REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTE DE MATHEMATIQUES ET D 'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'études

Présenté pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique.
Filière : Informatique.
Spécialité : Informatique Décisionnelle et Optimisation.

Par: NOUIS Rayhana Houda
BENZID Sabrina

THEME

Problème d'ordonnancement stochastique

Soutenu publiquement le : 12 / 06 / 2024 devant le jury composé de :

Pr. BOURAHLA Mustapha	**Université MB de M'sila**	**Président**
Pr. HEMMAK Allaoua	**Université MB de M'sila**	**Rapporteur**
Dr. THARAFI Abdellah	**Université MB de M'sila**	**Examinateur**

Promotion : 2023/2024

إهداء

قالتعالى : ﴿ قُلْ بِفَضلِ اللَّهِ وَ بِرَحْمَتِهِ فَبِذَلِكَ فَليفْرَحُوا ﴾

الحمد لله الذي ما سلكنا البدايات إلا بتوفيقه و ما حققنا الغايات إلا بفضله
الحمد لله حبًا و شكرًا و امتنانًا على البدء و حُسن الختام

أُهدي هذا النجاح لنفسي أولاً جزاء الصبر و العزيمة طيلة هذه السنوات ثم إلى كل من سعى معي لإتمام هذه المسيرة

إلى من قال الله فيهم .. ﴿وَ قَضَى رَبُّكَ أَلَّا تَعبُدُوا إِلَّا إِيَّاه وَ بِالْوَالِدَيْنِ إِحْسَانًا ﴾
والديّ ضياء طريقي و أساس حياتي فخري و داعمي الأول هذا نجاحكم و نتيجة جُهدكم ، أسأل الله أن يجعلني أسعى دائما لأكون فخرًا لكما
*و سببًا في فرحتكما و أنال رضاكما .. **أمي و أبي***

*إلى معلمتي ضماد الروح التي سادت محبتها في أرجاء أضلعي و يظل تحفيزها و وجودها دومًا داعمًا لي .. **أميرة منصور***

إلى من قال الله فيهم .. ﴿ سَنَشدُّ عَضُدَكَ بِأَخِيك ﴾
*رفقاء الخطوة الأولى الذين دعموني بصدق و محبة في كل مراحل دراستي .. **إخوتي و أخواتي***

*إلى رفيقتي و سندي في كل اللحظات حفظها الله و أسعدها ..**زوجة أخي***

*إلى من أنعم الله علي بصحبتهم و آمنوا بقدراتي و شاركوني لحظات الفرح و التحدي و بهم تكتمل لذة الوصول .. **رفيقات القرآن***

*إلى كل من كان له الفضل في تعليمي منذ بداية مسيرتي الدراسية إلى النهاية و لم يبخلوا علي بعلمهم و إرشاداتهم .. **أساتذتي الكرام***

أهديكم هذا الإنجاز المتواضع راجية من المولى أن ينفعني بما علمني و أن يكون هذا النجاح خطوة أولى في طريق العلم و المعرفة

فالحمد لله أولًا و آخرًا على ما وهبني

﴿ وَ آخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ﴾

ريحانة

إهداء

الحمد لله حبا و كراما و امتنانا على البدء والختام

أهدي بكل حب بحث تخرجي إلى نفسي العظيمة القوية التي تحملت كل العثرات، وتحملت مشقة الطريق، فشكراً لنفسي على الصبر والمثابرة.

إلى الذي زين اسمي بالألقاب، من دعمني بلا حدود وأعطاني بلا مقابل، إلى من علمني أن الدنيا كفاح وسلاحها العلم والمعرفة، داعمي الأول في مسيرتي وسندي، والدي العزيز، أدامك الله ظلا لنا.

إلى التي تعجز كل الكلمات عن وصفها، إلى التي كانت النور في عتمتي، إلى التي كان دعاؤها سر نجاحي، إلى أمي الحبيبة، متعها الله بالصحة والعافية، وجعلها دائماً سعيدة وفخورة بي.

إلى من عرفت معهم معنى الحياة، إخوتي محمد ومروة، وإلى أولاد أختي أمير ولؤي، الذين كانوا دائماً بجانبي ودعماً لي.

وإلى صديقاتي العزيزات، اللواتي كنّ لي خير سند ودعم في كل لحظة، أشكر كل واحدة منكن على وجودكن في حياتي.

شكرا لكل من ساندني وساعدني طوال رحلتي الدراسية، سواء بكلمة طيبة، أو نصيحة صادقة، أو دعم معنوي.

وفي الختام، أسأل الله التوفيق والسداد في كل خطوة أخطوها في المستقبل، وأن يجعل هذا البحث العلمي نافعاً ومفيداً للعلم.

وآخر دعوانا أن الحمد لله رب العالمين

صبرينة

شكر و عرفان

بسم الله الرحمن الرحيم

الحمد و الشكر لله رب العالمين الذي وفقنا وأعاننا على إتمام هذا البحث

يقول النبي ﷺ : ﴿ مَنْ لَا يَشكُرُ النَّاس لَا يَشْكُرُ الله ﴾

أولا نتقدم بخالص الشكر والعرفان إلى أستاذنا الفاضل و مشرفنا على هذا البحث،
*الدكتور " **هماك علاوة** " ، الذي لم يبخل علينا بوقته وجهده، وقدم لنا الدعم*
والتوجيه والإرشاد طوال فترة إعداد هذا العمل بكل إخلاص و تفاني.

" لقد كانت ملاحظاتك القيمة وتوجيهاتك السديدة هي النبراس الذي اهتدينا به في كل
خطوة من خطوات هذا البحث ..
فشكرًا لكَ على صبركَ وتحفيزكَ المستمر نسأل الله أن ينفع بكَ و بعلمك، بُوركت
جهودكَ و جزاكَ الله عنا كل خير "

و نتوجه بوافر الشكر والتقدير إلى أعضاء لجنة المناقشة
*الأستاذ " **بورحلة مصطفى** " و الأستاذ " **طرافي عبد الله** "*
لقبولهم مناقشة هذا العمل المتواضع و تقديم نصائحهم القيمة و إرشاداتهم الهادفة.

و الشكر موصول لكل من ساهم بمجهوداته و لم يتردد في مد يد العون لمساعدتنا في
إنجاز هذا البحث.

كما نود أن نعبر عن امتناننا لجميع أساتذتنا الكرام الذين كان لهم الفضل علينا في تعلمينا
منذ بداية مسيرتنا الدراسية إلى اليوم.

وفي النهاية، لا يسعنا إلا أن نشكر كل من كان له دور في دعمنا لوصولنا لهذا اليوم ،
سواءً كان دعمًا ماديًا أو معنويًا ، و نسأل الله أن يجزيهم عنا خير الجزاء.

Table des matières

Listes des figures

Liste des tableaux

Introduction Générale

Dans le monde complexe et rapide d'aujourd'hui, la planification efficace des projets et des tâches est devenue un défi majeur dans de nombreux domaines tels que la fabrication, la santé, les transports et les technologies de l'information. L'ordonnancement est le processus d'allocation des ressources disponibles (telles que les machines, les travailleurs, le temps) à diverses tâches de manière à garantir que les objectifs fixés sont atteints aussi efficacement que possible. Cependant, la planification est confrontée à des défis importants lorsqu'il existe des facteurs aléatoires ou incertains qui affectent les délais de traitement ou la disponibilité des ressources. D'où l'importance d'étudier les problèmes d'ordonnancement stochastique.

