الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة سعيدة د. مولاي الطاهر
كلية التكنولوجيا
قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Réseaux Informatique et Systèmes Répartis

Thème
Les colonies de fourmis pour des problèmes d'ordonnancement de type Job Shop

Présenté par :

Medani Nour

Berrague Aya Fatima Zahra

Dirigé par :

Dr. Mekour Mansour

Année universitaire 2022-2023


بسم الله الرحمن الرحيم

Remerciements

D’abord, je tiens à remercier Dieu clément et le miséricordieux de

M’avoir donné la force et le courage de mener à bien ce modeste travail.

J’exprime ma profonde reconnaissance et ma parfaite gratitude à

Mon encadreur

Dr : **Mekour Mansour**

Qui a témoigné de sa confiance et de son aide scientifique et

Par son expérience et sa compétence.

J’adresse de chaleureux remerciements à tous les

Enseignants et les travailleurs d’université de **Dr Moulay Tahar**

Je tiens à remercier particulièrement à tous les membres du

Département d'informatique

Pour leurs soutiens et leurs aides.

Je remercions aussi mes parents et mes amis pour leurs aides, leurs

Patiences, leurs compréhensions et leurs encouragements.

Medani Nour

Dédicaces

Je dédie ce modeste travail

A Mon Père et Ma Mère

A mes soeurs

Hakima, Anfal, Hadjer

A toute ma famille

Medani et Rahmani

A mon encadreur Dr. Mekour Mansour

A tous mes collègues et mes amis partout

Et en particulier Toufik, douaa,

Et en général Master 2 RISR.

Medani Nour

Remerciements

Je remercie avant tout ALLAH le tout Puissant de m'avoir donné le courage et la patience qui m'ont

permis d'accomplir Ce modeste travail.

Je tiens en premier lieu à exprimer

Ma Profonde gratitude et sincère reconnaissance Envers mon encadreur **Mekour Mansour**,

Maitre de conférence à l'université **Dr Moulay Tahar** SAIDA pour m'avoir Conseillé, dirigé pendant

La réalisation de ce travail.

Je remercie également tous les

Membres de jury pour l'honneur qu'ils me font en acceptant De juger ce travail.

Mes sincères remerciements s'adressent à toutes les personnes qui ont contribué au succès de mon

Projet fin d'étude.

Berrague Aya Fatima Zohra

Dédicaces

Je dédie ce modeste travail

A Mon Père et Ma Mère, A ma sœur Dhoha.

A mes frères Salah Eddinne, Ibrahim.

A Mon marié Kader.

A Ma Grand-mère (que Dieu lui

Miséricorde).

A toute ma famille

Berrague,Messadi,Chebab .

A mon encadreur Dr.makour Mansour

A tous mes collègues et mes amis partout

et en particulier Ikram.

Berrague Aya Fatima Zohra

Sommaire

Chapitre I : Les problèmes d'ordonnancement

Chapitre II : Techniques d'Optimisation des Colonies de fourmis

Chapitre III : Implémentation de ACO pour des problèmes d'ordonnancement de type job shop

Liste des figures

Chapitre I : Les problèmes d'ordonnancement

Chapitre III : Simulation

Liste des tableaux

Chapitre I : Les problèmes d'ordonnancement

Chapitre III : Simulation

Résumé

L'optimisation par colonies de fourmis (ACO) s'est avérée très efficace sur des problèmes de haute complexité (NP-difficiles) en optimisation combinatoire. Une implémentation d'un algorithme de modèle ACO connu sous le nom de Ant Elite System (EAS), pour un problèmes d'optimisation, est décrite. Approche adopte vise à réduire la latence en spécifiant les opérations immédiatement disponibles, mais en tenant compte du rare manque d'opérations disponibles. Les performances de l'algorithme sont évaluées par rapport à des problèmes de référence, et les résultats montrent que la qualité des solutions est bonne et que les solutions obtenues sont efficaces avec un très petit nombre d'évaluations de la fonction objectif.

Mots clés : colonie de fourmis, ACO.

ABSTRACT

The optimization of ant colonies (ACO) has proven to be very effective on high complexity problems (NP-difficult) in combinatorial optimization. An implementation of an ACO model algorithm known as the Ant Elite System (EAS), applied to optimization problems, is described. A proposed approach that aims to reduce latency by specifying the operations immediately available, but taking into account the rare lack of available operations. The performances of the algorithm are evaluated in relation to reference problems, and the results show that the quality of the solutions is good and that the solutions obtained are effective with a very small number of evaluations of the objective function.

Key words: ant colony, ACO.

ملخص

المذكرة تتناول موضوع مستعمرات النمل وتركز على مشاكل جدولة نوع متجر العمل. تبحث المذكرة في كيفية تحسين جدولة نوع متجر العمل في مستعمرات النمل وتحليل التحديات التي تواجهها.

تتضمن تحليلًا للطرق التقليدية لجدولة نوع متجر العمل في مستعمرات النمل وتحديد المشاكل الشائعة التي تواجهها. تشمل هذه المشاكل تأخر في تنفيذ المهام وعدم تحقيق الكفاءة القصوى في استخدام الموارد.

تتضمن المذكرة أيضًا دراسة تجريبية لتقييم فعالية الحل المقترح. يتم استخدام بيانات واقعية من مستعمرة نمل حقيقية لتحليل أداء الحل المقترح ومقارنته بالطرق التقليدية.

تتوصل ايضا إلى نتائج مهمة تؤكد فعالية الحل المقترح في تحسين جدولة نوع متجر العمل في مستعمرات النمل.و توضح أهمية تطبيق التقنيات الحديثة مثل الذكاء الاصطناعي والتعلم الآلي في مجال علم النمل وتحسين أداء المستعمرات.

باختصار يتناول الموضوع مستعمرات النمل ويركز على مشاكل جدولة نوع متجر العمل. تقدم الدراسة حلاً مقترحًا يستخدم تقنيات الذكاء الاصطناعي والتعلم الآلي لتحسين جدولة المهام وتحسين استخدام الموارد. توصلت المذكرة إلى نتائج إيجابية تؤكد فعالية الحل المقترح في تحسين جدولة نوع متجر العمل في مستعمرات النمل.

الكلمات المفتاحية: مستعمرات النمل, مشكلة تخطيط ورشة العمل.

Introduction Générale

Introduction Générale

Dans l'environnement commercial actuel en évolution rapide, l'allocation et la planification efficaces des ressources sont devenues un facteur clé pour optimiser les processus commerciaux et rester compétitif. Dans cet environnement dynamique, le défi de la planification du lieu de travail est devenu un problème d'optimisation intégré critique couvrant des secteurs allant de la fabrication traditionnelle à la prestation de services moderne. Essentiellement, la planification du lieu de travail implique la tâche complexe d'attribuer plusieurs tâches à des machines, des postes de travail ou des ressources spécifiques, tout en faisant face à un réseau complexe de contraintes et en s'efforçant de minimiser les délais d'exécution.

Dans ce contexte, le besoin d'approches novatrices pour aborder les complexités de la planification des horaires en milieu de travail est évident. Les méthodes traditionnelles ont souvent du mal à fournir des solutions satisfaisantes en raison des nombreuses variables et contraintes qui interagissent. Mais maintenant, une nouvelle perspective prend le devant de la scène, inspirée par l'ingéniosité d'Ali, l'architecte de la nature. Ces minuscules insectes ont démontré une incroyable capacité à naviguer dans des voies complexes et à trouver des itinéraires optimaux grâce à un comportement de recherche de nourriture collectif, ce qui en fait une source d'inspiration intéressante pour les techniques d'optimisation [2].

Cette recherche entreprend un voyage dans le domaine de la planification des horaires en milieu de travail, guidée par la vision d'améliorer à la fois efficace des processus de planification. Au cœur de son cœur se trouve l'intégration de l'optimisation des colonies de fourmis (ACO), un paradigme qui imite les processus décisionnels des fourmis pendant la recherche de nourriture. Le principe sous-jacent de l'ACO est la communication d'informations par le biais de sentiers de phéromones, qui guident les fourmis vers des chemins optimaux. En traduisant et en adaptant ces mécanismes aux défis de l'ordonnancement en milieu de travail, cette étude vise non seulement à résoudre des problèmes d'optimisation complexes, mais également à dévoiler des pistes innovantes pour générer des horaires optimaux dans un éventail de scénarios opérationnels [1].

Structurée comme suit:

- Chapitre 1: présente les problèmes d'ordonnancement, en analysant les difficultés liées à la planification efficace des tâches et en proposant des solutions pour optimiser les processus de gestion du temps.

- Chapitre 2: fournit détaille la conception et l'adaptation de l'ACO pour les problèmes d'ordonnancement. Il explique la formulation du problème, décrit l'algorithme ACO, décrit les heuristiques de construction de solutions et fournit des informations sur la mise en œuvre technique de l'algorithme, explorant les principes de l'ACO et décrivant les objectifs de l'étude.

- Chapitre 3: des études de cas et des montages expérimentaux sont présentés, mettant en évidence l'application pratique et la performance de l'approche proposée basée sur l'ACO en résumant les résultats.

Avec les bases posées dans cette introduction, nous nous lançons dans une exploration complète de la façon dont l'optimisation des colonies de fourmis peut révolutionner la planification du lieu de travail, ouvrant la voie à des processus améliorés et à une meilleure utilisation des ressources dans un large éventail d'industries. En approfondissant chaque chapitre, contribuant en fin de compte à l'avancement des méthodologies de planification et à l'optimisation des systèmes opérationnels du monde réel.

Chapitre I :

Les problèmes d'ordonnancement

1.1 Introduction aux problèmes d'ordonnancement

Un problème d'ordonnancement est une situation dans laquelle vous devez déterminer le meilleur ordre d'exécution pour des tâches ou des activités compte tenu de diverses contraintes et objectifs. L'objectif principal est de minimiser les coûts, de maximiser l'efficacité et de respecter des délais spécifiques tout en utilisant efficacement les ressources disponibles.

Ces problèmes surviennent généralement lorsque les ressources sont limitées et doivent être allouées de manière optimale pour effectuer un ensemble de tâches ou d'activités. Les contraintes incluent les dépendances des tâches, les contraintes de temps, les contraintes de capacité des ressources, les contraintes de priorité, etc.

La résolution de problèmes d'ordonnancement nécessite souvent l'utilisation d'algorithmes, de techniques d'optimisation et de modèles mathématiques pour trouver la séquence d'exécution optimale qui répond à des objectifs et à des contraintes spécifiques. En résumé, un problème d'ordonnancement est une situation dans laquelle il est nécessaire de déterminer l'ordre optimal d'exécution des tâches ou des activités pour atteindre un objectif particulier, compte tenu des contraintes. La résolution de ces problèmes nécessite l'utilisation d'algorithmes et de techniques d'optimisation pour trouver les meilleures solutions possibles [3].

1.2 Définition des Problèmes d'Ordonnancement

Un problème d'ordonnancement est un problème de planification et de gestion des ressources qui nécessite de déterminer l'ordre optimal d'exécution des tâches ou des activités compte tenu de certaines contraintes. Ces contraintes incluent les dépendances des tâches, les contraintes de temps, les contraintes de capacité des ressources, les contraintes de priorité, etc.

L'objectif principal des problèmes d'ordonnancement est de trouver une séquence d'exécution qui minimise les coûts, maximise l'efficacité et respecte des délais spécifiques tout en utilisant efficacement les ressources disponibles. Ces problèmes peuvent survenir dans divers domaines tels que la fabrication industrielle, la logistique, l'informatique et la gestion de projet.

La résolution de problèmes d'ordonnancement implique généralement l'utilisation d'algorithmes, de techniques d'optimisation et de modèles mathématiques pour trouver la séquence d'exécution optimale qui répond à des objectifs et à des contraintes spécifiques. Il existe différentes approches pour résoudre ces problèmes, allant des techniques heuristiques qui fournissent des solutions quasi optimales aux techniques précises qui garantissent des solutions optimales.

En résumé, un problème d'ordonnancement est un problème de planification et de gestion des ressources qui nécessite de trouver l'ordre d'exécution optimal des tâches ou des activités compte tenu de certaines contraintes. La résolution de ces problèmes nécessite l'utilisation d'algorithmes, de techniques d'optimisation et de modèles mathématiques pour trouver les meilleures solutions possibles [4] .

