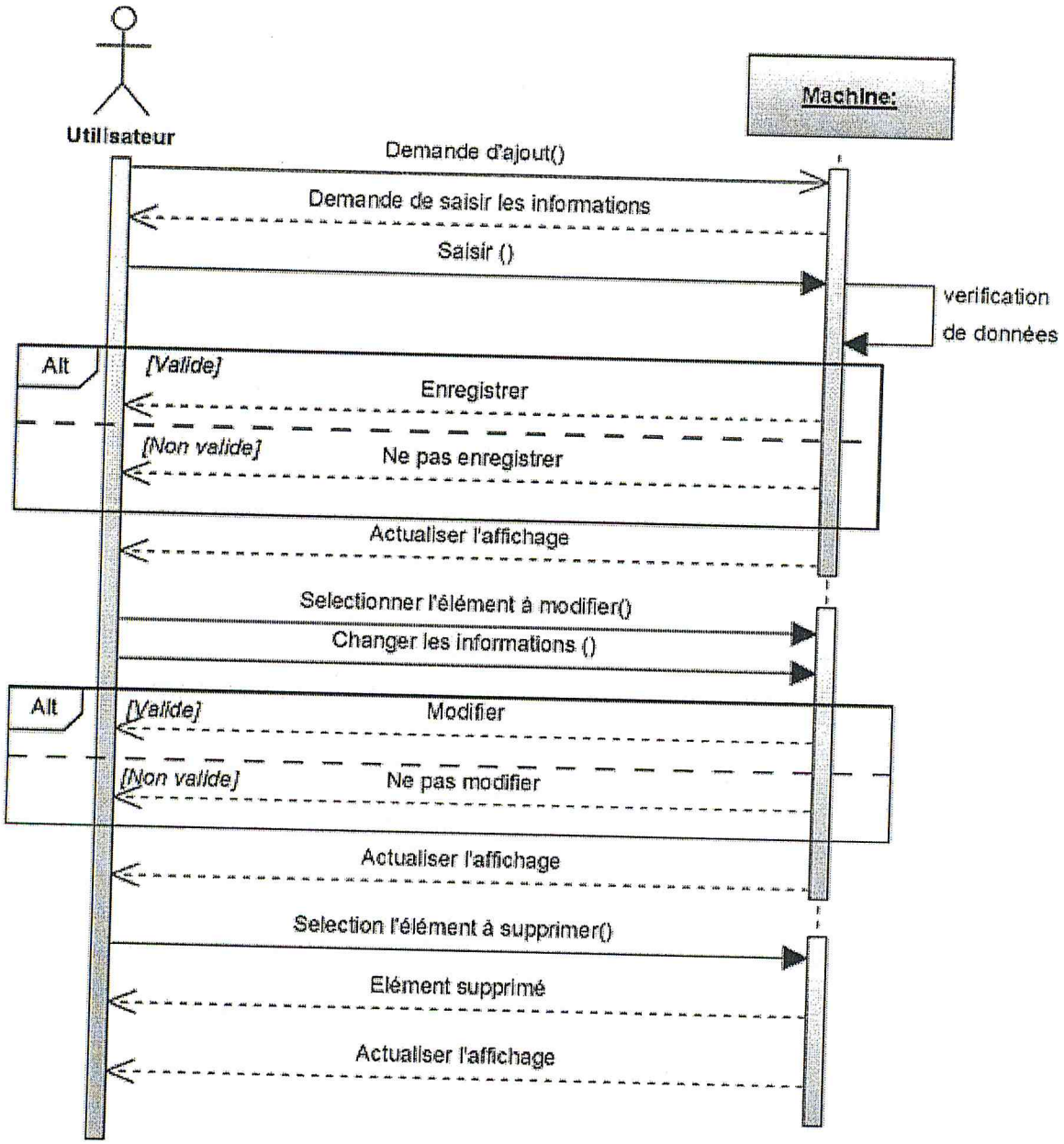

Figure III.15. Diagramme de séquence de gérer les machines.

III.6.1.c. Diagramme de séquence « Gérer les pièces»

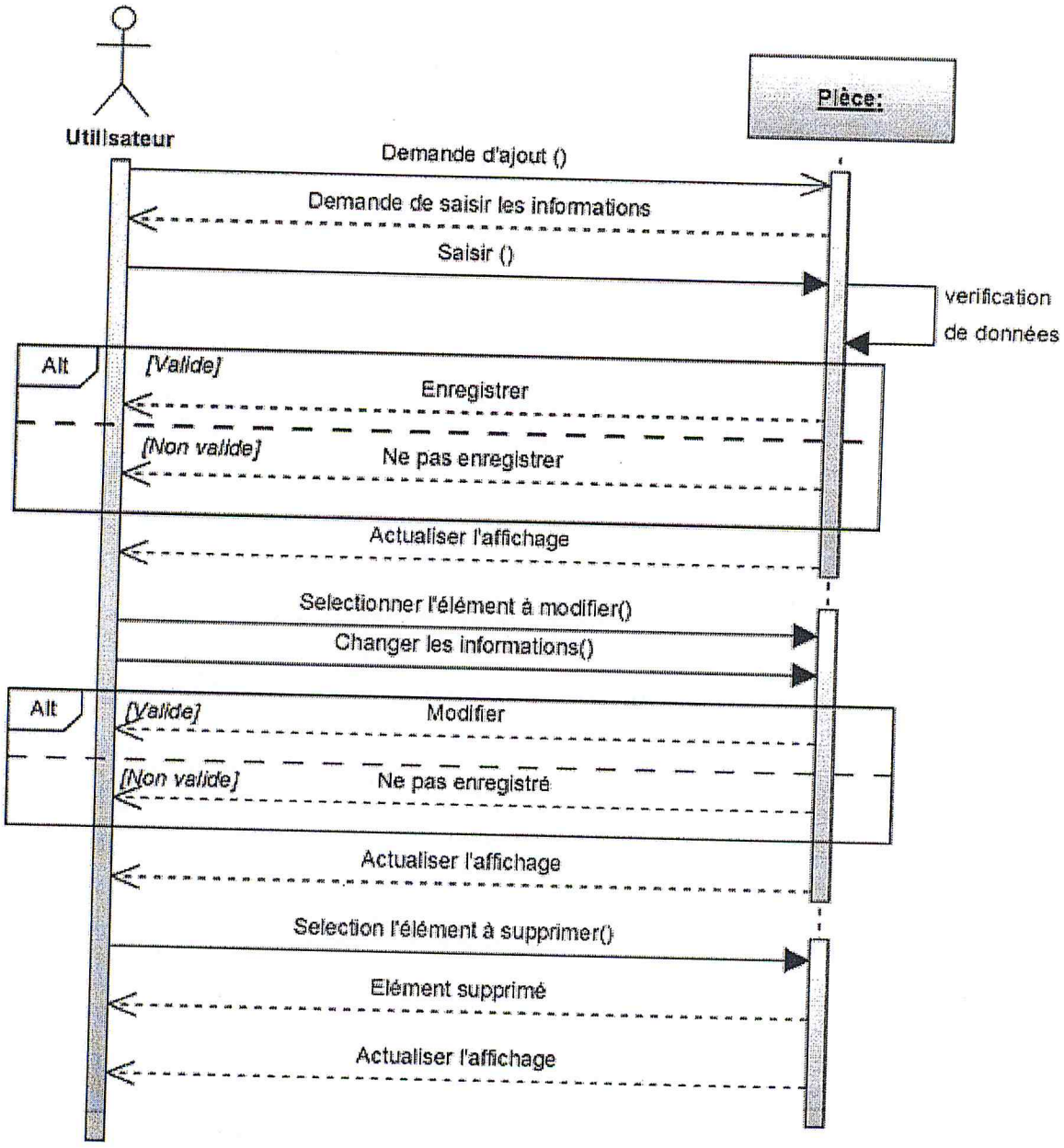

Figure III.16. Diagramme de séquence gérer les pièces.

III.6.1.d. Diagramme de séquence « Gérer les opérations»

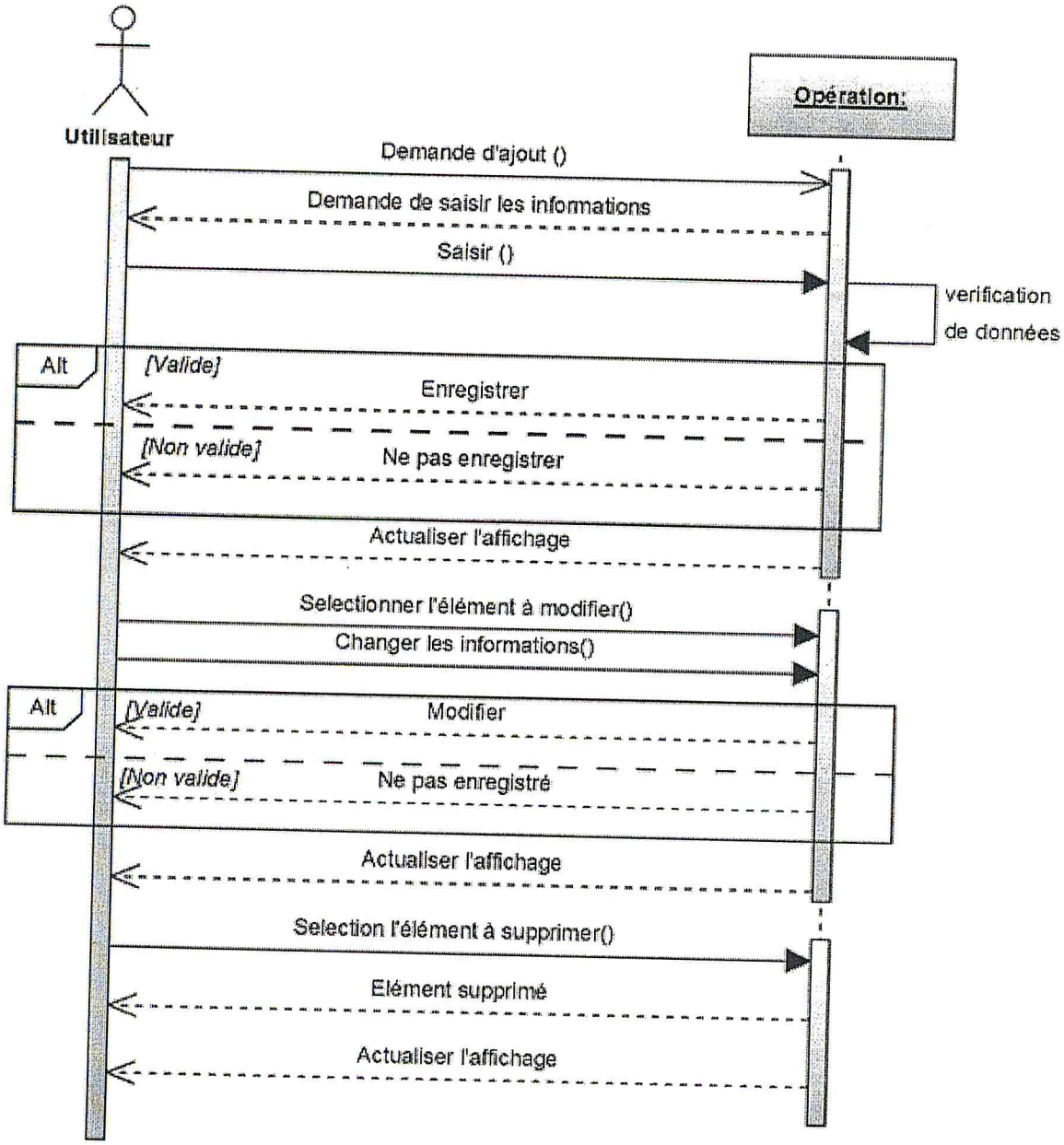

Figure III.17. Diagramme de séquence gérer les opérations.