L'ordonnancement stochastique représente une évolution importante dans le domaine de la planification, permettant aux organisations de relever les défis opérationnels de manière plus flexible et plus efficace. Parvenir à une planification optimale dans des conditions incertaines nécessite de l'innovation et l'utilisation d'outils et de modèles avancés pour analyser les données et s'adapter aux changements. Ce domaine ouvre de larges horizons pour la recherche et le développement, ce qui en fait un sujet passionnant et attrayant pour les chercheurs et ceux qui souhaitent améliorer l'efficacité dans divers secteurs.

La relation entre L'ordonnancement traditionnelle et L'ordonnancement stochastique est que cette dernière est une extension et une intégration de la première. Alors que l'ordonnancement traditionnelle part du principe de certitude, la l'ordonnancement stochastique ajoute une couche supplémentaire de réalisme et de flexibilité en incluant l'incertitude. Cela permet aux systèmes de s'adapter aux conditions changeantes, augmentant ainsi l'efficacité et la faisabilité des solutions proposées.

Le problème de planification stochastique est le problème de déterminer le calendrier optimal pour un ensemble de tâches nécessitant des ressources limitées, où certains paramètres tels que les temps de traitement ou la disponibilité des ressources sont incertains et suivent certaines distributions de probabilité.

Ce problème présente plusieurs complications : les temps de traitement réels des tâches peuvent différer des temps estimés en raison de facteurs aléatoires. Les ressources nécessaires

(telles que les machines ou les travailleurs) peuvent parfois être indisponibles en raison de pannes ou de changements inattendus dans leshoraires de travail. Il y a souvent plusieurs objectifs à atteindre, comme minimiser le temps d'exécution global (Makespan), minimiser la latence ou optimiser l'utilisation des ressources.

L'amélioration de la planification des tâches en situation aléatoire est un sujet crucial pour plusieurs motivations:Innovation dans les solutions La recherche sur l'ordonnancement stochastique est donc un domaine de recherche actif, offrant des opportunités pour le développement de nouveaux algorithmes et d'outils analytiques avancés. Également dans le but d'améliorer la flexibilité et d'atteindre la plus grande efficacité possible dans l'utilisation des ressources, ce qui conduit à une réduction des coûts et à une augmentation de la productivité. L'étude des problèmes aléatoires aide également à comprendre les risques associés à la planification et à développer des stratégies pour réduire leur impact.

Afin d'atteindre ces objectifs, nous avons proposé une structure de ce document composée de quatre chapitres.

Le premier chapitre passe en revue d'ordonnancement Stochastique en termes de définition, de formulation et d'éléments, et se termine par une discussion des différentes méthodes permettant de résoudre ces problèmes. Le deuxième chapitre est dédié à l'état de l'art du sujet de cette recherche.Dans le troisième chapitre, nous présenterons les concepts et modèles mathématiques de base dans le domaine de l'ordonnancement aléatoire et les méthodes utilisées en algorithmique. Quant au quatrième chapitre est consacré les étapes et résultats de l'implémentation.

CHAPITRE I

ÉLÉMENTSD'ORDONNANCEMENT STOCHASTIQUE

1. Introduction

Le problème d'ordonnancement Stochastique joue un rôle fondamental dans l'amélioration de l'efficacité des opérations et de la gestion des ressources dans des environnements caractérisés par l'incertitude. Ce chapitre couvre les aspects fondamentaux de la planification, depuis l'histoire de son développement et ses termes clés, jusqu'aux l'application pratique et différente types de problèmes de planification. Il passe également en revue la planification aléatoire en termes de définition et de formulation, et se termine par une discussion des différentes méthodes permettant de résoudre ces problèmes.

2. Historique

L'histoire de l'ordonnancement stochastique remonte aux débuts de la rechercheopérationnelle au XXe siècle. L'ordonnancement traditionnel se concentrait sur des modèles déterministes, où les variables étaient considérées comme fixes et prévisibles. Cependant, avec l'essor des techniques probabilistes et stochastiques, l'attention s'est tournée vers la modélisation des incertitudes et des variations aléatoires présentes dans de nombreux environnements réels.

Dans les années 1950 et 1960, des chercheurs comme Herbert Scarf ont commencé à explorer les modèles d'ordonnancement stochastique, reconnaissant que de nombreuses situations de planification et de production étaient soumises à des facteurs aléatoires tels que les temps de traitement variables, les pannes d'équipement, ou les fluctuations de la demande. Ces facteurs imprévisibles ont conduit à une évolution des modèles d'ordonnancement vers des approches probabilistes, où les décisions sont prises en tenant compte des incertitudes.

Depuis lors, l'ordonnancement stochastique a continué à se développer avec les progrès des méthodes mathématiques et informatiques. Des techniques telles que la simulation Monte Carlo, les processus stochastiques et les modèles de files d'attente ont été intégrées pour prendre en compte les aléas dans la planification des activités.

Aujourd'hui, l'ordonnancement stochastique est largement utilisé dans de nombreux domaines, notamment la production, la logistique, les services et la gestion de projet, où l'incertitude joue un rôle crucial. Il permet aux gestionnaires d'optimiser les décisions en tenant compte des risques et des variations, contribuant ainsi à une meilleure efficacité opérationnelle et à une gestion plus robuste des ressources.

3. L'ordonnancement

" Ordonnancer c'estprogrammer l'exécution d'une réalisation en attribuant des ressources auxtâches et en fixant leurs dates d'exécution."Donc l'ordonnancement constitue une solution au problème d'ordonnancement. Il décritl'exécution des tâches et l'allocation des ressources au cours du temps et vise à satisfaire un ouplusieurs objectifs.

La théorie de l'ordonnancement joue un rôle essentiel dans de nombreux secteurs d'activités : laconception (de bâtiments, de produits, de systèmes, . . .), l'administration (gestion d'emplois dutemps, gestion du personnel), l'industrie (gestion de la production), l'informatique (ordonnancementde processus, ordonnancement de réseaux).

On parle d'ordonnancement lorsqu'on fixe les dates de début ou de fin de chacune des tâches[17].

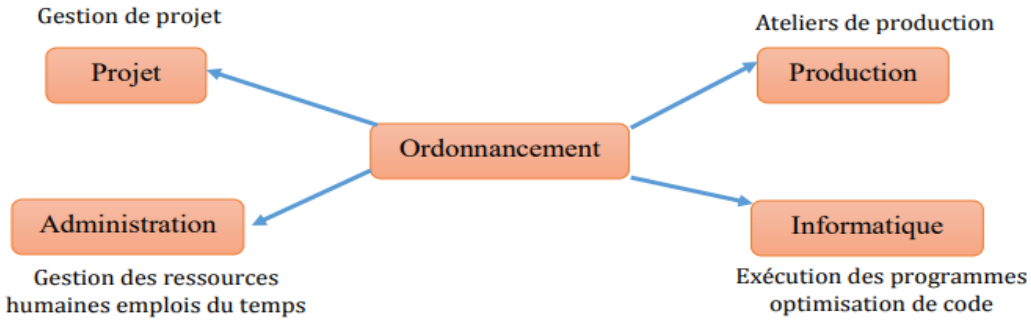

Figure 1- 1:Les domaines concernés par les problèmes d'ordonnancement [8].

4. Terminologied'ordonnancement

4.1.Tâches

Une tâche ; est une entité élémentaire de travail localisée dans le temps par unedate de début$\boldsymbol{t_i}$ou de fin $\boldsymbol{C_i}$, dont la réalisation nécessite une durée $\boldsymbol{p_i = C_i - t_i}$, et quiconsomme des moyens $\boldsymbol{k}$ avec une intensité $\boldsymbol{a_{ik}}$.

