République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire présenté pour l'obtention de diplôme de master
Option : Génie des Systèmes Informatiques (GSI)

Thème

Développement d'une application de pilotage dynamique des systèmes de production intégrant la technologie code à barres.

Président : Mme. Mancer Yacemine

Présenté par :
BOUHAOUYA Rahma
BELALI Chafia

Promoteur : Mr. DERRAR Hacene
Examinateur : Mr. Chemchem
Encadreur : Mr. GAHAM Mehdi

Organisme d'accueil : Centre de Développement des Technologies Avancées (CDTA)

2015-2016

Remerciement

Nous tenons à remercier la force divine, grâce à laquelle nous pouvons être guidés vers la bonne destination de croyance et de volonté.

Et tenons à remercier vivement tous ceux qui nous a orientées et nous a encouragées ,et pensons en particulier à notre encadreur GAHAM MEHDI pour sa confiance à nous sur ce travail et sa collaboration et son aide au cours du stage, et notre promoteur DERRAR HACENE d'avoir a bien suivi et dirigé notre travail et de faire profités de son savoir.

Nous n'oublions pas de remercier en particulier tous nos amis qui ont joué un rôle dans l'accomplissement de ce travail. Et précieux remerciements vont au président et membres de jury pour l'honneur qu'ils nous font en acceptent de juger ce travail.

De plus, nos remerciements seraient incomplets, si nous ne faisons pas mention de nos parents pour leur soutien, patience et sacrifice et que dieu le tout puissant les protègera.

Résumé

Dans la pratique industrielle, les critères de performance que sont les délais, la qualité et les coûts se traduisent au niveau opérationnel par des décisions à prendre sur l'affectation des ressources et les instants de début d'exécution des opérations. Ces décisions peuvent être prises seulement en temps réel au fur et à mesure de l'avancement de la production. Ceci ne peut être réalisé sans la mise en place de solutions informatiques permettant la gestion de l'ensemble des étapes inhérentes à ce processus.

C'est dans ce contexte précis que s'inscrit notre travail de fin d'études qui a pour objectif la conception et le développement d'un logiciel de pilotage dynamique du système de production basé sur un module d'ordonnancement. Ce module est basé sur l'implémentation d'un algorithme génétique afin de trouver une solution approchée au problème d'ordonnancement des tâches dans un atelier de type Job Shop Flexible. Ce problème est considéré comme une problématique difficile, sa complexité est de type NP-complet au sens fort. La fonction objectif est minimiser est le makespan représente avec des contraintes diverses qu'il faut absolument respecter pour aboutir à une solution réalisable et proches de l'optimum en un temps acceptable. Et enfin pour le suivi du processus de fabrication nous intégrons dans notre solution l'information temps réel en utilisons la technologie code à barre.

Mots clés : Ordonnancement, Job-shop, flexible, Algorithmes génétiques, Méta-heuristique, Makespan.

Résumé

Summary

In industrial practice, the performance criteria that are timely, quality and cost are translated at the operational level by decisions on the allocation of resources and the start times of execution. These decisions can only be taken in real time as and when the progress of the production. This can be achieved without the implementation of IT solutions for the management of all the steps involved in this process.

It is in this specific context that our work graduation which aims to design and development of a dynamic control software system of production based on a scheduling module. This module is based on the implementation of a genetic algorithm to find an approximate solution to the problem of scheduling tasks in a kind of workshop Flexible Job Shop. This problem is considered a difficult problem, its complexity is NP-complete type in the full sense. The objective function is to minimize the makespan is with the various constraints that must be strictly observed to achieve a workable solution and close to the optimum in an acceptable time. And finally to monitor the manufacturing process we integrate our solution in real-time information from the bar code technology.

Keywords: Scheduling, Job-shop, flexible, Genetic Algorithms, Metaheuristic, Makespan.

Résumé

الملخص

في الممارسة العملية الصناعية، تترجم معايير الأداء التي تتمثل في الوقت، الجودة والتكلفة في ورشة التوجيه أين يتم فيها اتخاذ قرارات على المستوى العملي، حيث تختص بتوزيع الموارد ولحظات بداية تنفيذ العمليات. حيث هذه القرارات لا يمكن اتخاذها إلا في الوقت الحقيقي، وعندما تقدم الإنتاج.

لهذه المشكلة، قد أجريت هذه المذكرة في سياق التعاون الصناعي.

يتمثل هدفنا في عمل برنامج للتحكم الديناميكي لنظم الإنتاج على أساس وحدة الجدولة،حيث تستخدم هذه الوحدة الخوارزميات الجينية (الأدلة العليا) لعلاج هذا النوع من جدولة الو رشات من نوع وظائف المتجر المرنة حيث تعتبر في البحوث القديمة كمشكل صعب للحل في مجال التركيبية و المعالجة على وجه خاص معقد من نوع NP-كاملة مستوى, makespan يمثل هدفنا الذي يقوم على تقليل من تعقد هذه الخوارزمية،تحوي هذه الخوارزمية على قيود مختلفة يجب مراعاتها بدقة للتوصل إلى حل عملي يجب مراعاتها بدقة للتوصل إلى حل عملي وعلى مقربة من الأمثل في وقت قصير جدا. في الأخير و من اجل متابعة دورة حياة برنامجنا للتصنيع من خلال استخدام المعلومات المتحصل عليها في الوقت الحقيقي باستخدام تكنولوجيا رمز البار كود.

مفتاح الكلمات :.الجدولة، وظائف المتجر المرنة، الخوارزميات الجينية ، الأدلة العليا

Chapitre 2 : Les méthodes de résolution « Les méta-heuristiques »

Chapitre 3 : Conception et réalisation

Chapitre 4 : Implémentation, expérimentations, résultats et discussions

Liste des figures

Liste des tableaux

Introduction Générale

Introduction générale

Les entreprises manufacturières évoluent aujourd'hui dans un environnement économique de forte concurrence, fortement dynamique et incertain. La globalisation des marchés, la concurrence accrue, la complexification des produits, leur durée de vie, toujours plus courte, impose à ces entreprises de disposer d'un haut niveau de performance et de flexibilité de l'outil de production associé à un haut niveau d'agilité et de réactivité des systèmes de gestion de la production.

Traditionnellement, les problèmes de conduite opérationnelle dans les industries manufacturières sont fortement causés par les problèmes d'ordonnancements et plus particulièrement les problèmes de type job shop. Le job shop consiste à allouer dans le temps des taches à des ressources qui existent en quantité limitée, tout en satisfaisant un ensemble de contraintes. **[1]**

Elargissant le spectre applicatif du job shop, le job shop flexible permet à une opération d'être réalisée avec plusieurs ressources avec des temps de généralement différents. Ainsi, le job shop flexible représente une version plus complexe que le job shop du fait qu'il implique la résolution, non pas seulement du problème d'ordonnancement des opérations sur les ressources, mais aussi la détermination de l'affectation optimale de ces dernières.

Par ailleurs, les problèmes de pilotage opérationnel et d'ordonnancement au sein des entreprises manufacturières sont rarement statiques. De nombreux facteurs tel que les pannes machines, l'arrivée de nouveaux produits, l'annulation d'une commande, les reprises de produits dues à des défauts de qualité, rendent le contexte fortement dynamique et incertain, ce qui accentue fortement la complexité du problème.

Les formulations idéalistes et simplifiées peuvent s'avérer très difficiles à traiter. Pour trouver un optimum, le seul moyen est souvent de faire une recherche exhaustive sur l'ensemble des solutions réalisables. La plupart des problèmes d'ordonnancement sont classés NP-difficiles. L'appartenance à la classe des problèmes de ce type signifie qu'on ne pourra probablement jamais résoudre ce problème d'une manière optimale. Il s'ensuit que la plupart des problèmes de taille industrielle (plusieurs milliers de tâches, ressources complexes et contraintes spécifiques) sont impossibles à résoudre de manière exacte. Par conséquent, les méthodes approchées constituent une alternative intéressante pour leur résolution **[1]**.

Dans le cadre de ce mémoire, nous nous focalisons sur l'ordonnancement d'atelier de type Job Shop Flexible par l'application de l'algorithme génétique, connu pour être un des meilleurs algorithmes pour la résolution de ce type de problème. Son efficacité a été particulièrement approuvée dans le domaine de la résolution des problèmes combinatoires complexes.

L'objectif de notre travail est de concevoir et développer d'un logiciel de pilotage dynamique des systèmes de production, basé sur un module d'ordonnancement type job shop flexible, par l'implémentation d'un algorithme génétique basé sur une fonction d'évaluation de makespan qui inclut le temps de transport entre les zones. Aussi, pour un meilleur suivi du processus de fabrication, nous avons intégré dans notre solution la technologie code à barre.

Notre mémoire est structuré en quatre chapitres. Dans le premier chapitre, nous ferons un état de l'art sur les travaux et les concepts principaux liés à la production et l'ordonnancement.

Le deuxième chapitre est consacré à la présentation des différentes méthodes d'optimisation pour la résolution le problème Job Shop Flexible allant des méthodes exactes aux méthodes approchées.

Dans le troisième chapitre, nous décrivons la méthode d'optimisation par l'algorithme génétique ainsi que les phases de conception du logiciel d'ordonnancement.

Dans le quatrième chapitre, nous présentons notre logiciel à travers des expérimentations effectuées sur des Benchmarks retenus comme échantillon de test afin de montrer l'efficacité de l'algorithme génétique implémenté.

Enfin, nous finalisons notre mémoire par une conclusion générale et nous proposerons quelques perspectives.

Chapitre 1 : Ordonnancement des activités de production.

1.1. Introduction

L'évolution des exigences du marché, qui se traduisent par une diversification des produits manufacturés, une réduction des délais de fabrication, une diminution des prix, nécessite le recours à l'informatique afin de gérer la diversité des informations à traiter et accélérer le processus de prise de décision.

Ce chapitre présente les notions fondamentales relatives à la gestion de production de manière générale et détaille par la suite la notion d'ordonnancement et les problèmes y relatifs.

1.2. Définition de la gestion de production

La gestion de la production consiste en la recherche d'une organisation efficace de la production des biens et services.

La gestion de production consiste donc à l'obtention d'un produit donné dont les caractéristiques sont connues en mettant en œuvre un minimum de ressources. Donc l'objectif majeur de la gestion de production est de pouvoir livrer un produit avec la qualité adéquate dans les meilleurs délais en respectant des contraintes tout en garantissant des bénéfices [2].

En fait, la gestion de production s'occupe d'un ensemble de problèmes liés à la production tels que la gestion des données, la planification, le contrôle (suivi) de la production, la gestion des stocks, la prévision, et l'ordonnancement...etc.

1.3. Organisation de la gestion de production

Les niveaux hiérarchiques de la gestion de production couramment retenus sont en nombre de trois : Stratégique, tactique et opérationnel.

- **Niveau stratégique**

Ces décisions consistent à déterminer la politique de l'entreprise et conditionner son avenir. Elles portent essentiellement sur la gestion des ressources durables, afin que celles-ci soient toujours suffisantes pour assurer la pérennité de l'entreprise.

- **Niveau tactique**

Leur rôle est d'assurer la liaison entre le niveau stratégique et le niveau opérationnel et de contrôler la bonne adéquation des ressources disponibles et des charges engendrées par les commandes ou les prévisions, mais sans modification profonde de la structure et du fonctionnement de l'entreprise.

- **Niveau opérationnel**

Ces décisions assurent le lancement des activités ainsi que les actions à même d'assurer la flexibilité nécessaire à la bonne conduite de la production [3].

1.4. Pilotage dynamique de système de production

Le pilotage dynamique permet à l'organisation de s'adapter de façon souple et rapide aux changements au sein de l'organisation et de son environnement.
Il est guidé par le délai d'obtention du produit, et le délai client (durée moyenne d'attente acceptée par les clients).

Dans le cadre du pilotage des systèmes de production, cinq fonctions essentielles sont définies et qui sont :

- **La planification**

L'horizon de la planification est le long terme. Elle définie le plan directeur de production (PDP), et de développer un compromis entre les objectifs commerciaux financiers de production. Ce plan intègre les commandes fermes et/ou prévisionnelles et les retards de livraison. Ce PDP constitue l'entrée du système de pilotage d'atelier.

- **la programmation**

Elle est chargée, à partir du plan directeur de production, d'établir un programme prévisionnel de production atelier par atelier, à capacité infinie ou suivant une charge globale admissible par l'atelier. Ce programme prend en compte les besoins dépendants et indépendants et calcule des besoins nets en fonction des stocks, des en-cours, des tailles des lots de fabrication. La programmation consiste essentiellement à décliner les objectifs de la planification en ordre de fabrication (OF) sur les différents ateliers et postes de travail et reste dans une logique de définition du quoi produire.

- **l'ordonnancement**

L'ordonnancement fournit un calendrier d'organisation du travail pour l'atelier fixant les dates de début et de fin de chaque tâche.
le séquencement des tâches consiste à déterminer l'ordre dans lequel les différentes tâches ou les différents produits passeront sur une ressource ou un groupe de ressources. Les dates de début et de fin de chaque tâche ne sont donc pas explicitement fixées. Dans les deux cas, l'objectif est de réaliser les OF(ordre de fabrication) du programme prévisionnel (délais,

quantité, qualité, etc.) .Tout en cherchant à optimiser l'utilisation des moyens de production en termes de charge.

- **la conduite**

La phase de conduite est chargée de réaliser la production prévue par l'ordonnancement. Comme on l'a vu, ce dernier fournit une solution réalisable à son niveau, solution qui n'est pas toujours une solution réalisable au niveau de la conduite.

Une des raisons de cet écart réside dans le caractère rigide des données et contraintes prises en compte pour établir l'ordonnancement prévisionnel.

Cette phase doit régler tous les problèmes non résolus par le niveau prévisionnel (charges ou contraintes locales). Elle doit aussi prendre en compte l'ensemble des contraintes de fabrication (contrôles de la qualité), les problèmes posés par les opérateurs (niveau de qualification, compétences, etc.), toutes présentes à ce niveau, et réagir aux aléas pour que la production prévue soit possible.

Les problèmes d'affectation de ressources doivent être résolus en temps réel (réponses aux questions quand et avec quoi produire). Il n'est pas étonnant que les performances réellement obtenues au sein d'un atelier soient souvent éloignées de celles initialement prévues par les phases prévisionnelles.

La conduite constitue la fonction charnière entre la gestion prévisionnelle et la gestion temps réel du système de production. Le paramètre temps et la rapidité de réponse sont primordiaux dans cette phase.

- **la commande**

Ce niveau, directement en relation avec le système de production, a un rôle d'interface et d'interpréteur. Sa tâche essentielle est de traduire un ordre en une séquence d'instructions compréhensibles par la partie opérative. Il est concrétisé soit par un opérateur pilotant une machine et assurant le suivi de réalisation, soit par un automatisme capable d'interpréter un ordre et de renseigner la conduite sur l'état d'avancement de celui-ci [4].

La figure suivante définit la fonction générale du pilotage dans la production :

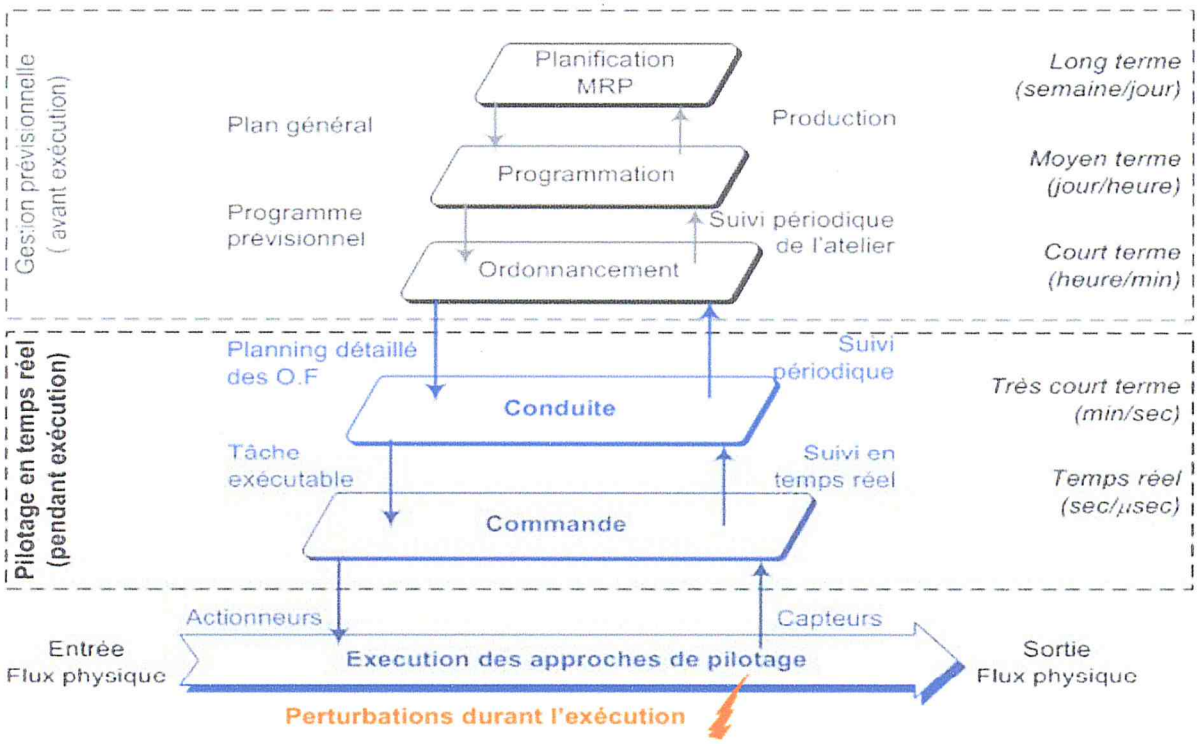


Figure 1.1: Fonction générale du pilotage dans la production [4].

1.5. L'utilisation de la technologie code à barres dans les ateliers industriels

La technologie du code à barre est l'une des plus utilisée à l'heure actuelle. En effet on la rencontre dans notre vie de tous les jours, que ce soit dans les supermarchés pour identifier les différents articles ou dans l'atelier industriel pour l'avancement de produit, il rester sur la pièce tout au long du processus de fabrication.

Dans notre travail on intègre la technologie de système code a barre pour le suivi de la production dans l'atelier, pour défini l'état de l'avancement de fabrication de produit **[5]**.
L'application de la technologie du code à barres à la gestion de production se réalise par un logiciel de gestion qui contrôle tous les processus de production et leurs différentes tâches.

Chaque travail réalisé dans l'usine est enregistré comme un repère dans l'application de gestion du contrôle de production en ligne, et donc on peut consulter immédiatement :

- ✓ L'origine des matières premières consommées.
- ✓ Avoir, en continu, des informations actualisées sur l'état de production, et d'aiguiller automatiquement un produit en fonction de ses caractéristiques, de ses défauts.
- ✓ Le temps utilisé pour chaque opération et machines.
- ✓ La consultation des incidents enregistrés sur le terrain.

- ✓ La détection l'état de l'outillage en cours d'utilisation lors de la fabrication du produit (le cas d'une panne d'une machine) [5].

1.5.1. Définition de code à barres

Un code à barres souvent appelé « code barre », représente la codification d'une information. Variant selon les algorithmes de codage, cette codification est optimisée selon les besoins pour encoder du texte, des chiffres, des caractères de ponctuation ou encore une combinaison de ces derniers. La représentation obtenue est optimisée pour une lecture optique. Les barres doivent contraster avec les espaces ce qui explique que celles-ci seraient souvent noires sur fond blanc. Il faut savoir que les lecteurs peuvent également codifier les informations.

Le code à barres est aujourd'hui la solution technique la plus utilisée pour saisir automatiquement une information [5].

1.6. L'ordonnancement dans les ateliers industriels

L'ordonnancement d'atelier consiste à organiser dans le temps le fonctionnement d'un atelier pour utiliser au mieux les ressources humaines et matérielles disponibles dans le but de fabriquer des produits de qualité dans les temps impartis.

1.6.1. Les éléments de problème d'ordonnancement

Dans la définition du problème d'ordonnancement, quatre éléments fondamentaux interviennent : Les tâches, les ressources, les contraintes et les critères [6].

1.6.1.1. Les tâches

Les tâches représentent toutes les opérations à effectuer pour la fabrication des produits. Une tâche, c'est à dire un ensemble d'opérations, requiert pour son exécution certaines ressources qu'il faut programmer de façon à optimiser un certain objectif.

Les paramètres d'une tâche sont :

- la duré opératoire P_i : c'est la duré d'exécution de la tâche.
- la date de disponibilité r_i : c'est la date de début au plus tôt.
- la date d'échéance d_i : c'est la date de fin au plus tard [6].

Notre cas (Job Shop flexible) appartient au problème non préemptif, chaque tâche est constituée d'un ensemble d'opérations liée entre elles par des contraintes technologiques.

Effectivement, en production manufacturier, on distingue souvent plusieurs phases dans l'exécution d'une tâche.

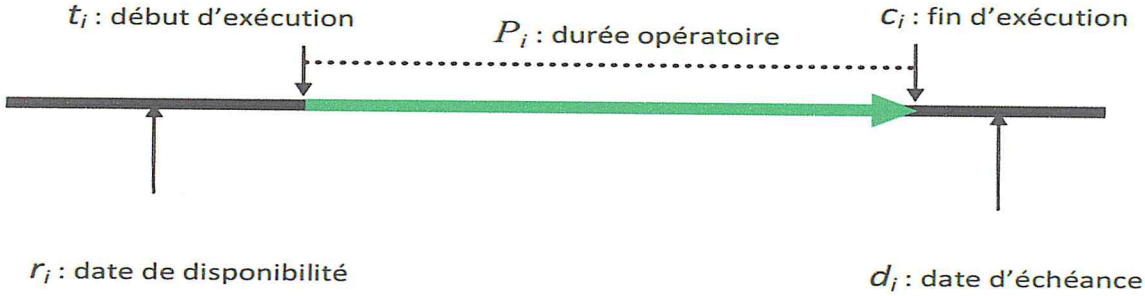


Figure 1.3 : Exemple d'une tâche.

1.6.1.2. Les ressources

Une ressource est un moyen requis, soit humain ou technique, utilisé pour réaliser une tâche. Ce moyen constitue l'élément nécessaire et indispensable pour la bonne exécution du processus de production. On distingue plusieurs types de ressources :

- **Ressources renouvelables et consommables**

Une ressource est renouvelable si après avoir été utilisée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machines, l'espace, etc.). La quantité de ressources utilisée à chaque instant est limitée. Dans le cas contraire, elle est consommable (matière première, budget, etc.), la consommation globale (ou cumul) au cours du temps est limitée.

- **Ressources disjonctives et cumulatives**

On distingue par ailleurs les ressources disjonctives (ou non partageables) qui ne peuvent exécuter qu'une tâche à la fois (machine-outil, robot manipulateur) et les ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches simultanément (équipes d'ouvriers, poste de travail **[6]**.

Dans notre problème (Job Shop flexible) les ressources sont **renouvelables** et **disjonctives.**

1.6.1.3. Les contraintes

Les contraintes représentent les conditions à respecter lors de la construction de l'ordonnancement pour qu'il soit réalisable. Elles rendent les problèmes d'ordonnancement plus difficiles car il faut les respecter lors de la résolution de ces problèmes.

Plusieurs types de contrainte doivent êtres respectées. On distingue notamment :

- **Les contraintes temporelles**

Les contraintes temporelles intègrent en général :

- Contraintes de temps alloué : Cette contrainte permet de représenter la limitation des valeurs possibles pour les dates des tâches.
- Contraintes de temps relatif : Cohérence technologique (gammes de fabrication) dans lesquelles il faut respecter le positionnement relatif entre les tâches.

- **Les contraintes de ressources**

Une contrainte de ressource représente le fait que les activités utilisent une certaine quantité de ressource, tout au long de leur exécution. Ces contraintes sont essentiellement soit :

- Contraintes d'utilisation des ressources : qui expriment la nature, la quantité et les caractéristiques d'utilisation de ces ressources.
- Contraintes de disponibilités des ressources : qui déterminent les quantités des ressources disponibles au cours du temps.

- **Contraintes d'affectation (Cas des problèmes d'ordonnancement flexible)**

Les contraintes d'affectation sont en général :

- Contrainte de domaine : Cette contrainte représente l'ensemble des ressources candidates pour l'exécution d'une tâche.
- Contrainte de différence : Cette contrainte impose l'utilisation de ressources différentes pour la réalisation d'un certain nombre de tâches [7].

1.6.1.4. Les critères

Les tâches sont soumises à des contraintes d'ordonnancement, c'est à dire une solution au problème ainsi défini, est évaluée vis-à-vis d'un ou de plusieurs objectifs à atteindre. L'objectif le plus souvent rencontré dans la littérature est l'obtention d'un ordonnancement optimisant un certain nombre de critères [6].

La détermination de ces objectifs s'avère souvent extrêmement délicate. En effet, on se trouve généralement en présence d'un ensemble d'objectifs plus ou moins contradictoires et dont l'importance relative est difficile à apprécier. Parmi ces critères, on peut citer :

- Le coût et la durée de réalisation.
- Le respect du délai d'exécution.
- La quantité de moyens nécessaires.
- La quantité de travail en attente.
- Le temps d'immobilisation des moyens.

Ces objectifs peuvent correspondre à des exigences quantitatives (valeurs à atteindre ou à ne pas dépasser) et se présentent sous forme de contraintes à respecter, ou bien à des exigences qualitatives s'exprimant sous forme de critères à optimiser. Parmi les critères les plus utilisés pour évaluer la qualité d'un ordonnancement obtenu, on distingue [8] :

➢ **La durée totale (Makespan)**

La durée totale de l'ordonnancement est égale à la différence entre la date d'achèvement de la tâche la plus tardive et la date de départ de la première tâche.

La minimisation de cette durée est le critère le plus souvent rencontré puisque ceci conduit inévitablement à une utilisation efficace des ressources.

➢ **Respect des dates au plus tard**

Dans beaucoup de problèmes réels, le respect au mieux des délais s'obtient en minimisant soit le plus grand retard soit la somme des retards.

➢ **Minimisation des coûts**

Ce genre de critère peut s'exprimer sous des formes très variées telles que, par exemple, la minimisation des stocks d'encours.

1.6.2. Organisation des ateliers industriels

Une classification très répandue des ateliers, du point de vue ordonnancement, est basée sur les différentes configurations des machines. Les modèles les plus connus sont ceux d'une machine unique, de machines parallèles, d'un atelier à cheminement unique ou d'un atelier à cheminement multiple.

- **Machine unique**

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. L'une des situations intéressantes où on peut rencontrer ce genre de configurations est le cas où on est devant un système de production comprenant une machine goulot qui influence l'ensemble du processus. L'étude peut alors être restreinte à l'étude de cette machine [9].

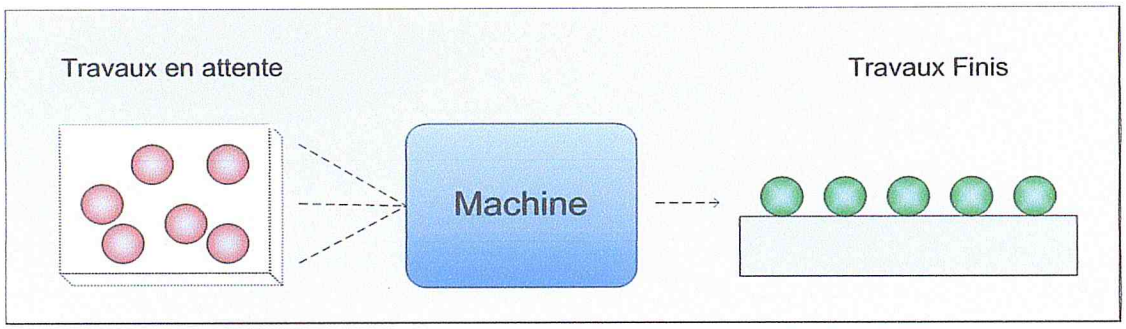


Figure 1.3 : Modèle a machine unique.

- **Machines parallèles**

Dans ce cas, on dispose d'un ensemble de machines identiques pour réaliser les travaux. Les travaux se composent d'une seule opération et un travail exige une seule machine. L'ordonnancement s'effectue en deux phases : la première phase consiste à affecter les travaux aux machines, et la deuxième phase consiste à établir la séquence de réalisation sur chaque machine [9].