III.6.1.e Diagramme de séquence « Gérer l'exécution des opérations»

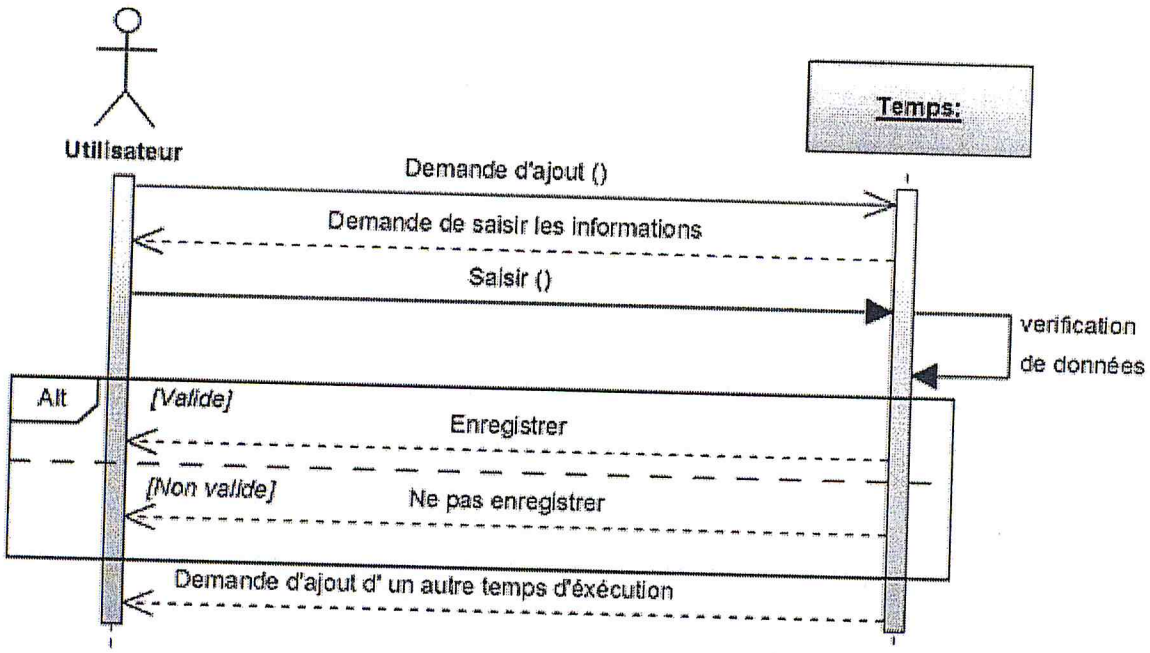

Figure III.18. Diagramme de séquence gérer l'exécution des opérations.

III.6.1.f. Diagramme de séquence « Gérer les opérateurs»

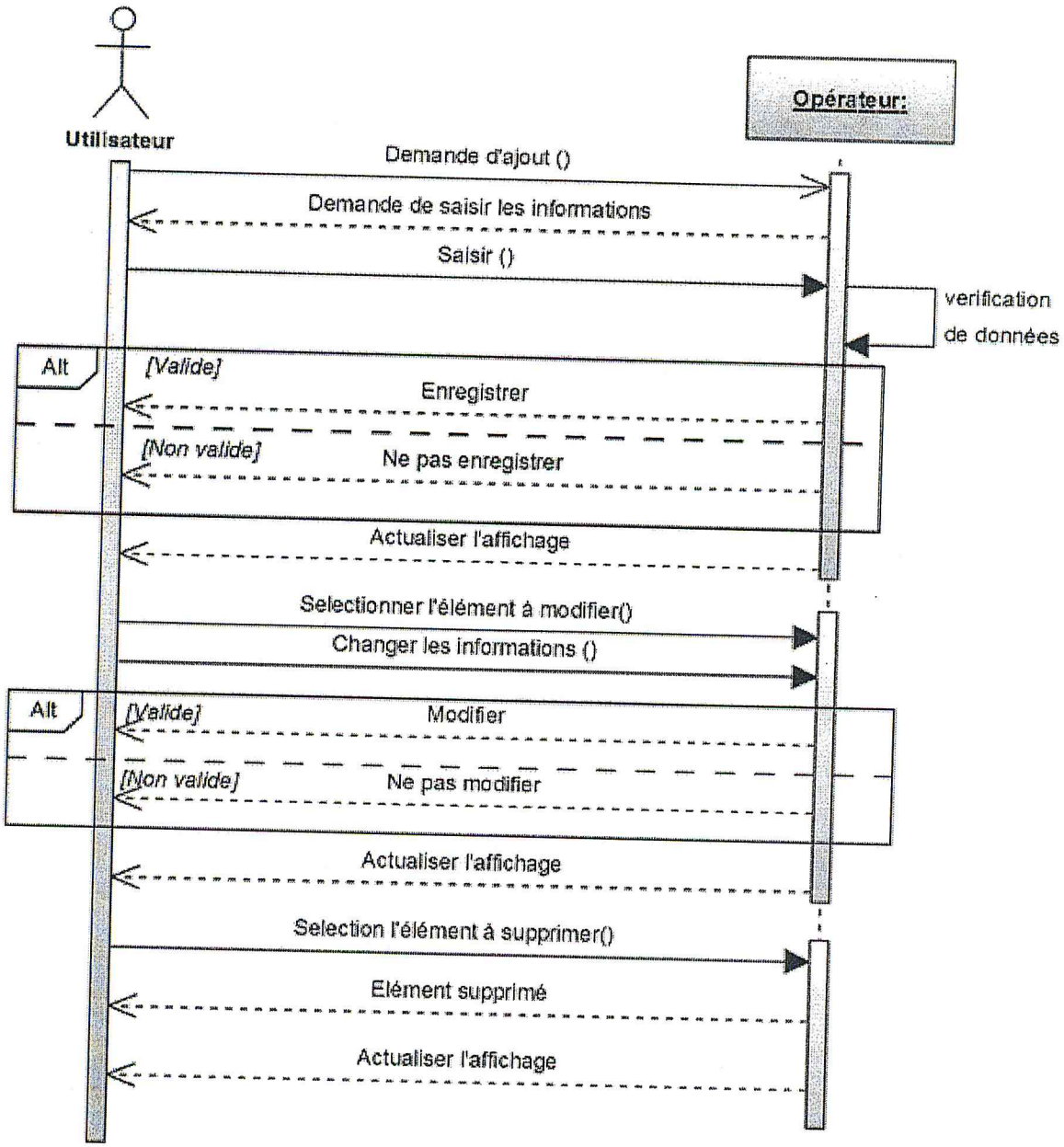

Figure III.19. Diagramme de séquence gérer les opérateurs.

III.7. Conception du système

Nous arrivons maintenant à la phase ultime de modélisation avec UML, après la modélisation des besoins puis l'organisation de la structure de la solution, la conception consiste à construire et à documenter précisément les classes, les tables et les méthodes qui constituent le codage de la solution.

III.7.1. Diagramme de classe

Les diagrammes de classes présentent un ensemble d'éléments de modèle statiques, leur contenu (structure interne) et leurs relations aux autres éléments. [37]

Une classe représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. Un attribut est un camp de classes, c'est-à-dire un type d'information contenu dans la classe.