Selon les problèmes, les tâches peuvent être exécutées par morceaux, ou doivent êtreexécutées sans interruption ; on parle alors respectivement de problèmes préemptifs etnon préemptifs.

Lorsque les tâches ne sont pas liées entre elles par des contraintes de cohérence technologique (par exemple contraintes d'enchaînement liées aux procédés de réalisation),elles sont dites indépendantes [10].

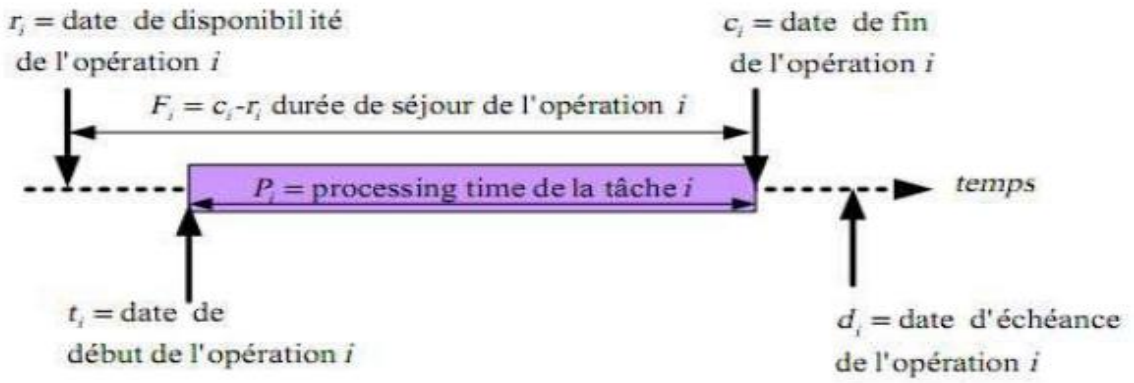

Figure 1- 2: Caractéristiques d'une tâche [8].

4.2.Lesresources

Une ressource $\boldsymbol{k}$ est un moyen technique ou humain destiné à être utilisé pour laréalisation d'une tâche, et disponible en quantité limitée, sa capacité $\boldsymbol{A_k}$. Plusieurs types de ressource sont à distinguer. Une ressource est renouvelable si, après avoir été utilisée par une ou plusieurs tâches, elle est à nouveau disponible en même quantité (les hommes, les machine, l'espace, l'équipement en général,…) ; la quantité de ressource utilisable à chaque instant est limitée. Dans le cas contraire, elle estconsommable (matière première, budget, …) ; la consommation globale (ou cumul) aucours du temps est limitée.

Qu'elle soit renouvelable ou consommable, la disponibilité d'une ressource peut àpriori varier au cours du temps. On distingue par ailleurs - principalement dans le cas de ressources renouvelables- les ressources disjonctives (on non partageables) qui ne peuvent exécuter qu'une tâche à la fois (machine-outil, robot manipulateur) et les ressources cumulatives (ou partageables) qui peuvent être utilisées par plusieurs tâches simultanément (équipe d'ouvriers, poste de travail). Dans notre travail, on s'intéressera au cas des ressources renouvelables disjonctives représenté par les machines [10]**.**

4.3.Les contraintes

Les contraintes expriment les restrictions que peuvent prendre conjointement les variables de décision. Donc les contraintes nous renseignent sur les limites imposées par l'environnement. On distingue plusieurs types de contraintes, qui sont [3] :

4.3.1. Les contraintes temporelles :

Pour la réussite d'un projet ou d'un plan de production, on doit se soumettre aux impératifs de production (réalisation) traduits par le respect d'un planning fixé avant. Au niveau global on parlera d'une date de lancement d'une tâche et d'une date de livraison. *A*un autre niveau de

détail plus fin, pour une tâche ou une opération. Ce dernier niveau se traduit par la définition des dates suivante:

_ Date de disponibilité : $\boldsymbol{r_i}$;

_ Date d'échéance : $\boldsymbol{d_i}$;

4.3.2. Les contraintes de précédence:

Une contrainte qui lie le début d'une activité à la fin d'un autre est appelée contrainte de succession ou de précédence. Les gammes opératoires sont un exemple de ces contraintes, d'autres contraintes sont imposées dans certain cas telles que les contraintes de synchronisation, de simultanéité, ou de recouvrements etc.

4.3.3. Les contraintes de ressources:

Les contraintes de ressources représentent le fait que les activités requièrent tout au long de leur exécution une certaine quantité de ressources.

- Les contraintes de ressources concernent les deux points suivants:
- L'utilisation de ces ressources, (leur nature, la quantité nécessaire, et les caractéristiques d'utilisation);
- La disponibilité et la quantité de ces ressources.

4.4.Les critères

Un critère correspond à des exigences qualitatives et quantitatives à satisfaire permettant d'évaluer la qualité de l'ordonnancement établi. Les critères dépendant d'une application donnée sont très nombreux; plusieurs critères peuvent être retenus pour une même application. Le choix de la solution la plus satisfaisante dépend du ou des critères préalablement définis, pouvant être classés suivant deux types, réguliers et irréguliers.

Les différents critères ne sont pas indépendants; certains même sont équivalents. Deux critères sont équivalents si une solution optimale pour l'un est aussi optimale pour l'autre et inversement [3]:

4.4.1. Les critères réguliers :

Constituent des fonctions décroissantes des dates d'achèvement des opérations. Quelquesexemplessontcités ci-dessous:

-La minimisation des dates d'achèvement des actions.

-La minimisation du maximum des dates d'achèvement des actions.

-La minimisation de la moyenne des dates d'achèvement des actions.

-La minimisation des retards sur les dates d'achèvement des actions.

-La minimisation du maximum des retards sur les dates d'achèvement des actions.

4.4.2. Les critères irréguliers:

Sont des critères non réguliers, c'est-à-dire qui ne sont pas desfonctions monotones des dates de fin d'exécution des opérations, tels que:

-La minimisation des encours.

-La minimisation du coût de stockage des matières premières.

-L'équilibrage des charges des machines.

-L'optimisation des changements d'outils.

La satisfaction de tous les critères à la fois est souvent délicate, car elle conduitsouvent à des situations contradictoireset à la recherche de solutions à des problèmes complexes d'optimisation.

5. Applications

Le problème d'ordonnancement, également connu sous le nom de problème d'emploi du temps ou de problème de planification, se retrouve dans de nombreux domaines différents. Voici quelques exemples d'application du problème d'ordonnancement dans divers domaines.

- **Industrie manufacturière :** Dans les usines, il est essentiel de planifier efficacement l'ordre des opérations pour optimiser la production, minimiser les temps d'attente et maximiser l'utilisation des ressources.
- **Logistique et transport:**Les entreprises de logistique doivent planifier les itinéraires de livraison, l'affectation des véhicules et le chargement des marchandises de manière à minimiser les coûts et à respecter les délais.
- **Informatique et planification des tâches**: Dans les centres de données, les algorithmes d'ordonnancement sont utilisés pour gérer l'exécution des tâches sur les serveurs afin d'optimiser les performances et de garantir une utilisation équilibrée des ressources.
- **Secteur médical:** Les hôpitaux doivent planifier les horaires des médecins et du personnel infirmier, ainsi que l'affectation des ressources telles que les salles d'opération et les équipements médicaux, pour assurer une prise en charge efficace des patients.
- **Projets de construction:**Les entreprises de construction doivent planifier les différentes étapes d'un projet, l'allocation des ressources humaines et matérielles, ainsi que la coordination des sous-traitants pour respecter les délais et les budgets.
- **Gestion des emplois du temps**: Dans les institutions éducatives, les entreprises et d'autres organisations, la planification des horaires des employés, des cours, des réunions et des événements nécessite souvent une résolution efficace du problème d'ordonnancement.