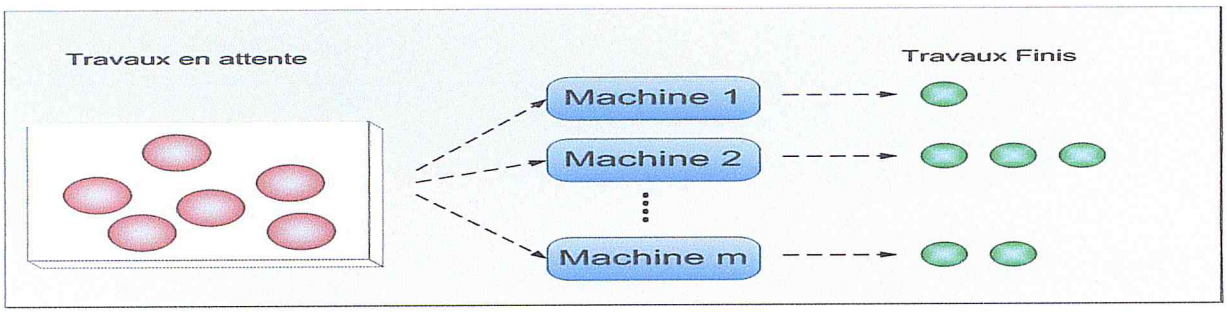


Figure 1.4 : Modèle a machines parallèles.

- **Ateliers à cheminement unique (Flow Shop)**

Un atelier à cheminement unique est un atelier où le processus d'élaboration de produits est dit « linéaire », c'est-à-dire lorsque les étapes de transformation sont identiques pour tous les produits fabriqués. Selon les types de produits élaborés, on distingue la production continue et la production discrète. La production continue est caractérisée par la fluidité de son processus et l'élimination du stockage. C'est le cas notamment dans les raffineries, les cimenteries, les papeteries. La production discrète de masse s'applique principalement aux produits de grande consommation fabriqués à la chaîne (ex, automobile, la majorité du domaine du textile, machines-outils…).

Dans les deux cas, les machines peuvent être dédiées à une opération précise, et sont implantées en fonction de leur séquence d'intervention dans la gamme de production.

L'un des objectifs principaux dans le cas d'atelier à cheminement unique est de trouver une séquence des tâches en main qui respecte un ensemble de contraintes et qui minimise le temps total de production **[9]**.

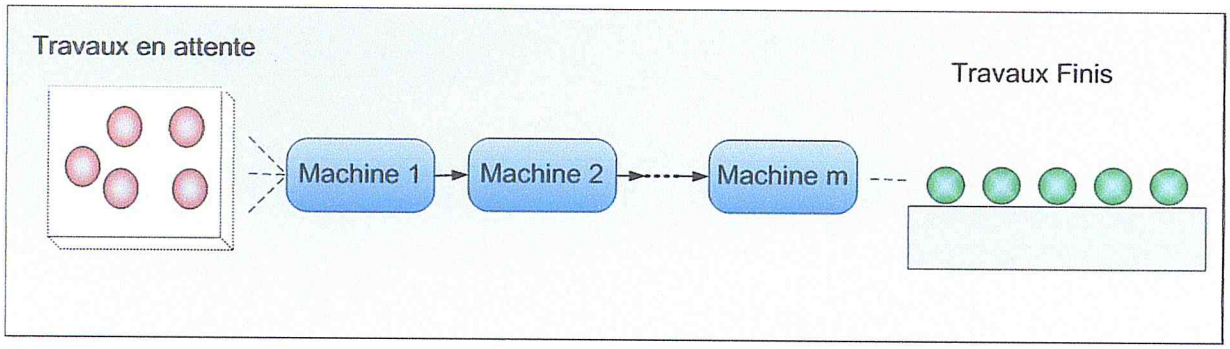


Figure 1.5 : Modèle d'ateliers à cheminement unique (Flow Shop).

- **Ateliers à cheminements multiples (Job Shop)**

Les ateliers à cheminements multiples (ACM) sont des unités manufacturières traitant une variété de produits individuels dont la production requiert divers types de machines dans des séquences variées. L'une des caractéristiques d'un atelier à cheminement multiple est que la demande pour un produit particulier est généralement d'un volume petit ou moyen. Une autre caractéristique est la variabilité dans les opérations et un mix produit constamment changeant. Ainsi, il est nécessaire que le système soit de nature flexible. Dans un sens général, la flexibilité est la capacité d'un système de répondre aux variations dans l'environnement.

L'objectif le plus considéré dans le cas d'un atelier à cheminements multiples est le même que celui considéré pour un atelier à cheminement unique, à savoir trouver une séquence de tâches sur les machines qui minimise le temps total de production **[9]**.

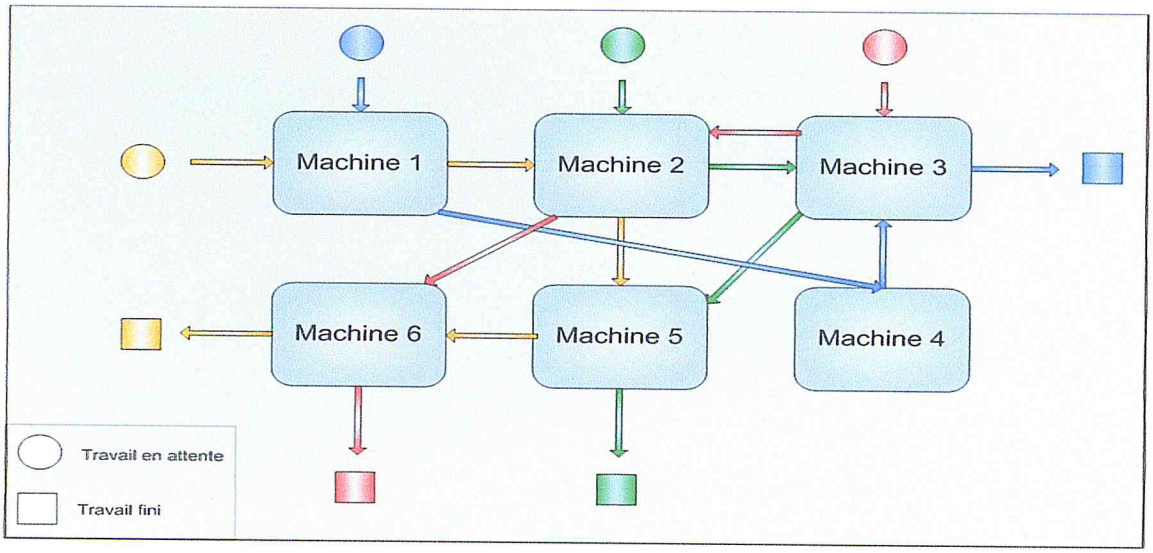


Figure 1.6 : Modèle d'ateliers à cheminements multiples (Job Shop).

1.7. Le Job Shop Flexible

Le Job Shop Flexible (FJSP) 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 [1].

1.7.1. Formulation du problème Job Shop Flexible

Les problèmes d'ateliers Job Shop flexibles peuvent être formulés comme suit:

- Soit un ensemble de n produits indépendants à réaliser sur m machine M_k, tel que $k = 1, 2, \ldots, m$.
- Chaque produit J_j est constitué d'une séquence de n_j opérations $O_{i,j}$, tel que i = 1, 2, ..., 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 $P_{i,j,k}$.

Le Job Shop Flexible 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 $TD_{i,j}$, et des temps de fin $TF_{i,j}$, de l'opération $O_{i,j}$ [1].

1.7.2. Les contraintes du Job Shop Flexible

Les contraintes du problème Job Shop Flexible sont des contraintes technologiques auxquelles sont soumis les tâches et les machines. Elles concernent l'utilisation des machines et les liens existant entre les opérations. Ces contraintes sont :

- Les machines sont indépendantes les unes des autre (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.
- 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 donné.
- 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 [1].

1.7.3. Représentation graphique d'une solution de Job Shop Flexible

Les représentations les plus courantes pour une solution d'un ordonnancement est le diagramme de Gantt et le graphe conjonctif. Dans notre cas nous avons utilisé le diagramme de Gantt.

➢ **Le diagramme de Gant**

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. Dans le cas des problèmes d'atelier, ce diagramme se compose de plusieurs lignes horizontales, chacune d'entre elles désigne une machines. Les opérations exécutées sur une machine donnée sont

représentées sous forme de barres ayant des longueurs proportionnelles à leurs temps opératoires **[10] (Figure 1.7).**

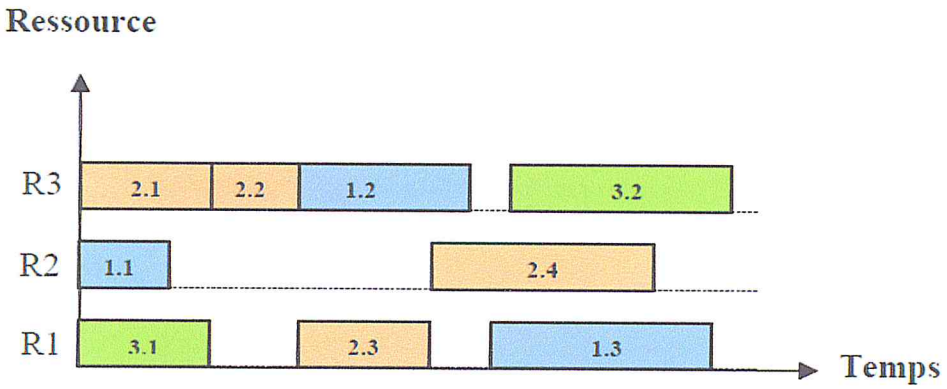


Figure 1.7 : Exemple d'un diagramme de Gantt.

1.8. Les approches de résolution des problèmes d'ordonnancement

1.8.1. Introduction générale aux méthodes de résolution

Les problèmes d'ordonnancement sont souvent faciles à définir mais sont généralement difficiles à résoudre. En effet, la plupart de ces problèmes appartiennent à la classe des problèmes NP-difficiles et ne possèdent donc pas de solution algorithmique efficace valable pour toutes les données **[11].**

Etant donnée l'importance de ces problèmes, de nombreuses méthodes de résolution ont été développées, ces méthodes peuvent être classées sommairement en deux grandes catégories :

- **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 grande taille. Parmi ces méthodes on distingue les méthodes de séparation et évaluation progressive **[11].**

- **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. 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 avantage d'être beaucoup moins contraignantes. Ainsi, dans de nombreux cas, les méthodes approchées deviennent la seule option performante envisageable [12].

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

- **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.

- **Les méthodes d'amélioration ou méta-heuristiques**

Le principe de ces méthodes n'est plus de construire un ordonnancement à partir des données initiales du problème, mais de modifier le résultat d'une solution admissible en vue d'améliorer la valeur de la fonction objectif. La plupart de ces méthodes utilisent la notion de voisinage de solution, on parle aussi de méthodes par voisinage.

A partir d'une solution initiale (générer aléatoirement ou avec une heuristique), une méthode par voisinage ou encore méthode de recherche locale réalise un processus itératif qui consiste à remplacer la solution courante par l'un de ses voisins en tenant compte de la fonction objectif. La méthode s'arrête et retourne la meilleure solution trouvée. Le critère d'arrêt peut être un nombre d'itération atteint ou une valeur de la fonction objective atteinte.

1.8.2. Comparaison entre méthodes exactes et méthodes approchées

On représente ci-dessous quelques différences entre les méthodes exactes et les méthodes approchées.

Méthodes approchées	**Méthodes exactes**
Efficacité pour les problèmes de grande taille	Efficacité pour des cas particuliers des problèmes de taille raisonnable
Espace mémoire raisonnable	Espace mémoire considérable
Durée de résolution très petite par rapport aux méthodes exactes	Durée de temps de résolution est généralement considérable pour les problèmes de grande taille
Aucune garantie d'optimalité	Garantie d'optimalité

Pratiquement simple à implanter et à comprendre	Pratiquement difficile à implanter et à comprendre.

Tableau 1.1 : Tableau de comparaison entre les méthodes exacte et méthodes approchées **[12].**

1.9. Les benchmarks de problème Job Shop Flexible

Les benchmarks sont des problèmes types construits par plusieurs auteurs pour tester les performances des approches de résolution, en procédant à des comparaisons sur les mêmes instances.

On ordonnancement d'ateliers de type Job Shop Flexible, il est souvent utile d'avoir des **Benchmarks** pour tester les modèles d'ordonnancement et surtout les méthodes de résolution appropriées **[13]**.

Les benchmarks de Job Shop Flexible présentent les caractéristiques suivantes :

- Chaque benchmarks est décrit par un ensemble de **n** tâches et **m** machines. Ce sont des problèmes **n** x **m**.
- Chaque tâche comprend un certain nombre d'opération tout dépond de la tâche.
- Chaque opération à son ensemble de choix de machine.
- Pour chaque choix, le numéro de machine ainsi que le temps d'exécution (nombre entier) de l'opération sur cette machine, sont spécifiées.
- Les solutions optimales de ces benchmarks sont connues ; ce qui constitue un bon outil pour les expérimentations et les comparaisons.

La figure suivante représente un exemple de benchmark : sur lequel nous constatons les caractéristiques générales des benchmarks.

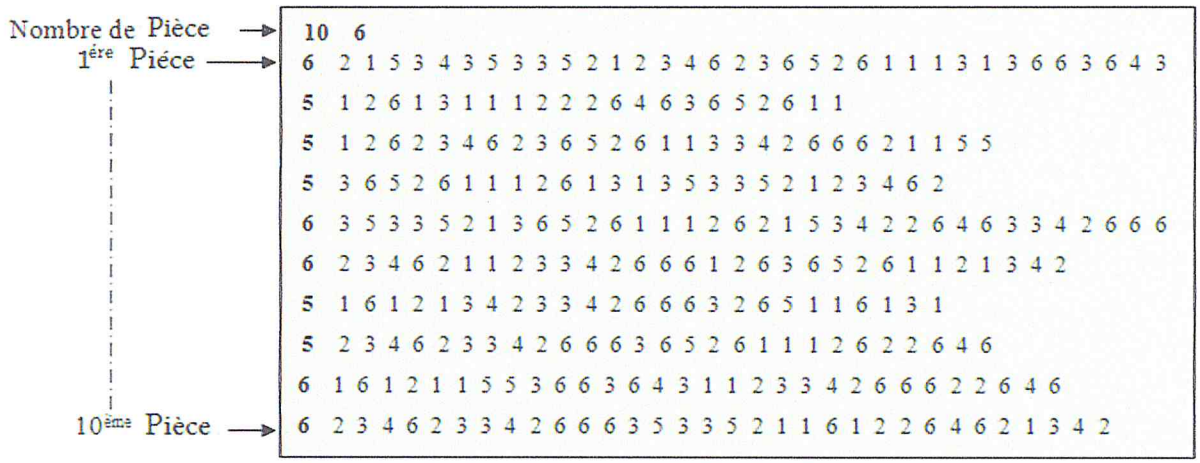
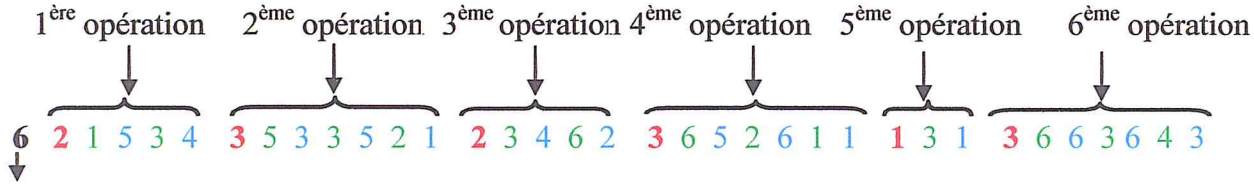
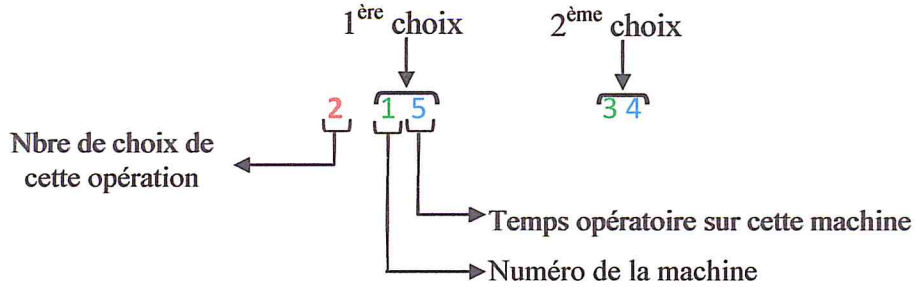


Figure 1.8 : Exemple de benchmark.

Exemple d'interprétation de la 1ère **tâche** de notre benchmark :



Exemple d'interprétation de la 1ère **opération** du 1ère **pièc** de **notre benchmark** :



Dans notre étude, nous utilisons cet échantillon de benchmark comme plateforme de teste, afin d'étudier la performance des stratégies méta-heuristiques employées dans la résolution du problème de Job Shop Flexible.

1.10. Les progiciels existants des systèmes de production

Depuis les années 1980, l'accroissement de la diversité et de la complexité des produits a progressivement conduit les entreprises à utiliser des progiciels informatiques dans le processus de production. Avec le développement considérable de l'informatique, ces

progiciels sont maintenant disponibles dans toutes les étapes du cycle de vie d'un système de production. On peut, cependant, noter le faible nombre d'outils informatiques existant dans le domaine de l'ingénierie de processus comparativement à ceux qui existent en exploitation de produit.

Dans cette phase d'exploitation de produit, on dispose de plusieurs outils adaptés aux divers niveaux de décision : ERP et GPAO pour les moyen et long termes en décisions stratégique et tactique, outils opérationnels d'atelier pour l'ordonnancement à court terme **[13]**.

Un logiciel de GPAO est un logiciel qui permet de gérer la production d'une entreprise. GPAO signifie Gestion de la Production Assistée par Ordinateur. Le logiciel GPAO est l'outil qui rend possible la gestion précise de l'ensemble des activités liées à la production. Cette gestion passe notamment par la gestion des stocks, la gestion des commandes et par la gestion de tous les articles qui jouent un rôle dans le processus de production **[13]**.

A titre d'illustration, nous prendrons deux exemples de logiciel GPAO.

1.10.1. Le progiciel GPAO Solune Alta

Le progiciel de GPAO **Solune Alta** permet de gérer les informations relatives aux devis de la facturation en passant par la production (Gestion commerciale, Achats, Sous-traitance, Stocks, Calcul de Besoins, Ordres de fabrication, Suivi d'atelier, Traçabilité, Gestion des documents Qualité, interface comptabilité, Planification simplifiée, Pointage des temps par Codes à Barres.)

Adapté et paramétrable, il satisfait les besoins de facilité d'utilisation, de rapidité et d'analyse de l'activité (bilans financiers par OF, par affaire, par client, par fournisseur, par produit et par composant) **[15]**.

1.10.2. Le progiciel PROCAD GPAO

PROCAD GPAO est un logiciel complet de gestion de production conçu pour répondre aux plusieurs fonctionnalités industrielles : achats, ventes, gestion des stocks, planification, nomenclature, gamme, modélisation des processus.

PROCAD GPAO est personnalisable et flexible afin de correspondre parfaitement à vos besoins fonctionnels métiers **[16]**.

Les systèmes excitant de pilotage de la production (logiciels d'ordonnancement) sont généralement réalisés avec une optique centralisée et statique au sens que ces systèmes

n'intègrent pas de vision dynamique et d'algorithmes de calcul de l'ordonnancement réactif aux aléas. Cette possibilité est donc très intéressante pour l'entreprise qui est soumise à des contraintes dynamiques du type aléas fréquent.

Nous proposons dans notre travail la réalisation d'un logiciel de pilotage temps réel du système de production en intégrant un module d'ordonnancement dynamique et un module de suivi de la production en utilisant la technologie code à barres.

1.11. Conclusion

Ce chapitre a été consacré à la présentation de la fonction gestion de production dans l'entreprise, plus particulièrement le problème d'ordonnancement. La première partie de ce chapitre avait pour objet de situer l'ordonnancement dans le contexte des fonctions de production et de gestion de production. La deuxième partie consistait en la définition du problème d'ordonnancement, les critères d'optimisation, et la classification de ces problèmes. Finalement, nous avons présenté dans ce chapitre le problème de type Job Shop Flexible, objet de notre projet de fin d'études.

Chapitre 2 : Les méthodes de résolution « Les méta-heuristiques ».

2.1. Introduction

La résolution optimale d'un problème d'ordonnancement consiste en la détermination d'une séquence d'exécution des tâches sur les machines de sorte que les critères d'optimisation atteignent leurs valeurs optimales.

Les méthodes de résolutions des problèmes d'ordonnancements puisent dans toutes les techniques de l'optimisation combinatoires (programmation mathématique, programmation dynamique, théorie des graphes …). Ces méthodes garantissent en général l'optimalité de la solution fournie. Mais, vu la complexité des problèmes d'ordonnancement, ces méthodes deviennent très vite inadaptées lorsque la taille du problème croit, d'où la nécessité d'utiliser des méthodes de résolution approchée, qu'on appelle méthodes heuristiques, généralement efficaces pour les problèmes NP- difficiles, dans la suite nous présenterons les méthodes de résolutions les plus connues.

2.2. 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 :

2.2.1. Les approches trajectoires

Ces algorithmes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans ces approches, se distingue:

- La méthode de descente.
- Le recuit simulé.
- La méthode Tabou [17].

2.2.2. Les approches populations (ou évolutionnaires)

Elles consistent à travailler avec un ensemble de solution simultanément, que l'on fait évoluer graduellement. L'utilisation de plusieurs solutions simultanément permet naturellement d'améliorer l'exploration de l'espace des configurations. Dans cette seconde catégorie, dans ces approches, on distingue :

- Les algorithmes génétiques [17].

2.3. Présentation des méta-heuristiques

Dans la suite nous présenterons les méthodes de résolutions les plus connues.

2.3.1. La méthode de descente

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 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 $N(x)$ de x. Si cette solution est meilleure que x, $f(x')$ 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 de descente**

  1: Initialisation: trouver une solution initiale **x**

  2: Répéter

  3: Recherche de voisinage: trouver une solution **x' ϵ N(x)**

  4: Si **f (x') < f(x)** alors

  5: x':= x

  6: Fin si

  7: Jusqu'à **f(y) ≥ f(x)**, quelque soit **y** ϵ **N(x)**

- **Avantages et inconvénients**

En général, l'efficacité des méthodes de recherche locale simple (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 **[18]**.

2.3.2. Le Recuit Simulé

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 [19].

- **Algorithme générale de Recuit Simulé**

1: Initialisation: trouver une solution initiale **x**, poser une température initiale **t**

2: Répéter

3: Recherche de voisinage: trouver une solution $x' \in N(x)$

4: Déterminer $\Delta C = f(x') - f(x)$

5: Obtenir $p \sim U(0.1)$

6: Si $\Delta C < 0$ ou $e^{\frac{\Delta C}{t}} > p$ 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**

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 locale 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 [19].

2.3.3. La Recherche Tabou

La méthode Tabou est une technique de recherche dont lequel est devenue très classique en optimisation combinatoire. Elle se distingue des méthodes de recherche locale simple par le recours à un historique des solutions visitées. 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» **[20]**.

- **Algorithme général de Recherche Tabou**

1: Choisir une solution initiale **i** dans **S** (l'ensemble des solutions)

Appliquer **i* = i** et **k = 0**

2: Appliquer **k = k+1** et générer un sous-ensemble de solution en **N (i,k)** pour que:

- Les mouvements tabous ne soient pas choisis.
- Un des critères d'aspiration **a (i,m)** soit applicable.

3: Choisir la meilleure solution **i'** parmi l'ensemble de solution voisines **N (i,k)**

Appliquer **i** = meilleur **i'**

4: Si **f (i) <= f (i*)**, alors nous avons trouvé une meilleure solution

Appliquer **i* = i**

5: Mettre à jour la liste **T** et les critères d'aspiration.

6: Si une condition d'arrêt est atteinte, stop. Sinon, retour à deuxième étape.

Condition d'arrêt: condition qui régira l'arrêt de l'algorithme (ex: arrêt après 20 itérations (k = 20). Avec **i*** : la solution optimale actuelle.

- **Avantages et inconvénients**

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.

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.

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.

2.3.4. Les algorithmes génétiques

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

- **Algorithme générale de l'algorithme génétique**

  1 : Initialisation: générer une population initiale $\mathbb{P}$ de solutions de taille $|\mathbb{P}| = n$

  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: Ajouter **y** à la nouvelle population.

  7: Jusqu'à critère d'arrêt satisfait

- **Avantages et inconvénients**

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 bonne solution en quelques itérations seulement.

Ils présentent un fort potentiel d'applications pratiques. D'ailleurs, ils sont de plus en plus utilisés et ce, dans de multiples domaines. Il faut dire qu'ils fournissent d'excellentes performances à de faibles coûts.

Ils permettent d'explorer des domaines possédant de très nombreuses solutions. Pour une population d'origine de N individus **[20]**.

En effet, il n'y a pas de garantie quant à l'obtention de la solution optimale au problème posé en un temps fini. L'utilisation de ces algorithmes est souvent coûteuse en temps de calculs (les algorithmes sont lents).

Pour la résolution du problème d'ordonnancement dans un système de production, nous avons implémenté l'algorithme génétique.

2.4. Conclusion

Ce chapitre a été consacré à la présentation des méthodes de résolution des problèmes d'ordonnancement, plus précisément les méta-heuristiques tout en mettant en exergue les avantages et les inconvénients des principaux algorithmes utilisés pour la résolution de ce type de problème. Nous avons, à cet effet, présenté d'une manière générale les méta-heuristiques les plus connues à savoir : la méthode de descente, le recuit simulé, la méthode Tabou, et les algorithmes génétiques. Ces derniers, compte tenu de leurs robustesses et leurs adaptations à notre problème, ont été implémenté dans notre solution.

Chapitre 3 : Conception et réalisation.

3.1. Introduction

Le but de notre travail est de concevoir et développer un logiciel d'ordonnancement industriel basé sur le modèle Job Shop Flexible intégrant un module de calcul basé sur l'implémentation d'un algorithme génétique. Dans ce chapitre sera consacré, d'une part, à l'étude et la conception du notre logiciel, et d'autre part, à la présentation des aspects relatifs à l'implémentation de cette méta-heuristique, notamment la fonction d'évaluation utilisée dans le but de trouver une solution optimale qui considère plus particulièrement le temps de transport entre les zones.

3.2. Le cycle de vie du logiciel

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.

Le modèle en cascade est un cycle de vie linéaire, séquentiel, Celui-ci a été défini dans les années 70, Ce cycle de vie est basé sur la production d'éléments livrables.

Dans ce modèle le principe est très simple : chaque phase se termine à une date précise par la production de certains documents ou logiciels. Les résultats sont définis sur la base des interactions entre étapes, ils sont soumis à une revue approfondie et on ne passe à la phase suivante que s'ils sont jugés satisfaisants [29]

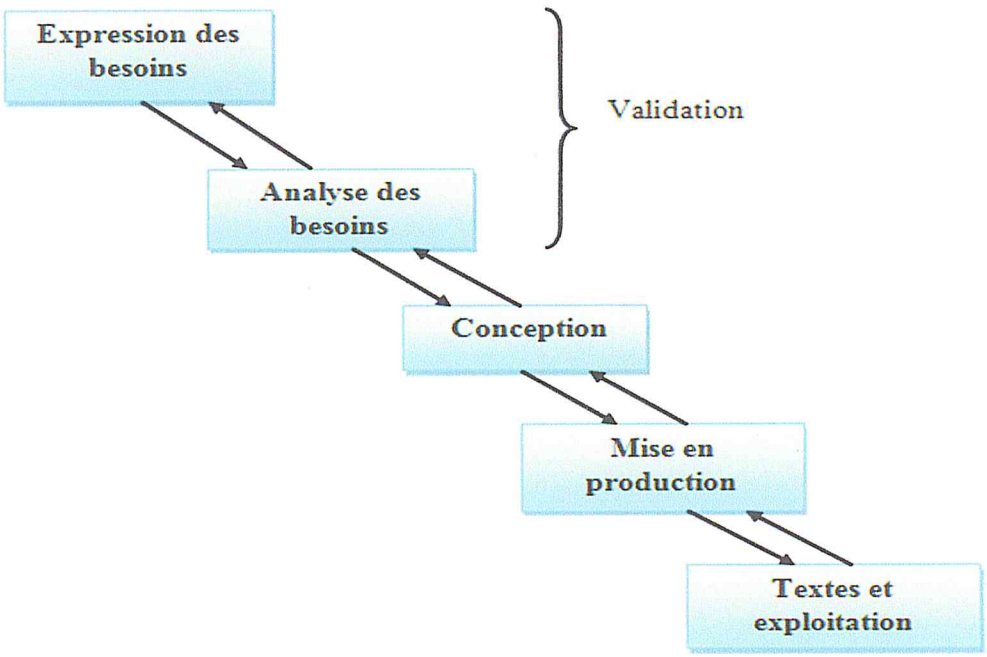


Figure 3.1: Modèle du cycle de vie en cascade.

3.2.1. La validation

A ce niveau, on parle de faisabilité, cette étape est représentée par les diagrammes suivants :

- Les diagrammes des cas d'utilisations.
- Les diagrammes des séquences.

Si la conception de ces diagrammes est faisable et validée on passe à la seconde phase du modèle en cascade.