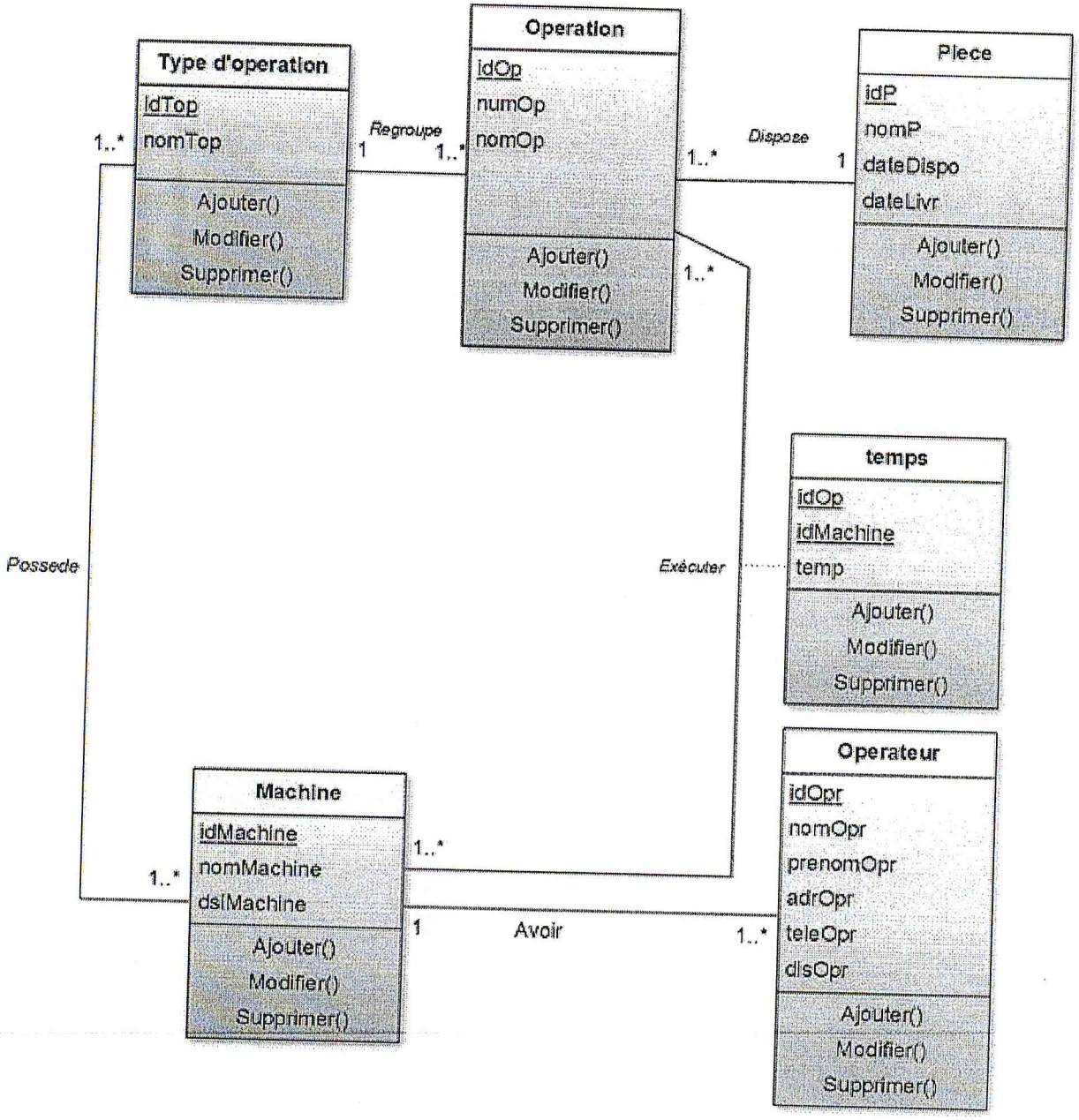

Figure III.20 : Diagramme de classes

III.7.1.a. Descriptions des données des classes

Nom de classe	Identifient	Attribut	Type
Type d'opération	IdTop	idTop	int
		nomTop	varchar
Machine	IdMachine	idMachine	int
		nomMachine	varchar
		disMachine	varchar
Opm	IdTop idMachine	idTop	int
		idMachine	int
Pièce	IdP	idP	int
		nomP	varchar
		dateDispo	date
		dateLivr	date
Opération	IdOp	idOp	int
		numOp	int
		nomOP	varchar
		idP	int
		idTop	int
Temp	IdOp idMachine	idOp	int
		idMachine	int
		temp	int
Operateur	IdOpr	idOpr	int
		nomOpr	varchar
		prenomOpr	varchar
		adrOpr	varchar
		teleOpr	int
		disOpr	varchar

Tableau III.11 : Descriptions des données

III.7.1.b. Dictionnaire des données des relations

Realtion	Collection	Identifiant	Cardinalité
Possede	Type d'opération Machine	idTop idMachine	1..* 1..*
Dispose	Piece Operation	IdP idOp	1 1..*
Regroupe	Operation Type d'operation	IdOp idTop	1..* 1
Exécuter	Operation Machine	IdOp idMachine	1..* 1..*
Avoir	Machine Operateur	idMachine idOpr	1 1..*

Tableau III.12 : Dictionnaire des données des relations

III.7.1.c. Passage Relationnelle :

Type d'opération (idTop, nomTop).

Machine (idMachine, nomMachine, disMachine).

Opm (idTop, idMachine).

Pièce (idP, nomP, dateDispo, dateLivr).

Opération (idOp, numOp, nomOp, idP*, idTop*).

Temp (idOp, idMachine, temp).

Opérateur (idOpr, nomOpr, prenomOpr, adrOpr, teleOpr, disOpr, idMachine*).

III.8. Conclusion

Dans ce chapitre, nous avons présenté le travail selon deux partie, dans la première partie on a abordé l'implémentation du noyau d'optimisation méta- heuristique (recherche Tabou modifiée). La seconde partie présente l''étude conceptuelle qui nous a permis de mettre en évidence les étapes nécessaires pour la création de l'application (ORdoRO). Cette étude nous a permis aussi de mettre en évidence les différentes classes du système.

Dans le chapitre suivant, nous allons implémenter et mettre en œuvre ce que nous avons proposé dans l'étude conceptuelle de notre système.

CHAPITRE 4

IMPLÉMENTATION

EXPÉRIMENTATIONS & RÉSULTATS

Inventer, c'est penser à côté
Albert EINSTEIN

IV.1. Introduction

Après avoir présenté dans le chapitre précédent la modélisation UML de notre application. L'objectif de ce chapitre est de présenter notre logiciel d'ordonnancement et donner une synthèse des résultats obtenus par application de la recherche Tabou au modèle de problème d'ordonnancement Job Shop Flexible.

Nous avons divisé ce chapitre en deux parties. La première est consacrée à la définition des outils de développement utilisés pour l'implémentation de notre système et à la présentation de l'application. Le but de la deuxième partie est de donner une synthèse des résultats obtenus par application de la recherche Tabou sur un ensemble de tests sur les "Benchmarks".

IV.2. Environnement de développement

IV.2.1. La structuration de données (MySQL)

Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. [38]

IV.2.2. Le langage de programmation choisi (JAVA)

Pour la réalisation de notre projet nous avons utilisé le langage de programmation Java

(SUN), sous l'éditeur NetBeans. (www.netbeans.org).

Nous avons choisi le langage JAVA pour les raisons suivantes :

- Nous sommes bien familiarisées avec les notions du langage JAVA;
- L'application peut s'exécuter sur n'importe quel système d'exploitation à condition d'avoir la machine virtuelle java installée sur la machine (portabilité).
- La disponibilité de la documentation et de l'assistance (forums).
- JAVA est un langage orienté objet qui met à la disposition du développeur plusieurs paquetages prêt à l'utilisation.

IV.3. Présentation d'ORdoRO

Notre outil, " ORdoRO "est un logiciel d'ordonnancement industriel.

IV.3.1. Interface d'accueil

Lors du lancement du logiciel,l'interface d'accueil apparait, portant le thème de notre projet nos noms respectifs, ainsi le bouton Entré (figureIV.1- 1).

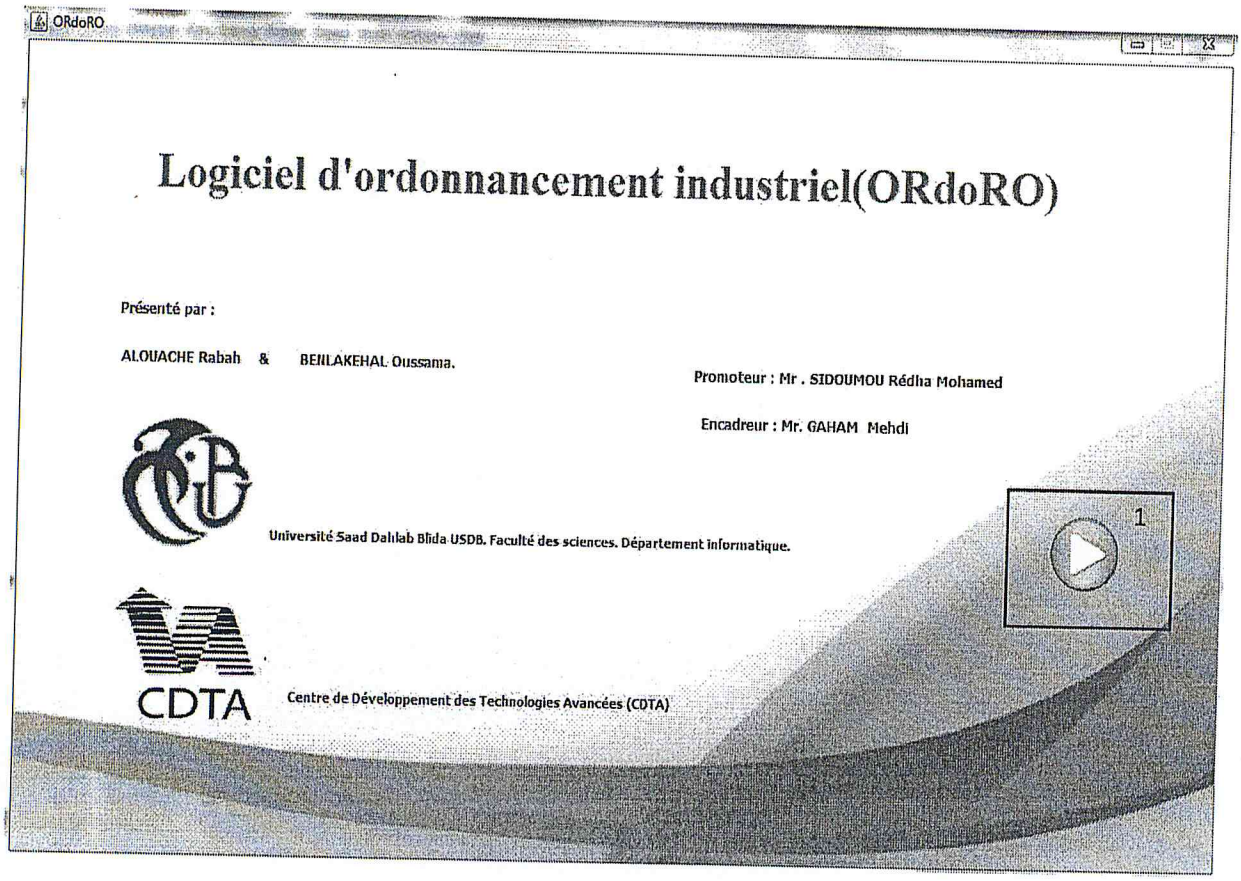

Figure IV.1. Interface d'accueil.