•**Fabrication de circuits intégrés :** L'industrie des semi-conducteurs utilise des algorithmes d'ordonnancement pour optimiser le processus de fabrication des puces électroniques, en minimisant les temps de production et en maximisant le rendement.

•**Planification de la production d'énergie:** Les entreprises énergétiques doivent planifier la production d'électricité en fonction de la demande prévue, en tenant compte des contraintes de stockage d'énergie et des capacités des centrales électriques pour garantir un approvisionnement stable et efficace.

•**Planification de la chaîne d'approvisionnement :** Les entreprises doivent organiser la production, le stockage et la distribution des produits de manière à minimiser les coûts tout en répondant à la demande des clients dans les délais impartis.

•**Systèmes de réservation et de planification en ligne** : Les plateformes de réservation de voyages, d'hébergement, de rendez-vous médicaux, etc., utilisent des algorithmes d'ordonnancement pour gérer les demandes et les ressources de manière efficace. Ces exemples illustrent la diversité des applications du problème d'ordonnancement dans de nombreux domaines, où il est utilisé pour optimiser les opérations, maximiser l'utilisation des ressources et garantir la satisfaction des clients ou des utilisateurs.

6. Les problèmes d'ordonnancement

Graham (1979) a présenté une le schéma de classification à trois champs $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$[3] :

•$\underline{\boldsymbol{\alpha}}$:**Environnement machine.**$\alpha = 1$ ou $\alpha = \alpha 1 \alpha 2$

•$\alpha 1$: concerne les caractéristiques spécifiques de l'environnement de la machine liées au problème d'ordonnancement, peut prendre différentes valeurs pour refléter les différentes contraintes ou caractéristiques de l'environnement.

Par exemple, dans un environnement à une seule machine, α1 pourrait indiquer des contraintes de ressources limitées, tandis que dans un environnement multi-machines, cela pourrait indiquer des contraintes de synchronisation entre les machines.

•$\alpha 2$:Cela désigne le nombre de machines.

Si α2 est une valeur entière positive, cela signifie que le nombre de machines est supposé constant pour le problème donné.

Si α2 est absent, cela signifie que le nombre de machines est supposé être arbitraire ou non spécifié dans la formulation du problème.

•$\underline{\boldsymbol{\beta}}$:**Les caractéristiques des tâches et les contraintes du problème**

•$\beta 1 = \text{pmtn}$ Si la préemption des tâches est autorisée, sinon $\beta 1$ est absent.

S'il y a des contraintes de précédence entre les tâches $\beta 2$ {prec, chain, tree}, sinon$\beta 2$ est vide.

•$\beta 3 = r_i$ Si les dates de début au plus tôt r_i(ou dates de disponibilité) des tâches ne sont pas forcément identiques, sinon $(j, ri = 0)\beta 3$ est absent.

•$\beta 4$ = jSi la tâche ou le job possède une date de fin obligatoire.[3]

•$\underline{\gamma}$:**Le (ou les) critère(s) à optimiser.** [3]

7. Les types d'ordonnancement

•**Ordonnancement statique et Ordonnancement dynamique :** Si l'ensemble des informations nécessaires à la résolution d'un ordonnancement est fixé au départ (ensembles des tâches, des contraintes, des ressources, etc.), on est alors devant un problème d'ordonnancement statique. Il y a une nuance entre la solution proposée qui est, généralement accompagnée d'un plan prévisionnel d'exécution des tâches, et l'exécution réelle de ces tâches. Si le plan n'est pas respecté et les objectifs sont modifiés, on est devant un problème d'ordonnancement dynamique qui nécessite une résolution d'une série de problèmes statiques et chaque étape doit débuter par une prise d'informations permettant d'actualiser le modèle à résoudre.

•**Ordonnancement admissible :** Un ordonnancement est dit admissible s'il respecte toutes les contraintes du problème (dates de début, dates de fin, précédence, contraintes de ressources, etc.)

•**Ordonnancement actif et Ordonnancement semi-actif** : Dans un ordonnancement actif, aucune tâche ne peut commencer au plus tôt et qui entraîne l'ordre relatif entre au moins deux tâches. Dans l'ordonnancement semi actif, on ne peut pas avancer une tâche sans modifier la séquence sur la ressource.

•**Ordonnancements sans retard :** Dans un ordonnancement sans retard, on doit exécuter la tâche qui est en attente à condition que la ressource soit disponible.

•**Ordonnancement préemptif et non préemptif :** Selon les problèmes, les tâches peuvent être exécutées sans interruption, c'est-à-dire si on commence l'exécution d'une tâche elle n'est pas interrompu jusqu'à son achèvement. On parle alors d'un ordonnancement préemptif. Par contre, si les tâches sont exécutées par morceaux, l'ordonnancement est appelé non préemptif[17].

8. L'ordonnancement Stochastique

L'ordonnancement stochastique permet de planifier les tâches d'un projet avec des durées incertaines, afin de minimiser la durée du projet attendu sans contrainte de ressources renouvelables et avec des relations de précédence. La méthode consiste en une génération aléatoire de tous les scénarios possibles en fonction des distributions de probabilités des durées des tâches. Afin de faire le tri dans l'ensemble des scénarios générés, il faut mettre en place des stratégies d'ordonnancement.

L'objectif général consiste à créer une politique d'ordonnancement qui minimise la durée attendue du projet. Fernandez (1995), Fernandez et al. (1996, 1998)et Pet-Edwards et al. (1998) ont modélisé le problème d'optimisation correspondant sous la forme d'une programmation stochastique multi étape.

Radermacher (1985) a défini les «earlystartpolicies » (politique du début au plus tôt). Ce concept minimise les solutions interdites, c'est à dire l'exécution de deux tâches sans relation de précédence simultanément alors qu'elle est utilisent les mêmes ressources. Igelmund and Radermacher (l 983a,b), ont introduit les politiques pré-sélectives (PRS).

Le principe est que pom chaque solution interdite il faut faire en sorte que l'une des activités commence quand l'autre est terminée.

Mohring et Stork (2000) ont introduit une représentation de présélection appelée «waiting conditions» (conditions d'attente). Ces conditions peuvent être modélisées par des conditions de précédences formulées avec des fonctions booléennes OU et ET. Il faut savoir que les politiques pré-sélectives sont limitées par la capacité de calcul des ordinateurs.

Ainsi Mohring and Stork (2000) ont défini les politiques de pré-sélections linéaires (LIN). Cette méthode vise à planifier dans un premier temps les ensembles d'ordonnancement qui respectent les conditions de précédence originelles. Au lieu d'utiliser des politiques d'ordonnancement pour trier les différents scénarios générés aléatoirement, on peut faire appel à des méthodes heuristiques. (Pet-Edwards, 1996; Golenko-Ginzburg and Gonik, 1997; Tsa i and Genm1il, 1998).Ces techniques sont encore émergentes [14].

L'ordonnancement stochastique est une approche de planification et d'ordonnancement dans laquelle les décisions sont prises en tenant compte de l'incertitude associée aux paramètres du système. Contrairement à l'ordonnancement déterministe, où les paramètres sont considérés comme fixes et connus, l'ordonnancement stochastique prend en compte la variabilité et l'incertitude qui peuvent survenir dans un système.

Dans un contexte d'ordonnancement stochastique, les variables telles que les temps de traitement des tâches, les délais de livraison, ou d'autres facteurs pertinents peuvent être modélisés comme des variables aléatoires plutôt que des valeurs déterministes. Cela permet de prendre des décisions qui sont robustes face à l'incertitude, en planifiant des solutions qui sont bonnes en moyenne ou qui minimisent les risques associés à différents scénarios possibles.