1.3 Exemples concrets de problèmes d'ordonnancement

Voici quelques exemples concrets de problèmes d'ordonnancement :

- Job Shop : Dans un environnement de job shop, différentes tâches doivent être exécutées sur différentes machines dans un ordre spécifique. Chaque tâche a une durée différente sur chaque machine et il peut y avoir des contraintes de précédence entre les tâches. L'objectif est de trouver la séquence d'exécution optimale qui minimise le temps total de production ou maximise l'utilisation des ressources.
- Flow Shop : Dans un environnement de flow shop, les tâches doivent être exécutées dans un ordre fixe sur une série de machines. Chaque tâche doit passer par toutes les machines dans le même ordre. L'objectif est de trouver la séquence d'exécution optimale qui minimise le temps total de production ou maximise l'utilisation des ressources.
- Scheduling de ressources : Dans ce type de problème, il s'agit de planifier et d'assigner efficacement des ressources (telles que des employés, des machines, des véhicules, etc.) à un ensemble de tâches ou d'activités. Les ressources peuvent avoir des contraintes de disponibilité, de capacité et de compétences. L'objectif est d'optimiser l'utilisation des ressources, de minimiser les coûts ou de respecter des délais spécifiques.

Ces exemples illustrent différentes situations dans lesquelles la planification et l'ordonnancement sont essentiels pour assurer une utilisation efficace des ressources et atteindre les objectifs spécifiques. La résolution de ces problèmes peut être complexe en raison du grand nombre de variables et de contraintes à prendre en compte, mais des méthodes avancées telles que les algorithmes génétiques, les algorithmes de recherche locale et les techniques de programmation linéaire peuvent être utilisées pour trouver des solutions optimales ou proches de l'optimalité[5] , [3] .

1.4 L'impact des problèmes d'ordonnancement

Les problèmes d'ordonnancement ont un impact significatif sur l'efficacité opérationnelle et la rentabilité des entreprises. Voici quelques-uns des principaux impacts :

- ✓ Utilisation efficace des ressources : En trouvant la séquence d'exécution optimale des tâches ou des activités, les problèmes d'ordonnancement permettent d'utiliser les ressources disponibles de manière plus efficace. Cela signifie que les ressources telles que les machines, les employés, les véhicules, etc., sont utilisées de manière optimale, ce qui réduit les temps d'attente, les temps d'arrêt et les goulots d'étranglement. Cela permet d'augmenter la productivité et de maximiser l'utilisation des ressources, ce qui se traduit par une meilleure efficacité opérationnelle.
- ✓ Réduction des coûts : Les problèmes d'ordonnancement permettent de minimiser les coûts liés à la production ou à l'exécution des activités. En optimisant la séquence d'exécution, il est possible de réduire les temps de production, les temps d'attente, les temps de mise en place, les coûts de main-d'œuvre, les coûts de stockage, etc. Cela se traduit par des économies de coûts significatives pour l'entreprise, ce qui améliore sa rentabilité.
- ✓ Respect des délais et des engagements : Les problèmes d'ordonnancement permettent de respecter les délais et les engagements vis-à-vis des clients. En optimisant la séquence d'exécution, il est possible de minimiser les retards, de respecter les dates de livraison prévues et de répondre aux exigences des clients en termes de délais. Cela renforce la satisfaction des clients, la fidélité et la réputation de l'entreprise, ce qui peut conduire à une croissance des ventes et à une augmentation de la rentabilité.
- ✓ Flexibilité et réactivité : Les problèmes d'ordonnancement permettent de gérer efficacement les changements et les imprévus. En disposant d'un plan d'ordonnancement optimisé, il est plus facile de réagir rapidement aux changements de demande, aux pannes de machines, aux absences de personnel, etc. Cela permet d'assurer une meilleure flexibilité opérationnelle, d'optimiser les ressources disponibles et de minimiser les perturbations dans la production ou l'exécution des activités.

En résumé, les problèmes d'ordonnancement ont un impact significatif sur l'efficacité opérationnelle et la rentabilité des entreprises. En optimisant la séquence d'exécution des tâches ou des activités, ils permettent une utilisation plus efficace des ressources, une réduction des coûts, le respect des délais et des engagements, ainsi qu'une meilleure flexibilité et réactivité face aux changements. Cela se traduit par une amélioration globale de la performance et de la rentabilité de l'entreprise [6] .

1.5 La complexité des problèmes d'ordonnancement

Les problèmes d'ordonnancement sont des problèmes algorithmiques qui impliquent la planification et la séquence d'activités, de tâches ou de ressources dans le but d'optimiser un critère spécifique. La complexité des problèmes d'ordonnancement varie en fonction de plusieurs facteurs, tels que le nombre de tâches, les contraintes de précédence, les ressources disponibles et les objectifs d'optimisation [15].

Les problèmes d'ordonnancement sont généralement classés en plusieurs catégories en fonction de leurs caractéristiques. Voici quelques exemples de problèmes d'ordonnancement et de leur complexité associée :

- *Ordonnancement à une machine :* Dans ce cas, on cherche à ordonnancer un ensemble de tâches sur une seule machine pour minimiser la durée totale d'exécution. La complexité de ce problème est généralement linéaire en fonction du nombre de tâches.
- *Ordonnancement à machines parallèles :* Lorsque les tâches peuvent être exécutées en parallèle sur plusieurs machines, la complexité peut varier en fonction du nombre de machines et de tâches. Certains problèmes d'ordonnancement parallèle peuvent être NP-difficiles, ce qui signifie qu'ils appartiennent à la classe de problèmes pour lesquels il n'existe pas d'algorithme efficace connu pour résoudre les instances de grande taille.
- *Ordonnancement avec contraintes de précédence :* Lorsque les tâches doivent être exécutées dans un certain ordre en raison de dépendances, la complexité peut augmenter en fonction du graphe de précédence. Certains problèmes, comme le Problème de Chemin Critique (CPM), peuvent être résolus en temps polynomial, tandis que d'autres, comme le Problème d'Ordonnancement de Projet (PSD), peuvent être NP-difficiles.
- *Ordonnancement avec ressources limitées :* Lorsque les tâches nécessitent l'utilisation de ressources spécifiques (par exemple, des machines, des travailleurs), la complexité peut augmenter en fonction du nombre de ressources et des contraintes de disponibilité.
- *Ordonnancement multi-objectif :* Lorsque plusieurs critères d'optimisation doivent être pris en compte (par exemple, minimiser le temps d'exécution et les coûts), la complexité peut augmenter en raison de la nature combinatoire des compromis entre les objectifs.

La complexité exacte dépendra donc du problème d'ordonnancement spécifique que vous considérez, ainsi que des contraintes et des objectifs associés. Certains problèmes sont solubles en temps polynomial, tandis que d'autres peuvent nécessiter des approches heuristiques ou des méthodes d'optimisation plus avancées pour résoudre des instances de grande taille [15].

2. Types de problèmes d'ordonnancement

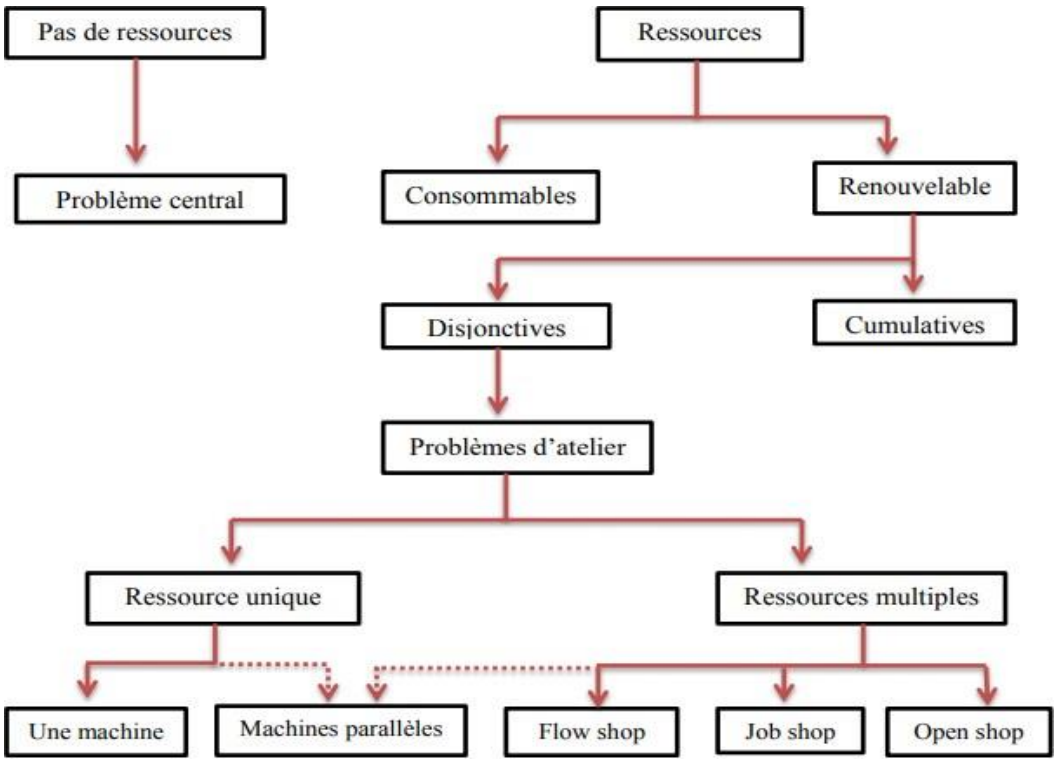

Figure I 1 : La Représentation typologie des problèmes d'ordonnancement [18].

2.1 Caractéristiques et les contraintes spécifiques

Explication des caractéristiques et des contraintes spécifiques à chaque type de problème d'ordonnancement

- **Problème de flow shop :**
  - Caractéristiques : Les tâches doivent être exécutées dans un ordre fixe sur une série de machines. Chaque tâche doit passer par toutes les machines dans le même ordre.
  - Contraintes spécifiques : Il n'y a généralement pas de contraintes de préemption, c'est-à-dire qu'une fois qu'une tâche a commencé à être traitée sur une machine, elle ne peut pas être interrompue. De plus, les temps d'exécution des tâches sont souvent constants [8] .
- **Problème de job shop :**
  - Caractéristiques : Les tâches doivent être exécutées sur un ensemble de machines, mais l'ordre d'exécution des tâches sur chaque machine peut varier. Chaque tâche a une séquence d'opérations à effectuer sur différentes machines.

- Contraintes spécifiques : Les tâches doivent respecter les contraintes de précédence, c'est-à-dire que certaines tâches doivent être terminées avant que d'autres ne puissent commencer. De plus, les temps d'exécution des tâches peuvent varier [8] .

➢ **Problème de Scheduling de ressources :**

- Caractéristiques : Ce type de problème concerne la planification des ressources pour l'exécution des tâches. Les ressources peuvent inclure des machines, des travailleurs, des véhicules, etc.
- Contraintes spécifiques : Les ressources sont limitées et doivent être allouées de manière optimale pour minimiser les temps d'attente ou maximiser la productivité. Il peut y avoir des contraintes de disponibilité des ressources, des contraintes de capacité, des contraintes de compétences, etc. [8] .

➢ **Problème de flow shop hybride :**

- Caractéristiques : C'est une combinaison du problème de flow shop et du problème de job shop. Certaines tâches doivent suivre une séquence fixe sur les machines, tandis que d'autres tâches peuvent varier leur ordre d'exécution sur certaines machines.
- Contraintes spécifiques : Il y a à la fois des contraintes de séquencement fixe et des contraintes de séquencement variable. Cela peut rendre le problème plus complexe, car il faut trouver un équilibre entre les deux types de contraintes [8] .

➢ **Problème de Scheduling sous contraintes :**

- Caractéristiques : Ce type de problème concerne la planification des tâches en tenant compte de contraintes spécifiques telles que les dépendances entre les tâches, les contraintes de ressources, les contraintes de temps, etc.
- Contraintes spécifiques : Les contraintes peuvent varier en fonction du contexte spécifique. Par exemple, il peut y avoir des contraintes de précédence strictes, des contraintes de disponibilité des ressources à des moments spécifiques, des contraintes de temps de traitement maximum, etc. La complexité du problème dépend des contraintes spécifiques qui doivent être respectées [8] .

2.2 Notation et classification des problèmes d'ordonnancement

Etant donné la différence des problèmes d'ordonnancement, nous utilisons couramment un formalisme de classification, permettant de distinguer les problèmes d'ordonnancement entre eux et de les classer. Ce formalisme, comporte trois champs : α, β, γ. Permettant de décrire les différentes entités d'un problème d'ordonnancement [2].

2.2.1 Champ α : Organisation des ressources

Le champ α décrit la structure des problèmes ordonnancement et se décompose on deux sous champs : a1 indique la nature des problèmes et a2 désire le nombre des machines.

Le paramètre α1 ∈ {1 ou ∅, P, Q, R, O, F, J, FH, JG, OG} caractérise le type de machines utilisées. Le paramètre α2 ∈ {∅, k} caractérise le nombre de machines utilisées dans le problème [2].

2.2.2 Champ β : les types de contraintes et caractéristiques du système

Le second champ β = β1 β2 β3 β4 β5 β6 β7 β8 décrit les types de contraintes prises en compte et les caractéristiques de la ressource [2].