3.2.1.1. Les diagrammes des cas d'utilisation

Un cas d'utilisation est une manière spécifique d'utiliser un système. Un cas d'utilisation est un diagramme qui consiste à définir le système du point de vue de l'utilisateur **[30]**.

➢ **Cas d'utilisation**

- Une manière d'utiliser le système.
- Une suite d'interactions entre un acteur et le système.
- Correspond à une fonction visible par l'utilisateur.
- Permet d'atteindre un but pour un utilisateur.

➢ **Le système**

- Est un ensemble de cas d'utilisation.
- Contient les cas d'utilisation mais pas les acteurs.
- Un modèle de cas d'utilisation permet de définir :
  - les fonctions essentielles du système.
  - les limites du système.
  - le système par rapport à son environnement.

➢ **Les acteurs**

- Élément qui interagit avec le système.
- Prend des décisions, des initiatives.
- Rôle qu'un utilisateur joue par rapport au système.

Dans notre travail, les diagrammes conçus sont :

❖ **Cas d'utilisation global**

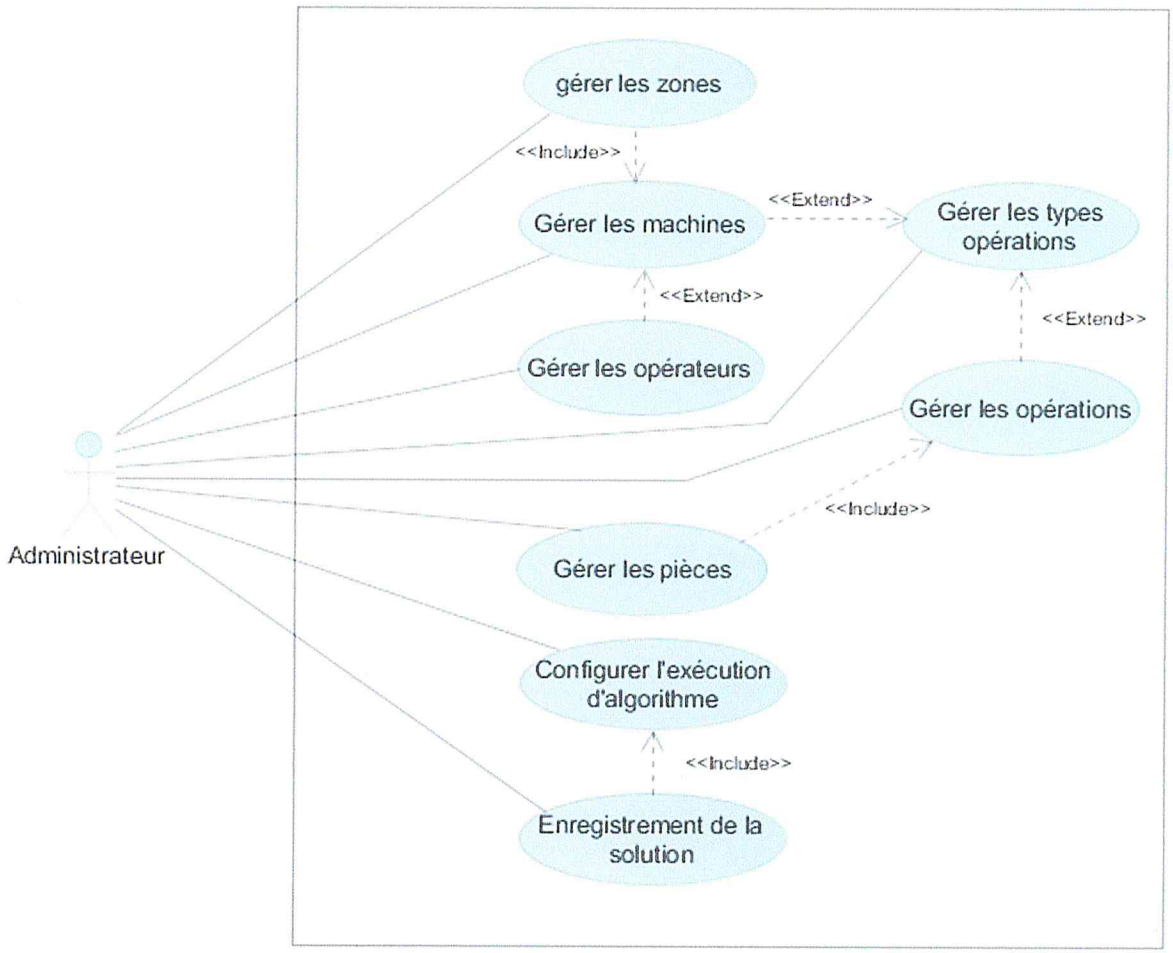


Figure 3.2: Cas d'utilisation global.

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

Cas d'utilisation	**Description**
Gérer les zones	L'administrateur peut gérer toutes les zones.
Gérer les machines	L'administrateur peut gérer toutes les différentes machines.
Gérer les types d'opérations	L'administrateur peut gérer toutes les différents types d'opérations.
Gérer les opérations	L'administrateur peut gérer toutes les différentes opérations.
Gérer les opérateurs	L'administrateur peut gérer toutes les opérateurs.
Gérer les pièces	L'administrateur peut gérer toutes les différentes pièces.

Configurer l'exécution d'algorithme	L'administrateur peut configurer l'exécution d'algorithme.
Enregistrement de la solution	L'administrateur peut enregistrer la solution finale de l'algorithme.

Tableau 3.1 : Description du cas d'utilisation global.

❖ **Les détails de cas d'utilisation global**

- **Gérer les zones**

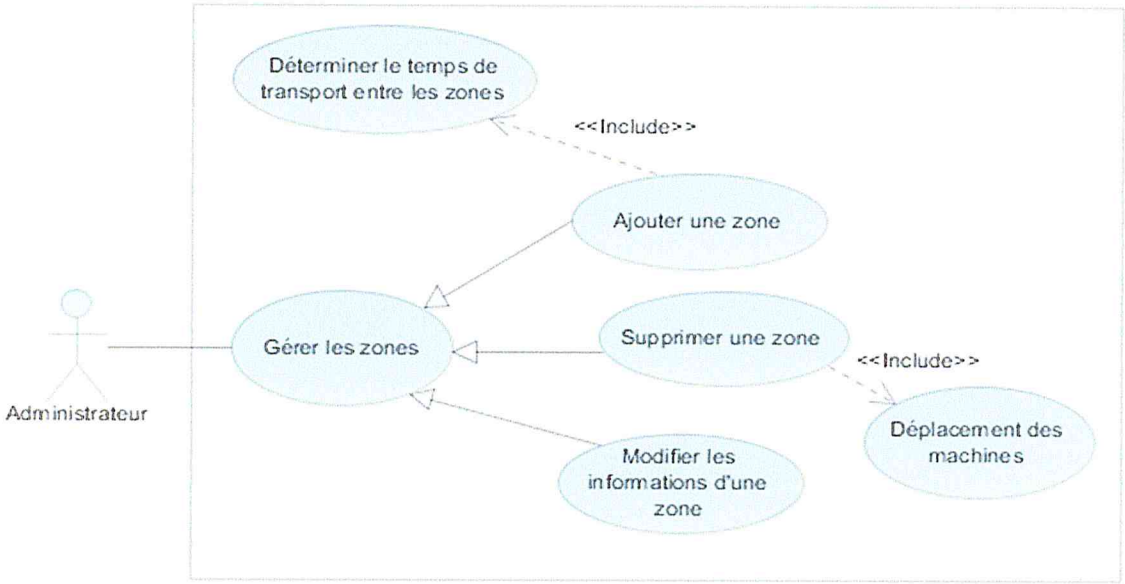


Figure 3.3: Cas d'utilisation « Gérer les zones ».

- **Description du cas d'utilisation « Gérer les zones »**

Cas d'utilisation	**Description**
Ajouter une zone	L'administrateur peut ajouter une zone, puis remplir leur propre information.
Supprimer une zone	L'administrateur peut supprimer une zone.
Modifier les informations d'une zone	L'administrateur peut modifier le nom d'une zone.
Déterminer le temps de transport entre les zones	L'administrateur doit déterminer le temps de transport entre deux zones.
Déplacement des machines	L'administrateur doit déplacer les machines à autre zone avant le supprimer.

Tableau 3.2: Description du cas d'utilisation « Gérer les zones ».

- **Gérer les types d'opération**

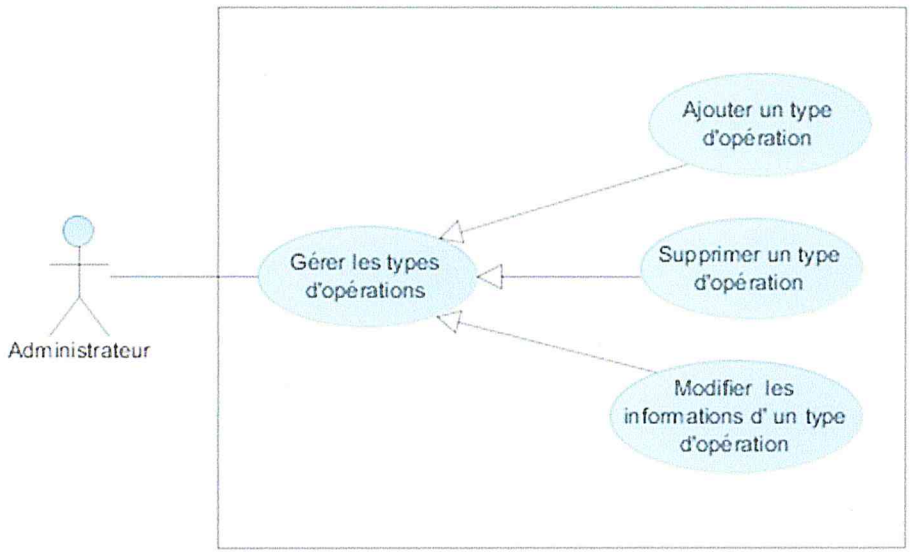


Figure 3.4: Cas d'utilisation « Gérer les types d'opération ».

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

Cas d'utilisation	**Description**
Ajouter un type d'opération	L'administrateur peut ajouter un type d'opération.
Supprimer un type d'opération	L'administrateur peut supprimer un type d'opération.
Modifier les informations d'un type d'opération	L'administrateur peut modifier les informations d'un type d'opération (erreur de frappe).

Tableau 3.3 : Description de cas d'utilisation « Gérer les types d'opération ».

- **Cas d'utilisation « Gérer les machines »**

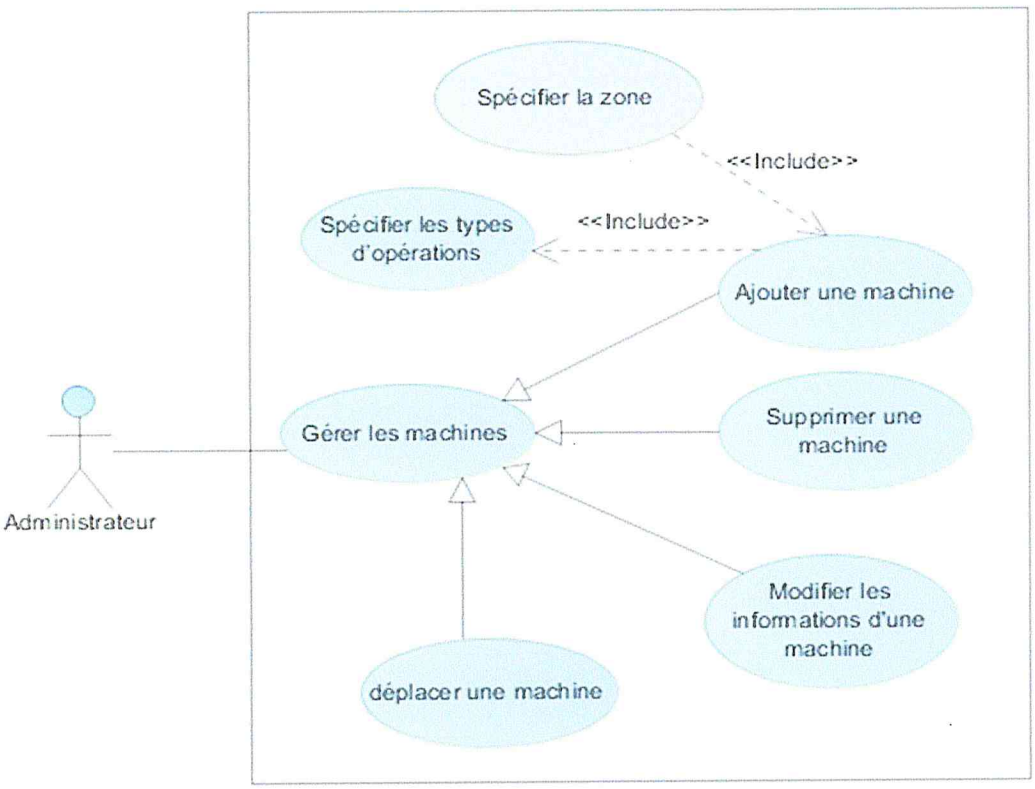


Figure 3.5: Cas d'utilisation « Gérer les machines ».

- **Description du cas d'utilisation « Gérer les machines »**

Cas d'utilisation	**Description**
Ajouter une machine	L'administrateur peut ajouter une machine, puis remplir leur propre information.
Supprimer une machine	L'administrateur peut supprimer une machine.
Modifier les informations d'une machine	L'administrateur peut modifier les informations d'une machine.
Spécifier le type d'opération	L'administrateur doit spécifier les types d'opérations de la machine qui est ajoutée.
Spécifier la zone	L'administrateur doit ajouter la machine dans la zone qui spécifier.
Déplacer une machine	L'administrateur peut déplacer une machine vers une autre zone.

Tableau 3.4: Description du cas d'utilisation « Gérer les machines ».

- **Gérer les opérateurs**

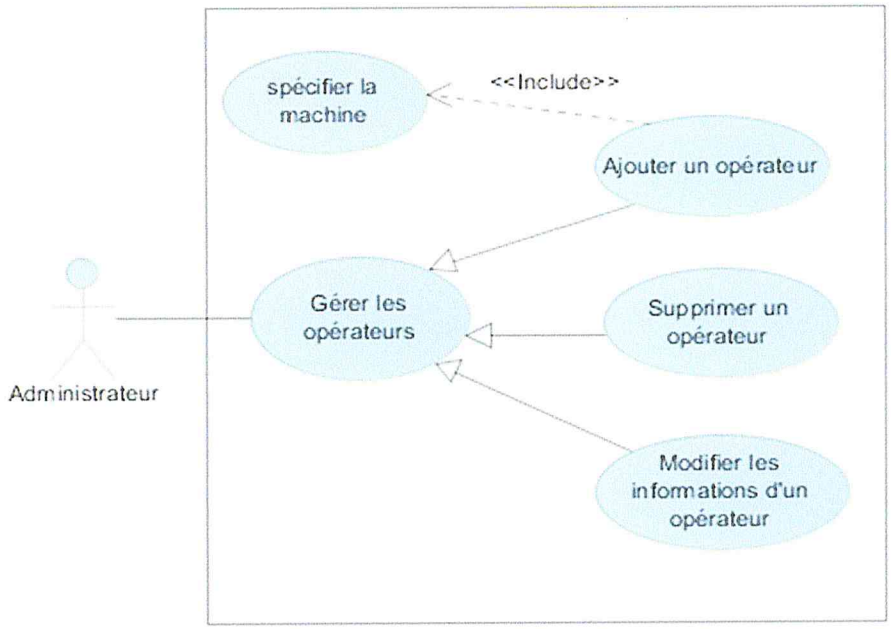


Figure 3.6 : Cas d'utilisation « Gérer les opérateurs ».

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

Cas d'utilisation	**Description**
Ajouter un opérateur	L'administrateur peut ajouter un opérateur.
Supprimer un opérateur	L'administrateur peut supprimer un opérateur.
Modifier les informations d'un opérateur	L'administrateur peut modifier les informations d'un opérateur (l'erreur de frappe).
Spécifier la machine	L'administrateur doit spécifier la machine qui compétent avec chaque opérateur.

Tableau 3.5 : Description de cas d'utilisation « Gérer les opérateurs ».

- **Gérer les pièces**

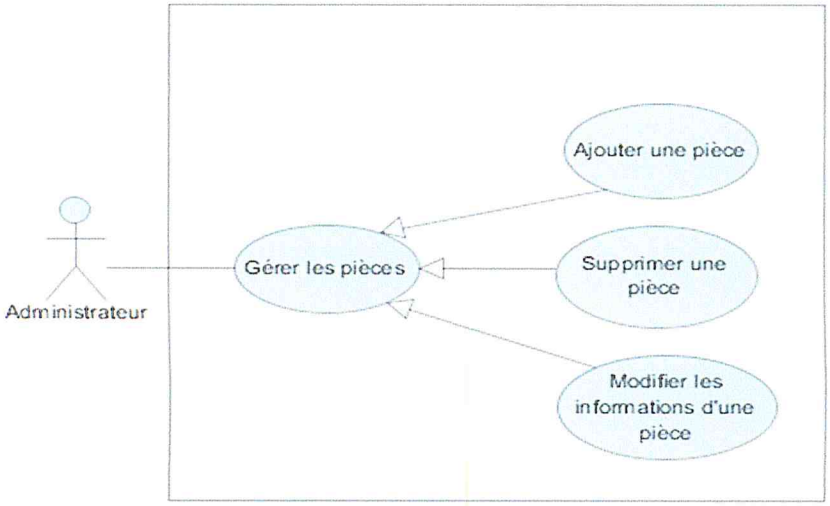


Figure 3.7 : Cas d'utilisation « Gérer les pièces ».

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

Cas d'utilisation	**Description**
Ajouter une pièce	L'administrateur peut ajouter une pièce.
Supprimer une pièce	L'administrateur peut supprimer une pièce.
Modifier une pièce	L'administrateur peut modifier les informations d'une pièce.

Tableau 3.6: Description du cas d'utilisation « Gérer les pièces ».

- **Gérer les opérations**

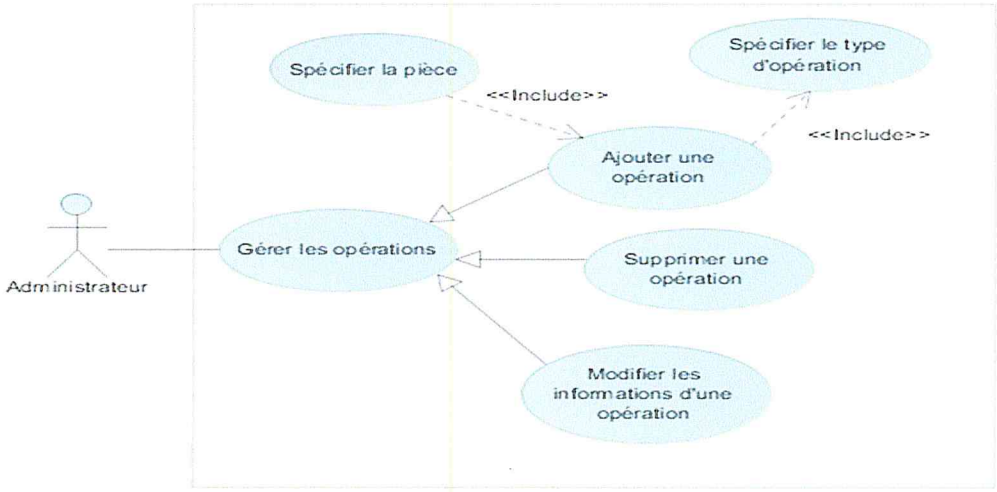


Figure 3.8 : Cas d'utilisation « Gérer les opérations ».

- **Description du cas d'utilisation « Gérer les opérations »**

Cas d'utilisation	**Description**
Ajouter une opération	L'administrateur peut ajouter une opération, puis remplir leur propre information.
Supprimer une opération	L'administrateur peut supprimer une opération.
Modifier une opération	L'administrateur peut modifier les informations d'une opération.
Spécifier type d'opération	L'administrateur doit spécifier le type d'opération de l'opération.
Spécifier la pièce	L'administrateur doit ajouter l'opération dans la pièce qui spécifier.

Tableau 3.7: Description du cas d'utilisation « Gérer les opérations ».

- **Configurer l'exécution d'algorithme**

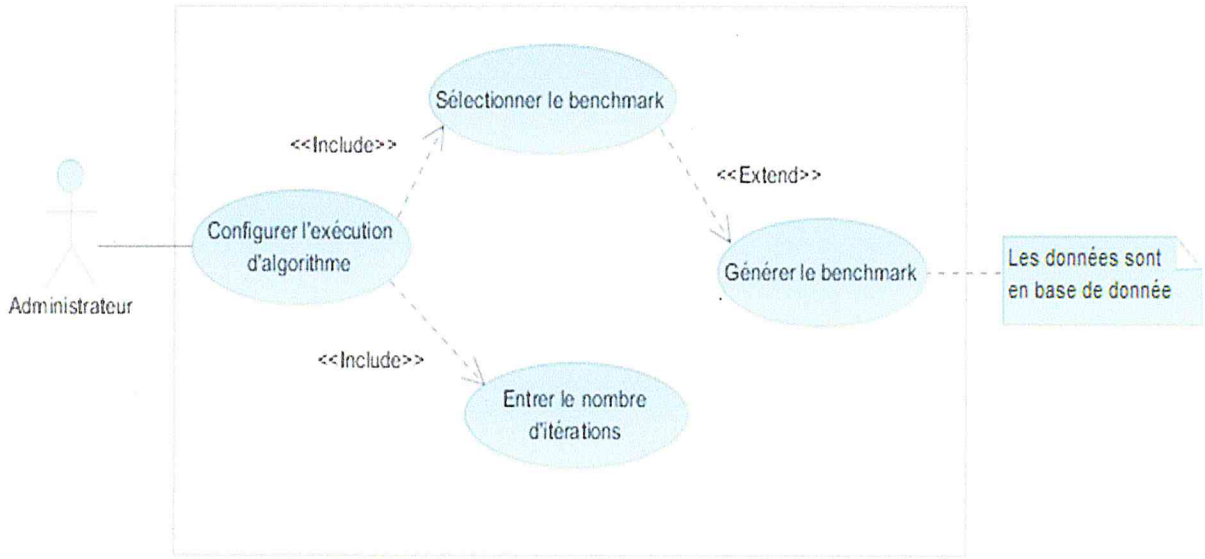


Figure 3.9 : Cas d'utilisation « Configurer l'exécution de l'algorithme ».

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

Cas d'utilisation	**Description**
Entrer le nombre d'itérations	L'administrateur doit entrer le nombre d'itérations pour l'exécution de l'algorithme.
Sélectionner le benchmark	L'administrateur doit choisir un benchmark.
Générer le benchmark	L'administrateur doit générer le benchmark à partir la base de données.

Tableau 3.8: Description du cas d'utilisation « Configurer l'exécution de l'algorithme ».

- **Enregistrement de la solution**

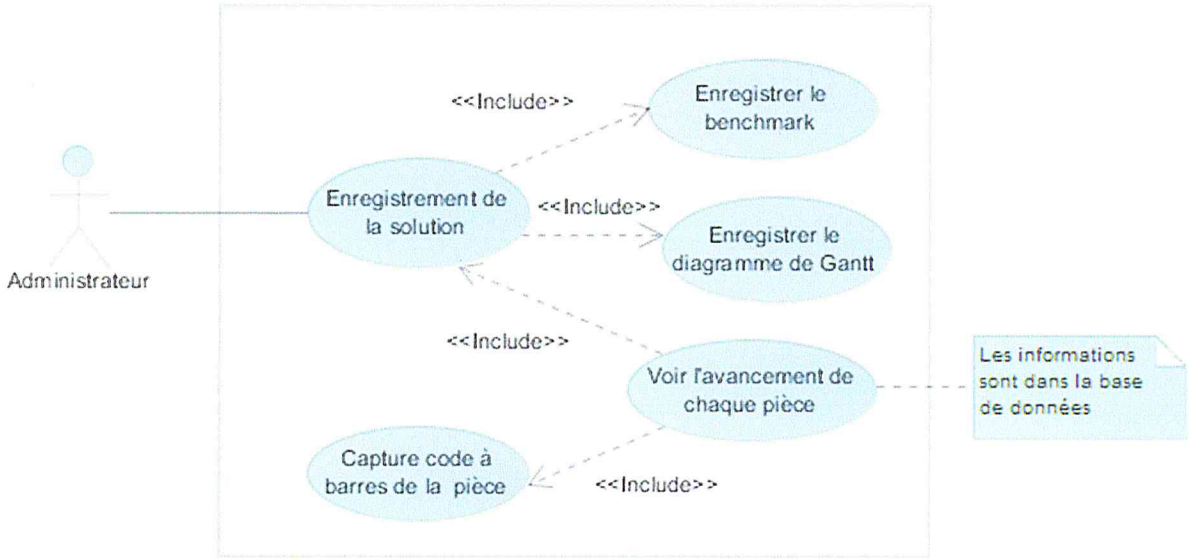


Figure 3.10 : Cas d'utilisation « Enregistrement de la solution ».

- **Description du cas d'utilisation « Enregistrement de la solution »**

Cas d'utilisation	**Description**
Enregistrer le benchmark	L'administrateur doit enregistrer le benchmark dans la base de données.
Enregistrer le diagramme de Gantt	L'administrateur doit enregistrer le diagramme de Gantt de la solution dans la base de données.

Voir l'avancement de chaque pièce	L'administrateur peut voir l'avancement de chaque pièce de la solution qui enregistre.
Capture code à barres de la pièce	L'administrateur doit capter le code à barres de la pièce pour afficher des informations sur cette pièce.

Tableau 3.9: Description du cas d'utilisation « Enregistrement de la solution ».

3.3.1.2. Diagrammes des séquences

Un diagramme de séquence permet de décrire les cas d'utilisation de façon à mettre en évidence les interactions entre les instances des classes (objets) du logiciel, et montre la séquence dans le temps des échanges de messages entre les objets participant à un scénario **[31]**.

Dans notre travail, les diagrammes de séquences sont décrits dans les sections suivantes :

- **Gérer les zones**

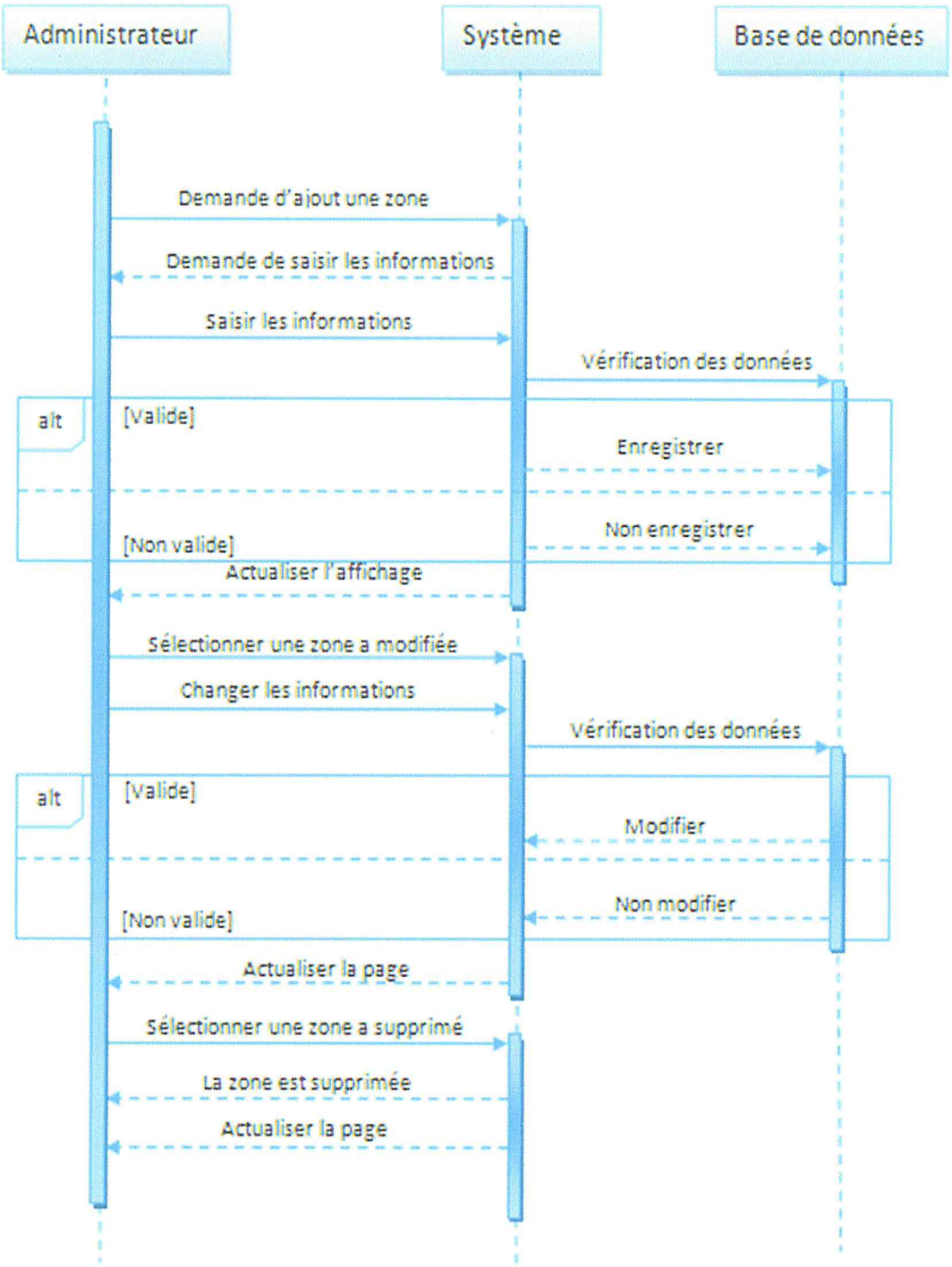


Figure 3.11: Diagramme de séquence « Gérer les zones ».