En appuyant sur le bouton (1) Entré, l'utilisateur passe à l'interface suivante (figureIV.2):

IV.3.2. Interface menu principal

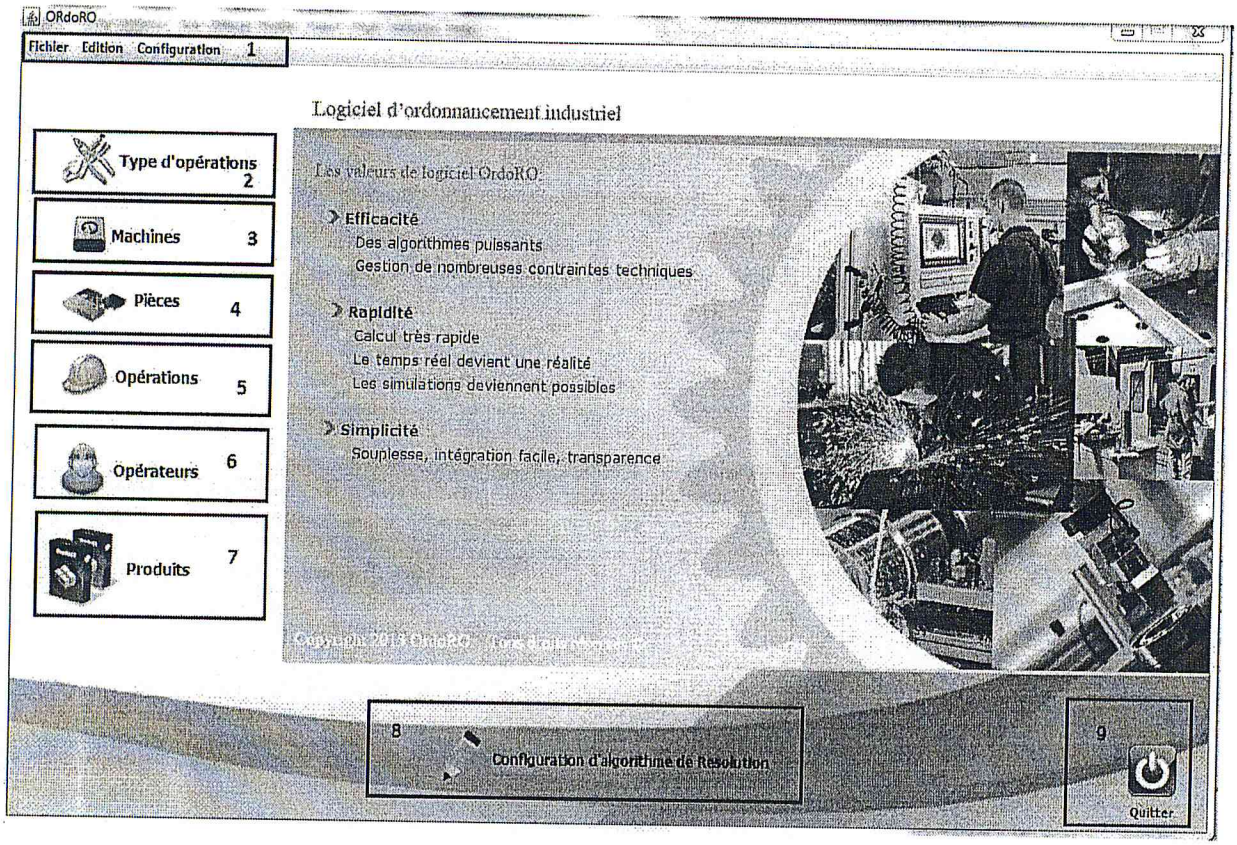

Figure IV.2. Interface menu principal.

Le menu de notre application contient neuf fonctionalités représentées par les boutons suivants :

1- La barre menu (Fichier ,Edition, Configuration).

2- Type d'opérations (Accéder à la fenetre type d'operations).

3- Machines (Accéder a la fenetre machines).

4- Pieces (Accédé a la fenetre pieces).

5- Opérations (Accédé a la fenetre opérations).

6- Opérateurs (Accédé a la fenetre opérateurs).

7- Produits (Accédé a la fenetre produits).

8- Configuration d'algorithme de resolution (Accédé a la fenetre configuration d'algorithme de resolution).

9- Quitter (Quitter l'application).

IV.3.3. Interface Type d'opération

La figure ci dessous représente un exemple de type d'opérations cette interface permet la gestion des types d'opérations en cliquant sur le bouton Type d'opérations (figure IV.2 -2) de l'interface menu principal, une nouvelle page sera ouverte, cette dernière affiche un formulaire qui montre les types d'opérations enregistrées dans la base de données. Les actions d'ajout, de modification, de suppression, d'impression et même le retour vers le menu principal peuvent être appliqués (figure IV.3 -3).

Il faut toute fois savoir qu'avant de cliquer sur les trois boutons ajouter, modifier et supprimer il est impératif que les champs de texte contiennent des informations correctes, dans le cas contraire une erreur survient.

Lorsque l'utilisateur clique sur une ligne du tableau (figure IV.3 -1). Les données de cette dernière seront affichées dans les champs de texte (figure IV.3 -2), afin de faciliter les actions de modification et de suppression.

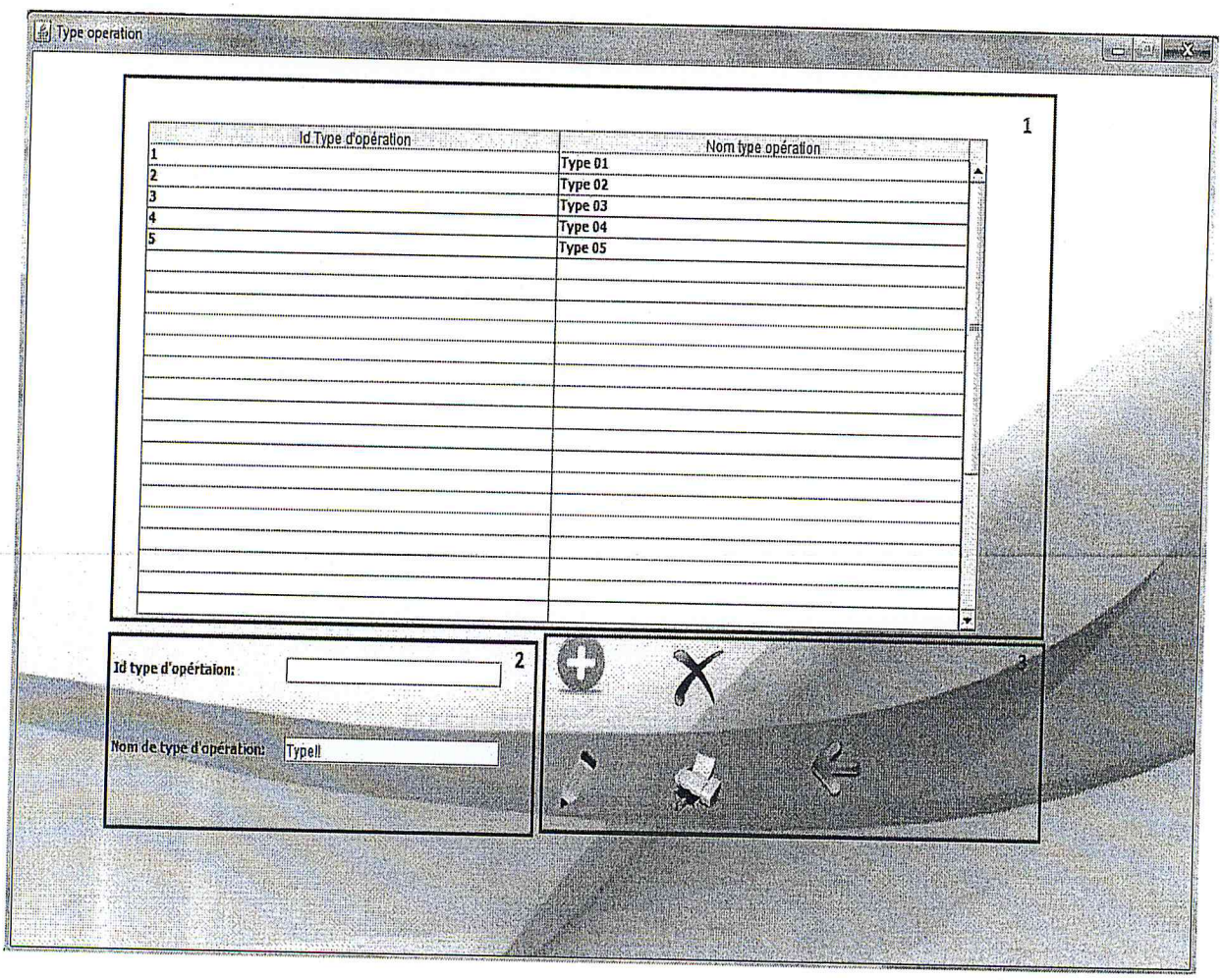

Figure IV.3. Interface Type d'opération.

IV.3.4. Interface Machine

La figure ci dessous représente un exemple de machine cette interface permet la gestion des machines en cliquant sur le bouton Machine (figure IV.2 -3). de l'interface menu principal, une nouvelle page sera ouverte, cette dernière contient un formulaire qui montre les machines enregistrées dans la base de données, Pour ajouter un type d'opération à une machine, il faut entrer l'idMachine enregistré, ou bien cliquer sur une ligne du tableau (figure IV.4 -1).A tout moment, l'utilisateur peut consulter les informations d'une des machines déjà enregistrées en cliquant sur la ligne du tableau représentant la machine souhaitée(figure IV.4 -1,2). Une fois les types d'opérations choisis, l'utilisateur valide sa modification en cliquant sur le bouton "ajouter types d'opérations"(figure IV.4 -5).

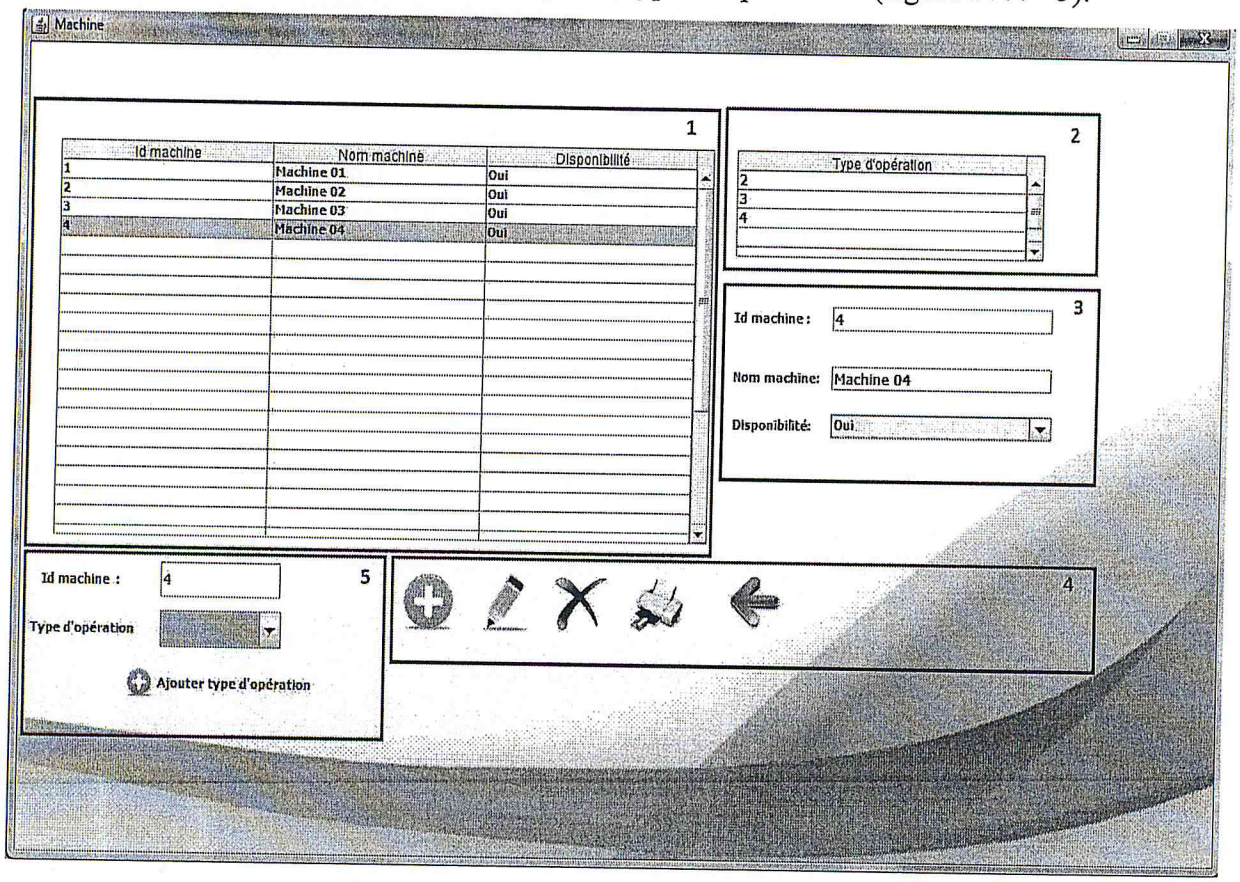

Figure IV.4. Interface machine.