2.2.3 Champ γ : fonction objectif

Le troisième champ, le champ γ, indique les fonctions objectives considérées ou les descriptions des critères d'évaluation d'un ordonnancement. Les objectifs visés sont liés à une bonne utilisation des ressources, une minimisation du délai global ou encore le respect d'un maximum de contraintes. Nous donnons ici les critères les plus couramment utilisés : Cmax (Makespan), TMax, … [2].

2.3 La représentation des problèmes d'ordonnancement

Il existe trois sortes de représentations possibles d'un problème d'ordonnancement:

Le diagramme de Gantt, le graphe Potentiel-Taches et la méthode PERT.

2.3.1 Le diagramme de Gantt

Le diagramme de Gantt est un outil permettant de modéliser la planification des taches Nécessaires à la réalisation d'un projet. Il s'agit d'un outil élabore en 1917 par Henry L. Gantt.

Etant donne la facilite relative de lecture des diagrammes de Gantt, cet outil est utilise Par la quasi-totalité des chefs de projet dans tous les secteurs. Il permet de représenter Graphiquement l'avancement du projet et constitue également un bon moyen de Communication entre les différents acteurs d'un projet. Le diagramme de Gantt présente

En ordonnée la liste des taches, notées 'Ti' a exécuter par les machines notées 'Mj' et en Abscisse l'échelle du temps [16].

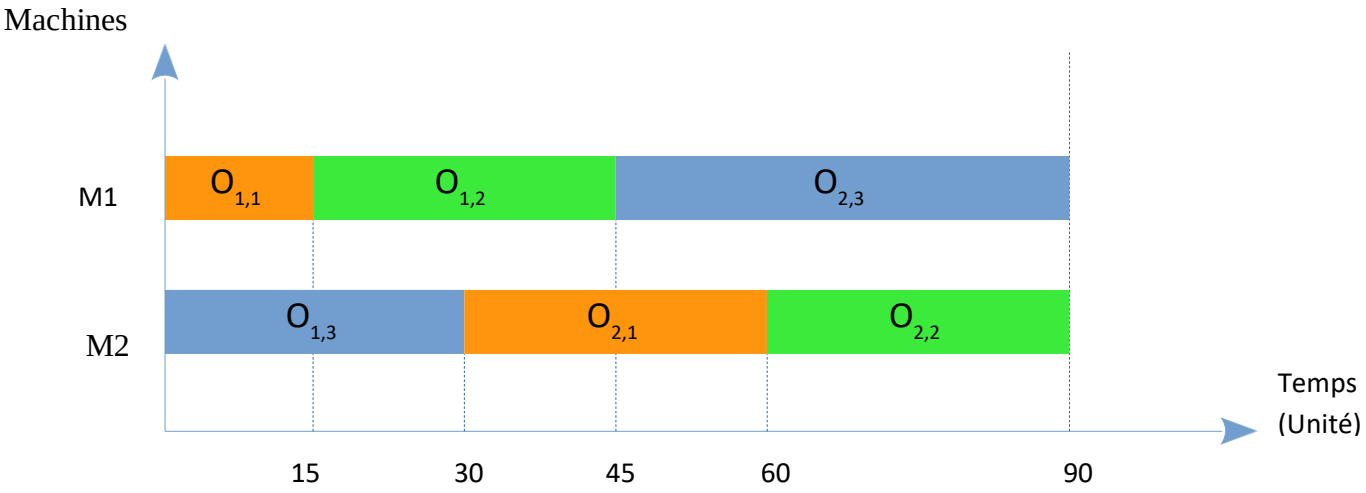

Figure I 2 : Diagramme de Gantt d'un ordonnancement [16].

2.3.2 Graphe potentiel-tâches

Cet outil graphique a été développé grâce à la théorie des réseaux de Pétri qui ont surtout servi à modéliser les systèmes dynamiques à événements discrets [16].

Dans ce genre de modélisation, les tâches sont représentées par des nœuds et les Contraintes par des arcs. Ainsi, les arcs peuvent être de deux types :

- Les arcs conjonctifs illustrant les contraintes de précédence et indiquant les durées des tâches
- les arcs disjonctifs indiquant les contraintes de ressources

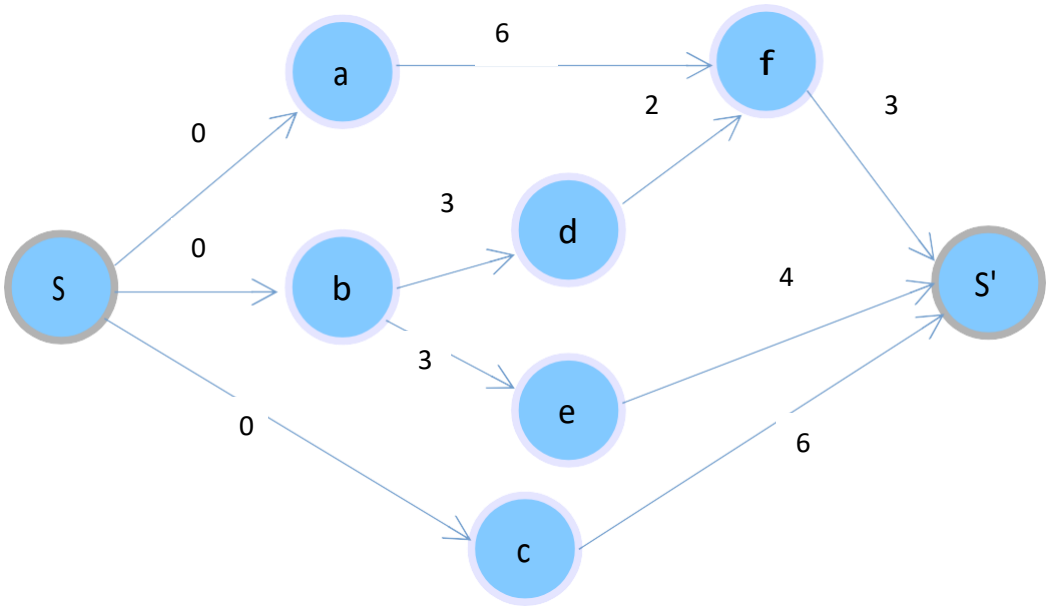

Figure I 3 : Graphe potentiel-tâches d'un ordonnancement [16].

2.3.3 Méthode PERT (Program Evaluation and Research Task)

Cette représentation, semblable à la précédente, permet de représenter une tâche par un arc, auquel est associé un chiffre qui représente la durée de la tâche. Entre les arcs, figurent des cercles, appelés sommets ou événements, qui marquent l'aboutissement d'une ou de plusieurs tâches. Ces cercles sont numérotés afin de suivre l'ordre de succession des divers événements.

2.4 Les problèmes d'ordonnancement déterministes vs stochastiques

Les problèmes d'ordonnancement déterministes et stochastiques diffèrent principalement par la nature des données et des paramètres utilisés pour la planification. Voici une illustration des différences entre les deux types de problèmes :

- **Problème d'ordonnancement déterministe :**
  - Données : Dans un problème d'ordonnancement déterministe, toutes les données nécessaires pour la planification sont connues et fixes. Cela signifie que les temps d'exécution des tâches, les contraintes de précédence, les capacités des ressources, etc., sont tous déterministes et ne varient pas.

- Planification : La planification consiste à trouver une séquence optimale des tâches et des ressources pour minimiser le temps total d'exécution, maximiser le débit ou atteindre tout autre objectif spécifique. Les contraintes et les objectifs sont définis de manière précise et ne changent pas au cours de la planification.

❖ **Problème d'ordonnancement stochastique :**

- Données : Dans un problème d'ordonnancement stochastique, certaines données sont incertaines et soumises à des variations aléatoires. Par exemple, les temps d'exécution des tâches peuvent varier en fonction de facteurs imprévisibles tels que les pannes de machines, les variations de la charge de travail, etc.
- Planification : La planification consiste à trouver une stratégie robuste qui gérer les variations aléatoires et minimiser les impacts négatifs sur les performances. Cela peut impliquer l'utilisation de techniques de prévision, de règles d'affectation flexibles, de politiques de réaffectation dynamique des ressources, etc. L'objectif est de maximiser la satisfaction des contraintes et des objectifs malgré l'incertitude.

En résumé, les problèmes d'ordonnancement déterministes se basent sur des données fixes et cherchent à trouver une solution optimale, tandis que les problèmes d'ordonnancement stochastiques prennent en compte l'incertitude et cherchent à trouver des stratégies robustes pour faire face aux variations aléatoires [9] .

3. Objectifs et contraintes des problèmes d'ordonnancement

3.1 Discussion des objectifs courants dans les problèmes d'ordonnancement

L'ordonnancement est un domaine fondamental dans la gestion des opérations et de la planification, utilisé dans une variété de contextes tels que la production, la logistique, l'informatique et bien d'autres. Les objectifs de l'ordonnancement varient en fonction du contexte et des contraintes spécifiques de chaque problème. Voici une discussion sur quelques objectifs courants dans les problèmes d'ordonnancement :

a) **Minimisation du temps d'exécution (makespan) :**L'objectif principal ici est de réduire la durée totale nécessaire pour terminer un ensemble de tâches. Cela peut être crucial dans des domaines tels que la production et la planification de projets, où une exécution plus rapide peut conduire à une meilleure efficacité et à des économies de coûts.

b) **Maximisation de l'utilisation des ressources :** Cet objectif vise à optimiser l'utilisation des ressources disponibles, comme la main-d'œuvre, les machines, les véhicules, etc. Une utilisation maximale des ressources peut conduire à une meilleure rentabilité et à une réduction du gaspillage.

c) **Réduction des délais (tardiness) :** Dans certains cas, l'accent est mis sur la minimisation des retards par rapport aux dates d'échéance souhaitées ou imposées. Cela est crucial lorsque le respect des délais est essentiel pour satisfaire les clients ou pour maintenir l'efficacité globale de l'entreprise.

d) **Minimisation des coûts :** Dans de nombreux problèmes d'ordonnancement, les coûts liés à l'utilisation des ressources, aux déplacements, aux retards, etc., peuvent être pris en compte. L'objectif est alors de minimiser les coûts totaux tout en respectant les contraintes opérationnelles.

a) **Équilibrage des charges (load balancing) :** Lorsque les ressources sont disponibles en quantités limitées et que les tâches diffèrent en termes de temps d'exécution ou d'exigences en ressources, l'équilibrage des charges vise à répartir équitablement les tâches entre les ressources pour éviter les goulots d'étranglement.

b) **Minimisation des temps d'attente :** Dans les systèmes où les tâches attendent d'être traitées, l'objectif est de minimiser les temps d'attente. Cela peut être important dans les files d'attente, les centres d'appels, etc.

c) **Maximisation de la priorité :** Certains problèmes d'ordonnancement exigent que certaines tâches soient traitées avec une priorité plus élevée que d'autres. Cela peut être lié à l'importance stratégique ou à l'urgence d'une tâche donnée.

d) **Minimisation des changements d'outils ou de configurations :** Dans les environnements de production et de fabrication, minimiser les changements d'outils ou de configurations peut réduire les temps d'arrêt et améliorer l'efficacité globale.

e) **Optimisation des séquences :** Dans certains cas, l'objectif est d'optimiser l'ordre dans lequel les tâches sont exécutées pour maximiser un certain critère, comme la satisfaction du client ou la séquence optimale pour une série d'opérations.

Il est important de noter que ces objectifs ne sont pas mutuellement exclusifs et peuvent être en conflit les uns avec les autres. La sélection de l'objectif approprié dépendra du contexte spécifique du problème, des contraintes opérationnelles et des priorités de l'entreprise. De plus, il existe des méthodes d'optimisation, des algorithmes et des techniques spécifiques pour aborder chacun de ces objectifs dans le contexte de l'ordonnancement [10] .

3.2 Exploration

Exploration des compromis et des défis liés à l'optimisation des objectifs tout en respectant les contraintes :

L'optimisation des objectifs tout en respectant les contraintes est l'un des défis majeurs dans le domaine de l'ordonnancement et de la planification. Cela implique de trouver un équilibre entre les objectifs concurrents tout en satisfaisant les contraintes opérationnelles. Cette situation donne souvent lieu à des compromis et soulève des défis complexes. Voici une exploration de ces compromis et défis :

1. Compromis entre objectifs : Dans la plupart des problèmes d'ordonnancement, il est rare que les objectifs soient mutuellement bénéfiques sans aucune concession. Par exemple, minimiser le temps d'exécution peut parfois entraîner une surutilisation des ressources, tandis que maximiser l'utilisation des ressources peut prolonger le temps d'exécution. Trouver un compromis entre ces objectifs concurrents nécessite une évaluation minutieuse des priorités et des objectifs de l'entreprise.

2. Complexité combinatoire : Les problèmes d'ordonnancement ont souvent une complexité combinatoire élevée, ce qui signifie que le nombre de solutions possibles augmente rapidement avec la taille du problème. L'exploration de toutes les solutions possibles pour respecter les contraintes peut être pratiquement impossible dans de nombreux cas. Cela nécessite l'utilisation d'algorithmes efficaces et d'heuristiques pour rechercher des solutions de haute qualité en un temps raisonnable.