- **Scénario**

  1. L'administrateur demande d'ajout d'une zone.
  2. Le système demande de saisir les informations.
  3. Vérification des données à la base de données, après l'enregistrement de la zone.
  4. L'administrateur sélectionne une zone pour modifier.

5. Le système demande de saisir les informations, après enregistrer les modifications dans la base de données.

6. L'administrateur sélectionne une zone pour supprimer.

7. Le système doit supprimer la zone.

- **Gérer les types opérations**

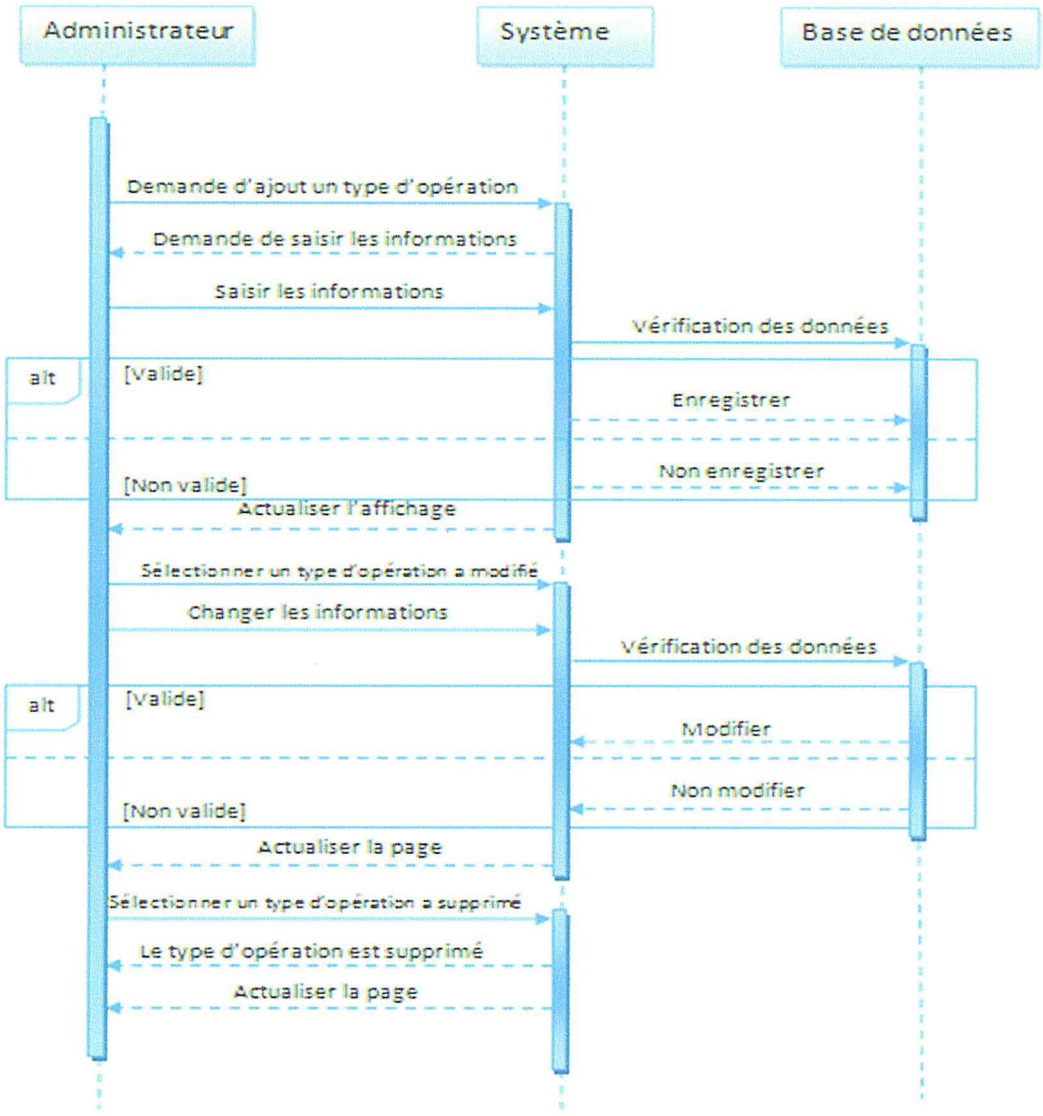


Figure 3.12: Diagramme de séquence « Gérer les types d'opérations ».

- **Scénario**

1. L'administrateur demande d'ajout d'un type d'opération.

2. Le système demande de saisir les informations.

3. Vérification des données à la base de données, après l'enregistrement de type d'opération.

4. L'administrateur sélectionne un type d'opération pour modifier.

5. Le système demande de saisir les informations, après enregistrer les modifications dans la base de données.

6. L'administrateur sélectionne un type d'opération pour supprimer.

7. Le système doit supprimer le type d'opération.

- **Gérer les machines**

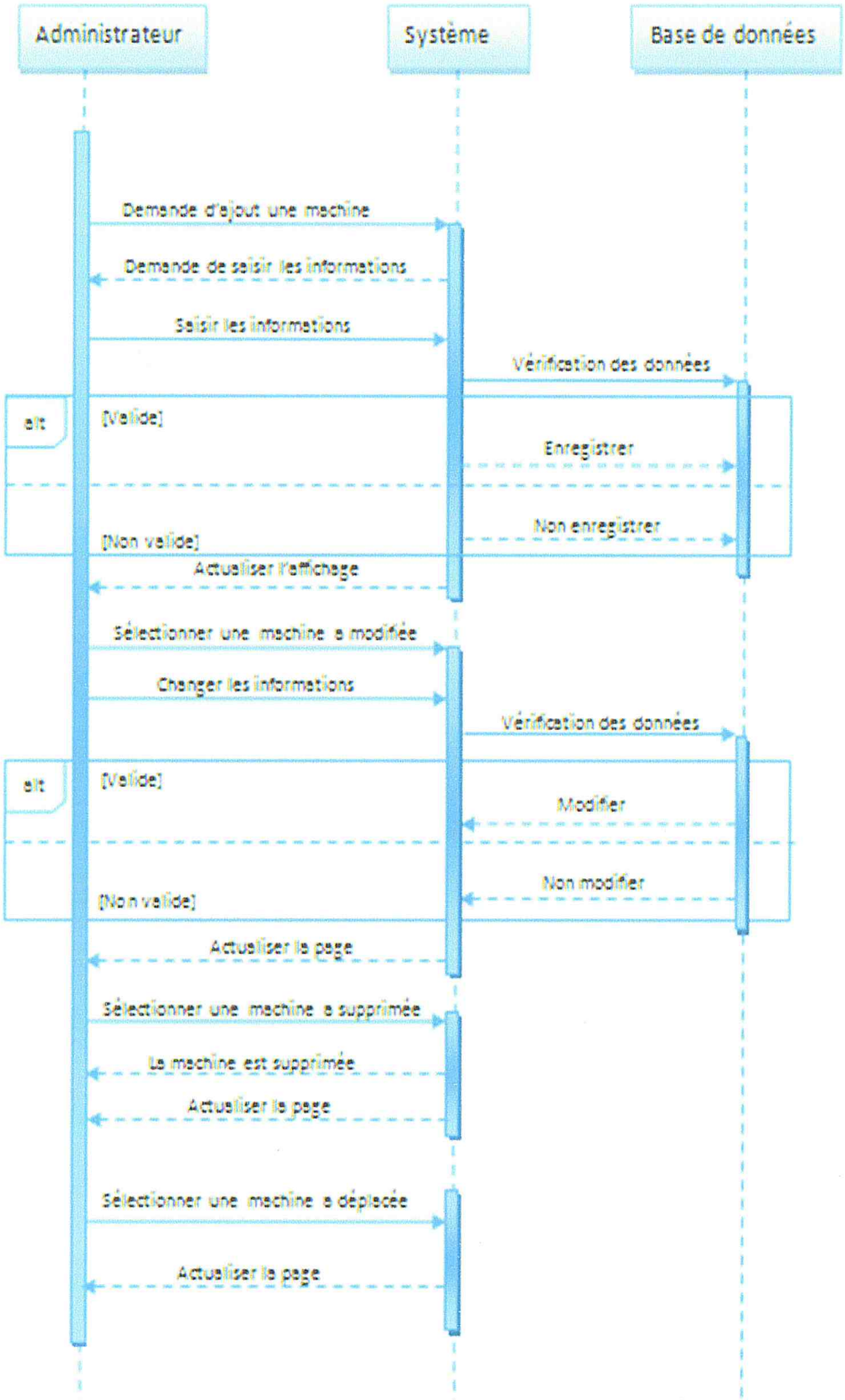


Figure 3.13 : Diagramme de séquence « Gérer les machines ».

- **Scénario**

  1. L'administrateur demande d'ajout d'une machine.
  2. Le système demande de saisir les informations.
  3. Vérification des données dans la base de données, après enregistrement de la machine.
  4. L'administrateur sélectionne la machine pour modifier.
  5. Le système demande de saisir les informations, après enregistrement les modifications dans la base de données.
  6. L'administrateur sélectionne la machine pour supprimer.
  7. Le système supprime la machine.
  8. L'administrateur sélectionne une machine pour déplacer.
  9. Le système déplace la machine.

- **Gérer les opérateurs**

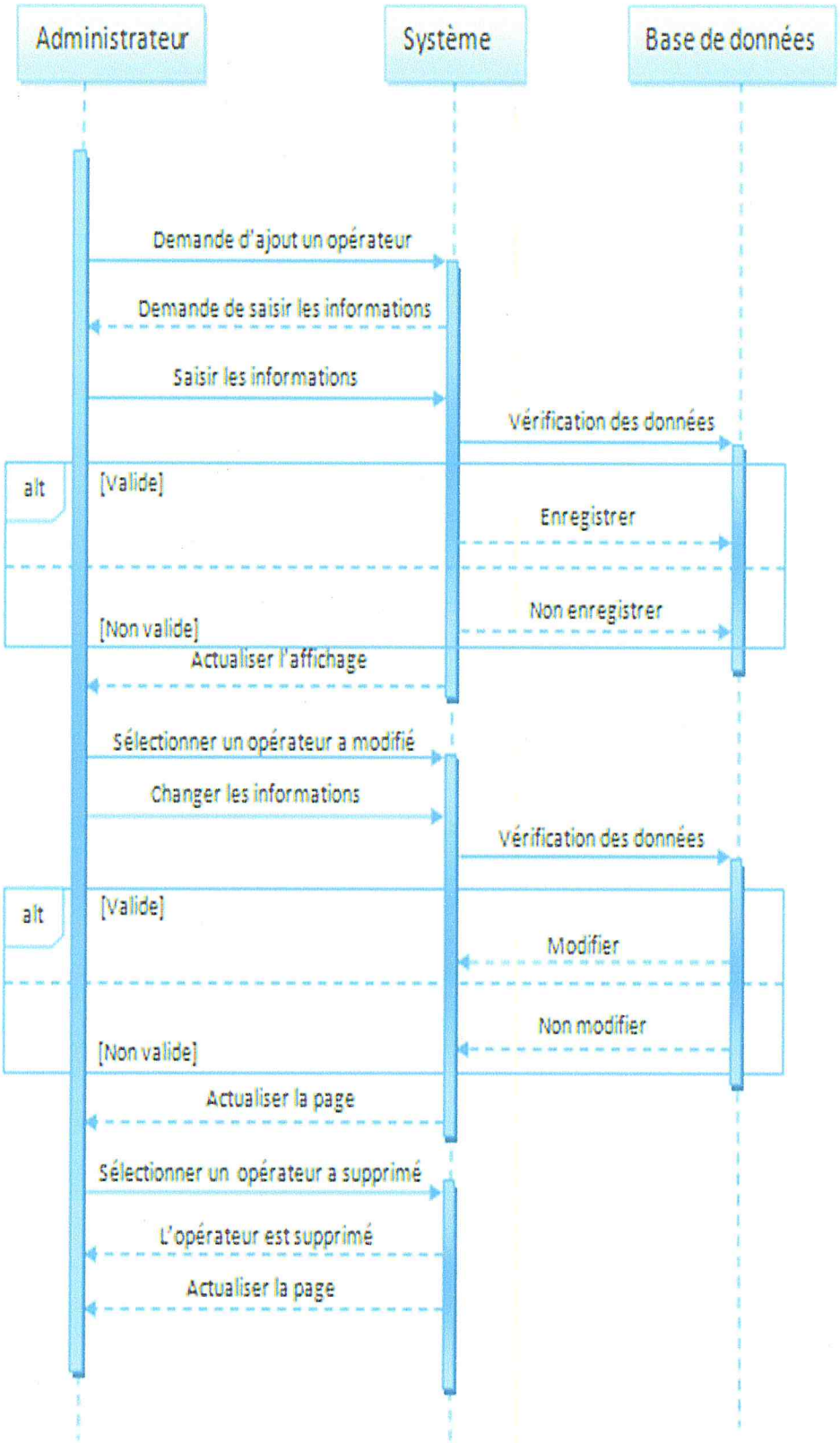


Figure 3.14 : Diagramme de séquence « Gérer les opérateurs ».

- **Scénario**

  1. L'administrateur demande l'ajout d'un opérateur.
  2. Le système demande de saisir les informations.
  3. Vérification des données dans la base de données, puis enregistrement des informations de l'opérateur.
  4. L'administrateur sélectionne un opérateur pour modifier.
  5. Le système demande de saisir les informations, puis enregistrer les modifications dans la base de données.
  6. L'administrateur sélectionne un opérateur pour supprimer.
  7. Le système supprime l'opérateur.

- **Gérer les pièces**

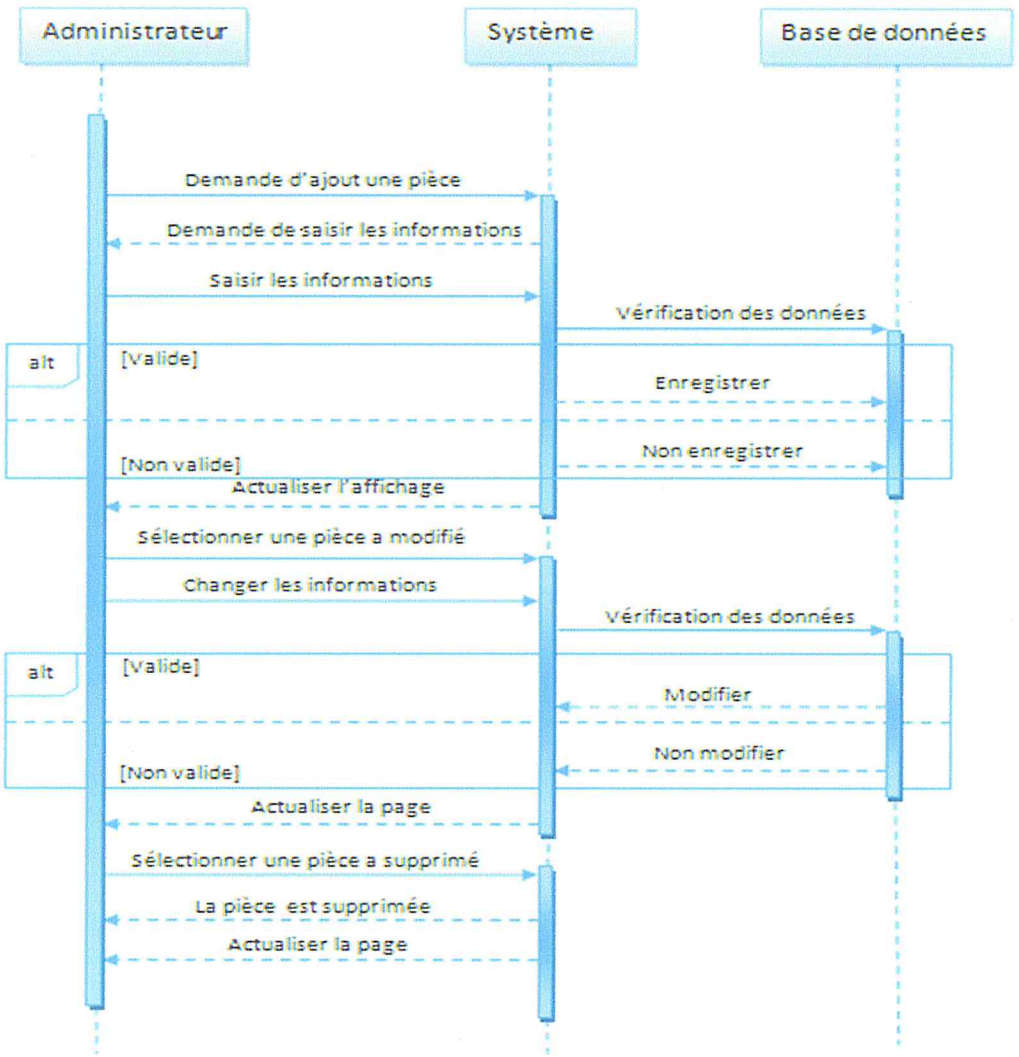


Figure 3.15 : Diagramme de séquence « Gérer les pièces ».

- **Scénario**

  1. L'administrateur demande l'ajout d'une pièce.
  2. Le système demande de saisir les informations.
  3. Vérification des données à la base de données, puis enregistrement de la pièce.
  4. L'administrateur sélectionne une pièce à modifier.
  5. Le système demande de saisir les informations, puis enregistrement les modifications dans la base de données.
  6. L'administrateur sélectionne une pièce à supprimer.
  7. Le système supprime la pièce.

- **Gérer les opérations**

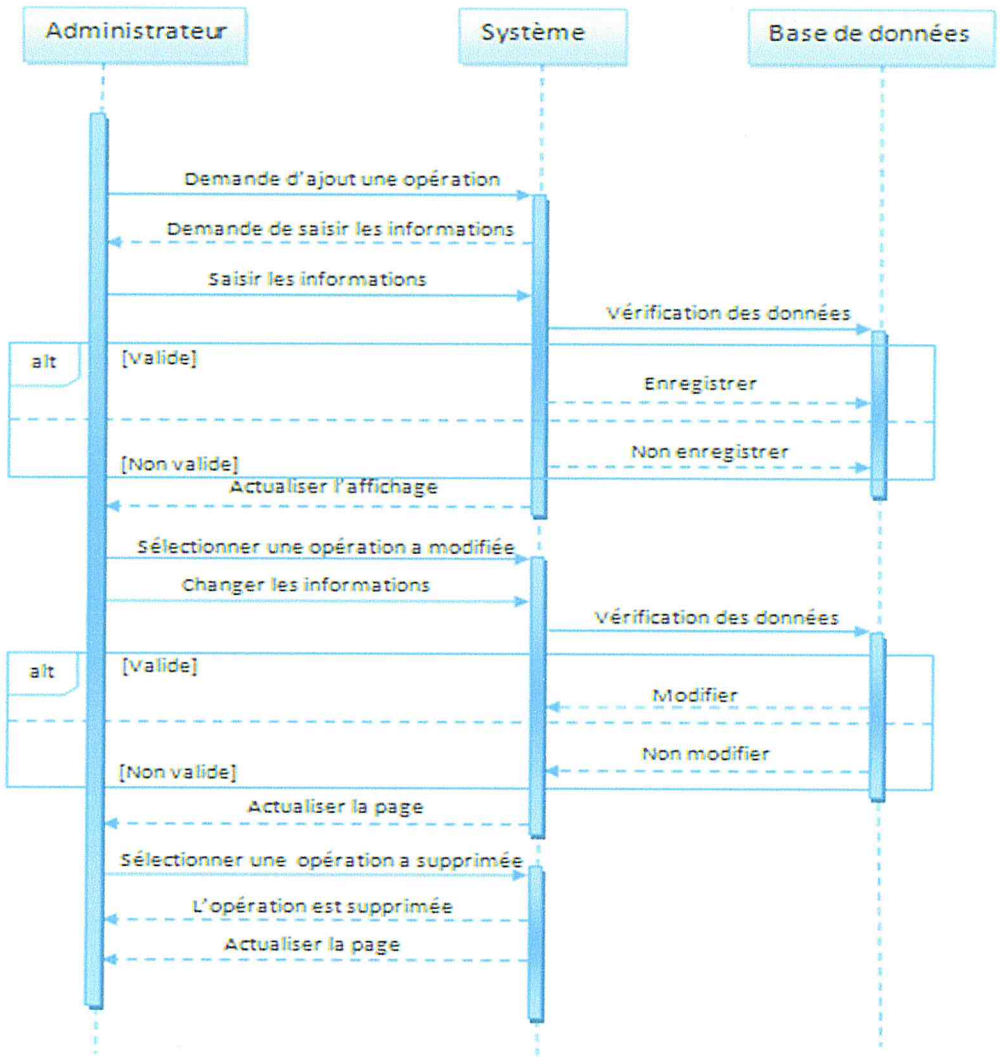


Figure 3.16: Diagramme de séquence « Gérer les opérations ».

- **Scénario**

  1. L'administrateur demande l'ajout d'une opération.
  2. Le système demande de saisir les informations.
  3. Vérification des données à la base de données, puis enregistrement de l'opération.
  4. L'administrateur sélectionne une opération à modifier.
  5. Le système demande de saisir les informations, puis enregistrement les modifications dans la base de données.
  6. L'administrateur sélectionne une opération à supprimer.
  7. Le système supprime l'opération.

- **Gérer l'exécution d'algorithme**

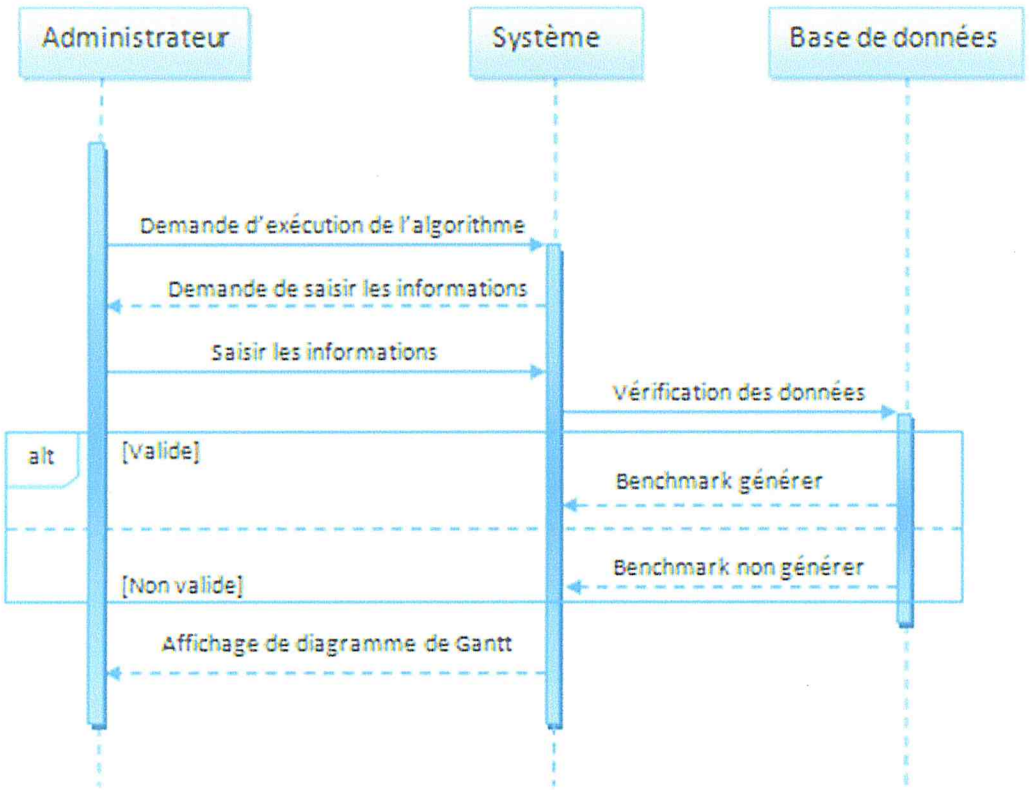


Figure 3.17: Diagramme de séquence « Gérer l'exécution d'algorithme ».

- **Scénario**

  1. L'administrateur demande d'exécuter l'algorithme.
  2. Le système demande de saisir les informations.

3. Vérification des données à la base de données, puis génération du benchmark.
4. Le système affiche le diagramme de Gantt.

- **Enregistrement de la solution**

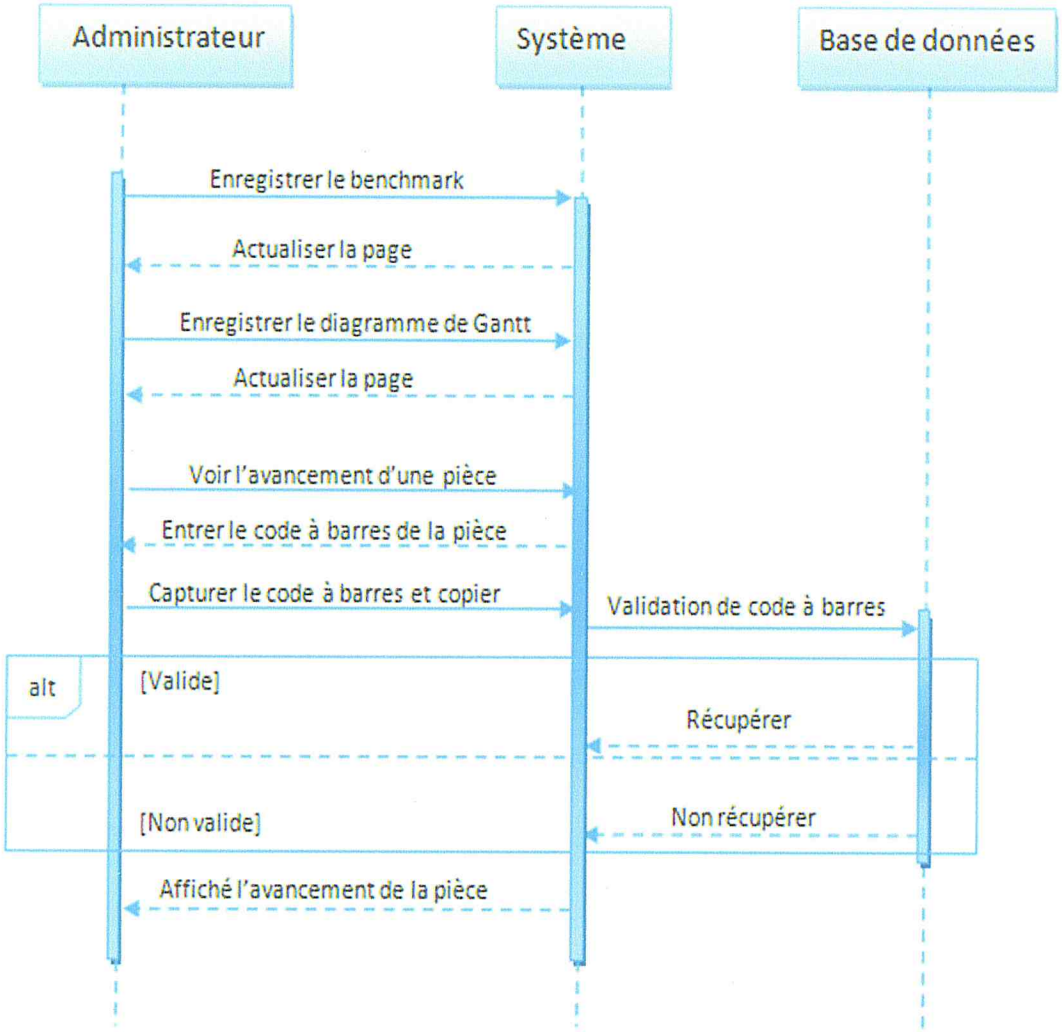


Figure 3.18: Diagramme de séquence « Enregistrement de la solution ».