Les techniques d'ordonnancement stochastique peuvent varier en fonction du type de système et des objectifs de planification. Elles peuvent impliquer l'utilisation de méthodes d'optimisation stochastique, de simulations Monte Carlo, ou d'autres approches probabilistes pour trouver des solutions efficaces et robustes.

Cette approche est couramment utilisée dans divers domaines tels que la gestion de la production, la logistique, les systèmes de transport, la finance, et d'autres domaines où l'incertitude joue un rôle important dans la prise de décision. En incorporant la dimension stochastique dans le processus d'ordonnancement, les entreprises peuvent mieux gérer les risques et améliorer leurs performances opérationnelles.

9. Formulation d'un problème d'ordonnancement stochastique

Un problème d'ordonnancement stochastique implique la planification de tâches ou d'activités dans un environnement où les temps d'exécution, les temps de traitement ou d'autres paramètres sont aléatoires ou incertains. Voici une formulation générale d'un tel problème :

Considérons un ensemble de tâches à ordonnancer. Chaque tâche j a les caractéristiques suivantes :

p_j**:**Le temps de traitement déterministe requis pour achever la tâche$j$.

r_j **:**Le temps d'arrivée de la tâche$j$ (par exemple, le temps auquel la tâche devient disponible pour être exécutée).

d_j **:**La date limite pour achever la tâche$j$.

Cependant, dans un problème d'ordonnancement stochastique, certaines de ces caractéristiques peuvent être aléatoires. Par exemple, p_j peut être une variable aléatoire représentant le temps de traitement réel de la tâche$j$.

L'objectif principal est généralement de minimiser une certaine fonction de coût, par exemple, le temps total de retard, la durée totale du projet, etc.

Une formulation mathématique courante d'un problème d'ordonnancement stochastique pourrait être un problème de programmation mathématique stochastique. Voici une formulation possible utilisant la programmation en nombres entiers :

Minimiser :

$$\sum_{j=1}^{n}(C_j - d_j)$$

Sous les contraintes :

Contraintes de précédence entre les tâches.

Contraintes de disponibilité des ressources.

Contraintes de temps de traitement stochastique, par exemple :

$$C_j \geq r_j$$

$$C_j \geq C_i + p_i \forall\ i \ \ tel\ que\ \ i\ précède\ j$$

Où $C_j$représente le moment de fin de la tâche j.

Cette formulation vise à minimiser le temps total de retard des tâches par rapport à leurs dates limites, en tenant compte de l'incertitude associée aux temps de traitement des tâches. Les détails spécifiques de la formulation dépendront des caractéristiques et des contraintes spécifiques du problème d'ordonnancement stochastique considéré.

10. Les méthodes de résolution du problème d'ordonnancement

10.1.Les méthodes exactes

Les méthodes exactes utilisent surtout deux approches de résolution très connues : la programmation dynamique et les procédures par séparation et évaluation. Ces méthodes sont souvent utilisées pour résoudre les problèmes combinatoires de manière exacte, en ordonnancement tout particulièrement. Ce sont des méthodes d'énumération implicite : L'énumération explicite construit toutes les solutions réalisables et retient une parmi les meilleures. L'énumération implicite consiste à explorer l'ensemble de toutes les solutions réalisables en éliminant des sous-ensembles de solutions moins intéressants sans avoir à les construire.

Nous pouvons citer trois approches particulièrement célèbres : La programmation dynamique introduite par Bellman dans les années 50, la méthode par séparation et évaluation (Branch and Bound en anglais : notée B&B) et l'algorithme de retour arrière (Backtracking)[3].

10.2.Les méthodes approchées

Une méthode approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles, D'un autre côté les algorithmes

d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires. Les méthodes approchées englobent deux classes : les heuristiques et les métaheuristiques. La particularité qui différencie les méthodes métaheuristiques des méthodes heuristiques c'est que les métaheuristiques sont applicables sur de nombreux problèmes. Tandis que, les heuristiques sont spécifiques à un problème donné.[3].

10.2.1. Les heuristiques

Une heuristique est une règle empirique simple basée sur l'expérience. C'est une technique de calcul approchée qui exploite d'une façon satisfaisant le problème à optimiser et qui fournit une solution admissible, non nécessairement exacte, dans un temps polynomial pour un problème d'optimisation difficile. En d'autres termes, une heuristique est un algorithme qui sacrifie en partie, la qualité de la solution au sens purement mathématique pour accélérer le processus de résolution. Elle est également une stratégie de bon sens pour se déplacerintelligemment dans l'espace des solutions, afin d'obtenir une solution approchée, la meilleure possible, dans un délai de temps raisonnable. Elle est là, à la place d'une méthode exacte qui donne la solution dans un temps de résolution exponentiel. Elle est dédiée, généralement, à un problème bien particulier. Les heuristiques sont dépendantes du problème à résoudre, principalement dans le choix du voisinage (donc dans le déplacement dans l'espace des solutions) [3].

De nombreuses méthodes heuristiques ont été proposées dans la littérature pour résoudre les problèmes d'ordonnancement d'atelier. On distingue :

- **FIFO** (First In First Out) : la première tâche qui vient est la première tâche ordonnancée.
- **SPT** (ShortestProcessing Time) : la tâche ayant le temps opératoire le plus court est traitée en premier lieu.
- **LPT** (LongestProcessing Time) : la tâche ayant le temps opératoire le plus important est ordonnancée en premier lieu.
- **EDD** (Earliest Due Date) : cet algorithme choisit parmi les tâches exécutables celle dont le délai est échu le plus tôt. Si aucune tâche n'est disponible, alors un temps libre est génér.
- **SRPT** (Shortest Remaining Processing Time) : cette règle, servant à lancer la tâche ayant la plus courte durée de travail restant à exécuter, est très utilisée pour minimiser les encours et dans le cas des problèmes d'ordonnancement préemptifs.

•**ST** (Slack Time) : à chaque point de décision, l'opération ayant la plus petite marge temporelle est prioritaire. Faute de disponibilité des ressources de production, cette marge peut devenir négative [3].

10.2.2. Métaheuristiques

Les métaheuristiques d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation. Les métaheuristiques s'efforcent de résoudre tout type de problème d'optimisation. Elles sont caractérisées par leur caractère stochastique, ainsi que par leur origine discrète. Elles sont inspirées par des analogies avec la physique (recuit simulé, recuit micro canonique), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Cependant, elles ont l'inconvénient d'avoir plusieurs paramètres à régler. Il est à souligner que les métaheuristiques se prêtent à toutes sortes d'extensions, notamment en optimisation mono-objectif et multi-objectif.

Les algorithmes, que nous considérons dans cet article, sont les algorithmes d'ordonnancement par liste et la méthode du recuit simulé et ses variantes. Pour évaluer le makespan d'un ordonnancement, nous exploitons un modèle de simulation déterministe[3].