3. Dépendance entre contraintes : Les contraintes peuvent être interdépendantes et parfois conflictuelles. Par exemple, satisfaire une contrainte de précédence peut entrer en conflit avec une contrainte de capacité. La prise en compte de ces dépendances ajoute une complexité supplémentaire à la résolution du problème.

4. Non-linéarités : Les problèmes d'ordonnancement peuvent présenter des relations non-linéaires entre les variables, ce qui rend la recherche de solutions optimales plus difficile. Les algorithmes traditionnels basés sur la programmation linéaire peuvent ne pas être suffisamment adaptés pour gérer ces non-linéarités.

5**. Exploration de l'espace des solutions :** Trouver la meilleure solution dans un espace de solutions complexe peut être difficile. Les algorithmes d'optimisation doivent équilibrer

l'exploration (recherche de nouvelles solutions) et l'exploitation (amélioration des solutions existantes) pour éviter de rester bloqués dans des solutions sous-optimales.

6. Incertitude : Les conditions opérationnelles peuvent changer, et les contraintes peuvent devenir moins strictes ou plus strictes au fil du temps. La planification initiale doit être flexible pour faire face à ces changements, ce qui peut introduire des incertitudes dans la recherche de solutions optimales.

7. Optimisation multi-objectif : Lorsque plusieurs objectifs sont à considérer, il peut être difficile de déterminer comment les pondérer. Les compromis entre les objectifs peuvent être difficiles à évaluer, et trouver la "meilleure" solution dans ce contexte peut être subjectif.

8. Heuristiques et Métaheuristiques : Étant donné la complexité des problèmes d'ordonnancement, il est souvent nécessaire de recourir à des heuristiques et des métaheuristiques, comme les algorithmes génétiques, les algorithmes de colonies d'abeilles, les méthodes de recherche tabou, etc. Cependant, trouver une heuristique efficace qui converge rapidement vers des solutions de haute qualité peut être un défi soi.

En résumé, l'optimisation des objectifs tout en respectant les contraintes est un défi complexe qui nécessite une approche soigneusement équilibrée entre la recherche de solutions optimales, la prise en compte des contraintes et la compréhension des compromis entre les différents objectifs. Des avancées continues dans les algorithmes d'optimisation et l'utilisation de techniques avancées d'intelligence artificielle contribuent à résoudre ces problèmes de manière plus efficace et précise [11] .

4. Approches pour résoudre les problèmes d'ordonnancement

4.1 Présentation des approches exacts

Il existe plusieurs approches classiques pour résoudre les problèmes d'ordonnancement, chacune avec ses avantages et ses limitations. Voici une présentation des principales approches utilisées :

4.1.1 Méthodes exactes : Les méthodes exactes visent à trouver la solution optimale en explorant l'ensemble complet des solutions possibles. Cela inclut les méthodes de recherche exhaustive (énumération), la programmation linéaire entière (PLNE) et la programmation en nombres mixtes (PNM). Les méthodes exactes sont efficaces pour de petits problèmes, mais leur complexité combinatoire les rend souvent impraticables pour des problèmes de grande taille [12]

4.1.2 Méthodes basées sur les règles : Ces méthodes exploitent des règles et des critères prédéfinis pour ordonner les tâches. Les règles peuvent être simples (ex. : traiter les tâches avec les délais les plus courts en premier) ou plus complexes (ex. : prioriser les tâches à faible temps d'exécution et haute priorité). Les méthodes basées sur les règles sont rapides et faciles à mettre en œuvre, mais elles peuvent ne pas garantir une solution optimale ou de haute qualité [14].

4.1.3 Méthodes de recherche locale : Les méthodes de recherche locale commencent avec une solution initiale et effectuent des mouvements locaux pour améliorer progressivement la solution. Les exemples incluent la recherche tabou, la descente de gradient et les algorithmes génétiques. Bien qu'elles puissent être rapides, ces méthodes peuvent parfois rester coincées dans des optima locaux [13].

4.1.4 Méthodes métaheuristiques : Les métaheuristiques sont des approches générales de résolution de problèmes qui guident la recherche vers des solutions de qualité en explorant l'espace des solutions de manière intelligente. Parmi les métaheuristiques populaires, on trouve les algorithmes génétiques, les essaims de particules, les colonies d'abeilles artificielles et la recherche de voisinage variable. Les métaheuristiques sont flexibles et peuvent être adaptées à différents problèmes d'ordonnancement ACO [15].

4.1.5 Méthodes basées sur la simulation : Pour les problèmes d'ordonnancement complexes et dynamiques, les méthodes basées sur la simulation peuvent être utilisées. Elles modélisent le système sous forme de simulation informatique et évaluent différentes stratégies d'ordonnancement en observant comment elles se comportent dans des conditions simulées. Cela permet d'analyser les performances de différentes solutions sans perturber le système réel [13].

4.1.6 Méthodes de planification par contraintes : Ces méthodes se concentrent sur la modélisation des contraintes opérationnelles et temporelles du problème à l'aide de contraintes logiques. Les solveurs de contraintes essaient de trouver une solution qui satisfait toutes les contraintes simultanément. Cela peut être utile pour des problèmes où les contraintes sont complexes et interdépendantes [13].

En pratique, la combinaison de plusieurs approches, comme l'utilisation d'une heuristique pour fournir une solution initiale à une méthode exacte ou métaheuristique, peut être très efficace pour résoudre des problèmes d'ordonnancement complexes. Le choix de la méthode dépendra de la taille du problème, de la disponibilité des ressources de calcul et de la complexité des contraintes spécifiques.

4.2 Explication des avantages et des limites de déférent approche

Approche	Avantages	Limites
Méthodes exactes	- Garantit une solution optimale - Convient aux problèmes de petite taille	- Impraticable pour les problèmes de grande taille - Complexité combinatoire élevée - Demande beaucoup de temps de calcul
Méthodes heuristiques	- Rapides, fournissent rapidement des solutions Acceptables - Adaptées aux problèmes de grande taille - Faciles à mettre en œuvre	- Solutions pas garanties d'être optimales - Qualité des solutions dépend de la stratégie utilisée - Sensibles aux paramètres et à la configuration du problème
Méthodes basées sur les règles	- Simples à mettre en œuvre - Rapides	- Peuvent donner des solutions sous-optimales - Ne prennent pas en compte toutes les contraintes et objectifs

Méthodes de recherche locale	- Convergent rapidement vers des solutions de qualité supérieure dans des espaces locaux	- Peuvent se coincer dans des optima locaux - N'explorent pas toujours l'espace global des solutions - Nécessitent parfois des mécanismes de diversification pour éviter les optima locaux
Méthodes métaheuristiques	- Combinent des stratégies intelligentes pour explorer l'espace des solutions de manière plus efficace - Flexibles et adaptables à différents problèmes	- Ne garantissent pas la solution optimale - Réglage des paramètres peut être complexe - Nécessitent souvent une expertise
Méthodes basées sur la simulation	- Modélisent des systèmes complexes et dynamiques - Permettent d'observer les performances de différentes stratégies d'ordonnancement dans un environnement contrôlé	- Reposent sur des hypothèses et des modèles - Résultats dépendent de la qualité des modèles et de la validation empirique

Tableau I 1 : les avantages et les limites de chaque approche pour résoudre les problèmes d'ordonnancement

Il est important de noter que le choix de l'approche dépendra du contexte spécifique du problème, des ressources disponibles et des objectifs à atteindre. Dans de nombreux cas, une combinaison intelligente de plusieurs approches peut fournir les meilleurs résultats [16].

4.3 Exemples d'algorithmes couramment utilisés

- **Algorithme de Johnson :** Cet algorithme est utilisé pour résoudre les problèmes d'ordonnancement de machines parallèles avec deux types de ressources (généralement des machines) et des tâches à exécuter. L'algorithme de Johnson trouve une séquence d'ordonnancement optimale en minimisant le temps de terminaison total (makespan). Il s'agit d'une méthode heuristique qui traite les tâches en fonction de leurs temps de traitement sur chaque type de ressource. Cet algorithme est souvent utilisé dans des contextes tels que la planification de la production et la gestion des ateliers.
- **Algorithme de séquencement de priorité (Priority Scheduling) :** C'est une approche basée sur les règles où les tâches sont ordonnées en fonction de leur priorité. Les tâches à haute priorité sont exécutées en premier. Cet algorithme est souvent utilisé dans les systèmes d'exploitation pour l'ordonnancement de processus ou de tâches. Cependant, il peut entraîner des problèmes de famine (où certaines tâches à faible priorité ne sont jamais exécutées).
- **Algorithme de Round-Robin :** Cet algorithme est principalement utilisé pour l'ordonnancement des processus dans les systèmes d'exploitation. Il fonctionne en attribuant un quantum de temps à chaque processus, et lorsque le quantum est écoulé, le processus en cours est suspendu et le processus suivant est exécuté. Cela permet de donner une chance égale à chaque processus de s'exécuter, mais cela peut entraîner une latence élevée pour les processus longs.
- **Algorithme de recherche tabou (Tabu Search) :** C'est une méthode métaheuristique utilisée pour résoudre des problèmes d'ordonnancement complexes en explorant l'espace des solutions. L'algorithme de recherche tabou utilise une liste tabou pour éviter de revisiter des solutions déjà explorées. Il permet d'éviter de rester coincé dans des optima locaux et peut fournir des solutions de meilleure qualité.
- **Algorithme génétique :** Les algorithmes génétiques sont des méthodes d'optimisation basées sur des processus inspirés de l'évolution biologique. Ils sont utilisés pour résoudre

une variété de problèmes d'ordonnancement en générant et en évoluant une population de solutions candidates. Les algorithmes génétiques permettent d'explorer de manière efficace l'espace des solutions et peuvent être adaptés pour traiter différents objectifs et contraintes.

- **Algorithme de recherche locale itérative :** Cette approche repose sur l'amélioration progressive d'une solution initiale. Elle explore les solutions voisines en effectuant des modifications locales, telles que l'échange de positions de tâches ou l'ajustement des horaires. L'algorithme recherche ensuite la meilleure solution parmi les voisins améliorés. C'est une méthode simple mais efficace pour améliorer les solutions existantes.
- **Algorithme de recherche en essaim de particules (Particle Swarm Optimization, PSO)** : Inspiré par le comportement social des oiseaux et des poissons, cet algorithme métaheuristique simule le mouvement de particules dans un espace de recherche. Les particules coopèrent pour trouver des solutions optimales en explorant et en exploitant l'espace des solutions de manière coordonnée.

Ces exemples ne représentent qu'une fraction des nombreux algorithmes et approches utilisés pour résoudre des problèmes d'ordonnancement. Le choix de l'algorithme dépendra du type de problème, des contraintes spécifiques et des ressources disponibles [9] .

5. conclusion

Nous avons discuté des différentes approches et méthodes utilisées pour résoudre ces problèmes d'ordonnancement. Nous avons notamment évoqué les algorithmes heuristiques, qui permettent de trouver des solutions rapidement mais qui ne garantissent pas l'optimalité, ainsi que les méthodes exactes, qui garantissent l'optimalité mais qui peuvent être plus coûteuses en termes de temps de calcul.

De plus, nous avons souligné l'importance de la modélisation mathématique dans la résolution des problèmes d'ordonnancement. En effet, la construction d'un modèle mathématique précis et adapté est essentielle pour obtenir des solutions optimales ou proches de l'optimalité. Nous avons également mentionné l'utilisation de logiciels spécialisés dans la résolution de ces problèmes, qui permettent de simplifier et d'automatiser les étapes de modélisation et de résolution.

Enfin, nous avons mis en évidence les enjeux et les défis liés aux problèmes d'ordonnancement. En effet, l'optimisation de l'ordonnancement peut permettre d'améliorer l'efficacité et la

rentabilité des processus de production, de réduire les coûts de transport ou encore d'optimiser l'utilisation des ressources. Cependant, la complexité de ces problèmes et le nombre important de contraintes à prendre en compte rendent leur résolution difficile et nécessitent l'utilisation de techniques avancées.

En conclusion, ce premier chapitre nous a permis de comprendre l'importance des problèmes d'ordonnancement dans de nombreux domaines d'activité. Nous avons également identifié les différentes approches et méthodes utilisées pour résoudre ces problèmes, ainsi que les enjeux et les défis qui y sont associés. Dans les prochains chapitres, nous approfondirons ces concepts et présenterons des études de cas pour illustrer la résolution de problèmes d'ordonnancement spécifiques

Chapitre II :

Techniques d'Optimisation des Colonies de fourmis

1.1 Introduction à l'Optimisation des Colonies de Fourmis (ACO)

L'optimisation des colonies de fourmis (ACO) est une technique d'optimisation inspirée du comportement des colonies de fourmis lorsqu'elles recherchent de la nourriture. L'ACO est utilisé pour résoudre des problèmes d'optimisation complexes, tels que la planification de tâches dans les ateliers de production.