- **Scénario**

1. L'administrateur enregistrer le benchmark.
2. L'administrateur enregistrer le diagramme de Gantt.
3. L'administrateur voir l'avancement de la pièce.
4. Le système demande de entrer le code à barres de la pièce.
5. L'administrateur capturé le code à barres de la pièce et entrer dans le système.
6. Le système affiché l'avancement de la pièce.

3.2.2. La conception

A ce stade du cycle de vie, on passe à la conception. Cette étape est représentée par le diagramme de classe.

3.2.2.1. Diagramme de classe

Les diagrammes de classes représentent un ensemble de classes, d'interfaces et de collaborations, ainsi que leurs relations. Ce sont les diagrammes les plus fréquents dans la modélisation des systèmes à objets. Ils présentent une vue de la conception statique d'un système [27].

La figure, ci-dessous, représente notre diagramme de classe.

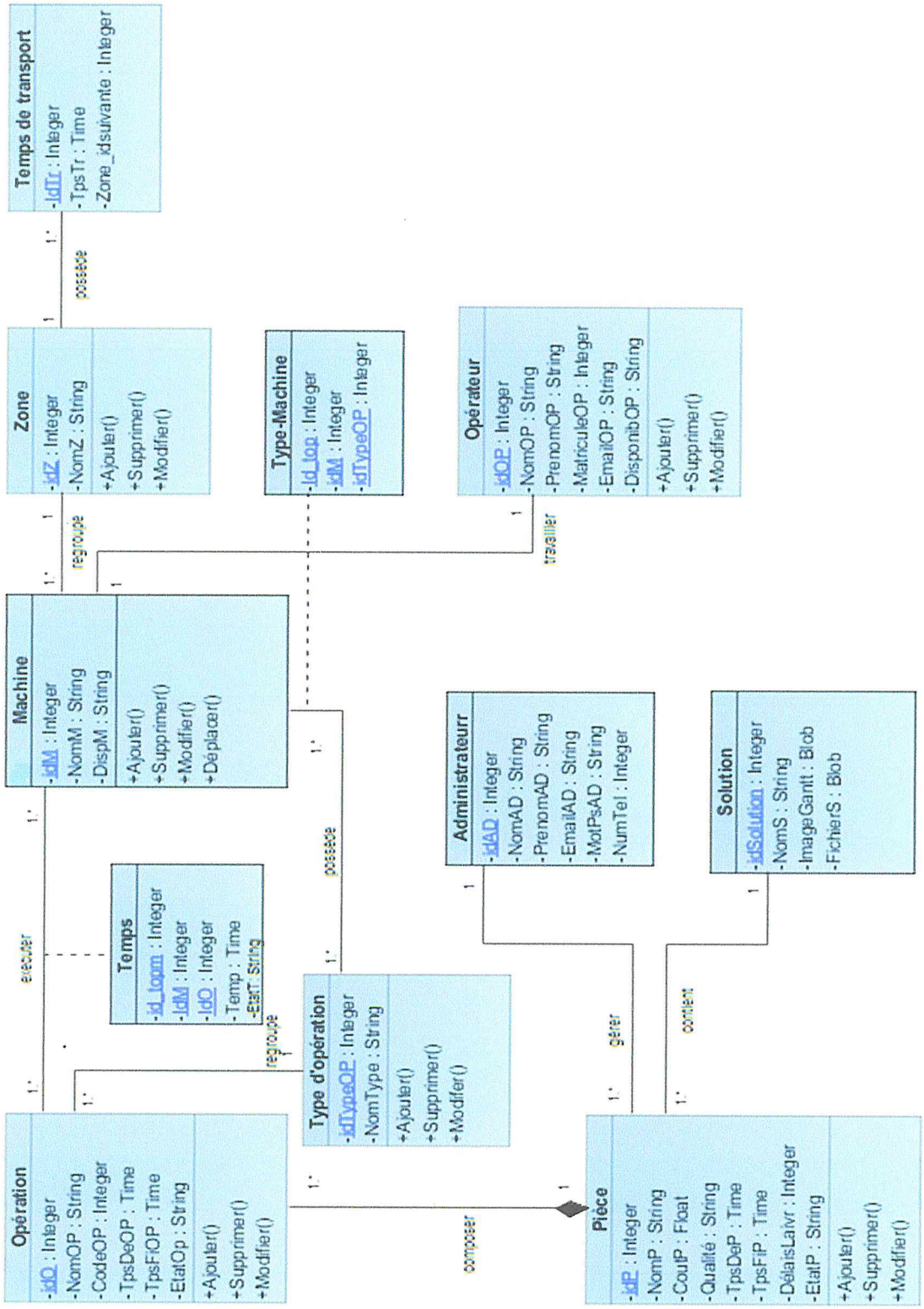


Figure 3.19 : Diagramme de classe.

- **Dictionnaire des données de classes**

Attribut	Description	Type
IdTypeOP	Identifiant du type d'opération	Integer
NomType	Nom du type d'opération	String
IdM	Identifiant de la machine	Integer
NomM	Nom de la machine	String
DispM	Disponibilité de la machine	String
IdZ	Identifiant de la zone	Integer
NomZ	Nom de la zone	String
IdT	Identifient du temps de transport	Integer
TpsTr	Temps de transport entre deux zones	Time
Zone_idsuivante	Numéro de la zone suivante	Integer
IdAD	Identifiant de l'administrateur	Integer
NomAD	Nom de l'administrateur	String
PrenomAD	Prénom de l'administrateur	String
EmailAD	Email de l'administrateur	String
MotPsAD	Mot de passe de l'administrateur	String
NumTel	Numéro de téléphone de l'administrateur	Integer
IdOP	Identifiant de l'opérateur	Integer
NomOP	Nom de l'opérateur	String
PrenomOP	Prénom de l'opérateur	String
MatriculeOP	Matricule de l'opérateur	Integer
EmailOP	Email de l'opérateur	String
DisponibOP	Disponibilité de l'opérateur	String
IdO	Identifiant de l'opération	Integer
NomO	Nom de l'opération	String
CodeO	Code de l'opération	Integer
TpsDeOP	Temps début de l'opération	Time
TpsFiOP	Temps fin de l'opération	Time
EtatOp	Etat de l'opération	String

IdP	Identifiant de la pièce	Integer
NomP	Nom de la pièce	String
QualitéP	Qualité de la pièce	Float
CoutP	Cout de la pièce	Float
TpsDeP	Temps début de la pièce	Time
TpsFiP	Temps fin de la pièce	Time
QuantitéP	Quantité de la pièce	Float
DélaisLivr	Délais de livraison de la pièce	Integer
EtatP	Etat de la pièce	String
Id_topm	Identifiant de temps d'opératoire	Integer
Temp	Temps opératoire pour une machine	Time
EtatT	Etat d'un choix machine	String
IdT	Identifiant du temps de transport	Integer
TpsTr	Temps de transport entre deux zones	Time
idSolution	Identifiant de la solution finale de l'algorithme	Integer
NomS	Nom de la solution finale de l'algorithme	String
ImageGantt	Image de diagramme de Gantt de la solution finale de l'algorithme	Blob
FichierS	Fichier qui présente la solution finale de l'algorithme	Blob
Id_top	Identifiant de type machine	Integer

Tableau 3.10: Dictionnaire des données de classe.

- **Description des données de classe**

Nom de class	Identifient	Attribut	Opération
Type opération	IdTypeOP	IdTypeOP NomType	Ajouter Supprimer Modifier
Machine	IdM	IdM	Ajouter

		NomM DispM	Supprimer Modifier Déplacer
Zone	IdZ	IdZ NomZ	Ajouter Supprimer Modifier
Administrateur	IdAD	IdAD NomAD PrenomAD EmailAD MotPsAD NumTel	
Opérateur	IdOP	IdOP NomOP PrenomOP MatriculeOP EmailOP	Ajouter Supprimer Modifier
Opération	IdO	IdO NomO CodeO TpsDeOP TpsFiOP EtatOp	Ajouter Supprimer Modifier
Pièce	IdP	IdP NomP QualitéP CoutP TpsDeP TpsFiP DélaisLivr EtatP	Ajouter Supprimer Modifier
Temps	Id_topm	Id_topm	

	IdO IdM	IdO IdM Temp EtatT	
Type-Machine	Id_top IdTypeOP IdM	Id_top IdTypeOP IdM	
Temps-Transport	IdT	IdT TpsTr Zone_idsuivante	

Tableau 3.11: Descriptions des données.

- **Dictionnaire des données des relations**

Realtion	**Collection**	**Identifiant**	**Cardinalité**
Regroupe	Zone Machine	IdZ IdM	1.* 1
Possède	Zone Tempe de trasport	IdZ IdT	1 1.*
Possède	Type d'opération Machine	IdTypeOP IdM	1.* 1.*
Dispose	Pièce Opération	IdP IdO	1 1.*
Regroupe	Type d'opération Opération	IdTypeOP IdO	1.* 1
Exécuter	Opération Machine	IdO IdM	1.* 1.*
Diriger	Administrateur Pièce	IdAD IdP	1.* 1
Possède	Pièce Solution	IdP IdSolution	1.* 1

Travailler	Opérateur	IdM	1
	Machine	IOP	1

Tableau 3.12 : Dictionnaire des données des relations.

- **Passage relationnelle**

Zone (IdZ).

Temps de transport (IdT ,# IdZ).

Machine (IdM, NomM,# IdZ).

Type d'opération (IdTypeOP, NomType).

Opération (IdO, NomOP,#IdP).

Opérateur (IdOP, #IdM).

Pièce (IdP, NomP, #IdAD).

Solution (IdSolution)

Administrateur (IdAD).

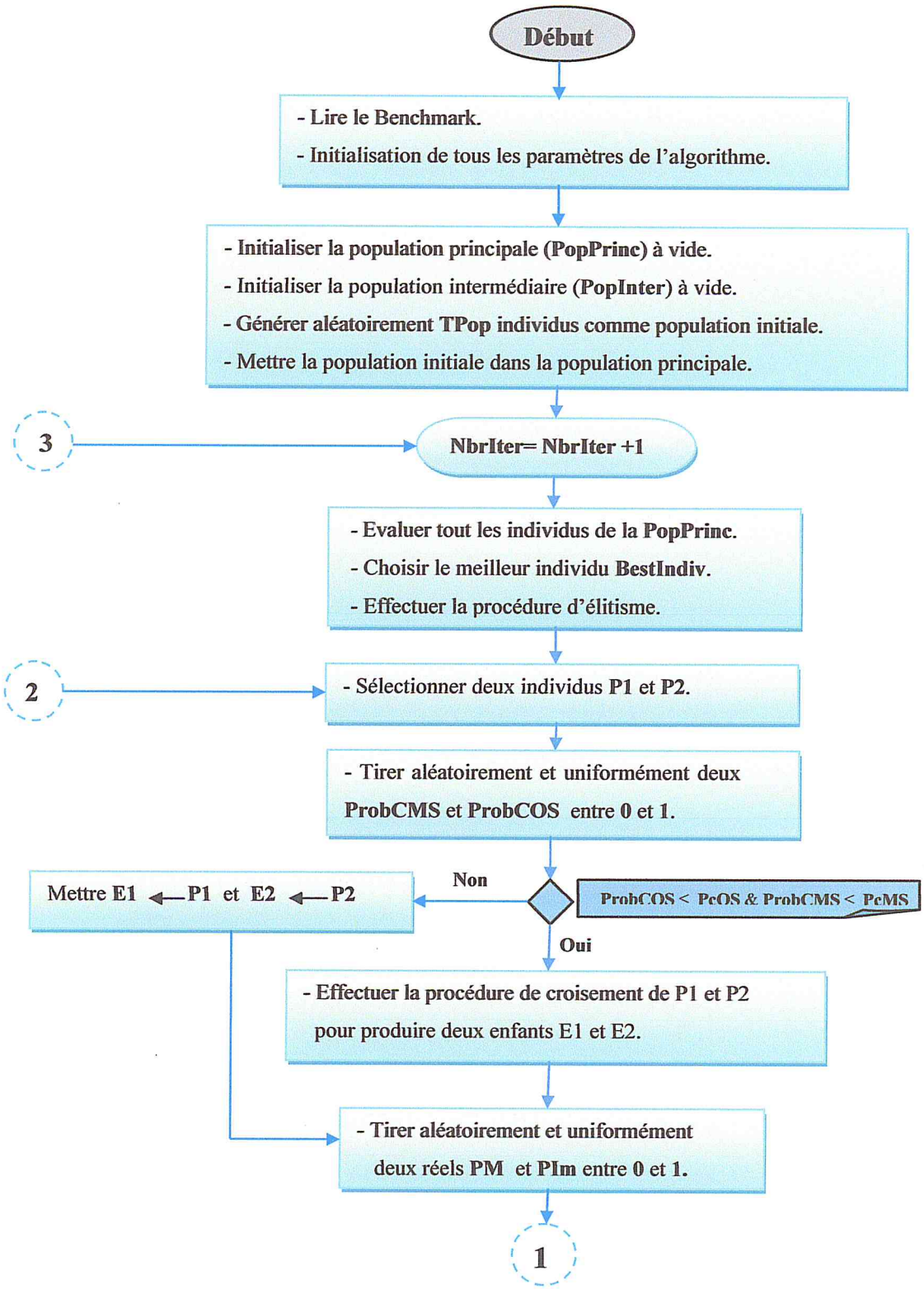
Temps (#IdM,# IdO).

Type-Machine (#IdM,# IdTypeOP).

3.3. Implémentation de l'algorithme génétique

Les algorithmes génétiques sont considérés par plusieurs chercheurs comme l'une des méthodes les plus adaptées au problème de Job Shop Flexible, même si ils ne permettent pas toujours d'atteindre l'optimum dans certains cas difficile **[22]**.

Notre application se base sur l'implémentation d'un algorithme génétique standard. Cependant, nous avons utilisée un codage et des opérateurs améliorés, basés sur des connaissances spécifiques du problème. L'algorithme implémenté est détaillé par l'organigramme de la **Figure 3.20**. Les différentes étapes de ce dernier sont présentées, en détail dans les sections suivantes.


Début
- Lire le Benchmark.
- Initialisation de tous les paramètres de l'algorithme.
- Initialiser la population principale (PopPrinc) à vide.
- Initialiser la population intermédiaire (PopInter) à vide.
- Générer aléatoirement TPop individus comme population initiale.
- Mettre la population initiale dans la population principale.
3
NbrIter= NbrIter +1
- Evaluer tout les individus de la PopPrinc.
- Choisir le meilleur individu BestIndiv.
- Effectuer la procédure d'élitisme.
2
- Sélectionner deux individus P1 et P2.
- Tirer aléatoirement et uniformément deux ProbCMS et ProbCOS entre 0 et 1.
Non
ProbCOS < PcOS & ProbCMS < PcMS
Mettre E1 ← P1 et E2 ← P2
Oui
- Effectuer la procédure de croisement de P1 et P2 pour produire deux enfants E1 et E2.
- Tirer aléatoirement et uniformément deux réels PM et PIm entre 0 et 1.
1

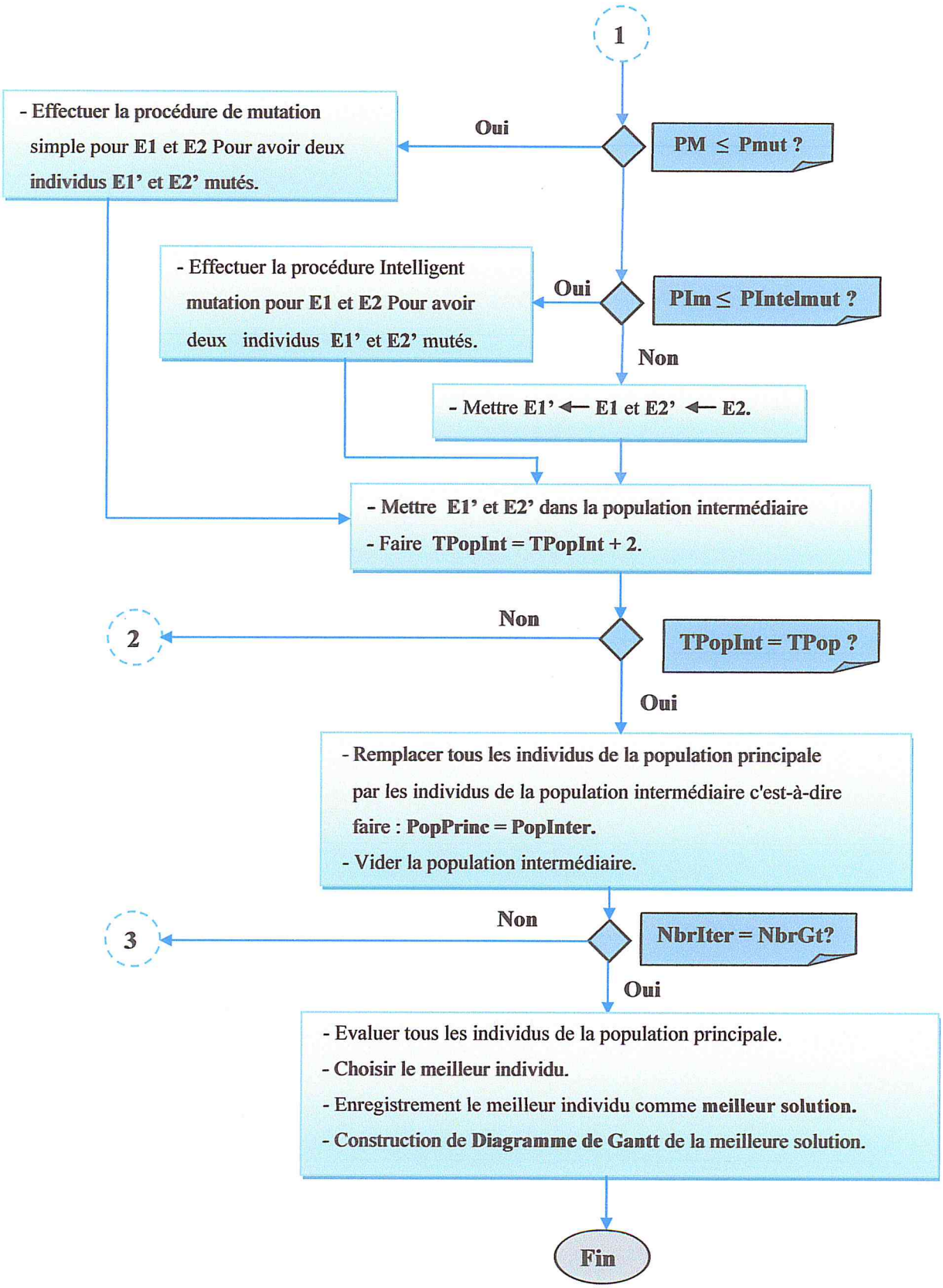


Figure 3.20 : Organigramme de l'algorithme génétique implémenté.

3.3.1. Les paramètres de l'algorithme génétique

1- Taille de la population **TPop**.

2- Nombre de génération **NbrGt**.

3- Probabilité de croisement de la partie (OS) du chromosome **PcOS**.

4- Probabilité de croisement de la partie (MS) du chromosome **PcMS**.

5- Probabilité de mutation **Pmut**.

6- Probabilité de l'intelligente mutation **PIntelmut**.

7- Elitisme **Elit**.

3.3.2. Codage de chromosome utilisé

L'individu est représenté par deux parties. Une partie pour coder la séquence des opérations « Opération sélection **OS** », et une autre pour le codage des choix des machines pour chaque opération « Machine sélection **MS** ». Chaque opération est affectée à son choix de la machine.

Le tableau suivant donne un exemple d'un problème de type Job Shop Flexible.

Jobs	Opération	M1	M2	M3	M4	M5
Produit 1	O_{11}	2	6	5	3	4
	O_{12}	-	8	-	4	-
	O_{13}	5	-	2	3	-
Produit 2	O_{21}	3	-	6	-	5
	O_{22}	4	6	5	-	-
	O_{23}	-	7	11	5	8
Produit 3	O_{31}	-	3	-	6	2
	O_{32}	4	5	2	-	4

Tableau 3.13 : Exemple d'un problème de type Job Shop Flexible.

- **Partie machine sélection (MS)**

Elle est constituée 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 jobs (produits) du problème (**Tableau 3.1**).

La figure suivante représente un chromosome de l'une des solutions du problème de l'exemple du (**Tableau 3.1**).

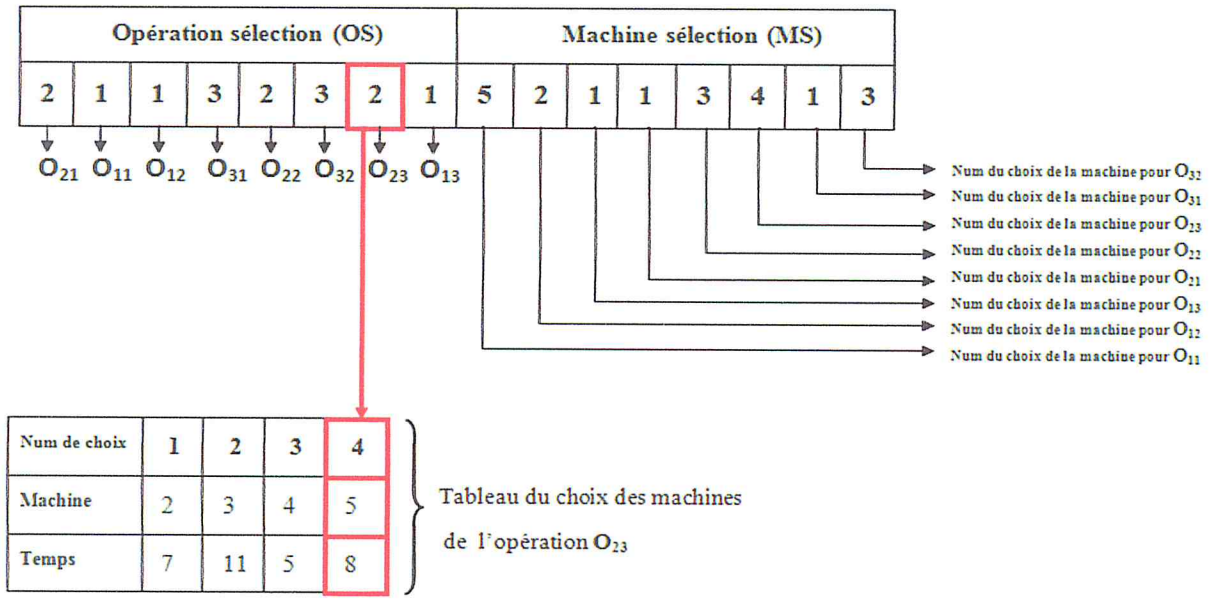


Figure 3.21 : Présentation d'une solution.

- **Partie opération sélection (OS)**

Dans cette partie, le codage utilisé est basé sur les opérations. Ce codage représente un ordonnancement comme une suite d'opérations. Chaque opération est codée par un gène dans la partie (OS) du chromosome tel que toutes les opérations d'un job portent le même symbole (le numéro de job). L'interprétation de ce symbole se fait suivant l'ordre de son occurrence dans la liste (**Figure 3.21**).

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

3.3.3. Les étapes de l'algorithme génétique

3.3.3.1 Génération de la population initiale

Dans cette étape, une population initiale doit être générée, où chaque chromosome représente une solution réalisable du problème. La taille de population initiale est définie par le paramètre **TPop**.

3.3.3.2. Evaluation des individus

Le seul critère utilisé pour l'évaluation des individus est le **makespan** de l'ordonnancement, en incluant le temps de transport. L'évaluation du ***fitness*** des chromosomes ce fait selon trois étapes :

- La construction de l'ordonnancement correspondant au chromosome.
- Le calcule du makespan de cet ordonnancement.
- L'attribution au chromosome de son makespan comme ***fitness***.

La figure suivante représente le diagramme de Gantt de la solution de **(Figure 3.3)**.A partir de ce diagramme on peut conclure le makespan (**makespan**=16).

Remarque : Le temps début de chaque opération d'une pièce est défini comme suivant :
Temps_Début_Opération = Max (Temps_Fin_Machine, Temps_Fin_Opération_Précédente).

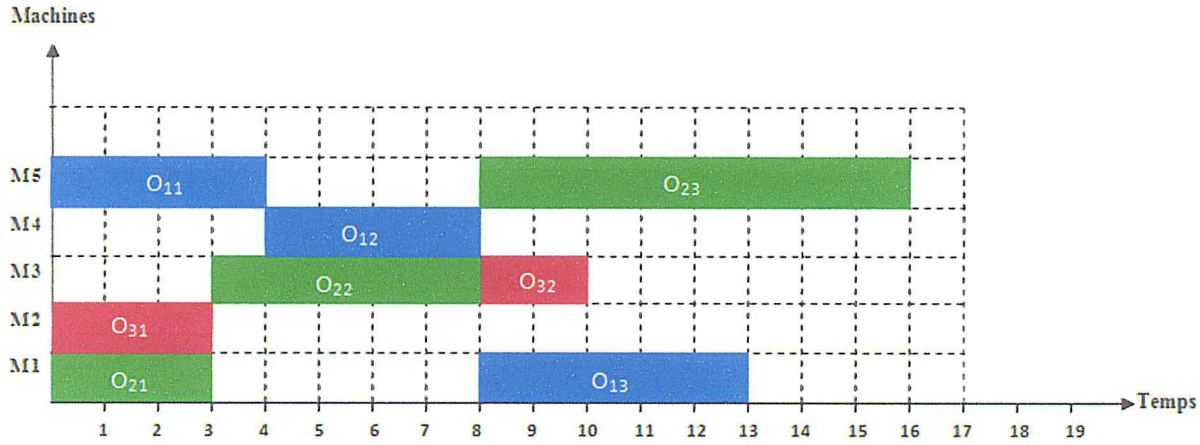


Figure 3.22 : Diagramme de Gantt de la solution.

3.3.3.3. Sélection des individus

La sélection permet d'identifier les individus susceptibles d'être croisés dans la population. Il existe plusieurs stratégies de sélection **[19]**.

La stratégie de la sélection des individus utilisée est illustrée par les étapes suivantes :

Étape 1 : - Sélectionner aléatoirement dans la population deux individus **Individu 1** et **Individus 2.**

- Choisir le meilleur individu entre **Individu 1** et **Individu 2** (Le choix est fait selon la fitness de chaque individu).
- Mettre **Parent 1** comme meilleure individu.

Étape 2 : - Sélectionner aléatoirement dans la population deux individus **Individu 3** et **Individu 4.**

- Choisir le meilleur individu entre **Individu 3** et **Individu 4** (Le choix est fait selon la fitness de chaque individu).
- Mettre **Parent 2** comme meilleure individu.

Étape 3 : Retourner **Parent 1** et **Parent 2** comme les deux individus sélectionnés.

3.3.3.4. L'élitisme

Afin de ne pas perdre le meilleur individu avant les opérations de la reproduction (Sélection, croisement, mutation), on procède à l'élitisme qui consiste à copier deux fois le meilleur individu dans la nouvelle génération.

3.3.3.5. Le croisement

Le croisement est un opérateur très important dans l'algorithme génétique. Ainsi, plusieurs types de croisements spécifiques au domaine sont proposés **[22]**.

Dans notre application un type de croisement est adapté pour chaque partie d'individu. La partie (OS) utilise un croisement basé sur les opérations et la partie (MS) utilise deux types de croisement, un croisement uniforme et un croisement en deux points.

➢ **Croisement uniforme**

La figure suivante présente un exemple de croisement uniforme de deux parties (MS) de deux parents **P1(MS)** et **P2(MS)** pour avoir deux parties (MS) de deux enfants **E1(MS)** et **E2(MS)**.

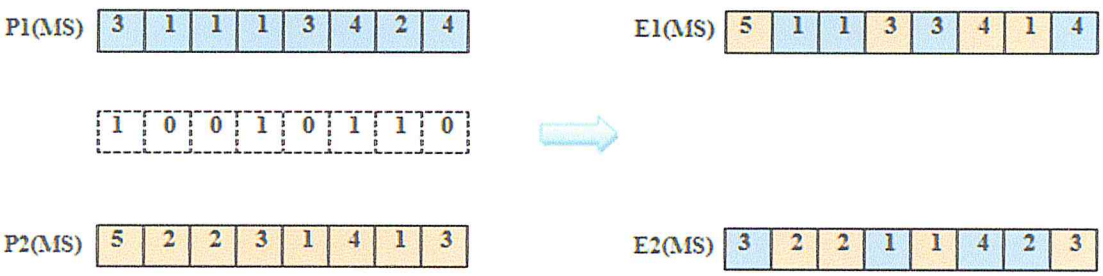


Figure 3.23 : Exemple de croisement uniforme de deux parties (MS) **[24]**.

➢ **Croisement en deux points**

La figure suivante montre un exemple de croisement en deux points de deux parties (MS) de deux parents **P1(MS)** et **P2(MS)** pour avoir deux parties (MS) de deux enfants **E1(MS)** et **E2(MS)**.