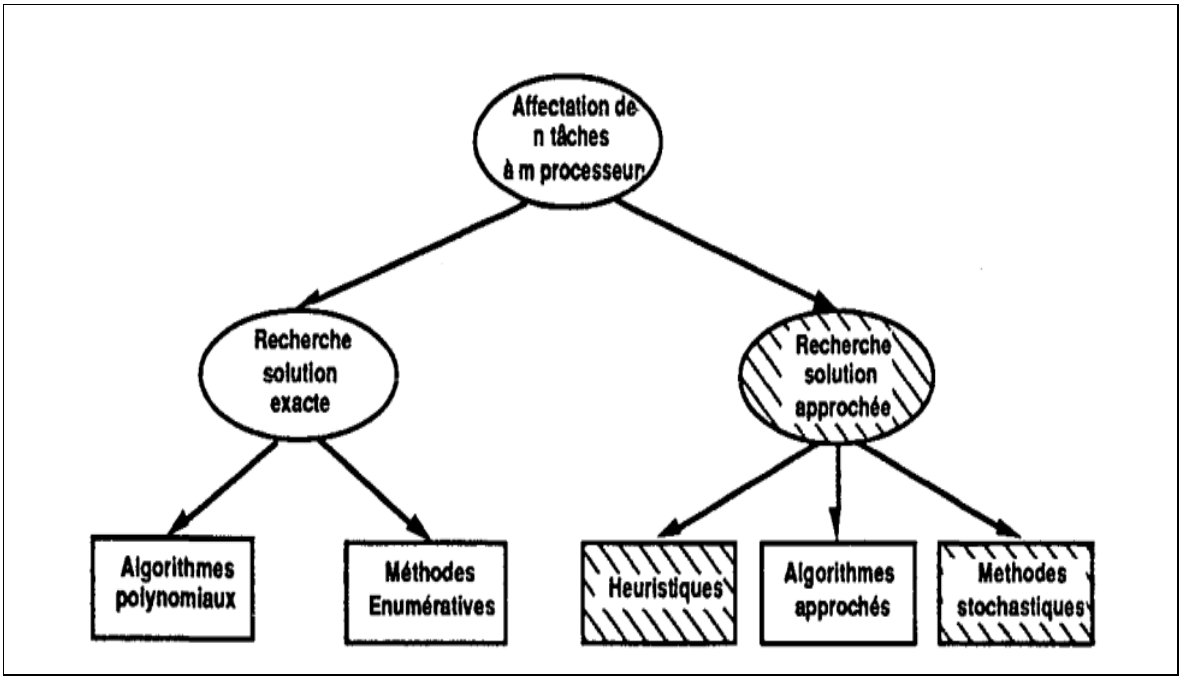

Figure 1- 3: Différentes méthodes de résolution du problème d'ordonnancement [16].

11. Conclusion

En résumé, ce chapitre a exploré l'importance de l'ordonnancement dans la gestion opérationnelle. Nous avons défini l'ordonnancement, discuté de son évolution historique et de ses applications pratiques. Les types d'ordonnancement et les problèmes associés ont également été abordés. En adoptant des approches efficaces d'ordonnancement, les organisations peuvent améliorer leur performance opérationnelle et leur compétitivité.

CHAPITREII

L’ÉTAT DE L’ART

1. Introduction

Dans ce chapitre, on représente des articles des chercheures sur notre sujet de thèse, et résumons donc chaque article.

2. Revue de la littérature

• **(2013) Alexandre Salch** Le cadre de cette thèse est l'étude de systèmes de production avec impatience. Ces systèmes sont modélisés comme des problèmes d'ordonnancement stochastiques avec des dates d'échéance. Dans la littérature, peu de résultats existent sur le contrôle optimal de ce genre de systèmes. C'est dans ce cadre que s'inscrit cette thèse. Il considère un système générique avec une machine, sur laquelle des tâches sont à exécuter. Les durées d'exécution, les dates d'échéance (ou durées d'impatience) et les dates de disponibilité des tâches sont des variables aléatoires. À chaque tâche est associé un poids et l'objectif est de minimiser l'espérance du nombre pondéré de tâches en retard.

Dans ces étude, Ils utilisent différentes modélisations, rendant compte des différentes contraintes régissant des systèmes réels. Notamment, nous faisons la différence entre l'impatience, le fait d'avoir attendu trop longtemps, et l'abandon, le fait de quitter le système suite à l'impatience. Dans la classe des politiques statiques, Ils donnent des ordonnancements optimaux pour des problèmes avec impatience. Dans la classe des politiques dynamiques avec préemption, Ils donnent.de nouvelles conditions garantissant l'optimalité d'une politique stricte pour des problèmes avec abandon. Ils donnent aussi une heuristique plus efficace que celles que l'on trouve dans la littérature. Enfin, Ils explorent des variantes et des extensions de ces problèmes, lorsque le système comporte plusieurs machines et lorsque la préemption n'est pas autorisée [1].

•**(2019) KARAM Zakaria** Le problème traité dans ce mémoire concerne l'insertion d'un novelle opération dans l'ordonnancement à machines parallèles en vue de minimiser le makespan. Les opérations peuvent être exécutées en parallèle sur plusieurs machines. Une méthode a été suggérée basée sur les algorithmes génétiques, et l'algorithme glouton ont été, ensuite, proposée pour la résolution de problèmes [8].

•**(2017) CHERGUI Abderrahmane DAHMANI AbdNaceur et ADDA ABBOU Abdellah** intéressent à l'adaptation et l'hybridation de l'algorithme de colonies d'abeilles (BCO) pour la résolution de problème d'ordonnancement Flow Shop. L'idée consiste à insérer

une méthode de recherche locale au niveau de l'algorithme de base pour améliorer l'intensification et la diversification de la technique de recherche de l'algorithme. Les résultats de simulation montrent que l'algorithme BCO hybride est plus performant que le BCO de base pour différentes classes de systèmes proposées [4]**.**

- **(2004)LATRECHE FAIZ** s'intéresse aux problèmes d'ordonnancement à cheminement unique (flow shop) et multiple (job shop). La partie théorique est consacrée à la représentation de ces deux types des problèmes avec quelque cas particuliers. Dans la partie pratique, on a proposé deux heuristiques A, B basés sur le même principe que CDS pour résoudre le problème général n/m/F/C_{max}puis on a programmé ces trois méthodes (A, B et CDS) pour les tester et comparer sur différents exemples [10].

- **(2017) ZAZGAD AHLEM** Ce travail fait un tour d'horizon sur l'ensemble des méthodes d'ordonnancement et plus précisément la méthode PERT. On s'intéresse à l'utilisation de la logique floue dans le cadre de problèmes pouvant s'apparenter aux problèmes d'ordonnancement de projet. Ce travail présente le calcul du chemin critique par trois méthodes PERT /CPM, PERT stochastique et PERT floue, et la comparaison entre ces méthodes [17].

- **(1994) S.Norre et Philippe CHRÉTIENNE** traite du problème d'affectation statique d'un ensemble de tâches, soumises à des contraintes de précédence, sur une architecture multiprocesseur à mémoire partagée. Deux types d'ordonnancement sont considérés : les ordonnancements déterministes (la durée d'exécution de chaque tâche est connue) et les ordonnancements stochastiques (la durée d'exécutionde chaquetâche est modéliséeparune loi de probabilité). Pour chacunde ces deux problèmesd'ordonnancement, ilsproposentdifférentesméthodesderésolutionetilsélaborent différentsmodèlesafind'évaluer l'efficacité des ordonnancements générés. La déterminationd'ordonnancements déterministes est réaliséeà l'aide de méthodesqui reposent surlesalgorithmes d'ordonnancement par liste et sur l'algorithme du recuit simulé et ses variantes ; les modèles sont des modèles de simulation déterministe finie. Pour les ordonnancementsstochastiques,ilsproposentmodèles de simulationdéterministe finie.Pour lesordonnancements stochastiques, ilsproposent des méthodesqui sont des adaptations du recuit simulé. Plusieurs modèles sont développés : un modèle d'analyse markovienne et des modèles de simulation stochastiques. Bien que ce problème soit NP-complet, ces méthodes fournissent des solutions satisfaisantes en des temps de calcul raisonnables [16].