IV.3.5. Interface Pièce

La figure ci dessous représente un exemple de pièce cette interface permet la gestion des pièces en cliquant sur le bouton Pièces (figure IV.2 -4) de l'interface menu principal, une nouvelle page sera ouverte, cette dernière contient un formulaire (figure IV.5 -1) qui montre les pièces enregistrées dans la base de données.

Lorsque l'utilisateur clique sur une ligne du tableau (figure IV.5 -1) les données de cette dernière seront affichées dans les champs de texte (figure IV.5 -2) afin de faciliter les actions de modification et de suppression.

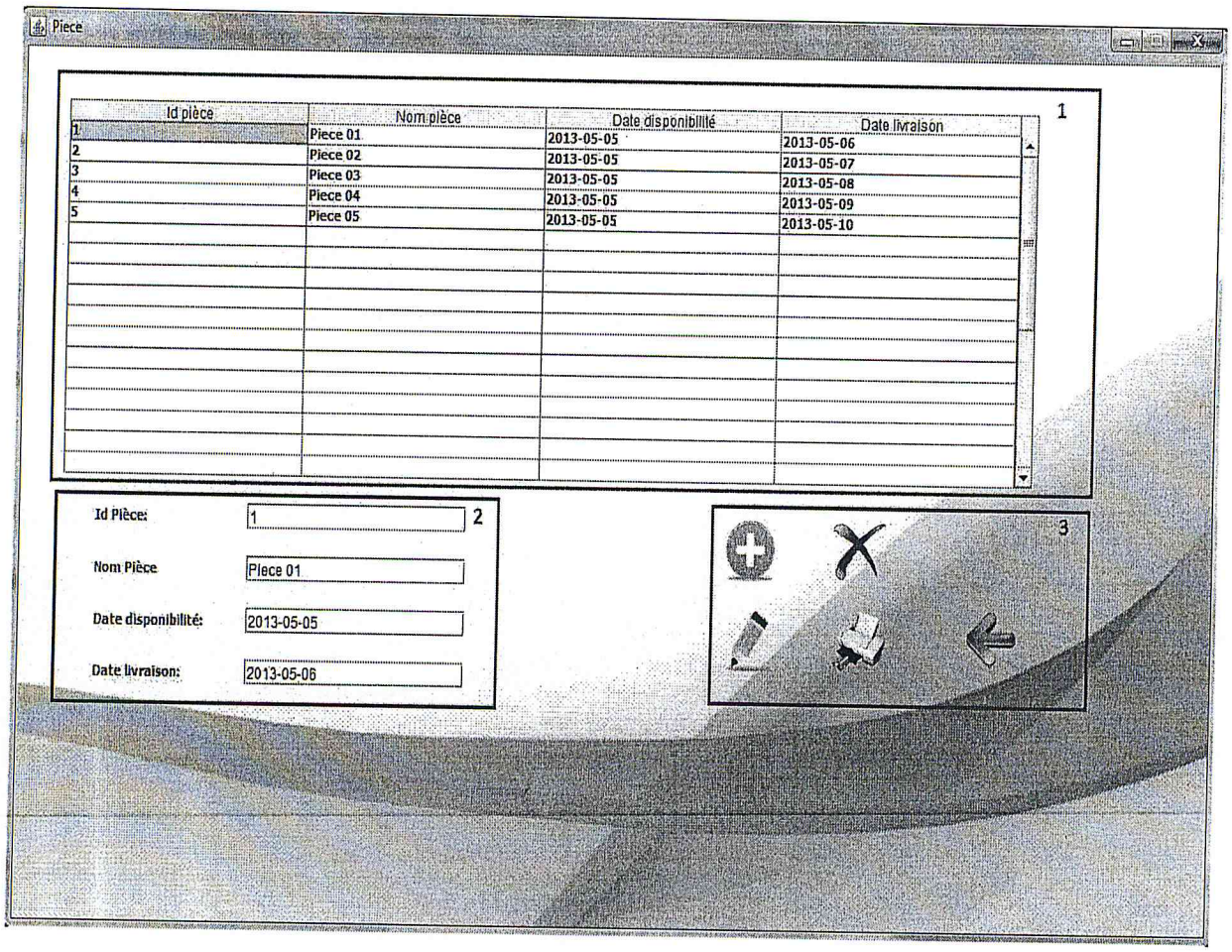

Figure IV.5. Interface pièce.

IV.3.6. Interface Opération

La figure ci dessous représente un exemple d'opération, cette interface permet la gestion des opérations en cliquant sur le bouton Opérations (figure IV.2 -5) de l'interface menu principal, une nouvelle page sera ouverte, cette dernière contient un formulaire (figure IV.6 -1) qui montre les opérations enregistrées dans la base de données. Les actions (figure IV.6 -4) d'ajout, modification, suppression, impression et même le retour vers le menu principal peuvent être appliqués.

Lorsque l'utilisateur clique sur une ligne du tableau (figure IV.6 -1) les données de cette dernière seront affichées dans les champs de texte (figure IV.6 -5) afin de faciliter les actions de modification et de suppression. Et afficher dans le second tableau (figure IV.6 -2) les différents temps d'exécutions sur différentes machines.

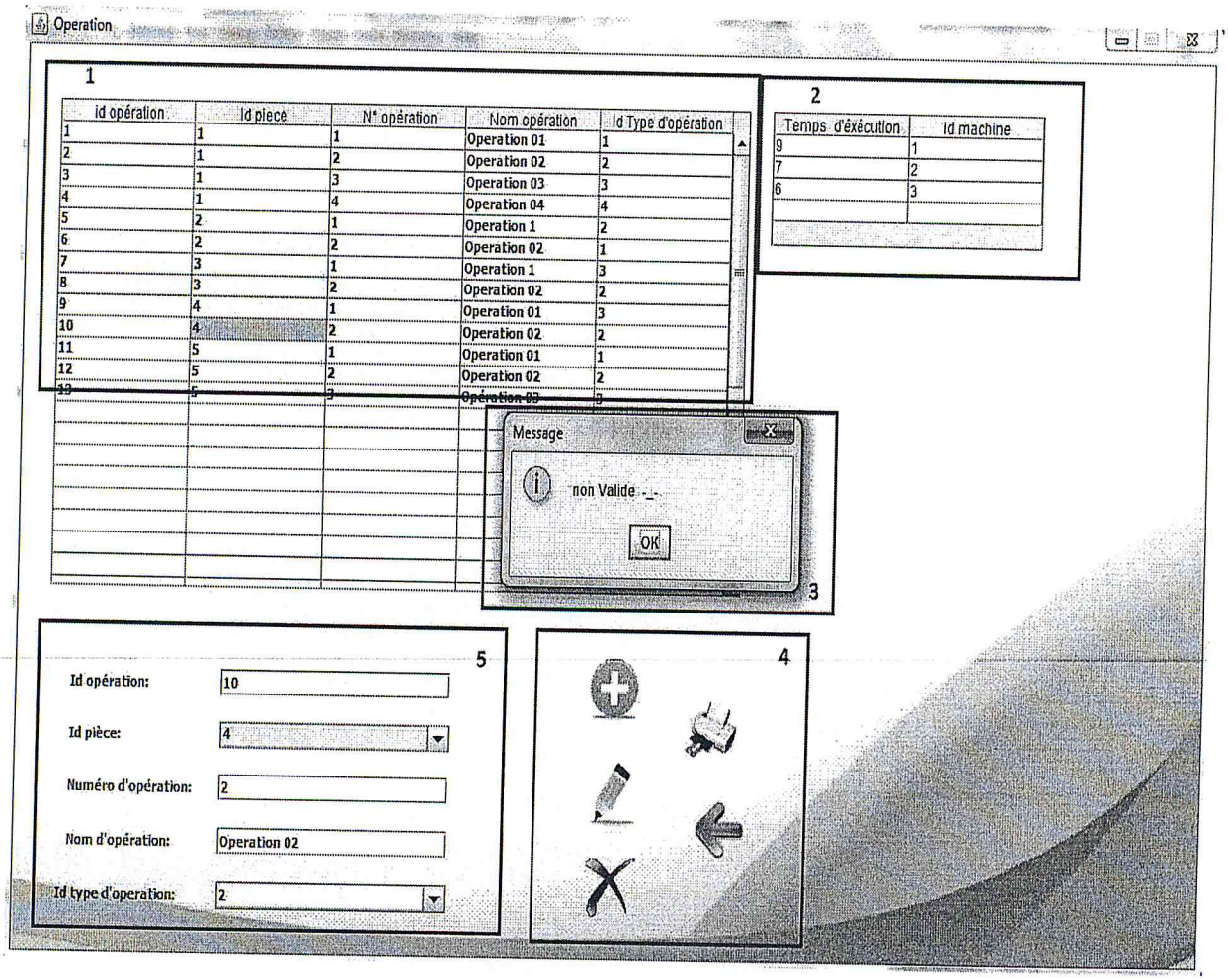

Figure IV.6. Interface d'opérations.

IV.3.7. Interface Opérateur

La figure ci dessous représente un exemple d'opérateurs, cette interface permet la gestion des opérateurs en cliquant sur le bouton Opérateurs (figure IV.2 -6) de l'interface menu principal, une nouvelle page sera ouverte, cette dernière contient un formulaire (figure IV.7 -1) qui affiche les types d'opérations enregistrées dans la base de donnée.