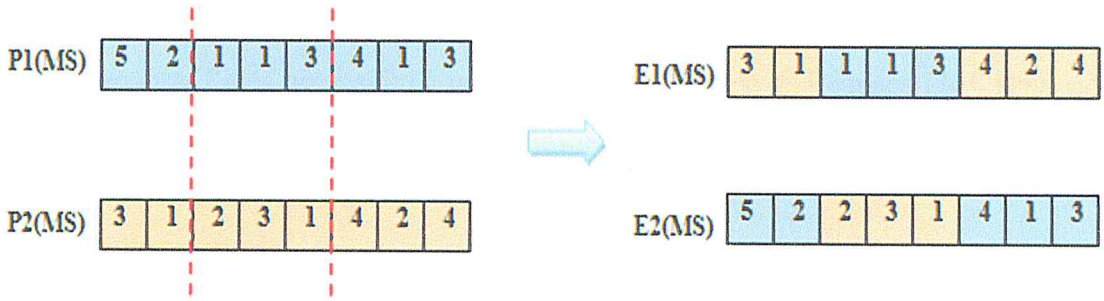


Figure 3.24 : Exemple de croisement en deux points de deux parties (MS) [24].

➢ **Croisement de la partie (OS)**

La stratégie de croisement adaptée à la partie (OS) est **POX,** qui est utilisé par **[Zhang et al., 08].**

Le principe de **POX** se résume comme suit :

A partir d'un parent **P1**:

- Détermination du point de croisement (une tâche est déterminée).
- Copier toutes les opérations de la tâche dans les mêmes emplacements dans l'enfant.
- Remplir les emplacements vides dans l'enfant avec les opérations abstenues du deuxième parent, dans leurs ordres d'apparition, en ignorant les opérations qui appartiennent à la tâche correspondant au point de croisement.

Remarque : Cette procédure montre le croisement de **P1(OS)** et **P2(OS)** pour avoir **E1(OS).** Alors pour avoir **E2(OS),** il suffit d'inverser les deux parents **P2(OS)** et **P1(OS)** et on applique la même stratégie **POX.**

La figure suivante représente un exemple de la stratégie **de POX**

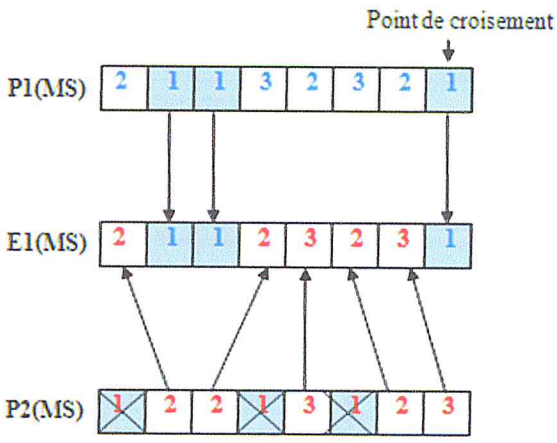


Figure 3.25 : Exemple de croisement POX de la partie (OS).

- **La mutation intelligence**

La stratégie de mutation intelligence est déterminée selon les étapes suivantes :

Étape 1 : Déterminer la machine la plus chargée.

Étape 2 : Tirer aléatoirement une opération qui à plus d'un choix de machine c'est-à-dire plusieurs choix de machine.

Étape 3 : Affecter cette opération à une autre machine sélectionné aléatoirement parmi ses chois de machine afin d'obtenir un individu muté **E1'**.

3.3.3.7. L'insertion

Après l'étape de mutation, on utilise une méthode d'insertion pour générer une nouvelle population. Dans notre travail, la stratégie suivie consiste à insérer les nouveaux individus (Enfants), qui ont été déjà créés par les opérateurs de croisement et mutation. L'ancienne génération qui contient les parents est supprimée.

3.3.3.8. Le critère d'arrêt

Le test d'arrêt joue un rôle très important dans le jugement de la qualité des individus.

Le critère d'arrêt utilisé est lorsque le nombre de génération sera égal au nombre fixé d'itération. Après **NbrGt** générations, l'algorithme génétique s'arrête et donne le meilleur chromosome qui possède un C_{max} minimum.

3.4. Conclusion

Dans ce chapitre nous avons présenté, d'une manière générale les algorithmes génétiques et leur application pour la résolution du problème d'ordonnancement : cas du problème Job shop Flexible. Ainsi qu'une présentation détaillée de la partie conception de notre solution notamment les diagrammes de cas d'utilisation, de classe, diagrammes de séquence.

Les phases de développement et de tests de notre solution seront détaillées dans le chapitre qui suit.

Chapitre 4 : Implémentation, expérimentations, résultats et discussions.

4.1. Introduction

Nous aborderons dans ce chapitre les aspects relatifs au développement et l'implémentation de notre solution. Ce chapitre est divisé en deux parties. La première partie est consacrée à la définition des outils de développement utilisés pour l'implémentation de notre solution ainsi que les fonctionnalités de notre application. L'objectif de la deuxième partie est de données une synthèse des résultats obtenus lors de l'application de l'algorithme génétique pour la résolution du problème d'ordonnancement Job Shop Flexible.

4.2. Environnement de travail

4.2.1. Matériels utilisés

- Un ordinateur portable HP I5.
- Un ordinateur portable ACER I3.

4.2.2. Le langage de programmation (Java EE)

Pour la réalisation de notre solution nous avons utilisé le langage de programmation Java EE 5. En effet, Java Platform, Enterprise Edition 5 (Java EE 5), vise à rendre le développement plus facile, tout en conservant la richesse de la plate-forme J2EE. Les services offerts tels que les services JavaServer Faces (JSF) de la technologie et Web API, Java EE 5 rend le codage plus simple, mais maintient la puissance qui a établi Java EE comme plate-forme de premier plan pour les services Web et le développement d'applications d'entreprise **[32]**.

- **Entreprise application**

Une application d'entreprise est une collection d'applications Web et de modules EJB configurés pour s'interagissent notamment lorsqu'ils sont déployés sur un serveur d'applications Java EE.

Une application d'entreprise n'a pas de fichiers source. Elle ne contient que des descripteurs de déploiement et d'autres fichiers de configuration. Au moment de la compilation, les fichiers d'archives (fichiers JAR et des fichiers WAR) pour chacun des modules de l'application de l'entreprise sont conçus et assemblés en une seule entité (EAR). Ce fichier est déployé sur le serveur d'application **[32]**.

La plate-forme Java EE est particulièrement intéressante dans des environnements fortement distribués. Cette simplification est rendue possible notamment par :

- l'utilisation des annotations.

- Gestion de transaction.
- Sécurité d'accès.
- la mise en œuvre de valeurs par défaut qui répondent à la plupart des besoins (configuration par exception).
- le descripteur de déploiement est facultatif.
- l'utilisation de JPA pour les Beans de type Entity.

La figure suivante représente l'architecture d'une entreprise application.

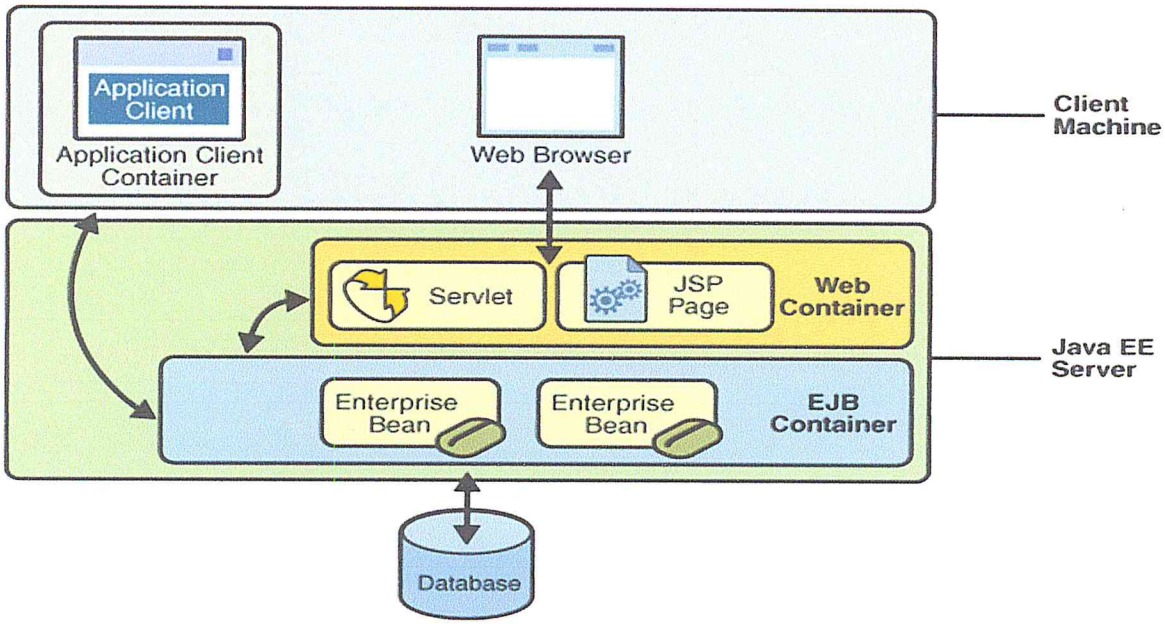


Figure 4.1 : L'architecture d'une entreprise application.

4.2.3. L'architecture logicielle (Le modèle MVC)

Le patron d'architecture logicielle modèle-vue-contrôleur, tout comme les patrons ou présentation, abstraction, contrôle est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

L'idée est de bien séparer les données, la présentation et les traitements. Il en résulte trois parties décrites comme suit :

- Une vue (Interaction avec l'utilisateur et la présentation des données).
- Un contrôleur (Contrôle des actions de l'utilisateur et des données).
- Le modèle (s'occupe de l'ordonnancement par l'algorithme implémenté, le stockage et la mise à jour des données).

Les différentes interactions entre le modèle, la vue et le contrôleur sont résumées par le schéma de la figure suivante :

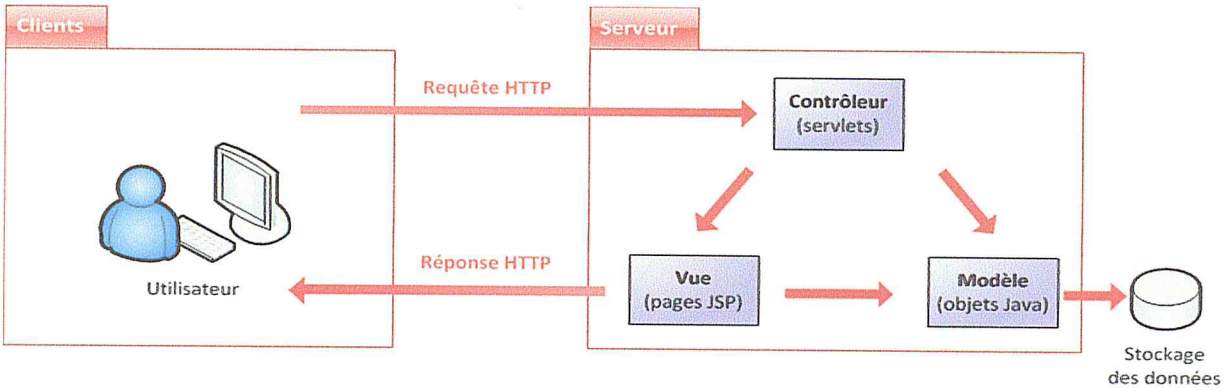


Figure 4.2 : Interaction entre le modèle, et la vue et le contrôleur.

- **La vue**

La vue fait l'interface avec l'utilisateur qui est représenté à travers des pages JSP. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir tous les actions de l'utilisateur (clic de souris, sélection d'une entrées, boutons, …). Ses différents événements sont envoyés au contrôleur. La vue n'effectue pas de traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

- **Le modèle**

Le modèle contient les données manipulées par le programme. Il représente algorithmique de l'ordonnancement, interactions avec la base de données, etc. Il décrit les données manipulées par l'application. Il regroupe la gestion de ces données et est responsable de leur intégrité. La base de données sera l'un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle ne s'occupent pas de la présentation.

- **Le contrôleur**

Le contrôleur est chargé de la synchronisation du modèle et de la vue, qui sont les servlets dans notre application. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier [32].

4.2.4. Les outils et langage utilisés

Certains outils et langages sont utilisés pour la réalisation de notre logiciel sont présenté comme suivant :

4.2.4.1. Netbeans

Netbeans est un environnement de développement intégré (EDI), Il permet de supporter différents langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP, HTML et Java afin de développé rapidement et facilement des applications. Il constitue par ailleurs une plate forme qui permet le développement d'application spécifiques .L'IDE Netbeans s'appuie sur cette plate forme. Il propose des fonctions de complétion, de contrôles syntaxiques et sémantiques, d'avertissements et de conseils, de reprise de codes

L'environnement de base comprend les fonctions générales suivantes:

- configuration et gestion de l'interface graphique des utilisateurs.
- support de différents langages de programmation.
- traitement du code source (édition, navigation, formatage, inspection..).
- fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder.
- accès et gestion de base de données, serveurs Web, ressources partagées [34].

4.2.4.2. GlassFish

GlassFish est un serveur d'applications Java EE dont le développement a été initié et est aujourd'hui encore dirigé par SUN. Il surpasse les capacités de Tomcat par exemple qui ne propose qu'un conteneur de JSP/servlet sans le support des EJB entre autres, et aussi est prend en charge Enterprise Edition (la plate-forme Java EE) et est construit sur un système modulaire. Les entreprises peuvent créer et déployer des applications Web avec le léger Java EE 6 Profil Web et facilement tirer parti de la puissance de la pleine plate-forme Java EE 6 pour les applications d'entreprise. [32]

4.2.4.3. HTML 5

HTML (HyperText Markup Language) est un langage informatique utilisé sur internet. Ce langage est utilisé pour créer des pages web. Il permet de mettre en forme du contenu. Les balises permettent de mettre en forme le texte et de placer des éléments interactif, tel des liens, des images ou bien encore des animations. Pour visualiser une page en HTML il est nécessaire d'utiliser un navigateur web. La dernière version développée est HTML5, qui offre une interface commune pour faire des éléments de chargement plus facile **[32]**.

4.2.4.4. CSS 3 (Cascading Style Sheets)

Les CSS (Cascading Style Sheets) sont des feuilles de style servent à mettre en forme documents web, type page HTML ou XML. Par l'intermédiaire de propriétés d'apparence (couleurs, bordures, polices, etc.) et de placement (largeur, hauteur, côte à côte, dessus-dessous, etc.), le rendu d'une page web peut être intégralement modifié sans aucun code supplémentaire dans la page web. La dernière évolution du langage CSS est CSS3 il apporte beaucoup de nouveautés attendues, comme les angles arrondis, les ombres, les dégradés, les transitions ou animations...etc **[35]**.

4.2.4.5. JavaScript

JavaScript est un langage de script incorporé dans un document HTML.C'est un langage orienté objet à prototype. Ce langage est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes du côté client, c'est-à-dire au niveau du navigateur et non du serveur web **[36]**.

4.2.4.6. SQL

Le SQL (Structured Query Language) est un langage de requête structurée permettant de traiter les opérations dans la base de données relationnelle.

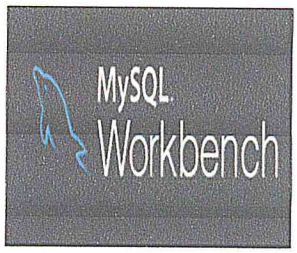
- définition les données : création des tables, des champs dans une base de données, insérer les données...etc.

- manipulation les données : sélectionner, insérer, modifier ou supprimer des données dans une table d'une base de données relationnelle.
- créer et gérer les connexions aux serveurs de base de données **[37]**.

4.2.4.7. MySQL

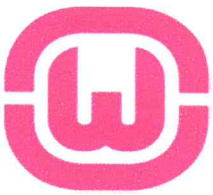
MySQL est un serveur de base de données relationnelle, qui supporte le langage de requêtes SQL pour le traitement des données dans la base de données. Il puisse être accessible par la plupart des langages de programmation, et est devenu une alternative populaire à des systèmes de bases de données en raison de sa vitesse et de fiabilité **[38]**.

4.2.4.8. MySQL Workbench



C'est un logiciel qui présente un bon environnement via des outils graphiques de modélisation de base de données MySQL, il fournit des outils d'administration complets pour la configuration de serveur ainsi il offre la possibilité d'exécuter des requêtes SQL sur les connexions de base de données et de créer rapidement de nouvelles bases de données MySQL **[39]**.

4.2.4.9. WampServer



WampServer est une plate-forme de développement web sous Windows pour des applications web dynamiques à l'aide du serveur Apache2, et un environnement qui fonctionne localement (sans se connecte à un serveur externe), comprenant deux serveurs (Apache et MySQL) **[40]**.

4.2.4.10. JDBC

JDBC (Java DataBase Connectivity) est une interface de programmation créée par Sun Microsystems. Elle permet aux applications Java d'accéder par le biais d'une interface commune à des sources de données pour lesquelles il existe des pilotes JDBC. Il s'agit d'une base de données relationnelle, et des pilotes JDBC sont disponibles pour tous les systèmes connus de bases de données relationnelles **[41]**.

4.2.4.11. WebCam

Une webcam, parfois cybercaméra ou webcaméra, est une caméra conçue pour être utilisée comme un périphérique d'ordinateur, et qui produit une vidéo dont la finalité n'est pas d'atteindre une haute qualité, mais de pouvoir être transmise en direct au travers d'un réseau, typiquement Internet. Cela dit, la norme HD (haute définition) tend à se généraliser avec l'Internet haut débit **[42]**.

4.2.4.12. Visuel paradigme

Visual Paradigm For UML : permet la création des diagrammes UML et des modèles qui en sont à l'origine. Ceux-ci peuvent alors générer du code dans un langage de programmation déterminé. Il propose également la création d'autres types de diagrammes, comme celui qui permet la modélisation des bases de données pouvant, lui aussi, générer des canevas d'applications basés sur des Framework et Pattern mais en plus, générer du code SQL qu'il peut ensuite déployer automatiquement dans différents environnements **[43]**.

4.3. Présentation de l'application

Dans cette partie nous présenterons les plus importantes fonctionnalités de notre application afin de tester si les objectifs fixés lors d'expression des besoins sont atteints.

Voici les principales interfaces dans notre application :

4.3.1. Interface principale

L'interface principale, ci-dessous, s'affiche lors du lancement du notre application.

Figure 4.3 : Interface principale.

L'interface principale de notre application contient un formulaire « **Login Administrateur** » réservé pour l'administrateur, pour la création des comptes utilisateurs et leurs mots de passe.

4.3.2. Interface du menu principale

Après l'inscription, l'administrateur saisie ses identifiants et clique sur le bouton « **Login** », pour accéder à l'interface suivante :

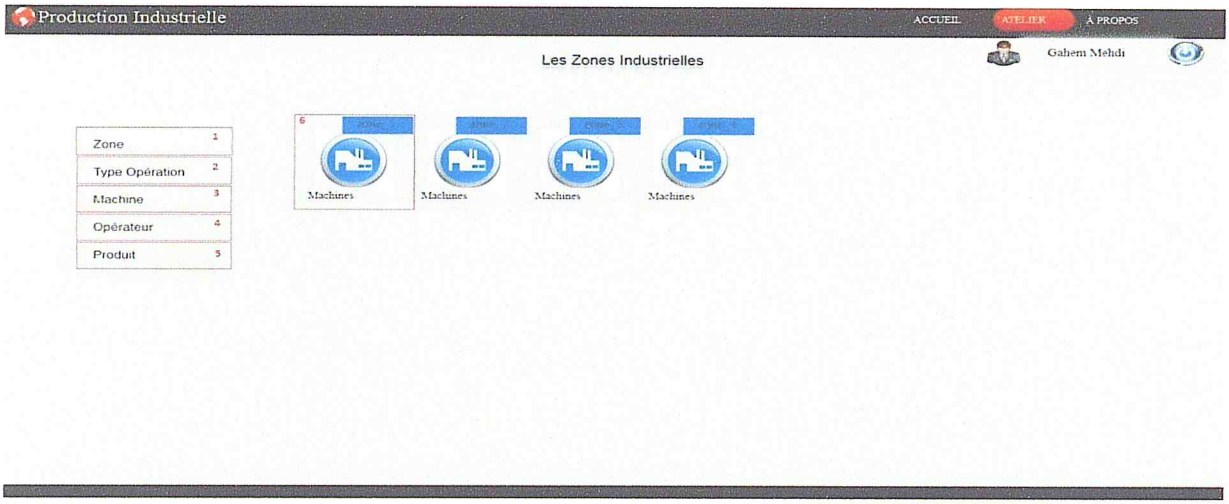


Figure 4.4 : Interface du menu principale.

Le menu de notre application contient les fonctionnalités qui peuvent être accédées à travers les boutons suivants :

1. Zone (Ajouter une zone, définir le temps de transport entre les zones).
2. Type d'opération (Lien vers la page « Les types d'opération »).
3. Machine (Lien vers la page « Les machines par rapport au type d'opération »).
4. Opérateur (Lien vers la page « Les opérateurs »).
5. Produit (Lien vers la page « Les pièces »).
6. La zone (On peut modifier ou supprimer cette zone a partir du bouton « le nom de zone », et aussi pour afficher les machines qui se trouvent dans cette zone et ce à partir du bouton « machines ».

4.3.3. Interface d'une zone

La figure suivante représente un formulaire de l'ajout d'une zone

Figure 4.5 : Ajout d'une zone.

Une fois l'ajout est effectué, la zone sera insérer automatiquement dans le menu principale. En cas ou l'administrateur devra modifier ou supprimer cette zone, il doit sléctionner le nom de cette zone une fois affichée. La figure suivante représente les actions de modification et suppression d'une zone.

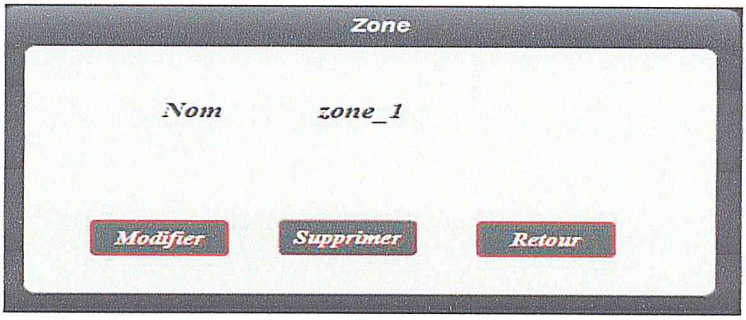


Figure 4.6 : Interface de modification et suppression d'une zone.

En cas de suppression d'une zone, un message sera affiché pour rappeler que la zone contient des machines, afin de déplacer ces machines sur autre zone ou les supprimer.

Après l'ajout de toutes les zones, il faut déterminer le temps de transport entre ces zones. Pour cela, il faut cliquer sur le bouton « Zone » du menu principal ensuite sur « Temps de transport ».

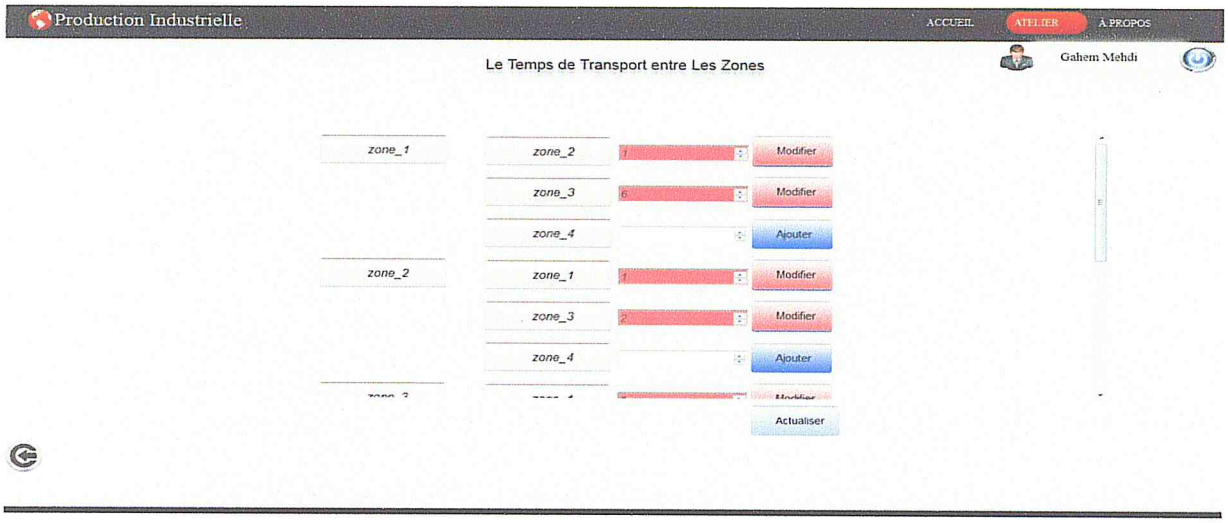


Figure 4.7 : Ajout de temps de transport entre les zones.

4.3.4. Interface des types d'opérations

La figure ci-dessous représente un exemple d'interface des types d'opérations, en cliquant sur le bouton « Type d'opération » de l'interface menu principale, une nouvelle page sera affichée, cette dernière contient les types d'opérations enregistrées dans l'application.

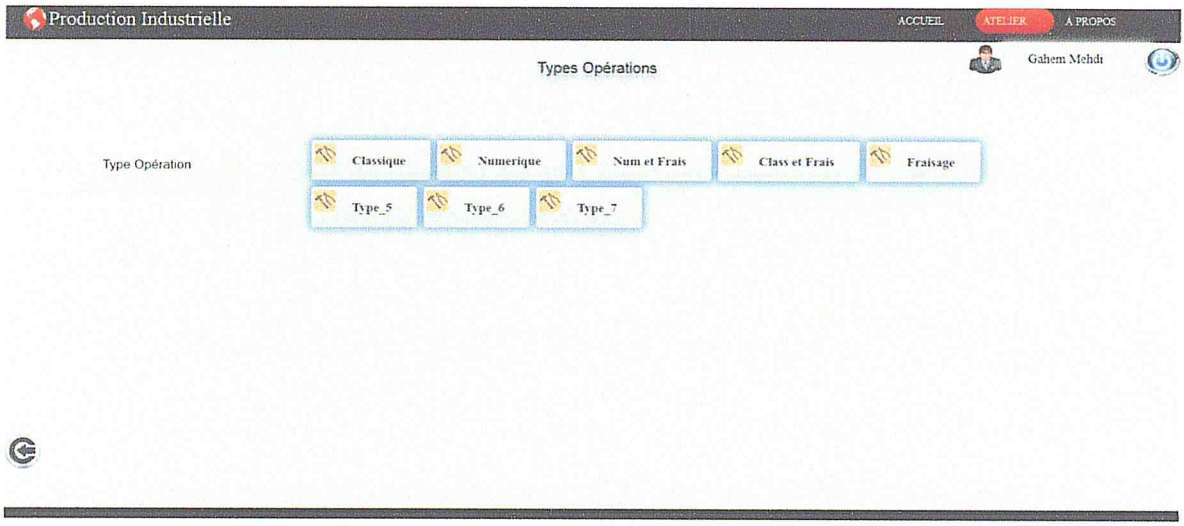


Figure 4.8 : Interface principale des types d'opérations.

L'ajout d'un type d'opération est réalisé à travers le bouton « Type Opération ». La figure suivante représente un formulaire de l'ajout d'un type d'opération.

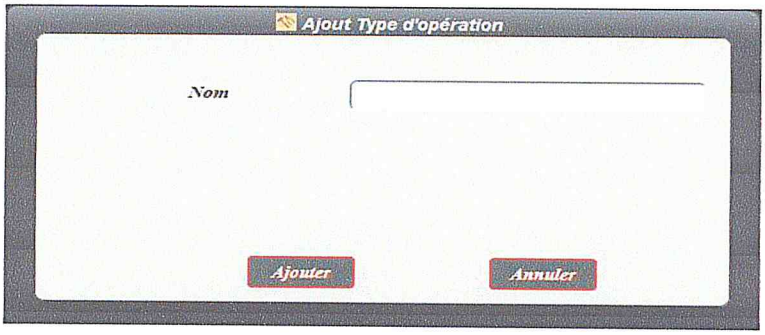


Figure 4.9 : L'ajout d'un type d'opération.

Le formulaire, ci-dessous, présente les actions de modification et suppression d'une opération.

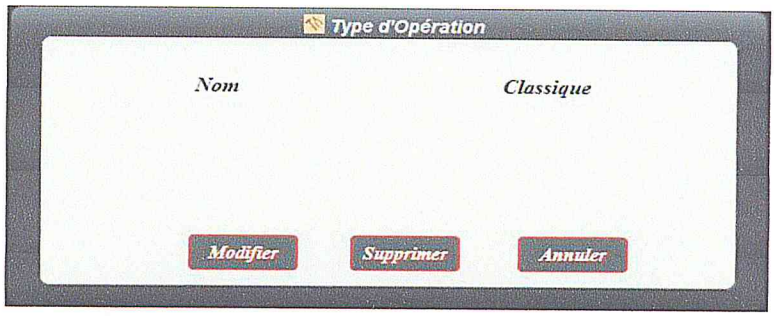


Figure 4.10 : Modification et suppression d'un type opération.