- **(2022) BENNAOUI BOUCHRA et MANNED MAROUA** Approche d'apprentissage automatique pour les problèmes de planification de machine unique Ce thème vise à concevoir

un algorithme d'apprentissage automatique pour aborder un NP-dur problème de planification de machine unique avec grande taille. Cette approche se compose de deux parties : les premiers par « apprentissage » visent à apprendre le système en fournissant au système un nombre significatif d'instances de petite taille résolues par une méthode exacte de programmation dynamique. La deuxième étape consiste à concevoir une approche d'apprentissage automatique pour aborder certaines instances de grande taille, au fur et à mesure que nous avançons, même ces instances fournissent notre système pour l'améliorer efficacement. Une comparaison avec la métaheuristique choisie est nécessaire pour justifier la contribution [2]**.**

•**(2015) DJEMIAT Sara**Dans ce mémoire elle a considéréune classe célèbre de problèmes d'optimisation combinatoire. Ceux sont les problèmes d'ordonnancement d'atelier. En fait elle a pris deux catégories que se soit pour les problèmes d'ordonnancement de tâches sur une seule ou plusieurs machines identiques parallèles ou ceux de Job-shop, Flow-shop ou Open-shop. Et puisque l'étude des problèmes simples de base (polynomiaux) a une importance majeure pour l'étude des problèmes complexes (NP-difficiles) elle s'est concentré à l'étude des premiers problèmes, i.e. ceux qu'on a pu déterminer un algorithme de complexité polynomiale pour leurs solutions.

Citons les problèmes à ressource unique

$1\,||\sum C_i$, $1\,||\sum F_i$, $1\,||\sum \omega_i C_i$,$1\,||\sum \omega_i F_i$, $1\,||L_{max}$, $1\,|(p_i = p)|C_{max}$,$1\,1\,||\sum U_i$.Alors pour les problèmes à ressource multiple citons le problème $F2||C_{max}$.

Les solutions de ceux-ci peuvent être utilisées comme heuristiques pour déterminer une solution admissible pour les mêmes problèmes avec d'autres contraintes supplémentaires.

Par conséquent les valeurs des solutions exactes trouvées peuvent être considérées comme solutions approchées pour les problèmes complexes. Il est important de noter aussi que la valeur de la solution optimale pour un problème polynomial pourrait être servie commeévaluation par défaut (excès) sur la fonction objective d'un problème NP difficile déduit de premier en imposant d'autres contraintes.

En fait et comme étude de cas on a considéré le problème $1\,||\sum \omega_i T_i$qui est NP difficile et on a essayé d'appliquer des algorithmes (heuristiques) telle que SWPT, EDD (optimales respectivement pour$1\,||\sum \omega_i C_i$, $1\,||L_{max}$).

Une étude numérique comparative entre la performance de ces méthodes a été faite en utilisant le langage Java [15].

•**(2021)ELBAGGOUR Messaoud**Dans ce mémoire, ils étudient la question de l'ordonnancement sur un groupe de machines parallèles. Tout d'abord, ils ont passé en revue un

ensemble de définitions, puis les solutions bien connues, et enfin ils ont donné quelques exemples de modèles d'ordonnancement [12].

- **(2005) MARTIN SKUTELLAet MARC UETZ**Ils examinent les problèmes d'ordonnancement de machines parallèles identiques, où les tâches sont soumises à des contraintes de précédence et à des dates de début, et où les temps de traitement des tâches sont régis par des distributions de probabilité indépendantes. Leur objectif est de minimiser la valeur attendue du temps total de terminaison pondéré. En s'appuyant sur une relaxation de programmation linéaire par Mohring, Schulz et Uetz [J. ACM, 46 (1999), pp. 924–942] et un algorithme d'ordonnancement de liste retardée par Chekuri et al. [SIAM J. Comput., 31 (2001), pp. 146–166], ils dérivent les premiers algorithmes d'approximation à facteur constant pour ce modèle [13].

- **(1999) C. R. Sox, J. L. PETER, B. ALAN et A. M. JOHN**Cet article passe en revue la littérature de recherche actuelle sur le problème de planification de lots stochastiques, qui concerne la planification de la production de plusieurs produits avec une demande aléatoire sur une seule installation ayant une capacité de production limitée et des changements importants entre les produits. La version déterministe de ce problème a reçu une couverture significative dans la littérature ; cependant, le problème stochastique n'a été abordé que récemment. De plus, toute une gamme de méthodes analytiques distinctes a été appliquée à ce problème. Cet article propose un cadre unificateur pour discuter de ces approches et offre des explications et des clarifications sur les différentes méthodes analytiques pour ce problème. Après avoir discuté de certaines des implications en matière de modélisation et de gestion de ce problème, une revue détaillée des stratégies de contrôle en temps continu et discret est donnée, et des domaines de recherche supplémentaires sont évoqués. (1999 Elsevier Science B.V. Tous droits réservés) [5].

- **(2020) LEBOUKH Amira** Dans ce mémoire elle a pu découvrir les notions les plus importantes de l'ordonnancement, avec les méthodes de résolution de leurs problèmes telles que les méthodes approchées auxquelles elle a recours lorsque nous ne pouvons pas les résoudre par les méthodes exactes, ainsi qu'en présentant certains types de problèmes d'ordonnancement comme le modèle à une opération [9].

- **(2017)KAHLOULA Tarek Yassine**Un problème de planification est de gérer et d'organiser et de contrôler les charges de travail de différentes activités en utilisant différentes ressources, en respectant les restrictions de l'environnement de travail afin de trouver des solutions minimales avec une bonne productivité. Cet article étudie les problèmes de planification de production multitâches sur des différents cas. L'objectif principal de ce travail

est de trouver de bonnes solutions pour les problèmes d'ordonnancement dans des scénarios plus pratiques, en construisant des algorithmes constructifs [7].

•**(2020) DAHMANE Khawla SILINI Loubna** Le besoin de services de communication augmente rapidement, car le service de communication mobile est synonyme d'un style de communication idéal permettant de communiquer à tout moment, n'importe où et avec n'importe qui.

Les réseaux mobiles ad hoc (MANET) sont un ensemble de nœuds mobiles connectés par des liaisons sans fil qui forment une topologie de réseau temporaire qui fonctionne sans station de base et administration centralisée.

Ils étudientun problème de routage de réseau dans cette thèse étudie le routage de chemin le plus court en ligne sur des réseaux à sauts multiples.

Les coûts ou délais de liaison varient dans le temps et sont modélisés par des processus aléatoires indépendants et répartis de manière identique, dont les paramètres sont initialement inconnus. Les paramètres, et donc le chemin optimal, ne peuvent être estimés qu'en acheminant les paquets à travers le réseau et en observant les retards réalisés. L'objectif est de trouver une politique de routage qui minimise le regret (la différence cumulée de retard attendu) entre le chemin choisi par la politique et le chemin optimal inconnu.

Ils formulentle problème comme un problème d'optimisation combinatoire des bandits. Nous présentons des résultats montrant qu'une politique KL-SR (KLbased Source-Routin) est optimale pour le problème de routage par NS2, TCL[11].