Lorsque l'utilisateur clique sur une ligne du tableau (figure IV.7 -1) les données de cette dernière seront affichées dans les champs de texte (figure IV.7 -3) afin de faciliter les actions de modification et de suppression.

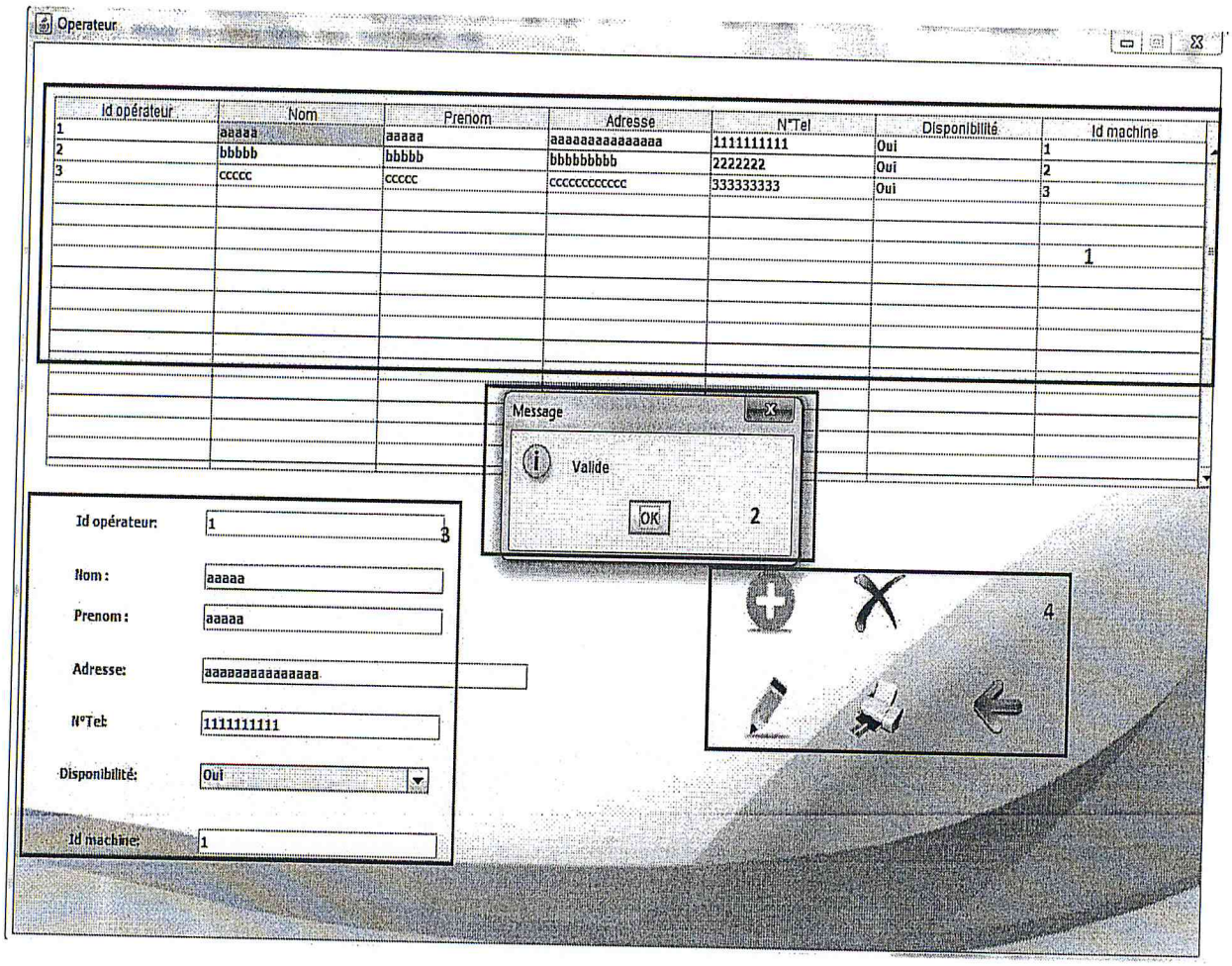

Figure IV.7. Interface opérateurs.

IV.3.8. Interface Produits

La figure ci dessous représente un exemple de produit en cliquant sur le bouton Produit (figure IV.2 -7) de l'interface menu principale, une nouvelle page sera ouverte, cette dernière contient un formulaire (figure IV.8 -1) qui montre toutes les données enregistrées dans la base de données et un champ de texte pour le nombre d'itération (figure IV.8 -3)et un bouton (figure IV.8 -4) qui génère le Benchmark, l'algorithme de la recherche tabou et affiche le diagramme de Gantt. Si l'utilisateur souhaite une recherche sur les données de tableau un champ de texte est proposé afin de faciliter la recherche, les données seront afficher sur le tableau(figure IV.8 -2),.

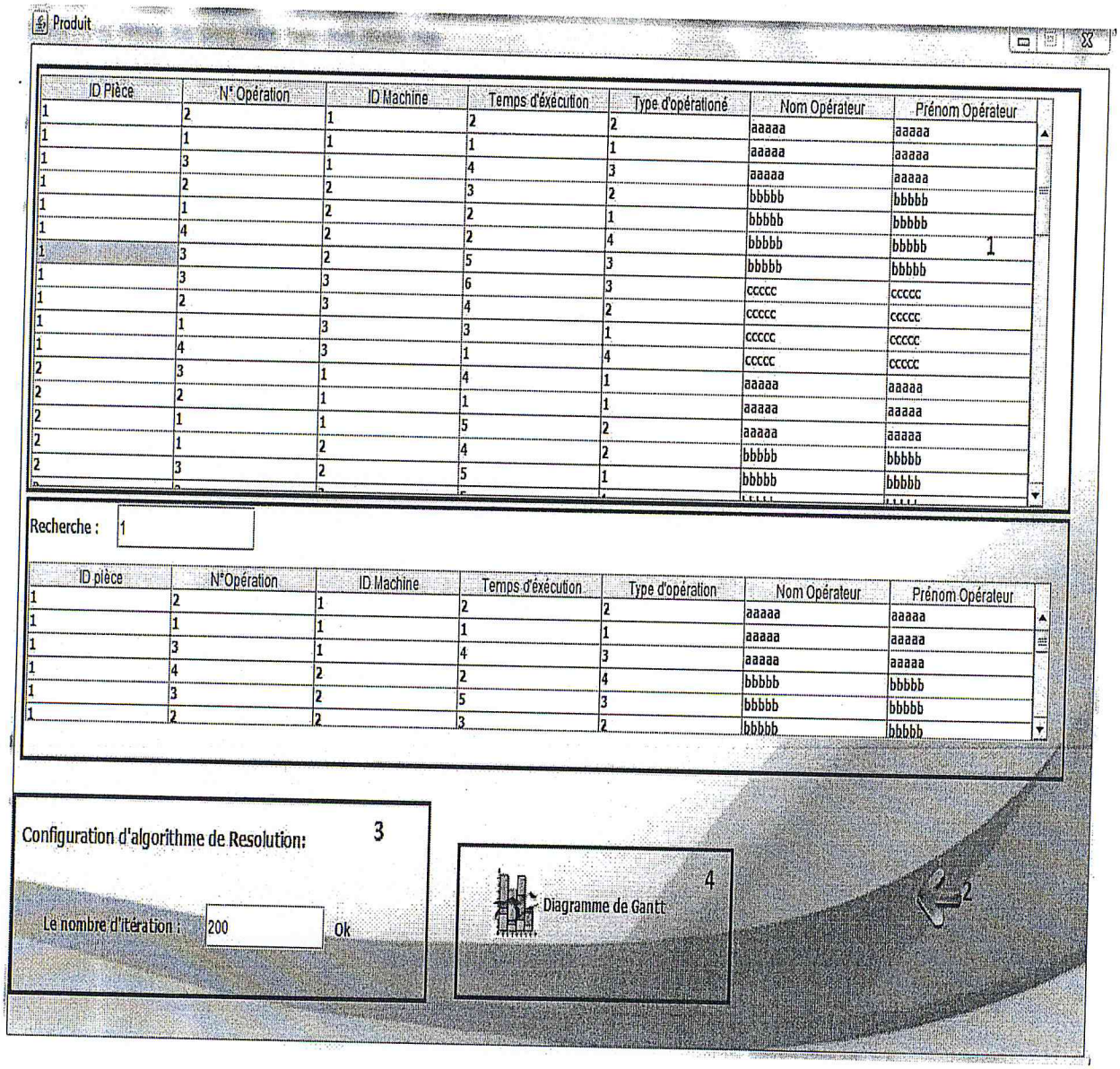

Figure IV.8. Interface produit.

IV.3.9. Interface configuration d'algorithme de résolution

La figure ci dessous représente des testes des Benchmarks en cliquant sur le bouton configuration d'algorithme de résolution (figure IV.2 -8) de l'interface menu principal, une nouvelle page sera ouverte, cette dernière contient un champ de texte pour le nombre d'itérations (figure IV.9 -1) ,une combo box ,un simple clique sur un teste de la liste permet de choisir le Benchmark à utiliser (figure IV.9 -2) . Un bouton qui génère l'algorithme de la recherche Tabou et afficher le diagramme de Gantt, après avoir saisi toutes les données (Figure IV.9 -2).