4.3.5. Interface des machines

La figure ci-dessous représente un exemple d'interface des machines. En cliquant sur le bouton « Machines » une nouvelle page sera affichée. Cette dernière contient les machines enregistrées dans la zone sélectionnée.

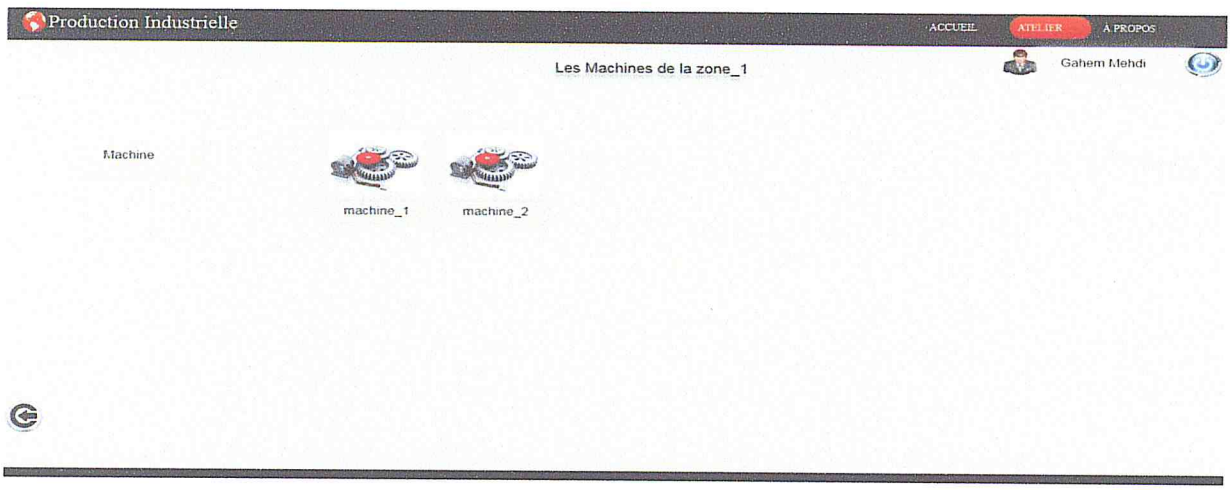


Figure 4.11 : Interface des machines d'une zone.

La figure ci-dessous représente un exemple de formulaire de l'ajout d'une machine.

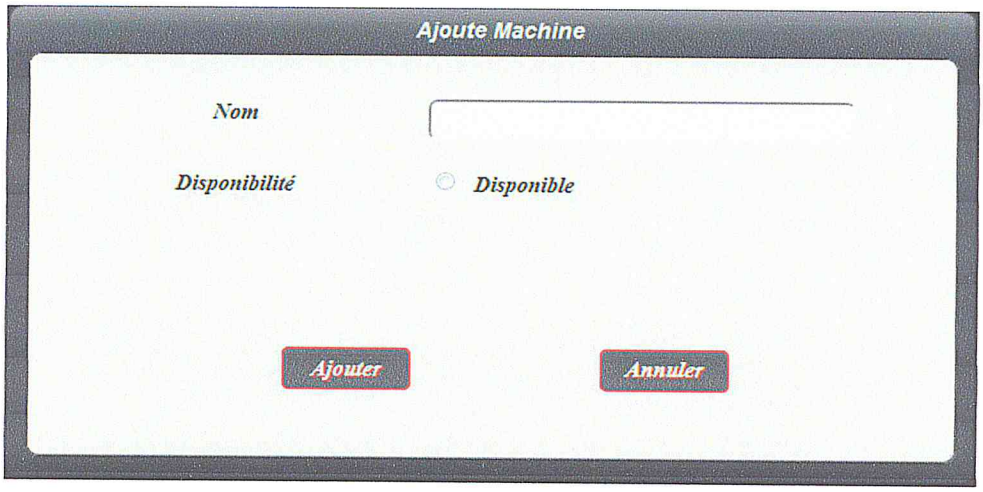


Figure 4.12 : L'ajout d'une machine.

Une fois la machine ajoutée, il faut définit ces types d'opérations en cliquant sur le nom de cette machine ensuite « type opération », une nouvelle page sera affichée et qui représente les types opérations enregistrées dans la base de données de notre application.

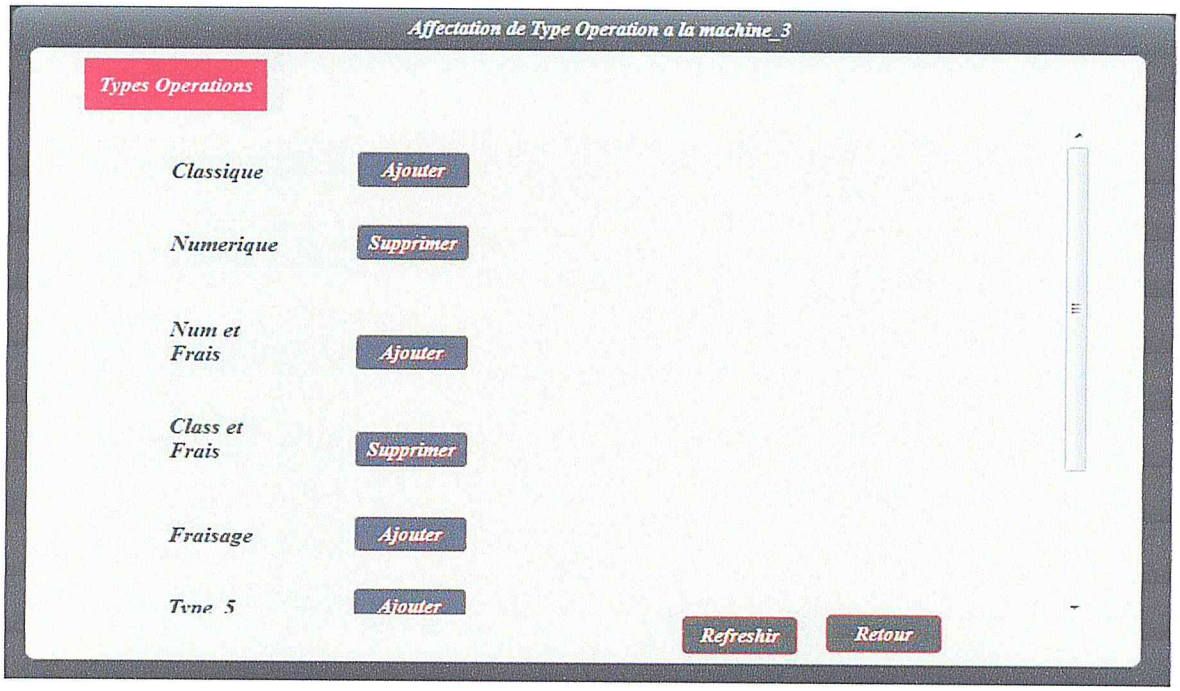


Figure 4.13 : Affectation des types d'opérations à une machine.

La figure suivante représente le classement des machines par type d'opérations.

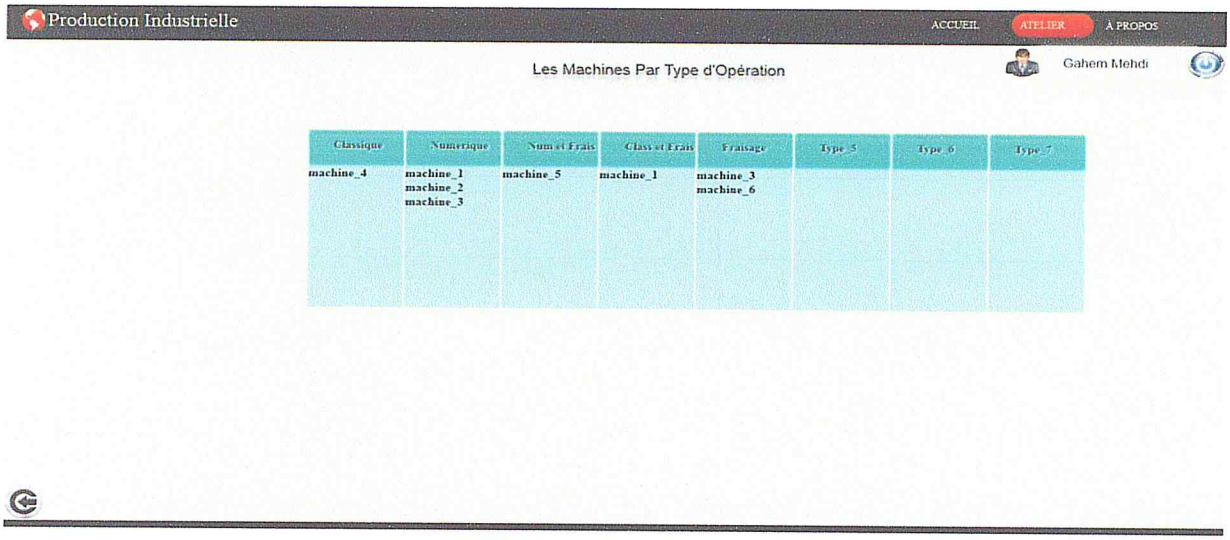


Figure 4.14 : Classement des machines par type d'opération.

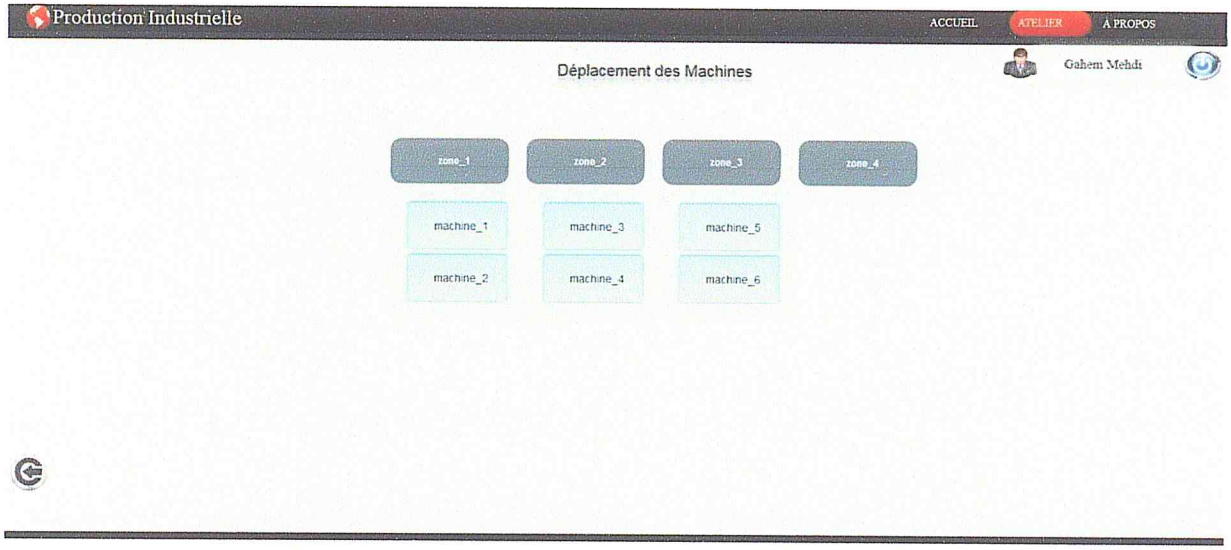


Figure 4.15 : Déplacement des machines.

4.3.6. Interface des opérateurs

La figure ci dessous représente un exemple d'interface des opérateurs. En cliquant sur le bouton « Opérateurs » du menu principal, une nouvelle page sera affichée. Cette dernière contient les opérateurs enregistrés dans l'application.

Figure 4.16 : Interface principale des opérateurs.

La figure ci-dessous représente un exemple de formulaire de l'ajout d'un opérateur qui s'affiche lorsqu'on clique sur le bouton « Opérateur » dans le menu principal des opérateurs.

Figure 4.17 : L'ajout d'un opérateur.

Les actions de modification et de suppression peuvent être appliquées une fois un nom d'opérateur est sélectionné.

Figure 4.18 : Modification et suppression d'un opérateur.

Chaque opérateur est affecté à une machine. Pour ce faire on sélectionne un opérateur en cliquent sur son et on l'affecte à une machine via le bouton « Choix Machine ».

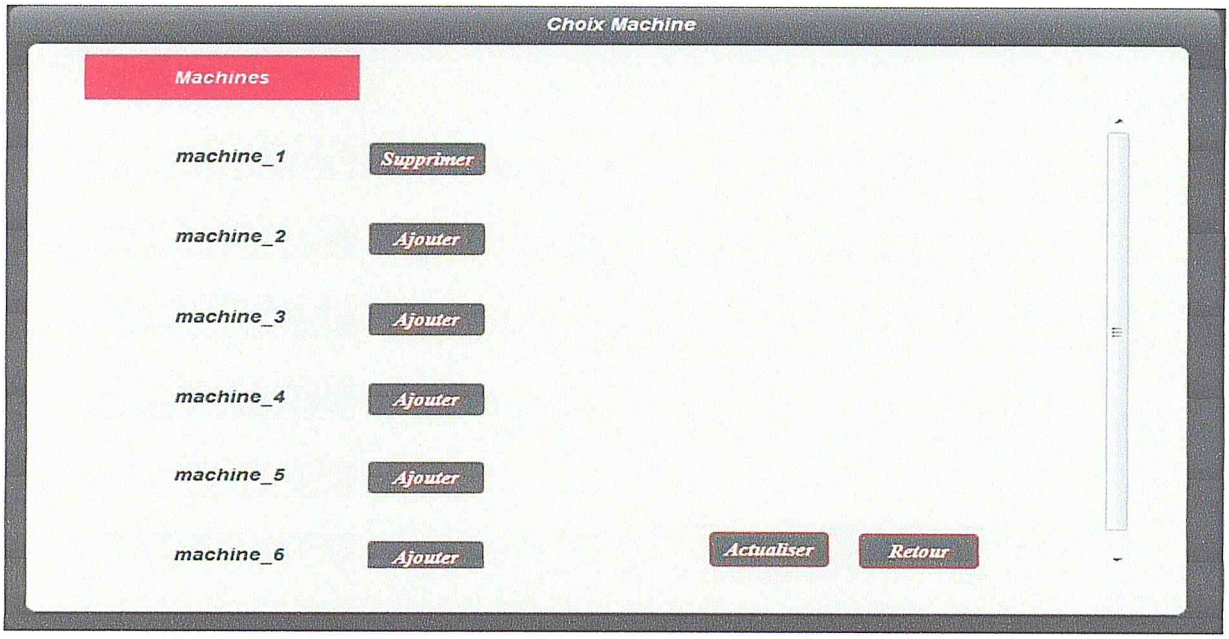


Figure 4.19 : L'affectation d'une machine à un opérateur.

4.3.7. Interface de produit

La figure ci-dessous représente un exemple d'interface de produit. En cliquant sur le bouton « Produit » de l'interface menu principal, une nouvelle page sera affichée. Cette dernière contient les pièces enregistrées dans l'application.

Figure 4.20 : L'interface principale des pièces.

La figure ci-dessous représente un exemple de formulaire de l'ajout d'une pièce.

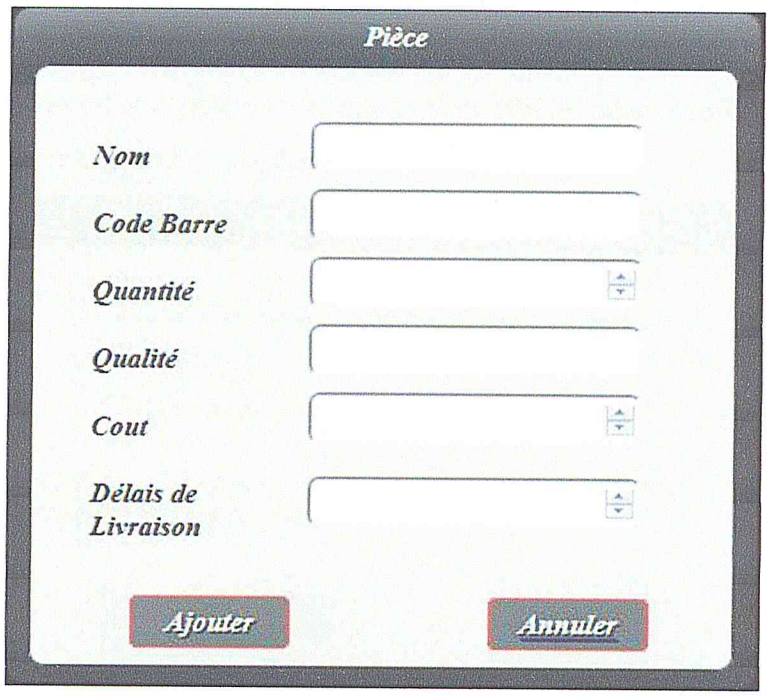


Figure 4.21 : L'ajout d'une pièce.

Les actions de modification et de suppression sont réalisées à travers selon le formulaire ci-après.

4.3.8. Interface des opérations

La figure ci-dessous représente un exemple d'interface d'opération. En cliquant sur le bouton « Opérations » une nouvelle page sera affichée. Cette dernière contient les opérations enregistrées dans l'application.

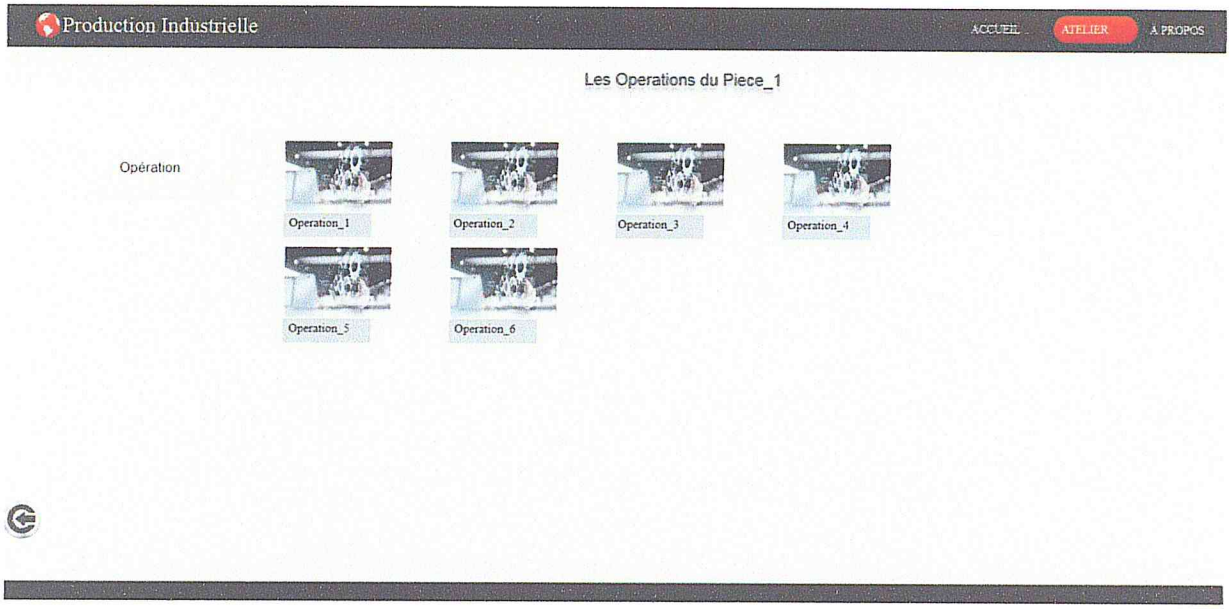


Figure 4.24 : Interface principale des opérations.

La figure ci-dessous représente un exemple de formulaire de l'ajout d'une opération qui s'affiche en cliquant sur le bouton « Opération » du menu principale des opérations. Pour chaque opération on détermine son type opération.

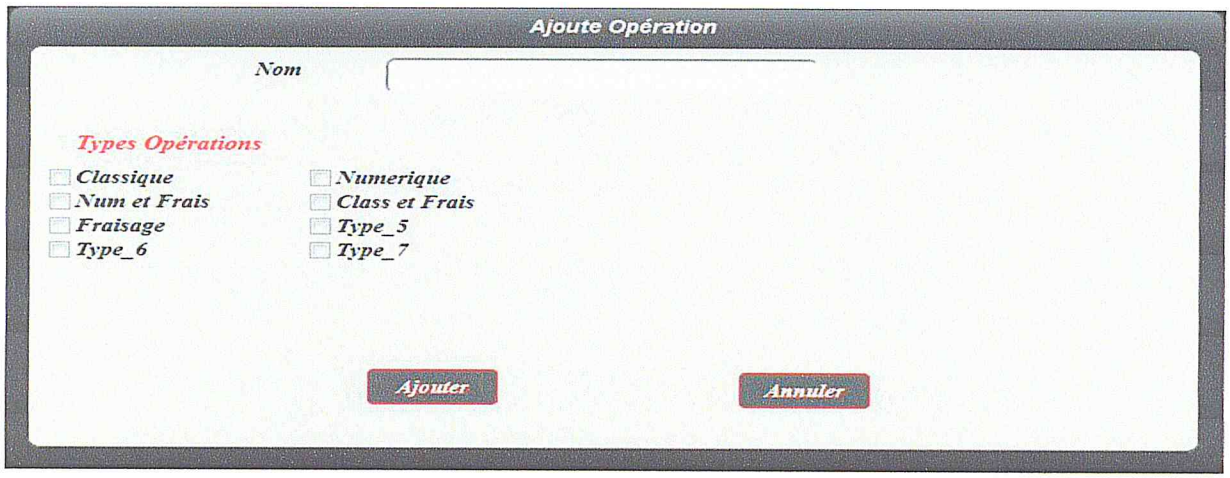


Figure 4.25 : L'ajout d'une opération.

Les actions de modification et suppression peuvent être réalisées en sectionnant le nom la pièce.

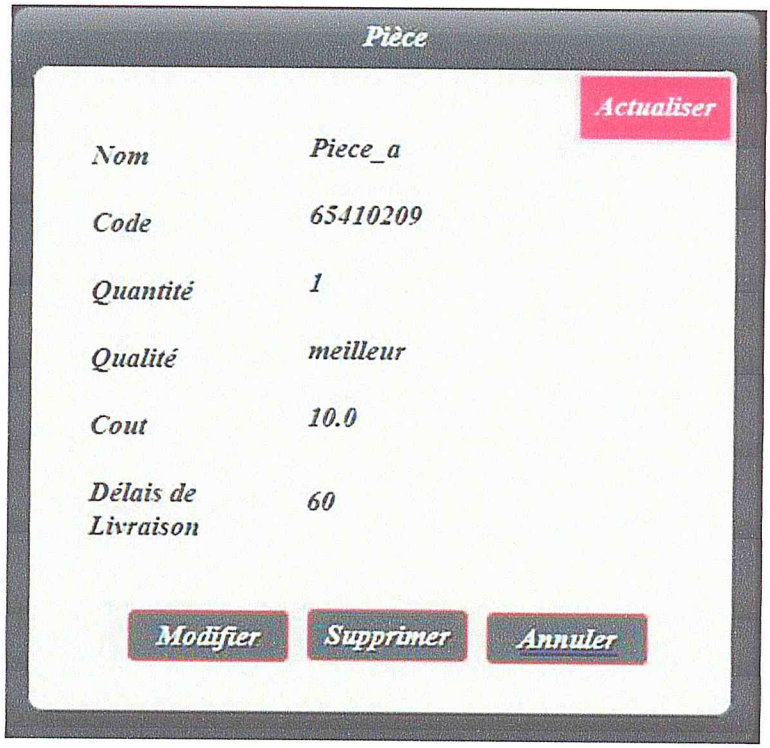


Figure 4.22 : Modification et suppression d'une pièce.

En cas de suppression d'une pièce, un message sera affiché pour confirmer la suppression et rappeler que la pièce contient des opérations à exécuter.

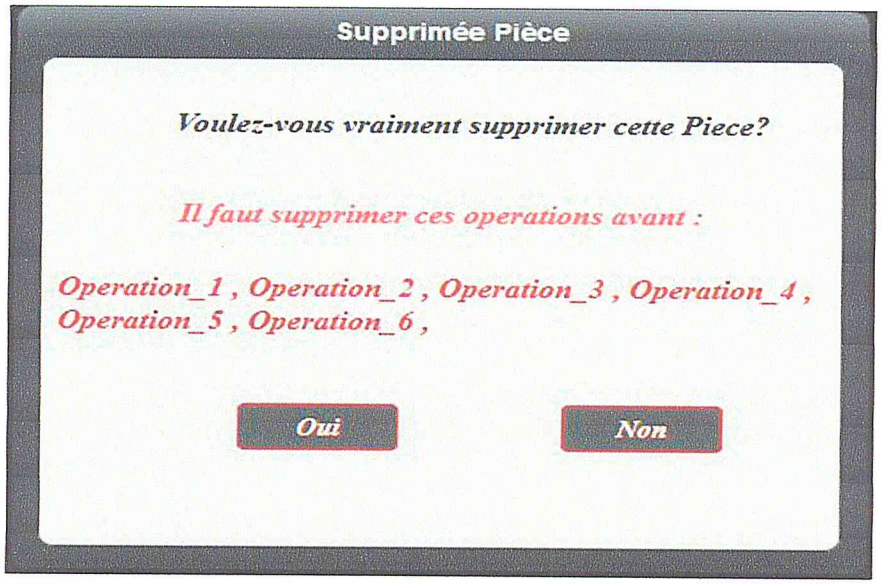


Figure 4.23 : Confirmation de suppression d'une pièce.

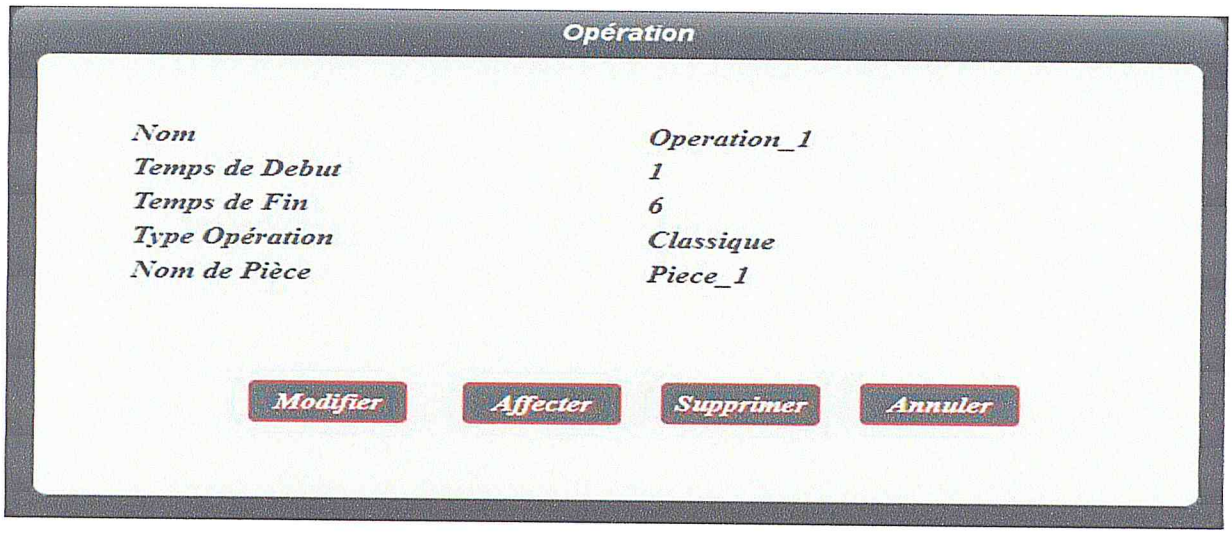


Figure 4.26 : La modification et suppression d'opération.

Une fois opération ajoutée, il faut affecter chaque opération à ces machines et déterminer le temps opératoire sur ces machines.