L'ACO repose sur deux principes clés : les phéromones et les heuristiques. Les fourmis communiquent entre elles en déposant des phéromones sur le sol, qui servent de signal pour guider les autres fourmis vers la source de nourriture. Les phéromones sont également évaporées avec le temps, ce qui permet aux fourmis de découvrir de nouvelles sources de nourriture. Les heuristiques sont des règles ou des stratégies qui aident les fourmis à prendre des décisions lorsqu'elles rencontrent des choix multiples.

Dans le contexte de l'ACO pour la planification des ateliers, les tâches sont représentées comme des nœuds et les machines comme des arêtes d'un graphe. Les fourmis construisent des solutions en sélectionnant des tâches selon des règles spécifiques, en tenant compte des phéromones déposées sur les arêtes et des heuristiques basées sur les caractéristiques des tâches et des machines. Les solutions construites par les fourmis sont ensuite évaluées et les phéromones sont mises à jour en fonction de leur qualité.

L'ACO présente plusieurs avantages pour la résolution des problèmes de planification des ateliers. Il est capable de trouver des solutions de qualité en explorant efficacement l'espace de recherche. De plus, l'ACO peut s'adapter aux changements dans l'environnement et trouver des solutions robustes. Cependant, l'ACO peut être sensible aux paramètres du problème et nécessite une configuration appropriée pour obtenir de bons résultats.

En résumé, l'optimisation des colonies de fourmis est une technique d'optimisation inspirée du comportement des fourmis. Elle est utilisée pour résoudre des problèmes d'optimisation complexes, tels que la planification des ateliers, en utilisant des phéromones et des heuristiques pour guider la recherche de solutions optimales. L'ACO présente des avantages tels que la capacité de trouver des solutions de qualité et de s'adapter aux changements, mais nécessite une configuration appropriée pour obtenir de bons résultats [17].

1.2 Concepts et principes clés

Les concepts et principes clés des algorithmes ACO (Ant Colony Optimization) sont les suivants :

1. ***Les fourmis virtuelles :*** Les fourmis virtuelles sont utilisées pour explorer l'espace de recherche et construire des solutions itérativement. Chaque fourmi suit des règles spécifiques pour choisir les nœuds suivants en fonction des phéromones déposées sur les arêtes et des heuristiques locales.
2. ***Les phéromones :*** Les phéromones jouent un rôle crucial dans les algorithmes ACO. Les fourmis déposent des phéromones sur les arêtes qu'elles empruntent, et ces phéromones sont mises à jour en fonction de la qualité des solutions construites. Les autres fourmis utilisent ces phéromones comme une indication de la qualité des chemins et sont plus susceptibles de choisir les arêtes avec des niveaux de phéromones plus élevés.
3. ***Les heuristiques locales :*** Les heuristiques locales fournissent des informations supplémentaires aux fourmis pour guider leur choix des nœuds suivants. Ces heuristiques sont généralement basées sur les caractéristiques des tâches et des machines dans le problème d'optimisation.
4. ***Construction de solutions :*** Les fourmis virtuelles construisent des solutions itérativement en choisissant les nœuds suivants en fonction des phéromones et des heuristiques locales. La construction de solutions peut être réalisée de manière stochastique ou déterministe, en fonction de la configuration de l'algorithme.
5. ***Mise à jour des phéromones :*** Les niveaux de phéromones sont ajustés en fonction de la qualité des solutions trouvées. Les arêtes empruntées par les fourmis qui ont construit de bonnes solutions reçoivent généralement une augmentation de phéromones, tandis que celles empruntées par les fourmis ayant construit de mauvaises solutions voient leurs phéromones diminuer.
6. ***Itérations et convergence :*** Les algorithmes ACO sont itératifs et effectuent plusieurs cycles de construction de solutions et de mise à jour des phéromones. Au fur et à mesure des itérations, les fourmis explorent de plus en plus les chemins avec des niveaux de phéromones plus élevés, ce qui peut conduire à la convergence vers une solution optimale.

7. ***Paramètres et configuration :*** Les algorithmes ACO nécessitent une configuration appropriée pour obtenir de bons résultats. Les paramètres tels que la quantité de phéromones déposées, l'importance des heuristiques locales, la taille de la colonie de fourmis, etc., doivent être ajustés en fonction du problème spécifique.

En utilisant ces concepts et principes clés, les algorithmes ACO peuvent trouver des solutions de qualité en explorant efficacement l'espace de recherche. Ils sont adaptatifs et peuvent s'ajuster aux changements dans l'environnement du problème, ce qui les rend utiles pour résoudre une variété de problèmes d'optimisation [18].

1.3 Domaines d'application de l'ACO

Les algorithmes ACO (Ant Colony Optimization) ont été appliqués avec succès dans de nombreux domaines, notamment :

- ***Problèmes de routage :*** Les algorithmes ACO sont souvent utilisés pour résoudre des problèmes de routage tels que le problème du voyageur de commerce (TSP) et le problème de routage de véhicules (VRP). Ils peuvent trouver des solutions de haute qualité en explorant efficacement l'espace de recherche des chemins optimaux.
- ***Problèmes d'ordonnancement :*** Les algorithmes ACO peuvent être utilisés pour résoudre des problèmes d'ordonnancement complexes tels que le problème de l'ordonnancement de production, le problème de l'ordonnancement de projet, etc. Ils peuvent trouver des solutions efficaces en optimisant la séquence des tâches ou des activités.
- ***Problèmes de clustering :*** Les algorithmes ACO peuvent être utilisés pour résoudre des problèmes de clustering, où l'objectif est de regrouper des objets similaires dans des groupes. Ils peuvent trouver des solutions de clustering de haute qualité en explorant l'espace de recherche des configurations de groupes.

- ***Problèmes de conception de réseaux :*** Les algorithmes ACO peuvent être utilisés pour résoudre des problèmes de conception de réseaux tels que le problème du routage dans les réseaux de télécommunications, le problème de conception de réseaux de distribution d'électricité, etc. Ils peuvent trouver des configurations de réseau optimales en optimisant les chemins ou les connexions.

- ***Problèmes de planification :*** Les algorithmes ACO peuvent être utilisés pour résoudre des problèmes de planification tels que la planification de projets, la planification de ressources, etc. Ils peuvent trouver des solutions de planification efficaces en optimisant l'allocation des ressources et la séquence des activités.
- ***Problèmes d'optimisation combinatoire :*** Les algorithmes ACO peuvent être utilisés pour résoudre une variété de problèmes d'optimisation combinatoire tels que le problème du sac à dos, le problème de l'arbre de Steiner, etc. Ils peuvent trouver des solutions optimales en explorant l'espace de recherche de manière efficace.

Ces domaines d'application ne sont qu'une sélection des nombreux domaines dans lesquels les algorithmes ACO ont été appliqués avec succès. Leur capacité à trouver des solutions de qualité et à s'adapter aux changements les rendent utiles pour résoudre une large gamme de problèmes d'optimisation [17].

2. Modélisation Des problèmes d'ordonnancement de type job shop

2.1 Définition et caractéristiques

Les problèmes d'ordonnancement de type job shop sont des problèmes d'optimisation combinatoire qui se posent dans le domaine de la planification de la production. Dans ce type de problème, il y a plusieurs tâches (jobs) qui doivent être exécutées sur une série de machines dans un ordre spécifique.

Voici les caractéristiques principales des problèmes d'ordonnancement de type job shop :

I. ***Jobs :*** Un problème de job shop implique un ensemble de jobs, où chaque job est composé d'un ensemble de tâches à exécuter. Chaque tâche doit être traitée sur une machine spécifique et a une durée d'exécution donnée.

II. ***Machines :*** Un problème de job shop implique un ensemble de machines, où chaque machine peut effectuer une ou plusieurs tâches spécifiques. Chaque tâche doit être exécutée sur une machine spécifique, et chaque machine ne peut exécuter qu'une seule tâche à la fois.

III. ***Contraintes de précédence :*** Les tâches d'un job doivent être exécutées dans un ordre spécifique, défini par des contraintes de précédence. Par exemple, la tâche B ne peut commencer qu'après l'achèvement de la tâche A.

IV. ***Objectif d'optimisation :*** L'objectif principal des problèmes d'ordonnancement de type job shop est généralement de minimiser la durée totale de production, également appelée

makespan. Il s'agit du temps écoulé depuis le début du premier job jusqu'à la fin du dernier job.

V. ***Complexité :*** Les problèmes d'ordonnancement de type job shop sont connus pour être NP-difficiles, ce qui signifie qu'il est difficile de trouver une solution optimale en un temps raisonnable pour des instances de grande taille. Cela rend l'utilisation d'algorithmes d'optimisation, tels que les algorithmes ACO, très utile pour résoudre ces problèmes.

La résolution des problèmes d'ordonnancement de type job shop est un défi important dans le domaine de la planification de la production. Les algorithmes d'optimisation, tels que l'ACO, peuvent être utilisés pour trouver des solutions de haute qualité en explorant efficacement l'espace de recherche des ordonnancements possibles [4] .

2.2 Formulation du problème et représentation mathématique

La formulation du problème d'ordonnancement de type job shop peut être décrite comme suit :

A. Données d'entrée :

- Un ensemble de jobs, noté J = {J1, J2, ..., Jn}, où chaque job Ji est composé d'un ensemble de tâches à exécuter.
- Un ensemble de machines, noté M = {M1, M2, ..., Mk}, où chaque machine Mj peut effectuer une ou plusieurs tâches spécifiques.
- Une matrice de temps de traitement, notée T, où T (i, j) représente la durée d'exécution de la tâche i du job j sur la machine correspondante.

B. Variables de décision :

- Une séquence d'ordonnancement, notée S = {S1, S2, ..., Sm}, où chaque Sj représente la tâche j à exécuter.
- Une séquence d'affectation de machines, notée A = {A1, A2, ..., Am}, où chaque Aj représente la machine attribuée à la tâche j.

C. Contraintes :

- Chaque tâche doit être exécutée exactement une fois.
- Chaque machine ne peut exécuter qu'une seule tâche à la fois.
- Les contraintes de précédence doivent être respectées, c'est-à-dire que certaines tâches doivent être exécutées avant d'autres.

D. Objectif :

- Minimiser la durée totale de production (makespan), qui est le temps écoulé depuis le début du premier job jusqu'à la fin du dernier job.

La représentation mathématique du problème d'ordonnancement de type job shop peut varier en fonction de l'approche utilisée pour le résoudre. Cela peut inclure des équations pour calculer le temps de début et de fin de chaque tâche, des contraintes de précédence, des contraintes d'affectation de machines, etc.

Par exemple, une formulation mathématique simple peut être :

Minimiser makespan = max (Fin(Jn))

Sous contraintes :

- Pour chaque tâche j du job i, le temps de début est supérieur ou égal à la fin de la tâche précédente : Début(j) ≥ Fin (j-1)
- Pour chaque tâche j du job i, le temps de fin est égal au temps de début plus la durée de traitement : Fin(j) = Début(j) + T (j, A(j))

Cependant, il est important de noter que la formulation mathématique précise peut varier en fonction des détails spécifiques du problème et de l'approche de résolution utilisée [8] .

2.3 Défis et complexités de l'ordonnancement de type job shop

Les problèmes d'ordonnancement de type job shop présentent plusieurs défis et complexités :

- ***Complexité algorithmique :*** Les problèmes d'ordonnancement de type job shop sont connus pour être NP-difficiles, ce qui signifie qu'il est difficile de trouver une solution optimale en un temps raisonnable pour des instances de grande taille. La complexité algorithmique élevée rend la résolution de ces problèmes très difficile.
- ***Nombre élevé de combinaisons possibles :*** Le nombre de combinaisons possibles d'ordonnancements pour les tâches et les machines peut être très élevé, en particulier pour des instances de grande taille. Cela rend l'exploration de l'espace de recherche extrêmement complexe et nécessite des algorithmes d'optimisation efficaces pour trouver des solutions de haute qualité.
- ***Contraintes de précédence :*** Les contraintes de précédence entre les tâches peuvent rendre la planification plus complexe. Certaines tâches doivent être exécutées avant d'autres, ce qui ajoute une dimension supplémentaire à la recherche de la meilleure séquence d'ordonnancement.