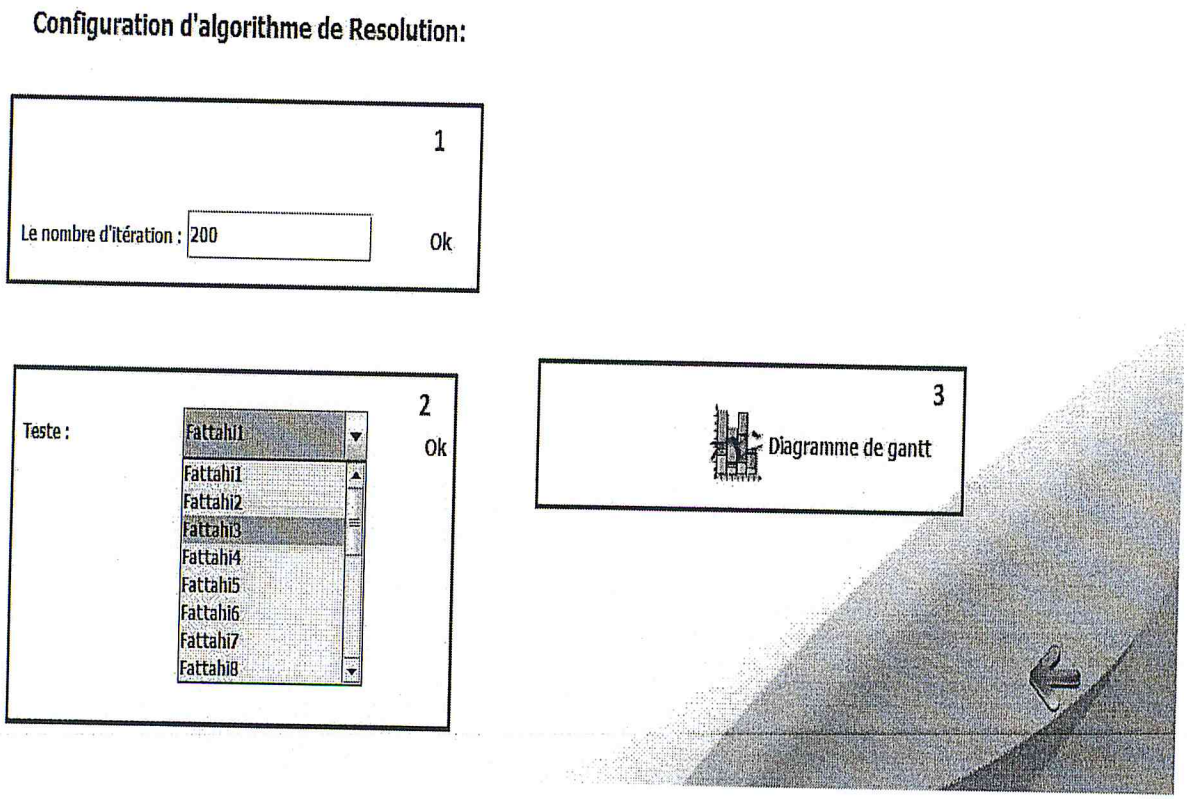

Figure IV.9. Interface de configuration d'algorithme de résolution.

IV.4. Application de la Recherche Tabou sur ORdoRO

Parmi les nombreuses expérimentations effectuées on a choisi quelques unes d'entre elles pour montrer l'influence des paramètres sur la recherche Tabou qu'on a implémenté.

IV.4.1. Le nombre d'itérations

La recherche Tabou à cause de son comportement stochastique, profite toujours de toute itération supplémentaire.

IV.4.2. Les Benchmarks

Pour tester l'efficacité d'une méthode de résolution, il est préférable d'utiliser des Benchmarks. Donc avant de parler des résultats de nos testes. Nous allons définir, dans ce qui suit, ce qu'est un benchmark. [6]

IV.4.2.1. Définition

Les Benchmarks sont des problèmes-types construits par plusieurs auteurs pour tester les performances des approches de résolution, en procédant surtout à des comparaisons sur les mêmes instances. Il s'agit d'instances théoriques (à l'instar des benchmarks des autres problèmes), formulées pour servir à l'étude de Job Shop Flexible [6].

Un test réalisé par ORdoRO

- Teste réalisation avant d'utiliser l'algorithme de recherche Tabou (Modifié).

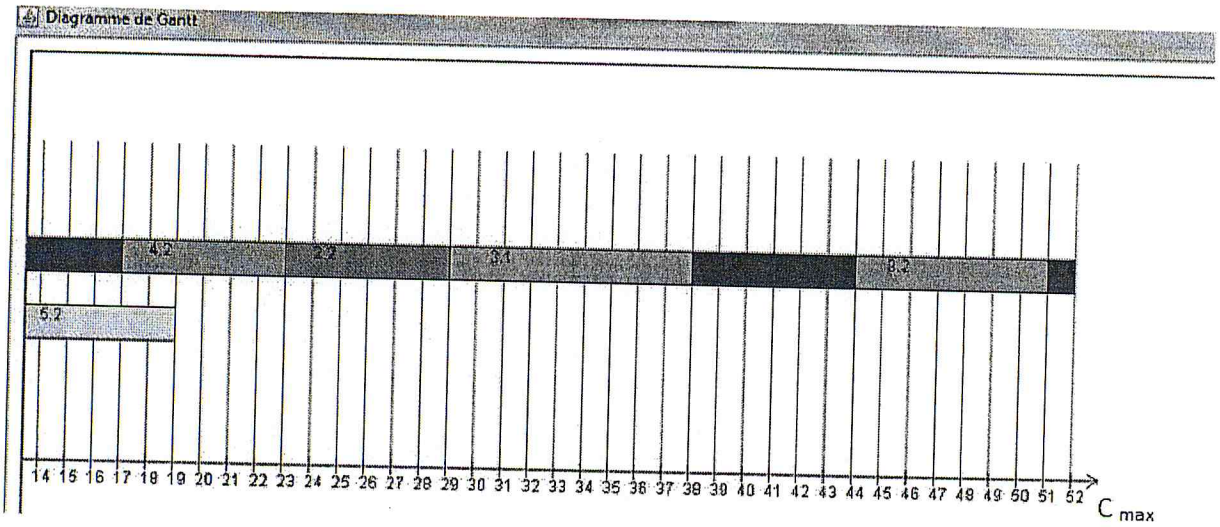

Figure IV.10. Diagramme de Gantt représente une solution obtenu avant d'utilisé notre recherche Tabou.

- Test réalisé avec l'algorithme Tabou (modifié).

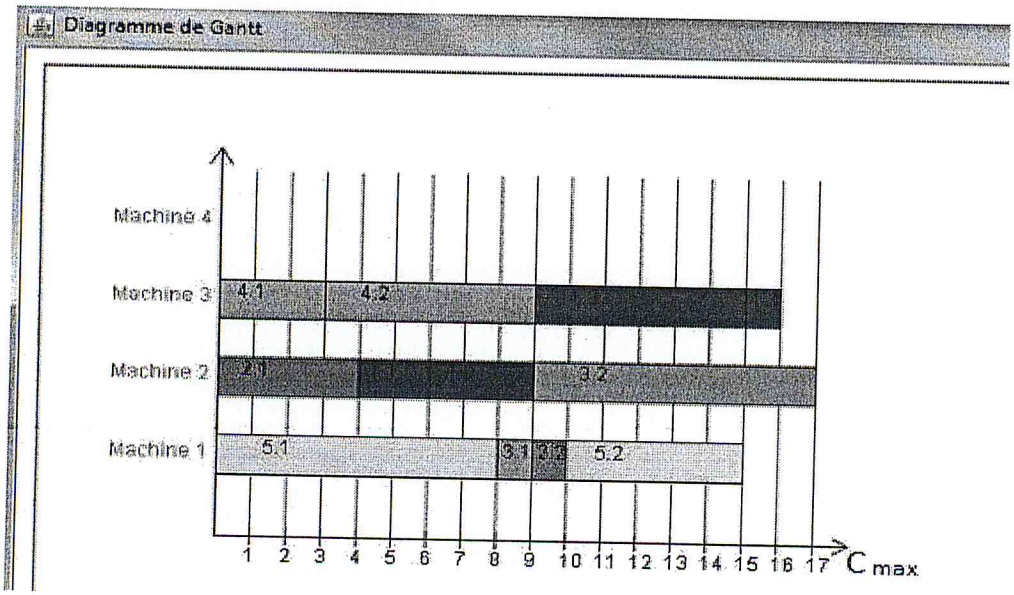

Figure IV.11. Diagramme de Gantt représente une solution obtenu après d'utilisé notre recherche Tabou.

Apres exécution d'ORdoRO avec notre algorithme de recherche Tabou (modifié), le diagramme de Gantt (Figure IV.11) montre largement l'optimisation de Makespan comparé à l'exécution d'ORdoRO sans cet algorithme (Figure IV.10).

IV.4.3. Les résultats

Une comparaison de nos résultats obtenus par les différentes approches implémentées avec d'autres résultats obtenus par des chercheurs est représentée par le tableau (Tableau IV.1) suivant.

Avec un nombre d'itérations=500.

Problème	AG	RS	RT	RT (Modifié)
SFJS1	66	66	66	66
SFJS2	107	107	107	107
SFJS3	221	221	221	221
SFJS4	355	355	390	355
SFJS5	119	119	137	119
SFJS6	320	320	320	320
SFJS7	397	397	397	397
SFJS8	253	253	253	253
SFJS9	210	215	215	210
SFJS10	516	516	617	516
MFJS1	468	488	548	470
MFJS2	448	478	457	475
MFJS3	466	599	606	548
MFJS4	554	703	870	726
MFJS5	514	674	729	590
MFJS6	634	856	816	726
MFJS7	881	1066	1048	1095
MFJS8	891	1328	1220	1070
MFJS9	1094	1148	1124	1105
MFJS10	1286	1546	1737	1505

Tableau IV.1 : Résultats obtenus avec nombres d'itérations=500.

D'après le tableau4.1 ci-dessus, on remarque qu'avec un nombre d'itérations =500 nous avons obtenu des résultats intéressants par rapport au temps d'exécution.

Remarque : Pour des raisons d'espace dans le tableau précédent on a utilisé les abréviations suivantes :

AG : Les résultats obtenus par Algorithme Génétique sur le problème d'ordonnancement Job Shop Flexible. [26]

RS : Les résultats obtenus par Recuit Simulée sur le problème d'ordonnancement Job Shop Flexible. [26]

RT : Les résultats obtenus par Recherche Tabou sur le problème d'ordonnancement Job Shop Flexible. [26]

RT (Modifié): Les résultats obtenus par notre Recherche Tabou sur le problème d'ordonnancement Job Shop Flexible.

Lorsqu'on compare nos résultats avec les résultats trouvés par d'autres chercheurs, nous sommes arrivés à l'affirmation que notre travail abouti à un résultat satisfaisant.

IV.5. Conclusion

Dans ce chapitre nous avons présenté brièvement l'environnement de programmation. Nous avons aussi décrit l'interface d'ORdoRO avec toutes les fonctionnalités qu'elle permet d'accomplir. Les résultats de différentes expérimentations réalisées sur un ensemble de benchmarks. L'objectif est d'explorer les performances des stratégies qu'on a utilisées, d'un côté, et de valider notre implémentation des méta-heuristiques adaptées à la résolution du problème de Job Shop Flexible d'un autre coté.

CONCLUSION GÉNÉRALE

Conclusion Générale

Les problèmes de l'ordonnancement sont présents dans tous les secteurs de l'économie et constituent une fonction importante en gestion de production. Un problème d'ordonnancement consiste à allouer dans le temps des taches à des ressources qui existent en quantité limitée, tout en satisfaisant un ensemble de contraintes.