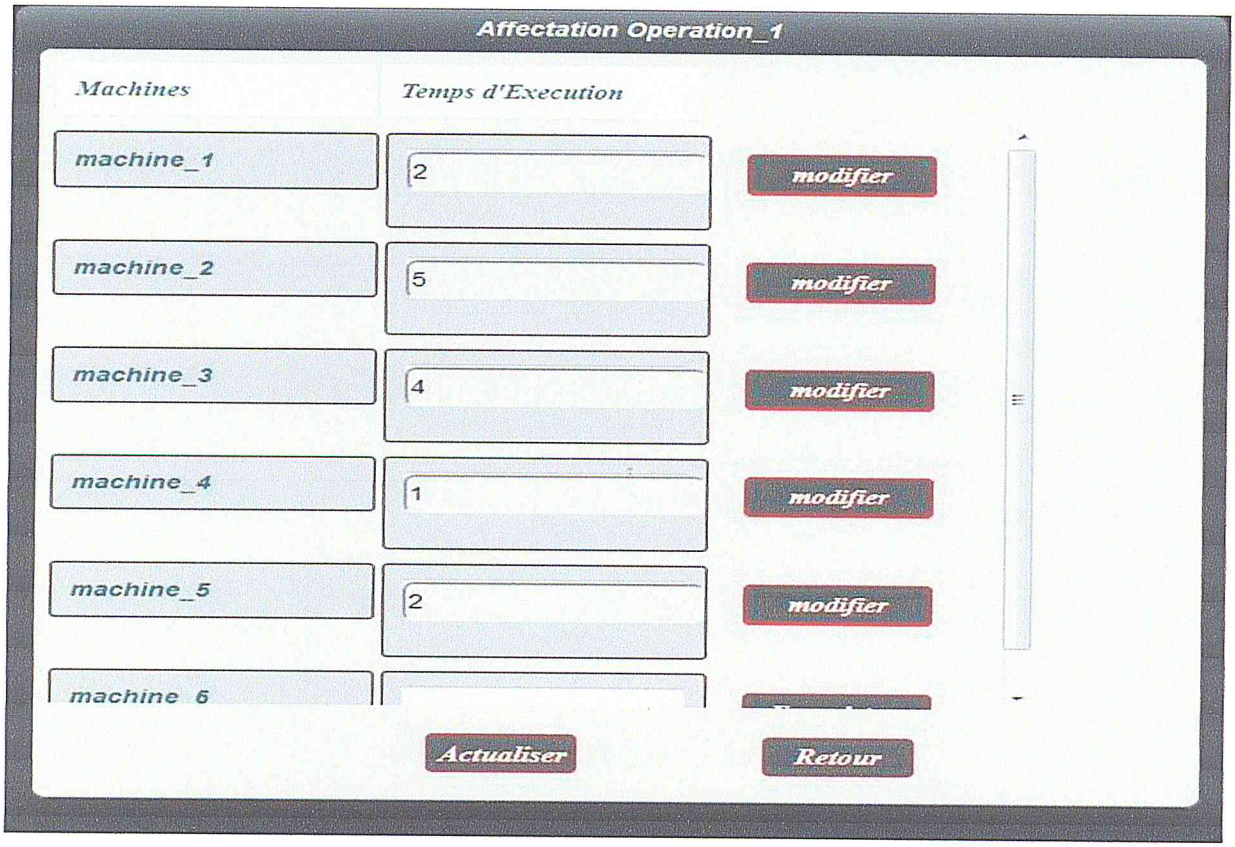


Figure 4.27 : L'affectation d'une opération sur les machines.

Pour la résolution de notre problème d'ordonnancement (générer le fichier benchmark) , il faut la cliquer sur le bouton « Affectation machines » dans la page des pièces, et donner le nom au fichier et cliquer sur le bouton « Télécharger »

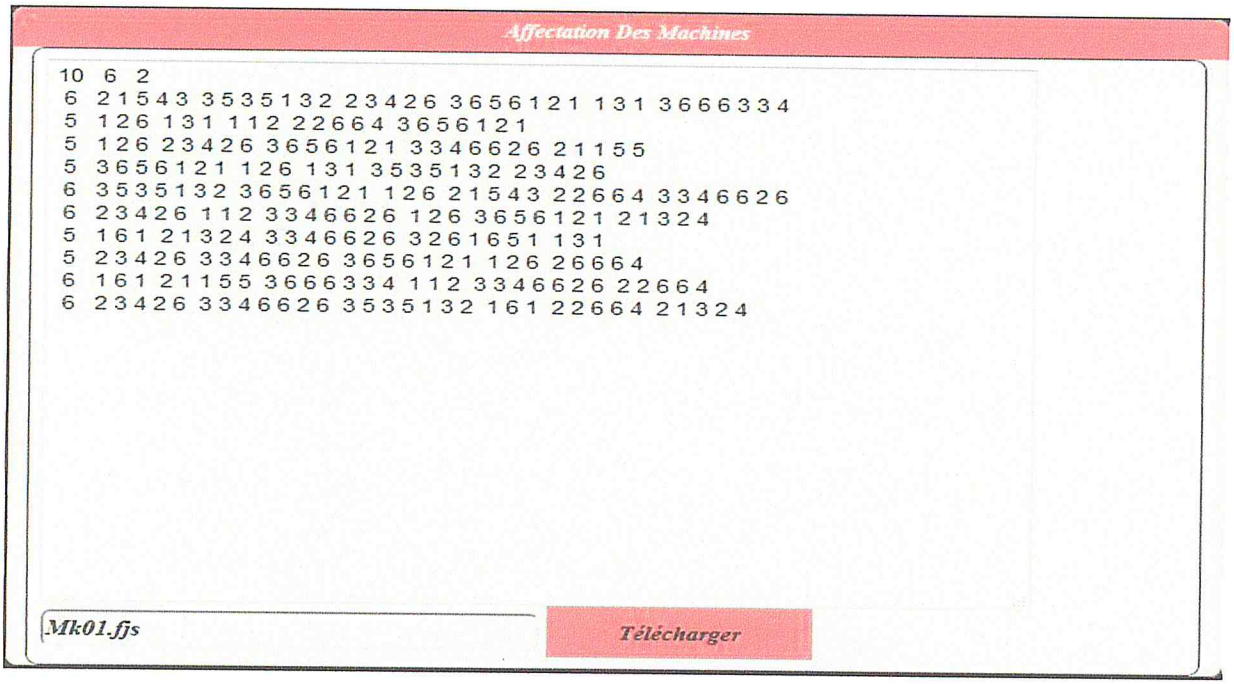


Figure 4.28 : Fichier benchmark.

4.3.9. Interface configuration d'algorithme de résolution

Après l'enregistrement du benchmark, on procède à la configuration de l'algorithme génétique. Un message sera affiché pour consulter la configuration.

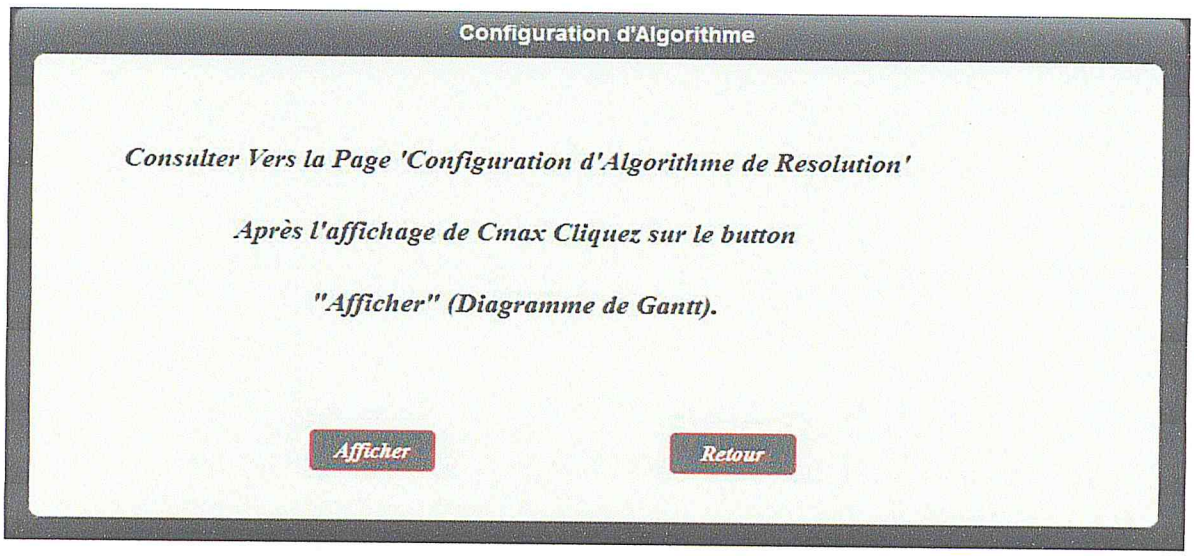


Figure 4.29 : Message de configuration d'algorithme.

La figure suivante représente la fenêtre de configuration d'algorithme de résolution, tout d'abords on saisit le nombre d'itération et on choisit le benchmark à utiliser. Ensuite on lance l'algorithme en cliquent sur le bouton « Run ». Après la génération de l'algorithme le résultat (**Cmax**) sera affiché dans la fenêtre de configuration.

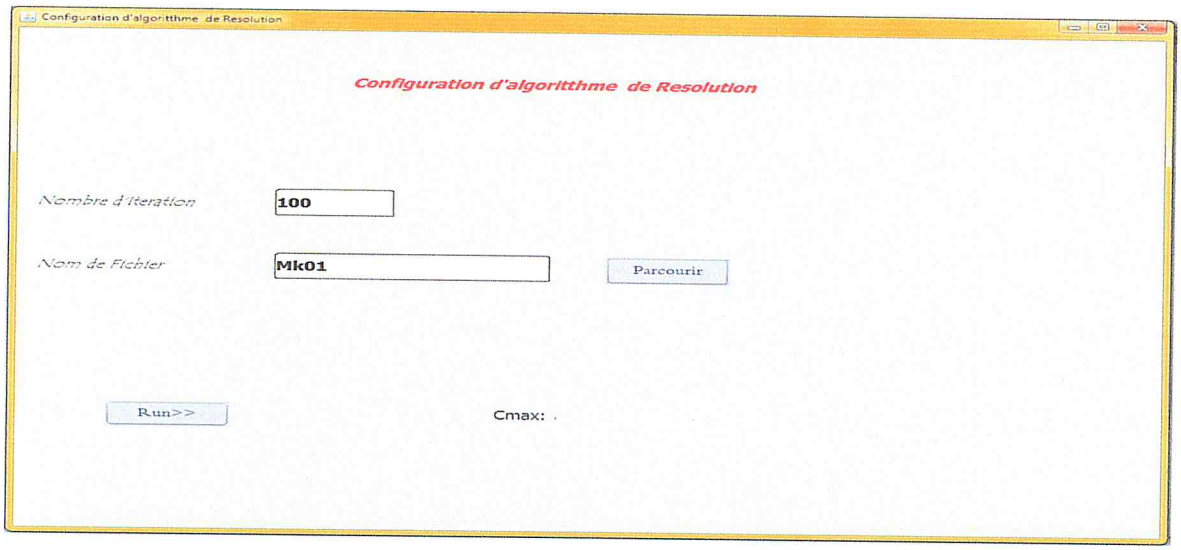


Figure 4.30 : Configuration de l'algorithme de résolution.

Pour l'affichage de diagramme de Gantt (résultat de l'algorithme de résolution), il faut cliquer sur le bouton « Afficher » (**Figure 4.29**).

Figure 4.31 : Diagramme de Gantt de la résolution de l'algorithme.

Après l'affichage de la solution, on lui on donne un et on le télécharge, qui se enregistrer automatiquement dans le dossier de téléchargement.

4.3.10. Interface de l'avancement des pièces

Après de l'enregistrement de la solution. Celle-ci sera affichée automatiquement dans une nouvelle page pour avoir plus de détails sur l'avancement de la fabrication des pièces.

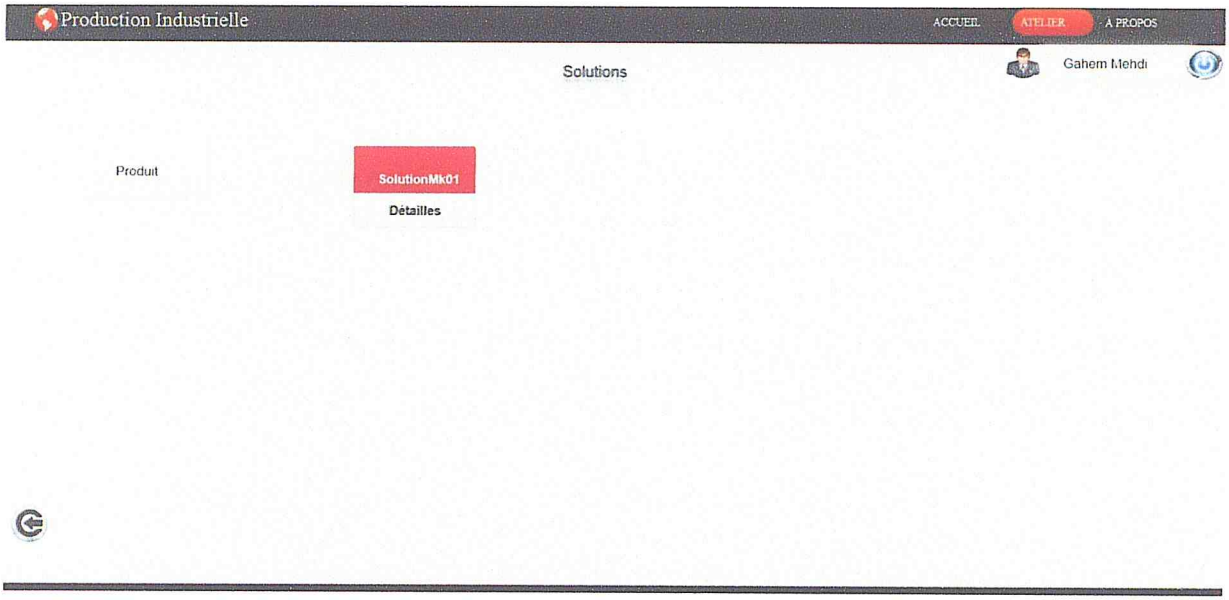


Figure 4.32 : Interface des solutions.

Tout d'abord il faut cliquer sur le bouton « produit » pour actualiser chaque pièce et pour définir l'état final de toutes les pièces qui ont été finalisées (le temps début et de fin de chaque pièce).

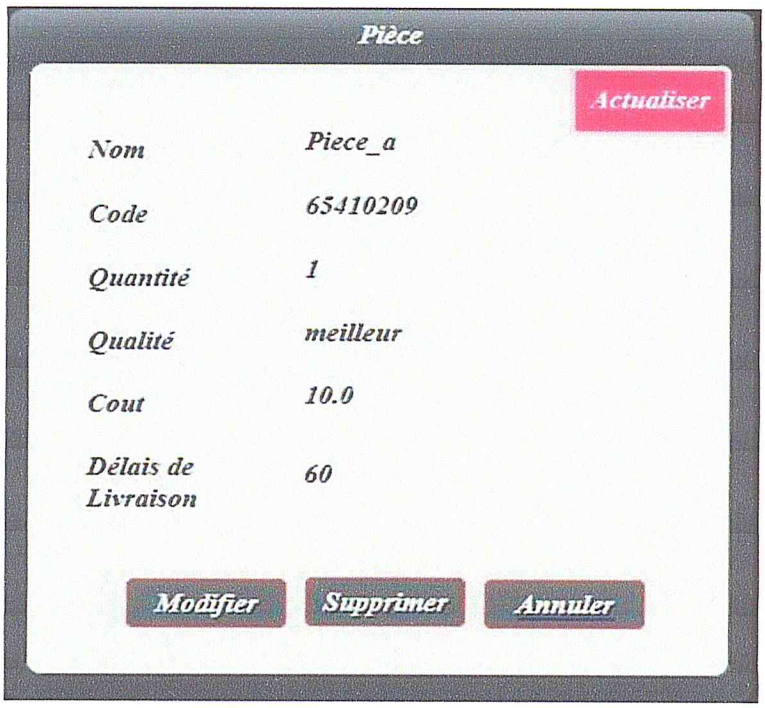


Figure 4.33 : Actualisation des pièces.

Pour voir les détailles de chaque solution, il faut cliquer sur le bouton « Détailles » de la solution.

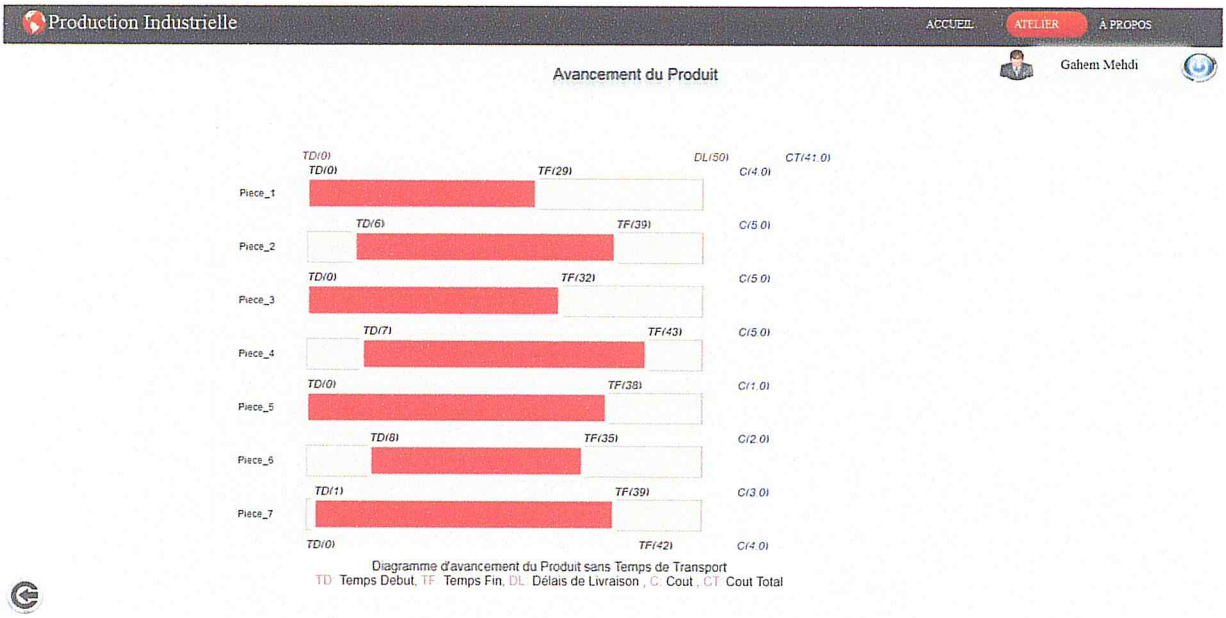


Figure 4.34 : Les détailles des pièces de produit.

4.3.11. Affichage de la solution

Pour voir le diagramme de Gantt et le fichier benchmark des solutions réalisées il faut :

- Cliquer sur le bouton « diagramme de Gantt» qui s'affiche sous format image.
- Cliquer sur le bouton « Fichier » pour l'ouvrir.

Figure 4.35 : Affichage d'une solution.

4.3.12. Détails de chaque pièce à travers le code à barres

Pour avoir les détails de chaque pièce à travers les informations issues de son code à barres, on lance d'abord la fenêtre de capture du code à barre de la pièce.

Figure 4.36 : Fenêtre de capture du code à barre.

Après le capture, on copie le code capturé dans le champ de texte et on clique sur « afficher » (**Figure 4.27**).Une nouvelle page sera affichée présentant le diagramme de l'avancement travaux réalisés sur la pièce.

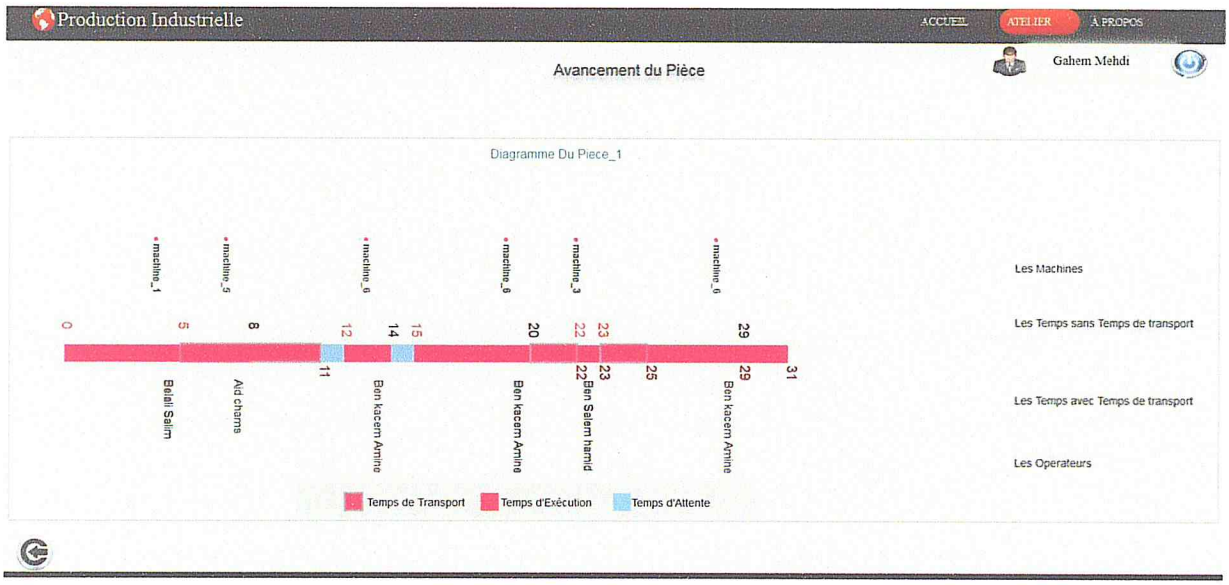


Figure 4.37 : L'avancement d'une pièce.

4.6. Conclusion

Dans ce chapitre nous avons présenté succinctement l'environnement de programmation et de réalisation de notre solution tout en détaillant les fonctionnalités des interfaces conçues et les modalités de l'implémentation de notre algorithme génétique. Dans ce chapitre, nous avons également présenté les résultats de différentes expérimentations réalisées sur un ensemble de problème liés à l'ordonnancement des tâches dans un système de gestion de la production. L'objectif visé est d'explorer les performances des stratégies que nous avons utilisées, d'un côté, et de valider l'implémentation de notre algorithme génétique pour la résolution du problème de Job Shop Flexible.

Conclusion Générale

Conclusion générale

Les nouvelles technologies de l'information sont utilisées dans les entreprises industrielles, pour améliorer le processus de production et de lutter contre les contraintes multiples qui entravent la performance de ces entreprise en terme de compétitivité et de réactivité. Les nouvelles technologies imposent de nouvelles stratégies et de nouveaux processus de production. Des industries informationnelles émergent dans les secteurs de l'industrie manufacturière à travers de nouvelles solutions et équipements favorisant ainsi l'accroissement de la productivité et la rentabilité de ces entreprises.

C'est dans ce contexte précis, que s'inscrit le travail réalisé dans ce mémoire et dans lequel nous avons réalisé une application informatique de pilotage dynamique des processus production par le traitement du problème d'ordonnancement des ateliers de type Job Shop Flexible. Aussi, nous avons conçu et implémenté un algorithme génétique basé sur une nouvelle fonction d'évaluation qui inclut le critère temps de transport entre les zones, et aussi pour plus de rapidité d'intégration de l'information sur le suivie de production toute en intégrant la technologie code à barre qui permet le suivie de fabrication des pièces dans un système de production.

En perspectives de ce travail, nous envisageons, d'une part, d'ajouter d'autres contraintes comme la disponibilité des opérateurs (emploi de temps), ou machines (en panne) pour un meilleur suivi de l'avancement du processus de fabrication des pièces, et d'autre part, utiliser la technologie Java Web Start pour lancer les fenêtres graphique (JFrame, JApplet,…) depuis une page JSP sans accéder à la classe de la fenêtre graphique.

Bibliographie

[1] [Saad et al., 07] I. Saad, H. Boukef, P. Borne, The comparison of criteria aggregative approaches for the multi-objective optimization of flexible job-shop scheduling problems, 18(2007), p.603-608.

[2] [Giard, 03] V. Giard, Gestion de la production et des flux, 2-7178-4498-8(2003).

[3] [Cauvin, 05] A. Cauvin, Analyse, modélisation et amélioration de la réactivité des systèmes de décision dans les organisations industrielles, (2005).

[4] [Le Moigne, 74] JL. Le Moigne, Les systèmes de décision dans les organisations, (1974).

[5] [Noyel et Pisaneschi, 11] M. Noyel, T. Pisaneschi, Mise en place d'un système RFID pour une entreprise de panneaux laqués haute finition, (2011).

[6] [Mesghouni, 99] K. Mesghouni, Applications des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de la production, Thèse de Doctorat, Université des Sciences et Technologies de Lille, Laboratoire d'Automatique et d'Informatique Industrielle de Lille, (1999).

[7] [Montanari, 74] U. Montanari, Networks of Constraints: Fundamental Properties and Applications to Picture Processing, Artificial Intelligence, Vol. 7, p. 95-132(1974).

[8] [Carlier et Chrétienne, 88] Carlier, J., Chrétienne, P., « Problème d'ordonnancement, modélisation, complexité et algorithmes ». Masson, (1988).

[9] [Selt, 08] O. Selt, Méta-heuristiques, pour résoudre les problèmes d'ordonnancement des tâches sur des machines parallèles, Thèse de magistère, Université de M'SILA, p.5-14(2008).

[10] [Gérard et Abécassis, 03] C. Gérard, D. Abécassis, Gestion de projet-diagramme de Gantt, Université de Lorraine, (2003), p.4-5.

[11] [Gary et Johnson, 79] M.R. Gary, D.S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, W.H. Freeman and Company, (1979).

[12] [Lin et Kernigham, 73] S. Lin, B.W. Kernigham, An efficient heuristic for the traveling-salesmanproblem, 21(1973), p. 498-516.

[13] [Brandimarte, 93] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, 3(1993), p.157-183.

[14] [Courtois et al., 03] A. Courtois, C. Martin-Bonnefous, M. Pillet, Gestion de production. Les Ed. d'Organisation, (2003).

[15] http://www.solune.com/ (Consulté le 12-2015).

[16] http://www.procad-gpao.com/ (Consulté le 12-2015).

[17] [Gendreau et Potvin, 05] M. Gendreau, J. Y. Potvin, Metaheuristics in combinatorial optimization, 1(2005), p.189-213.

[18] [Cea, 68] J. Cea, Les méthodes de «descente» dans la théorie de l'optimisation, 3 (1968), p. 79-101.

[19] [Dréo et al., 05] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, Métaheuristiques pour l'optimisation difficile, Eyrolles, (2005).

[20] [Glover, 90] F. Glover, Tabu search, part II , 2 (1990), p. 4-32.

[22] [Vilcot, 07] G. Vilcot, Algorithmes approchés pour les problèmes d'ordonnancement multicritères de type Job Shop Flexible et Job Shop multi ressource, Thèse de Doctorat, Université François-RABELAIS, (2007).

[23] [Nearchou, 04] A.C. Nearchou, The effect of various operators on the genetic search for large scheduling problems, International Journal of Production Economics, 2(2004), p. 191–203.

[24] [Souquet et Radet, 04] A. Souquet, F.G. Radet, Algorithmes génétiques, (2004), p.18-19.

[25] [Zhang et al., 08] G. Zhang, Y. Shi, L. 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, (2008), p.369-372.

[26] [Durand et al., 94] N. Durand, J.M. Alliot et J. Noailles, Algorithmes génétiques : un croisement pour les problèmes partiellement séparables, Journées Evolution Artificielle Francophones, JEAF, Toulouse, 1994.

[27] [Muller et Gaertner, 00] P.A. Muller, N. Gaertner, Modélisation objet avec UML, Paris Eyrolles, 514 (200).

[28] [Jacobson et al., 00] I. Jacobson, G. Booch, J. Rumbaugh, Le processus unifié de développement logiciel, Eyrolles, (2000).

[29] [Hugues, 02] A. M. Hugues, Différents modèles de cycle de vie, Génie logiciel, (2002).

[30] [Benoît et Aomar, 10], C. Benoît, O. Aomar, UML2, 3e édition, Yann Thierry-Mieg, (2010).

[31] [Frédéric, 14] V. Frédéric, Les Diagrammes de Séquence en phase de, Polytech : Paris-Sud, (2014).

[32] [Goncalves, 11] A. Goncalves, Java EE 5, Editions Eyrolles, (2011).

[33] http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/ (Consulté le 01-2015).

[34] https://netbeans.org/features/index.html (Consulté le 12-2015).

[35] http://www.wifeo.com/lexique/definition-css-49.html (Consulté le 02-2016).

[36] https://www.techopedia.com/definition/3929/javascript-js (Consulté le 02-2016).

[37][Melton, 98] J. Melton, Database language sql, Springer Berlin Heidelberg, (1998), p. 103-128).

[38] http://www.futura-sciences.com/tech/definitions/internet-mysql-4640/ (Consulté le 02-2016).

[39] https://www.mysql.fr/products/workbench/ (Consulté le 02-2016).

[40] http://techterms.com/definition/wamp (Consulté le 02-2016).

[41]http://www.journaldunet.com/encyclopedie/definition/340/34/20/java_database_connectivity.shtml (Consulté le 02-2016).

[42] https://fr.wikipedia.org/wiki/Webcam#cite_note-2 (Consulté le 04-2016).

[43] http://c2.com/cgi/wiki?VisualParadigm (Consulté le 12-2015).