- ***Variabilité des temps de traitement :*** Les temps de traitement des tâches peuvent varier considérablement, ce qui rend la planification plus difficile. Il est nécessaire de prendre en compte ces variations pour optimiser l'ordonnancement et minimiser le temps total de production.
- ***Gestion des ressources :*** Les problèmes d'ordonnancement de type job shop impliquent la gestion efficace des ressources, telles que les machines et les travailleurs. Il est important de prendre en compte les contraintes de capacité et d'affectation des ressources pour obtenir un ordonnancement réalisable.
- ***Incertitude et dynamisme :*** Dans certains cas, les problèmes d'ordonnancement de type job shop peuvent être soumis à des variations imprévues, des interruptions ou des changements de priorités. La gestion de l'incertitude et du dynamisme ajoute une complexité supplémentaire à la résolution de ces problèmes.

les algorithmes génétiques, les algorithmes à colonies de fourmis (ACO) ou les méthodes de recherche locale sont résolution efficace pour des problèmes d'ordonnancement de type job shop Ces algorithmes peuvent être utilisés pour explorer efficacement l'espace de recherche et trouver des solutions de haute qualité, même pour des instances de grande taille et des problèmes complexes [19] .

3. ACO pour des problèmes d'ordonnancement de type job shop

3.1 Techniques ACO pour des problèmes d'ordonnancement de type job shop

Les algorithmes à colonies de fourmis (ACO) sont des techniques d'optimisation populaires qui peuvent être adaptées pour résoudre des problèmes d'ordonnancement de type job shop. Voici comment les techniques ACO peuvent être adaptées pour résoudre ces problèmes :

- ***Représentation des solutions :*** La première étape consiste à définir une représentation appropriée pour les solutions du problème d'ordonnancement de type job shop. Cela peut être réalisé en utilisant une matrice ou un graphe pour représenter l'ordonnancement des tâches sur les machines.
- ***Construction du graphe des phéromones :*** Un graphe des phéromones est construit pour représenter les informations partagées par les fourmis lors de la recherche de solutions. Chaque arc du graphe représente une transition entre deux tâches, et la quantité de phéromones déposées sur chaque arc est mise à jour en fonction de la qualité des solutions trouvées par les fourmis.

- ✓ ***Déplacement des fourmis :*** Les fourmis sont placées sur des tâches initiales et se déplacent d'une tâche à l'autre en suivant des règles spécifiques. Les règles de déplacement peuvent être basées sur la quantité de phéromones présente sur les arcs, ainsi que sur des heuristiques locales pour guider le choix des tâches suivantes.
- ✓ ***Mise à jour des phéromones :*** Après que toutes les fourmis ont terminé leur déplacement, les phéromones sont mises à jour pour refléter la qualité des solutions trouvées. Les arcs qui ont été empruntés par les fourmis lors de la construction de solutions de haute qualité reçoivent une quantité plus élevée de phéromones, tandis que les arcs qui ont été moins empruntés reçoivent une quantité plus faible de phéromones.
- ✓ ***Exploration et exploitation :*** Les algorithmes ACO combinent l'exploration de l'espace de recherche pour découvrir de nouvelles solutions potentiellement meilleures et l'exploitation des informations partagées par les fourmis pour améliorer les solutions existantes. Cela permet de trouver progressivement des solutions de plus en plus optimales au fil du temps.

- ✓ ***Critère d'arrêt :*** L'algorithme ACO s'arrête lorsque certaines conditions prédéfinies sont satisfaites, telles que le nombre maximum d'itérations atteint, la stagnation de la qualité des solutions ou l'atteinte d'une solution optimale connue.

En adaptant les techniques ACO pour des problèmes d'ordonnancement de type job shop, il est important de prendre en compte les contraintes spécifiques du problème, telles que les contraintes de précédence, les contraintes de capacité des machines et les contraintes d'affectation des ressources. Ces contraintes peuvent être intégrées dans les règles de déplacement des fourmis et les mises à jour des phéromones pour garantir la faisabilité des solutions générées [21] .

3.2 Règles de mise à jour des pistes de phéromones et heuristiques pour ACO

Les règles de mise à jour des pistes de phéromones et les heuristiques sont des éléments clés pour guider le comportement des fourmis dans les algorithmes à colonies de fourmis (ACO). Voici quelques exemples de règles de mise à jour des pistes de phéromones et d'heuristiques utilisées dans les adaptations ACO pour les problèmes d'ordonnancement de type job shop :

1. ***Mise à jour des pistes de phéromones :***

- ***Évaporation :*** Les pistes de phéromones présentes sur les arcs peuvent s'évaporer progressivement pour éviter une accumulation excessive de phéromones. Cela peut être réalisé en multipliant les valeurs des pistes de phéromones par un facteur d'évaporation à chaque itération de l'algorithme.
- ***Dépôt local :*** Lorsqu'une fourmi emprunte un arc, elle peut déposer une quantité de phéromones locale sur cet arc. La quantité de phéromones déposée peut être proportionnelle à la qualité de la solution trouvée par la fourmi.
- ***Dépôt global :*** Après que toutes les fourmis ont terminé leur déplacement, les arcs empruntés par les fourmis lors de la construction des meilleures solutions peuvent recevoir une quantité supplémentaire de phéromones. La quantité de phéromones déposée peut être proportionnelle à la qualité de la meilleure solution trouvée parmi toutes les fourmis.

2. Heuristiques :

- ***Règle de choix de tâche :*** Lorsqu'une fourmi doit choisir la prochaine tâche à exécuter, elle peut utiliser une heuristique pour évaluer la qualité potentielle de chaque tâche. Cette heuristique peut être basée sur des informations telles que le temps de traitement restant pour chaque tâche, la priorité des tâches ou d'autres critères spécifiques au problème.
- Règle de choix de machine : Lorsqu'une fourmi doit choisir la machine sur laquelle exécuter la prochaine tâche, elle peut également utiliser une heuristique pour évaluer la qualité potentielle de chaque machine. Cette heuristique peut être basée sur des informations telles que la charge de travail actuelle de chaque machine, les temps de traitement des tâches sur chaque machine ou d'autres critères spécifiques au problème.

L'adaptation des règles de mise à jour des pistes de phéromones et des heuristiques dépendra des caractéristiques spécifiques du problème d'ordonnancement de type job shop. Il est important de prendre en compte les contraintes de précédence, les contraintes de capacité des machines et les contraintes d'affectation des ressources lors de la conception de ces règles pour garantir la faisabilité des solutions générées par l'algorithme ACO[17] .

3.3 Équilibre exploration vs exploitation dans ACO pour l'ordonnancement

L'équilibre entre l'exploration et l'exploitation est un aspect crucial dans les algorithmes à colonies de fourmis (ACO) pour l'ordonnancement. L'exploration consiste à découvrir de nouvelles solutions potentiellement meilleures, tandis que l'exploitation consiste à utiliser les

informations partagées par les fourmis pour améliorer les solutions existantes. Un bon équilibre entre ces deux aspects est essentiel pour trouver des solutions de haute qualité.

Dans le contexte de l'ordonnancement, l'exploration peut être réalisée en permettant aux fourmis d'explorer différentes séquences de tâches et d'affectations de machines. Cela peut être réalisé en utilisant des règles de choix probabilistes qui permettent aux fourmis de sélectionner des tâches et des machines de manière aléatoire ou en utilisant des heuristiques pour guider la recherche vers des régions de l'espace de recherche moins explorées.

D'autre part, l'exploitation peut être réalisée en utilisant les informations partagées par les fourmis pour renforcer les solutions de haute qualité. Cela peut être réalisé en mettant à jour les pistes de phéromones sur les arcs empruntés par les fourmis lors de la construction des meilleures solutions. Les fourmis sont ainsi encouragées à suivre les pistes de phéromones plus fortes, ce qui permet de renforcer les solutions prometteuses et d'exploiter les connaissances collectives des fourmis [17].

Il est important de trouver un bon équilibre entre l'exploration et l'exploitation pour éviter les pièges locaux et trouver des solutions de haute qualité. Si l'exploration est trop dominante, les fourmis peuvent se disperser excessivement et ne pas converger vers des solutions optimales. D'un autre côté, si l'exploitation est trop dominante, les fourmis peuvent être piégées dans des solutions locales et ne pas découvrir de meilleures solutions potentielles.

L'équilibre entre l'exploration et l'exploitation peut être ajusté en utilisant des paramètres tels que le taux d'évaporation des pistes de phéromones, les heuristiques de choix de tâche et de machine, ainsi que les règles de dépôt local et global de phéromones. Ces paramètres peuvent être adaptés empiriquement ou à l'aide de techniques d'optimisation pour trouver le meilleur équilibre pour un problème d'ordonnancement spécifique [21] .

4. Implémentation d’ACO pour des problèmes d'ordonnancement de type job shop

4.1 Considérations de conception pour la mise en œuvre de l'ACO

Lors de la mise en œuvre de l'algorithme ACO (Ant Colony Optimization), il y a plusieurs considérations de conception à prendre en compte pour garantir son bon fonctionnement et son efficacité. Voici quelques-unes de ces considérations :

1) ***Représentation des solutions :*** Il est important de choisir une représentation appropriée pour les solutions du problème d'ordonnancement. Cela peut inclure la représentation des séquences de tâches, des affectations de machines, des temps de début et de fin, ainsi que d'autres informations spécifiques au problème. La représentation choisie doit être adaptée pour permettre aux fourmis de construire et de manipuler efficacement les solutions.
2) ***Construction des solutions :*** L'algorithme ACO repose sur la construction progressive des solutions par les fourmis. Il est donc important de concevoir des règles de construction qui permettent aux fourmis de construire des solutions de manière efficace et de manière à respecter les contraintes du problème. Cela peut inclure des règles de choix de tâche et de machine basées sur des heuristiques, ainsi que des mécanismes pour gérer les contraintes de précédence et de capacité des machines.
3) ***Mise à jour des pistes de phéromones :*** Les pistes de phéromones sont mises à jour par les fourmis pour refléter la qualité des solutions trouvées. Il est important de concevoir des règles de mise à jour appropriées pour garantir que les pistes de phéromones évoluent de manière à guider efficacement la recherche. Cela peut inclure des mécanismes de dépôt local et global de phéromones, ainsi que des stratégies pour ajuster les taux d'évaporation des phéromones.
4) ***Paramètres de l'algorithme :*** L'algorithme ACO comporte plusieurs paramètres qui doivent être ajustés pour obtenir de bons résultats. Cela peut inclure le nombre de fourmis, le taux d'évaporation des phéromones, les heuristiques de choix de tâche et de machine, ainsi que d'autres paramètres spécifiques au problème. Il est important de trouver les valeurs optimales de ces paramètres en utilisant des techniques d'optimisation ou des expérimentations empiriques.
5) ***Critère d'arrêt :*** Il est également important de définir un critère d'arrêt approprié pour l'algorithme ACO. Cela peut être basé sur le nombre d'itérations, le temps d'exécution, la convergence des solutions ou d'autres critères spécifiques au problème. Le critère d'arrêt doit être choisi de manière à permettre à l'algorithme de trouver des solutions de qualité tout en évitant une exécution excessive.

En résumé, la mise en œuvre de l'algorithme ACO pour l'ordonnancement nécessite une attention particulière aux considérations de conception telles que la représentation des solutions, la construction des solutions, la mise à jour des pistes de phéromones, les paramètres de l'algorithme et le critère d'arrêt. En tenant compte de ces considérations, il est possible de concevoir un algorithme ACO efficace et adapté au problème d'ordonnancement spécifique [21].

4.2 Sélection des paramètres spécifiques au problème

La sélection des paramètres spécifiques au problème dans l'algorithme ACO dépend de la nature du problème d'ordonnancement que vous souhaitez résoudre. Voici quelques paramètres couramment utilisés et des conseils pour les sélectionner :

- ***Nombre de fourmis :*** Le nombre de fourmis détermine le niveau d'exploration de l'algorithme. Un nombre plus élevé de fourmis permet généralement une exploration plus large de l'espace de recherche, mais peut également augmenter le temps d'exécution. Il est important de trouver un équilibre entre l'exploration et l'exploitation en ajustant le nombre de fourmis en fonction de la taille du problème et du temps disponible pour l'exécution.

- ***Taux d'évaporation des phéromones :*** Le taux d'évaporation des phéromones contrôle la vitesse à laquelle les pistes de phéromones s'estompent. Un taux d'évaporation plus élevé permet une exploration plus rapide de l'espace de recherche, tandis qu'un taux plus faible favorise l'exploitation des solutions de haute qualité. La sélection du taux d'évaporation dépend de la nature du problème et de la vitesse à laquelle les solutions de haute qualité émergent.
- ***Heuristiques de choix de tâche et de machine :*** Les heuristiques de choix de tâche et de machine guident les fourmis dans la construction des solutions. Ces heuristiques peuvent être basées sur des informations telles que les temps de traitement des tâches, les capacités des machines, les contraintes de précédence, etc. Il est important de choisir des heuristiques qui sont adaptées au problème spécifique et qui favorisent la construction de solutions de haute qualité.
- ***Paramètres de dépôt de phéromones :*** Les paramètres de dépôt de phéromones déterminent la quantité de phéromones déposée par les fourmis lors de la construction des solutions. Ces paramètres peuvent être ajustés pour favoriser l'exploitation des solutions de haute qualité ou pour encourager l'exploration de nouvelles régions de l'espace de recherche. La sélection de ces paramètres dépend de la nature du problème et des objectifs de recherche.
- ***Critère d'arrêt :*** Le critère d'arrêt détermine quand l'algorithme doit s'arrêter. Cela peut être basé sur le nombre d'itérations, le temps d'exécution, la convergence des solutions ou

d'autres critères spécifiques au problème. Le critère d'arrêt doit être choisi de manière à permettre à l'algorithme de trouver des solutions de qualité tout en évitant une exécution excessive.