Le problème d'ordonnancement de type Job Shop Flexible est connu dans la littérature comme un problème NP-difficile [5]. De ce fait, l'utilisation de méthodes exactes en vue de l'obtention de solutions optimales semble non réaliste. Le recours à des méthodes approchées comme les heuristiques est donc devenues incontournable. Parmi ces méthodes, le paradigme des Méta-heuristiques s'impose comme une approche très prometteuse. En effet, en plus de leur adaptabilité aux différents problèmes combinatoires, les méta-heuristiques ont l'avantage de ne parcourir qu'une faible fraction de l'espace de solution pour parvenir à une solution acceptable. Ce qui réduit nettement les temps de calcul.

L'objectif de notre travail est la conception et la réalisation d'un logiciel d'ordonnancement industriel basé sur le model job shop flexible et le développement d'un noyau algorithmique de résolution efficace pour ce problème. Pour ce faire nous avons développé une recherche Tabou modifiée basée sur une nouvelle structure de voisinage qui permet la détermination de l'ensemble des solutions voisines à partir de représentations codées des solutions non pas à partir de structures de données complexes. Cette approche permet particulièrement de réduire les temps de calcul ce qui est important pour les logiciel applicatifs d'ordonnancement industriel.

Dans ce travail, nous avons traité le problème d'ordonnancement de type Job Shop Flexible, ainsi, nous avons introduit différentes méthodes de résolution exactes et approchées pour ce problème.

Après, nous avons décrit, d'une manière détaillée, l'approche Tabou implémentée, et les aspects conceptuels du logiciel d'ordonnancement(ORdoRO).

Enfin, nous avons présenté notre logiciel, plusieurs expérimentations sont effectuées sur des Benchmarks retenus comme échantillons de tests pour prouver l'efficacité de l'approche Tabou proposée et implémentée.

Ce travail nous a permis de constater que la méthode de la recherche Tabou est très intéressante pour la résolution du problème d'ordonnancement de type Job Shop Flexible.

Son application à notre problème a fait émerger plusieurs résultats importants:

-L'emploi d'une mémoire trop courte ou trop langue donne pour plusieurs instances de mauvaises solutions. La taille trop courte risque de favoriser le cyclage .Alors que, la taille trop grande peut bloquer la recherche en mettant tous les mouvements tabous.

- Les résultats ont permis de conclure l'efficacité de la recherche Tabou, qui reste à améliorée en terme d'efficience

-Pour des raisons de simplification, nous avons appliqué le makespan comme critère d'optimisation. Toutefois, ce critère peut ne plus être performant dans certaines situations.

- L'emploi d'une stratégie multicritères semble être plus bénéfique.

-A cause de son comportement, « profite » toujours de toute itération supplémentaire.

A l'issue du travail présenté dans ce mémoire, différentes pistes formant les directions futures à cette recherche, sont envisagées:

-Il est certain que cette étude présente des limites qui doivent être soulevées .Tout d'abord, l'échantillon des problèmes qui a servi à comparer les méthodes est faible. En ce sens, la résolution d'un plus grand nombre de problèmes, mais aussi des problèmes de taille et de natures différentes, serait souhaitable pour compléter cette analyse, Le recours à une telle analyse peut mieux renforcer les résultats obtenus.

-Une autre direction de recherche envisagée à travers ces études consiste à l'application des Meta-heuristiques à l'ordonnancement intégré avec d'autres systèmes : la maintenance, la chaine logistique, etc.

Bibliographie

[1] Vincent GIARD, «Gestion de la production» 2eme édition, Economica, Paris, 1998.

[2] Letouzey A., «Ordonnancement interactif basé sur des indicateurs : Applications à la gestion de commandes incertaines et à l'affectation des opérateurs», Thèse de Doctorat, Institut National Polytechnique de Toulouse, 2001.

[3] GOTHA, «Les problèmes d'ordonnancement», RAIRO-Recherche Opérationnelle 27, page 77-150, 1993.

[4] LOPEZ P. et ROUBELLAT F ., «Ordonnacement de la production», Hermes Sciences, IC2 Productique, 2001.

[5] VACHER J-PH ., «Un système adaptif par agents avec utilisation des algorithmes génétiques multi-objectifs : Application à l'ordonnancement d'atelier de type job-shop NxM», thèse de Doctorat, Université du Havre, 2000.

[6] BENRABAH Nessrine & REKKACHE Omar, «Recuit simulé pour le problème d'ordonnancement de type Job Shop Flexible», Projet de fin d'études, En vue de l'obtention du diplôme de master en Engineering en Recherche Opérationnelle, Université des Sciences et de la technologie Houari Boumediene, 2012.

[7] SELT O ., «Méta-heuristiques, pour résoudre les problèmes d'ordonnancement des tâches sur des machines parallèles», thèse de magistère, Université de M'SILA, 2008.

[8] I. Saad, H. Boukef, P. Borne, « The comparison of criteria aggregative approaches for the multi-objective optimization of flexible job-shop scheduling problems ». Fourth Conference on Management and Control of Production and Logistics, MCPL 2007, Sibiu, pp. 603-608, 2007.

[9] BRUCKER P, JURISH B. KRAMER A., «Complexity of scheduling problems with multi-prupose machines», Annals of Operations Research, 1997.

[10] HAO J-K., GALINIER Ph., MICHEL H., «Méta-heuristiques pour l'optimisation combinatoire et l'affectation sous contraintes», Revus d'intelligence artificielle, 1999.

[11] FERRAH Djahida et BABA Kahina, «Approche évolutionnaire pour la résolution du problème d'ordonnancement de type Job Shop Flexible Dynamique», Mémoire pour l'obtention d'un diplôme d'ingénieur d'état en informatique. Option : Système d'information, Université Saad Dahlab, Blida, 2010.

[12] Brucker P., Schlie R, « Job shop scheduling with multi-purpose machines», Computing 45, 369–375.1990.

[13] Brandimarte P.J« *Routing and scheduling in a flexible job shop by tabu search*», Annals of Operations of Research 41, 157-183, 1993.

[14] Paulli J. «*A hierarchical approach for the FMS scheduling problem*». European Journal of Operational Research, 86, 32-42, 1995.

[15] Barnes J.W., Chambers J.B. « *Flexible job shop scheduling by tabu search*» Graduate Program in Operations and Industrial Engineering, The University of Texas at Austin, Technical Report Series, ORP96-09, 1996.

[16] Fisher H., Thompson G.L., «*Probabilistic learning combinations of local jobshop scheduling rules* ». In Industrial Scheduling, J.F. Muth and G.L. Thompson (eds.), Prentice-Hall, Englewood Cliffs NJ. 225-251, 1963.

[17] Lawrence S «*Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques*». Graduate School of Industrial Administration, Carnegie-Mellon University, 1984.

[18] Dauzère-Pérès S., Paulli, J., «Solving the general job-shop scheduling problem.Management». Report Series, vol. 182. Erasmus University Rotterdam, Rotterdam School of Management, Rotterdam, 1997.

[19] Mastrolilli M., Gambardella L. M., «*Effective neighbourhood functions for the flexible job shop problem*», Journal of Scheduling 3, 3–20, 2000.

[20] Kacem I., Hammadi S., Borne P., «*Approach by localization and multiobjective evolu-tionary optimization for flexible job shop scheduling problems*», IEEE Transactions on Systems, Man and Cybernetics, Part C 32(1), 408–419 , 2002.

[21] Guohui Zhang, Yang Shi, Liang Gao «*AGenetic Algorithm and Tabou Search for Solving Flexible Job Shop Schedules* » The State Key Laboratory of Digital Manufacturing Equipment and Technology Huazhong University of Science & Technology, Wuhan, 430074, China : gaoliang@mail.hust.edu.cn.

[22] Geoffrey VILCOT, « *Algorithmes approchés pour les problèmes d'ordonnancement multicritères de type Job Shop Flexible et Job Shop multiressource* »
Thèse de Doctorat, Université François-RABELAIS, Tours, Ecole doctorale : Santé, sciences et technologies, 2007.

[23] Gao J., Gen M., Sun L., Zhao X., «*A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems»,* Computers and Industrial Engineering 53, 149-162, 2007.

[24] Pezzella F, Morganti G, Ciaschetti G (2008). «A genetic algorithm for the Flexible Job-shop Scheduling Problem». Computers and Operations Research, 35(10), pp. 3202-3212.

[25] J.B. Chambers, « Classical and flexible job shop scheduling by tabu search ». Thèse de Doctorat, Université du Texas, Austin, USA, 1996.

[26] Fattahi, P, M. Saidi Mehrabad and F.Jolai, 2007. «Mathematical modeling and heuristic approaches to flexible job shop scheduling problems». J. Intel. Manuf, 18: 331-342.

[27] Mohammad R. Raeesi N., Ziad Kobti, 2012. «A memetic algorithm for Job Shop scheduling using a critical-path-based local search heuristic». School of computer Science, University of Windsor. Windsor, ON N9B-3P4, Canada.

[28] Luca Maria Gambardella, Monaldo Mastrolilli « *Effective Neighborhood Functions for the Flexible Job Shop Problem* » IDSIA : **I**nstituto **D**all Molle di **S**tudi sull'**I**ntelligenza **A**rtificiale C.so Elvezia 36, 6900 Lugano, Switzerland {monaldo, luca}@idsia.ch, http://www.idsia.ch , 1999.

[29] Duvivier D, « *Etude de l'hybridation des méta-heuristiques, Application à un problème d'ordonnancement de type Job Shop* », Thèse de Doctorat, Université du littoral côte d'Opale, LIL, Calais 2000.

[30] K.Schmidt, «*Using Tabu Search to Solvethe Job Shop Scheduling Problem with Sequence Dependent Stup Times »,* Communication, 2001.

[31] J.STEFFE « Cours UML», ENITA de Bordeaux ,mars 2005.

[32] Muller,S « Modélisation objet avec UML », EYROLLES, 2002.

[33] Bres,B « Atelier de génie logiciel », Masson ,1993.

[34] P, Gérard. «Processus de développement logiciel », Université de Paris ,13 2008.

[35] Site officiel de Java, «Définition d'un scénario de diagramme de séquence» [En ligne] (Page consulter le : 10/03/2013). http://www.istantic.com/v2/programmation/Java/Generalites/Generalites.htm.

[36] Site officiel d'éclipse, «Définition de diagramme de séquence» [En ligne] (Page consulter le : 12/03/2013). http://www.eclipse.org .

[37] Site officiel des objets, «Définition de diagramme de classe » [En ligne] (Page consulter le : 15/03/2013). http://support.objecteering.com/objecteering6.1/help/fr/objecteering_uml_modeler/diagrams/class_diagrams.htm.

[38] Site officiel de MySQL, «Définition de MySQL» [En ligne] (Page Consulter le : 19/04/2013). http://www.mysql.com/.