Il est important de noter que la sélection des paramètres spécifiques au problème peut être un processus itératif. Il est recommandé de commencer par des valeurs par défaut et de les ajuster en fonction des résultats obtenus lors des premières exécutions. L'utilisation de techniques d'optimisation ou d'expérimentations empiriques peut également aider à trouver les meilleures valeurs de paramètres pour votre problème d'ordonnancement spécifique [22].

4.3 Études de cas et exemples concrets de mise en œuvre de l'ACO

Voici quelques exemples concrets de mise en œuvre de l'algorithme ACO dans le domaine de l'ordonnancement :

- ***Ordonnancement de production :*** L'algorithme ACO peut être utilisé pour optimiser l'ordonnancement des tâches dans un environnement de production. Par exemple, dans une usine de fabrication, les tâches peuvent être les différentes étapes de production et les machines peuvent être les ressources disponibles. L'objectif est de minimiser le temps de production total en trouvant la séquence optimale des tâches sur les machines. L'algorithme ACO peut être utilisé pour trouver cette séquence en utilisant les heuristiques appropriées pour guider les fourmis dans la construction des solutions.
- ***Ordonnancement de projet :*** L'algorithme ACO peut également être utilisé pour optimiser l'ordonnancement des activités dans un projet. Les activités peuvent avoir des contraintes de précédence et des dépendances, et les ressources disponibles peuvent être limitées. L'objectif est de minimiser la durée totale du projet en trouvant la séquence optimale des activités et la répartition optimale des ressources. L'algorithme ACO peut être utilisé pour trouver cette séquence en utilisant des heuristiques basées sur les contraintes de précédence et les capacités des ressources.
- ***Ordonnancement de véhicules :*** L'algorithme ACO peut être utilisé pour optimiser l'ordonnancement des véhicules dans des problèmes de logistique et de transport. Par exemple, dans le problème du voyageur de commerce (TSP), l'objectif est de trouver la séquence optimale de villes à visiter pour un voyageur de commerce qui doit passer par toutes les villes une fois et revenir à son point de départ. L'algorithme ACO peut être utilisé pour trouver cette séquence en utilisant les distances entre les villes comme heuristiques pour guider les fourmis dans la construction des solutions.

- ***Ordonnancement des opérations de maintenance :*** L'algorithme ACO peut être utilisé pour optimiser l'ordonnancement des opérations de maintenance dans des systèmes complexes tels que les centrales électriques, les réseaux de télécommunication, etc. L'objectif est de minimiser les temps d'arrêt des systèmes en trouvant la séquence optimale des opérations de maintenance. L'algorithme ACO peut être utilisé pour trouver cette séquence en utilisant des heuristiques basées sur les priorités des opérations de maintenance et les contraintes de disponibilité des ressources.

Ces exemples montrent comment l'algorithme ACO peut être appliqué à différentes problématiques d'ordonnancement. La clé du succès réside dans la sélection des heuristiques appropriées, des paramètres spécifiques au problème et du critère d'arrêt, en fonction de la nature du problème et des objectifs de recherche [23].

5. Évaluation et Analyse de la Performance

5.1 Métriques pour évaluer les performances d'ordonnancement

Lors de l'évaluation des performances d'un algorithme d'ordonnancement, plusieurs métriques peuvent être utilisées pour mesurer l'efficacité et la qualité des solutions obtenues. Voici quelques-unes des métriques couramment utilisées :

1. ***Temps d'exécution :*** Cette métrique mesure le temps nécessaire à l'algorithme pour trouver une solution. Un temps d'exécution court est généralement souhaitable, car il permet de trouver rapidement une solution optimale ou proche de l'optimal.
2. ***Makespan :*** Le makespan est la durée totale nécessaire pour terminer toutes les tâches dans l'ordonnancement. Il s'agit d'une mesure de l'efficacité de l'ordonnancement, où un makespan plus court indique une meilleure performance.
3. ***Tardiness :*** La tardiness mesure le retard accumulé des tâches par rapport à leurs dates d'échéance. Une tardiness plus faible indique un meilleur respect des délais.
4. ***Flowtime :*** Le flowtime est la durée totale nécessaire pour terminer toutes les tâches, y compris les temps d'attente. Il s'agit d'une mesure de l'efficacité de l'ordonnancement, où un flowtime plus court indique une meilleure performance.
5. ***Utilisation des ressources :*** Cette métrique mesure le taux d'utilisation des ressources disponibles dans l'ordonnancement. Une utilisation plus équilibrée et efficace des ressources est généralement souhaitable.

6. ***Énergie consommée :*** Dans certains problèmes d'ordonnancement, notamment dans le contexte de l'ordonnancement de machines parallèles, la consommation d'énergie peut être une métrique importante à considérer. Une consommation d'énergie plus faible indique une meilleure performance.

7. ***Nombre de contraintes violées :*** Cette métrique mesure le nombre de contraintes du problème qui sont violées dans l'ordonnancement. Un nombre plus faible de contraintes violées indique une meilleure performance.

Il est important de choisir les métriques appropriées en fonction du contexte spécifique du problème d'ordonnancement. Certaines métriques peuvent être plus pertinentes que d'autres en fonction des objectifs et des contraintes de votre problème. Il est également possible d'utiliser plusieurs métriques simultanément pour obtenir une évaluation plus complète des performances de l'ordonnancement [24] .

5.2 Analyse comparative de l'ACO avec des algorithmes traditionnels

L'ACO (Ant Colony Optimization) est une métaheuristique basée sur le comportement des fourmis lors de la recherche de nourriture. Comparée aux algorithmes traditionnels, l'ACO présente plusieurs avantages et caractéristiques distinctes :

I. ***Exploration globale :*** L'ACO est capable d'explorer l'espace de recherche de manière globale, ce qui lui permet de trouver des solutions potentiellement meilleures que les algorithmes traditionnels qui se concentrent souvent sur une recherche locale.

II. ***Adaptabilité :*** L'ACO peut s'adapter à des environnements dynamiques et changeants, car il utilise des mécanismes d'apprentissage basés sur les interactions entre les fourmis. Il peut ainsi ajuster ses choix et ses comportements en fonction des informations collectées au fur et à mesure de la recherche.

III. ***Recherche heuristique :*** L'ACO utilise des heuristiques pour guider la recherche vers des solutions potentiellement meilleures. Ces heuristiques sont généralement basées sur des informations spécifiques au problème, telles que les distances, les contraintes de précédence, etc. Cela permet à l'ACO de trouver des solutions de meilleure qualité plus rapidement que les algorithmes traditionnels.

IV. ***Parallélisme naturel :*** L'ACO se prête bien à la parallélisation, car les fourmis peuvent explorer l'espace de recherche de manière indépendante. Cela permet d'accélérer le processus de recherche et d'obtenir des solutions plus rapidement.

V. ***Robustesse :*** L'ACO est généralement robuste aux variations des paramètres et des conditions du problème. Il peut trouver des solutions de qualité même lorsque les conditions sont incertaines ou complexes.

Cependant, il convient de noter que l'ACO peut également présenter certaines limites par rapport aux algorithmes traditionnels :

a. ***Temps d'exécution :*** L'ACO peut nécessiter plus de temps d'exécution que les algorithmes traditionnels, en particulier pour des problèmes de grande taille ou complexes. Cela est dû à la nature stochastique de l'ACO et à la nécessité d'explorer l'espace de recherche de manière globale.
b. ***Sensibilité aux paramètres :*** L'ACO comporte plusieurs paramètres qui doivent être réglés de manière appropriée pour obtenir de bonnes performances. Une mauvaise configuration des paramètres peut entraîner une convergence lente ou des solutions de qualité inférieure.
c. ***Complexité de mise en œuvre :*** La mise en œuvre de l'ACO peut être plus complexe que celle des algorithmes traditionnels en raison de la nécessité de gérer les interactions entre les fourmis, les phéromones, les heuristiques, etc.

En conclusion, l'ACO présente des avantages significatifs par rapport aux algorithmes traditionnels, notamment en termes d'exploration globale, d'adaptabilité et de recherche heuristique. Cependant, il peut également présenter des limitations en termes de temps d'exécution, de sensibilité aux paramètres et de complexité de mise en œuvre. Le choix entre l'ACO et les algorithmes traditionnels dépendra donc du contexte spécifique du problème, des contraintes et des objectifs de recherche [25].

5.3 Analyse de sensibilité et réglage des paramètres pour ACO

L'analyse de sensibilité et le réglage des paramètres sont des étapes importantes dans l'utilisation de l'ACO (Ant Colony Optimization) pour résoudre un problème spécifique. Voici les étapes générales à suivre :

- ***Identification des paramètres :*** Tout d'abord, identifiez les paramètres de l'ACO qui ont un impact sur les performances de l'algorithme. Ces paramètres peuvent inclure le nombre de fourmis, la quantité initiale de phéromones, les taux d'évaporation des phéromones, les heuristiques utilisées, etc.
- ***Définition des plages de valeurs :*** Pour chaque paramètre, définissez une plage de valeurs possibles à tester. Par exemple, vous pouvez définir une plage de 10 à 100 pour le nombre de fourmis, et une plage de 0,1 à 0,9 pour le taux d'évaporation des phéromones.
- ***Conception de l'expérience :*** Concevez une expérience pour évaluer les performances de l'ACO en utilisant différentes combinaisons de paramètres. Vous pouvez utiliser des métriques de performance telles que la qualité de la solution trouvée, le temps d'exécution, etc.
- ***Exécution de l'ACO avec différentes configurations de paramètres :*** Exécutez l'ACO en utilisant différentes combinaisons de paramètres dans la plage de valeurs définie. Collectez les résultats de chaque exécution, y compris les métriques de performance.
- ***Analyse des résultats :*** Analysez les résultats pour identifier les combinaisons de paramètres qui donnent les meilleures performances. Vous pouvez utiliser des techniques d'analyse statistique pour comparer les performances et identifier les tendances.
- ***Réglage des paramètres :*** Sur la base de l'analyse des résultats, sélectionnez les meilleures combinaisons de paramètres et ajustez-les si nécessaire. Par exemple, si vous constatez que des valeurs plus élevées pour le nombre de fourmis conduisent à de meilleures performances, vous pouvez augmenter cette valeur.
- ***Validation croisée :*** Pour valider les performances de l'ACO avec les paramètres réglés, exécutez l'algorithme sur un ensemble de problèmes de test différents. Assurez-vous que les performances sont cohérentes et satisfaisantes pour un large éventail de cas.

L'analyse de sensibilité et le réglage des paramètres peuvent être itératifs, nécessitant plusieurs cycles d'exécution et d'ajustement. Il est également important de noter que les paramètres optimaux peuvent varier en fonction du problème spécifique à résoudre. Par conséquent, cette étape nécessite de la patience et de l'expérimentation pour obtenir les meilleurs résultats possibles avec l'ACO [26].

6. conclusion

Dans ce deuxième chapitre de notre recherche sur les techniques d'optimisation des colonies de fourmis, nous avons exploré en détail cette approche basée sur le comportement des fourmis pour résoudre des problèmes d'optimisation. Nous avons tout d'abord présenté les bases de cette méthode, en expliquant comment les fourmis communiquent entre elles grâce à des phéromones et comment elles peuvent trouver des solutions efficaces à des problèmes complexes.

Ensuite, nous avons examiné les différentes étapes de l'algorithme des colonies de fourmis, à savoir l'initialisation, la construction des solutions, la mise à jour des phéromones et l'évaporation des phéromones. Nous avons souligné l'importance de la balance entre l'exploration et l'exploitation dans la recherche de solutions optimales, ainsi que l'influence des paramètres tels que l'intensité des phéromones ou la probabilité de choisir une solution.

De plus, nous avons discuté des extensions et des améliorations de l'algorithme des colonies de fourmis, telles que l'utilisation de l'élitisme pour favoriser les meilleures solutions, l'introduction de la diversification pour éviter les solutions stagnantes, ou encore l'ajout de mécanismes de recherche locale pour améliorer les solutions déjà construites.

Nous avons également abordé les avantages et les limites de cette approche. En effet, les colonies de fourmis offrent une grande flexibilité et une capacité à trouver des solutions de qualité dans des environnements complexes et dynamiques. Cependant, elles peuvent être sensibles aux paramètres choisis et nécessitent une fine calibration pour obtenir des résultats satisfaisants.

En conclusion, ce deuxième chapitre nous a permis de comprendre les principes fondamentaux des techniques d'optimisation des colonies de fourmis. Nous avons exploré les différentes étapes de l'algorithme, ainsi que les extensions et les améliorations possibles. Dans les prochains chapitres, nous approfondirons ces concepts et présenterons des études de cas pour illustrer l'efficacité de ces techniques dans la résolution de problèmes d'optimisation spécifiques.

Chapitre III :

Implémentation de ACO pour des problèmes d'ordonnancement de type job shop

1.1introduction

La mise au point sur l'application spécifique de cette méthode pour résoudre des problèmes d'ordonnancement complexes. Le job shop est un problème classique dans le domaine de l'ordonnancement, où un ensemble de tâches doit être exécuté sur un ensemble de machines, chacune ayant ses propres contraintes et temps d'exécution. Nous explorerons les différentes étapes de l'implémentation de l'ACO pour résoudre ce type de problème, en mettant l'accent sur la construction des solutions, la mise à jour des phéromones et l'évaluation des performances. Nous discuterons également des avantages et des limites de cette approche pour les problèmes d'ordonnancement de type job shop.

2. Méthodologie

La figure 01 montre la méthodologie proposée que nous utilisions pour résoudre le problème job shop scheduling en utilisant l'algorithme ACO.

Figure III 1 : la méthodologie utilise

Les instances de test utilisées pour les expériences se trouvent dans quatre bases de données. Ces instances proviennent d'une contribution à la [OR-Library par Dirk C. Mattfeld et Rob J.M. Vaessens]. Cette base de données originale contient un ensemble de 82 instances de test Jolb Shop. Les 4 bases de données que nous avons choisies sont décrites dans le tableau 1

Base de Donnée	Nombre d'instance
la01	10x5 instance
la40	15x15 instance
la29	20x10 instance
ft06	6x6 instance

Tableau III 1 : Les 4 bases de données

Description de la base de données :

Les quatre base de données représente des séquences d'opérations pour des problèmes d'ordonnancement de type Job Shop dans l'application "Les colonies de fourmis". Chaque ligne correspond à une séquence de tâches, avec des numéros de job et d'opération associés. Ces données sont essentielles pour optimiser les processus de planification ET de gestion du temps (la01, la40, la29, ft06)

2. Les outils matériels et logiciel utilisez

2.1 Matériels

Nous avons implémenté l'algorithme ACO dans un ordinateur avec une configuration telle que décrite dans le tableau 02

Matérielles	Specifications
CPU	2x Intel Xeon @ 2.20GHz
RAM	8.00 GO
HARD DISK	256 GO

Tableau III 2 : Spécifications matérielles de L'ordinateur

Ence qui concerne les algorithmes et le langage de programmation utilisés,

2.2 Logiciels

La programmation en langage python

Python 3.6 :

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages. Nouveaux modules intégrés: Python 3.6 a introduit plusieurs nouveaux modules intégrés, y compris des secrets pour générer des nombres aléatoires sécurisés, enum pour créer des types énumérés et taper des indices de type.

Syntaxe et sémantique améliorées: Python 3.6 a introduit diverses améliorations de la syntaxe et de la sémantique du langage, telles que l'utilisation de traits de soulignement dans les littéraux numériques pour une meilleure lisibilité, la préservation de l'ordre des dictionnaires et des messages d'erreur plus précis.

Notre algorithme ACO utilisez posse des paramètres spécifiques comme représenté dans le tableau 03 pour s'exécuter sur toutes les bases de données,

parameters	la description
"seed" : 0	Random seed qui permit de répliquer les resultants
"ALPHA" : 1	Poids exponentiel de phéromone sur les probabilities de marche
"BETA" : 1	Poids exponentiel de la désirabilité sur les probabilités de marche
"init_pheromone" : 0.999	Phéromoneinitiale pour tous les bords
"pheromone_constant" :1	Constante qui aide à calculer la contribution des phéromones de bord
"min_pheromone" : 0.001	Valeurminimale de phéromone d'un bord
"evaporation_rate": 0.91	Tauxd'évaporation des phéromones par cycle
"ant_numbers" : 20	Nombre de fourmismarchanten cycle
"cycles" : 20	Nombre de cycles

Tableau III 3 : description des paramètres

Description des paramètres :

Les paramètres spécifiques de cette base de données sont les suivants :

- "seed" : Numéro de graine aléatoire permettant de reproduire les résultats

- "ALPHA" : Poids exponentiel de la phéromone sur les probabilités de marche

- "BETA" : Poids exponentiel de la désirabilité sur les probabilités de marche

- "init_pheromone" : Valeur initiale de phéromone pour tous les bords

- "pheromone_constant" : Constante utilisée pour calculer la contribution des phéromones de bord

- "min_pheromone" : Valeur minimale de phéromone d'un bord

- "evaporation_rate" : Taux d'évaporation des phéromones par cycle

- "ant_numbers" : Nombre de fourmis marchant par cycle

- "cycles" : Nombre de cycles effectués dans l'algorithme

3. Résultats

Notre algorithme ACO est exécuté avec les mêmes paramètres avec 20 fourmis pendant 20 cycles sur les quatre bases de données

Les résultats de chaque base de données sont présentés ci- dessous Les résultats des instances de test la01

Présentation de résultats en tableau : les résultats des instances de test la 01

la base de données	cycles	Les résultats		
		le coût total	la durée moyenne	valeur de référence
La 01	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	666 666 666 666 666 666 666 666 666 666 666 666 666 666 666 666 666 666 666 666	738.7 739.05 739.05 745.75 742.35 750.7 748.8 760.6 764.15 764.15 761.75 756.85 755.25 755.65 759.55 758.3 758.9 755.05 758.9 755.05	860 860 860 860 860 860 860 898 898 898 898 898 898 860 860 860 860 860 860 860

Tableau III 4 : le résultat de la base de données la01

Description des résultats :

Les résultats obtenus pour la base de données "la01" montrent que l'algorithme ACO a été exécuté avec succès pendant 20 cycles, utilisant 20 fourmis. Les valeurs affichées représentent les résultats obtenus pour chaque cycle. Chaque résultat est un tableau comprenant trois valeurs : le premier nombre représente le coût total de la solution trouvée, le deuxième nombre est la durée moyenne de la solution, et le troisième nombre est une valeur de référence pour la solution. Les résultats varient d'un cycle à l'autre, indiquant que l'algorithme explore différentes solutions pour trouver la meilleure. Les valeurs obtenues montrent une certaine stabilité dans les performances de l'algorithme, avec des coûts et des durées moyennes qui restent relativement constants.

4. conclusion

En conclusion, ce troisième chapitre nous a permis de comprendre l'importance de la simulation dans la résolution de problèmes complexes. Nous avons exploré les différentes étapes de la modélisation et de la simulation, ainsi que les avantages et les limites de cette approche.

Conclusion Générale

Conclusion Générale

Les colonies de fourmis ont démontré leur efficacité dans la résolution de problèmes d'ordonnancement de type job shop. Grâce à leur capacité à communiquer et à coopérer, les fourmis sont capables de trouver des solutions optimales en termes de temps et de ressources.

L'utilisation de phéromones pour marquer les chemins les plus courts et les plus efficaces a permis de guider les fourmis vers des solutions optimales. De plus, l'ajout de règles de décision basées sur la qualité des solutions trouvées a permis d'améliorer encore davantage les résultats.

Les colonies de fourmis ont également montré leur adaptabilité et leur robustesse face aux changements de l'environnement. Elles sont capables de s'ajuster rapidement aux nouvelles contraintes et de trouver de nouvelles solutions en conséquence.

Cependant, malgré leurs avantages, les colonies de fourmis ne sont pas exemptes de limites. La complexité des problèmes d'ordonnancement de type job shop peut rendre la recherche de solutions optimales difficile et nécessiter des ressources supplémentaires. De plus, la mise en place d'un système de colonies de fourmis peut être complexe et nécessiter une expertise technique.

En conclusion, les colonies de fourmis sont une approche prometteuse pour résoudre les problèmes d'ordonnancement de type job shop. Leur capacité à trouver des solutions optimales, leur adaptabilité et leur robustesse en font une méthode intéressante à explorer pour améliorer l'efficacité des processus d'ordonnancement.

Références

[1] Andrea LUONG Thé Van MARILL Guillaume .Optimisation par colonies de fourmis COSTANZO . 19mai2006.

[2] Melle Bendada Meriem , Mme Addou Narimane Batoulle. Gestion des Emplois du Temps ,Mémoire de fin d'études, Approche Graphique Réalisé .23 Juin 2016.

[3] Chapitre3 Ordonnancement sous contraintes, consulté le 15 février 2023.

[4,9] GOTHA. Les problèmes d'ordonnancement Revue française d'automatique, d'informatique et de recherche opérationnelle.Rechercheopérationnelle,tome27,no1(1993),

[5] Chapitre4 Ordonnancement d'atelier, consulté le 10 février 2023.

[6] Jean-Pierre Fontaine. L'ordonnancement des opérations est stratégique, ,Fév 14, 2019

[7] Julien Fondrevelle . l'institut national Résolution exacte de problèmes d'ordonnancement de type flow shop de permutation en présence de contraintes d'écarts temporels entre opérations Thèse doct-ora ,21 Nov 2005

[8] Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables ,l'université Tlemcen , Thèse de docteur, le 12 décembre 2018.

[9,4] GOTHA. Les problèmes d'ordonnancement Revue française d'automatique, d'informatique et de recherche opérationnelle.Rechercheopérationnelle, tome27, no1(1993), p.77-150.

[10] L'essentiel à comprendre de l'ordonnancement informatique, Publié le 18 mai 2021 à 12 h 35 min - Mis à jour le 18 mai 2021.

[11] Billaut Jean-Charles. Résolution par des algorithmes exacts et approchés de problèmes intégrés d'ordonnancement de la production et de tournées de véhicules. thèse de Docteura. le 8 décembre 2020,

[12] Méthodes exactes en optimisation combinatoire, , consulté le 10 mars 2023 .

[13] DanielCosminPorumbel.AlgorithmesHeuristiquesetTechniquesd'Apprentissage-Applicationsau Problème de Coloration de Graphe. Informatique Université d'Angers ,2009 Français

[14] logistique conseil, Articles, Gestion-production,Regles-priorite 2013.

[15] M. Sakarovitch,« Graphes et Programmation Linéaire, Edition Hermann, Paris, 1984.

[16] Y. Bahmani, Optimisation multicritère de l'ordonnancement des activités de la production et de la maintenance intégrées dans un atelier Job Shop, Batna, Thèse de Doctorat, 2017.

[15] faculté des sciences de la nature et de la vie · faculté snv · Publication des Enseignants · Bibliothèque · Emplois du temps Biologie Appliquée ·

[16] GOTHA Les problèmes d'ordonnancement, tome27,no1(1993).

[17] Jimmy Wales et Larry Sanger le 15 janvier 2001, Wikipédia Site web

[18] Optimisation par colonies de fourmis COSTANZO Andrea LUONG Thé Van MARILL Guillaume

[19] «Algorithmes, Modèles et Principes de base d'ordonnancement de la production», Dr SOUIER Mehdi, École supérieure en sciences appliquées –Tlemcen- Cours et exercices

[20] *articles* Les algorithmes de colonies de fourmis *labellisées en 2007.*

[21] Optimisation par colonies de fourmis, COSTANZO Andrea, LUONG Thé Van, MARILL Guillaume, 19mai2006.

[22] Fatima Benbouzid Sitayeb, Christophe Varnier, Noureddine Zerhouni. Résolution du problème de l'ordonnancement conjoint production/maintenance par colonies de fourmis..6èmeConférence Francophone de Modélisation et SIMulation,MOSIM'06.Modélisation, Optimisation et Simulation des Systèmes:défisetopportunités.,Apr2006,Rabat,Maroc.10p.hal-00327506

[23] L'université de Constantine, Ordonnancement D'atelier , Chapitre III : ALGORITHMES D'ORDONNANCEMENT,

[24] Ordonnancement sous contraintes, Chapitre3, 2013

[25] Ilhem Boussaid. Perfectionnement de métaheuristiques pour l'optimisation continue. Autre .Université Paris-Est ;Université des sciences et de la technologie Houari Boumediene(Alger;1974-..),2013. Français.NNT:2013PEST1075.tel-00952774

[26] Madjid Khichane, Patrick Albert, Christine Solnon .Un modèle réactif pour l'optimisation par colonies de fourmis: application à la satisfaction de contraintes .Cinquièmes Journées Francophones de Programmation par Contraintes,Orléans,juin2009,Jun2009,France.pp.195-205.hal-00387847

[27] Jacques Kerneis ,Michaël Huchette. Comment les représentations graphiques parlent-elles du monde industriel? Les manuels de technologie vus comme des espaces de communication .2016.hal01143557

[18] N. Mouhoub, Algorithmes de construction de graphes dans les problèmes d'ordonnancement de projet, Sétif, Thèse de Doctorat, 2011


the end