3. Conclusion

Les chercheurs dans ces études ont apporté des contributions variées et importantes dans le domaine des problèmes d'ordonnancement, en abordant une gamme de défis et d'innovations. En modélisant les systèmes de production avec des variables aléatoires, ils ont démontré comment réduire le retard des tâches grâce à des stratégies d'ordonnancement optimales et à des politiques dynamiques efficaces. Les chercheurs ont rencontré plusieurs défis, tels que la complexité des modèles stochastiques et les contraintes de ressources, mais ils les ont surmontés

en utilisant des algorithmes hybrides, l'analyse markovienne et des modèles de simulation. Les résultats qu'ils ont obtenus ont montré l'efficacité des solutions proposées et l'importance de l'application de techniques comme l'apprentissage automatique et les algorithmes génétiques pour améliorer la performance dans les problèmes d'ordonnancement complexes. Ces recherches ouvrent de nouvelles perspectives pour améliorer l'efficacité des systèmes de production et fournir des solutions pratiques aux défis industriels réels.

CHAPITRE III

LES PROBLÈMESD'ORDONNANCEMENTSTOCHASTIQUE

1. Introduction

Ce chapitre se concentre sur un problème fascinant : l'ordonnancement stochastique. Il s'agit d'un domaine de la gestion des opérations où nous devons planifier l'exécution de tâches sur une machine, mais avec une particularité intrigante : les temps d'exécution des tâches sont aléatoires. Cette incertitude nécessite une approche analytique et décisionnelle spécifique pour trouver des ordonnancements optimaux tout en tenant compte du risque associé à chaque plan. Nous explorerons les concepts clés, les modèles mathématiques et les défis pratiques de ce domaine passionnant.

2. Problématique

Consiste à trouve ordonnancement de tâche sur machine avec des temps d'exécution Aléatoire consiste objectif.

La problématique de notre travail réside dans la recherche d'un ordonnancement optimal des tâches sur une machine, en tenant compte de la variabilité des temps d'exécution. L'objectif est de minimiser le temps de complétion maximal $\mathbf{min}\, C_{max}$, tout en respectant les contraintes.

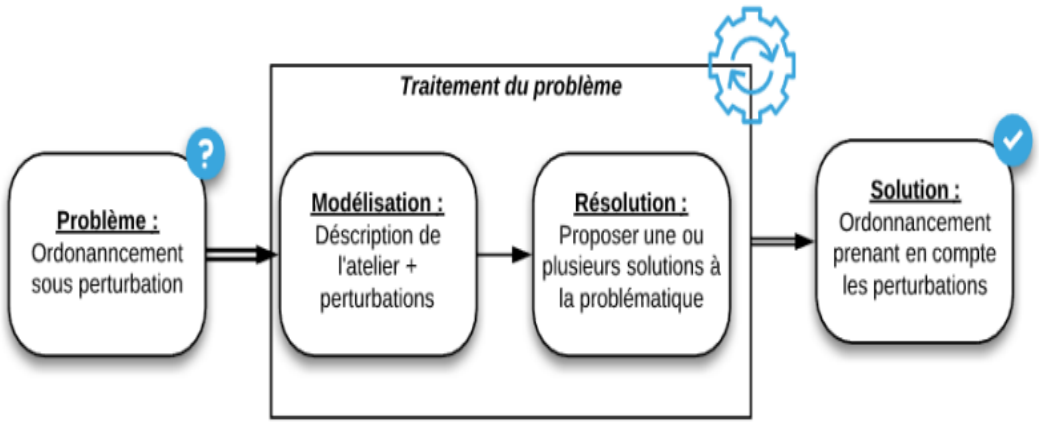

Figure 3- 1: Méthode de traitement de la problématique [6].

3. Contexteet motivation

Le "problème d'ordonnancement stochastique" fait référence à une classe de problèmes d'ordonnancement où les temps d'exécution des tâches sont soumis à des incertitudes ou des variations aléatoires. Contrairement aux problèmes d'ordonnancement déterministes, où les durées des tâches sont fixes et connues à l'avance, les problèmes d'ordonnancement stochastique prennent en compte l'incertitude inhérente aux processus de production ou aux opérations.

Les motivations pour étudier les problèmes d'ordonnancement stochastique sont multiples :

•**Robustesse et résilience :** Les environnements de production réels sont souvent soumis à des perturbations aléatoires telles que des pannes de machines, des variations de la demande ou des retards dans les fournitures. La planification robuste face à ces incertitudes est cruciale pour maintenir l'efficacité opérationnelle.

•**Optimisation sous contrainte :** Dans de nombreux contextes, il est nécessaire d'optimiser l'ordonnancement des tâches tout en respectant diverses contraintes, telles que les délais de livraison, les capacités des ressources et les coûts.

•**Amélioration des performances :** En tenant compte des incertitudes, il est possible de concevoir des stratégies d'ordonnancement qui maximisent les performances globales, telles que la minimisation du temps d'attente, la maximisation de la production ou la réduction des coûts.

•**Prise de décision éclairée :** Les problèmes d'ordonnancement stochastique fournissent un cadre pour prendre des décisions éclairées en tenant compte des risques et des incertitudes. Cela peut aider les gestionnaires à élaborer des plans d'action robustes et adaptatifs.

•**Applications diverses :** Les problèmes d'ordonnancement stochastique trouvent des applications dans de nombreux domaines, notamment la production manufacturière, la logistique, les systèmes de transport, les réseaux de télécommunications et les soins de santé.

En résumé, l'étude des problèmes d'ordonnancement stochastique vise à développer des méthodes et des modèles permettant de planifier efficacement les opérations dans des environnements incertains, afin d'améliorer les performances opérationnelles et de prendre des décisions éclairées face à l'incertitude.

4. Formulation mathématique

4.1 Variables de décision

$\boldsymbol{\pi}$**:** Séquence d'ordonnancement des tâches.

$\boldsymbol{s_j}$**:** Temps de début d'exécution de de la tâche j.

$\boldsymbol{C_j}$ **:** Date de fin (d'achèvement) de la tâche j.

4.2 Notation

$\boldsymbol{J} = \{\mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{n}\}$: Ensemble des tâches à ordonnancer.

$\boldsymbol{P_j}$ **:** Variable aléatoire représentant le temps d'exécution de la tâche $\boldsymbol{j}$, avec une fonction de distribution de probabilité $\boldsymbol{F_j(t)}$.

$\boldsymbol{n}$**:** le nombre de tâches.

$\boldsymbol{m}$**:** le nombre de machines.

j:L'indice de la tâche.

d_j:Date de fin la tâche j.

C_{max}: Makespan.

4.3 Fonction Objectif

$$C_{max} = max(C_i)$$

4.4 Contraintes

- Pas de préemption.
- Machines identiques.
- Tâches indépendantes.

4.5 Formulation de POS

$$Min\ C_{max}$$

5. Complexité du Problème

La complexité du problème d'ordonnancement stochastique est généralement classée comme étant **NP-difficile**. Cela signifie que, bien qu'il soit facile de vérifier si une solution donnée est valide (dans le cas de la vérification de l'ordonnancement d'une séquence de tâches), il peut être extrêmement difficile de trouver une solution optimale en un temps raisonnable, surtout à mesure que la taille du problème augmente.

Le terme "**NP-difficile**" signifie que le problème est au moins aussi difficile à résoudre que les problèmes **NP**, ce qui inclut un large éventail de problèmes de décision pour lesquels une solution peut être vérifiée en temps polynomial. Cependant, cela ne garantit pas que le problèmeestNP-complet, ce qui signifierait qu'il est aussi difficile que tous les autres problèmes **NP**.

La complexité du problème d'ordonnancement stochastique dépend de facteurs tels que le nombre de tâches, la distribution de probabilité associée aux temps d'exécution de chaque tâche, les contraintes spécifiques du problème (telles que l'absence de préemption ou l'identité des machines), et les techniques algorithmiques utilisées pour le résoudre. En raison de sa nature stochastique, des méthodes d'optimisation spécifiques doivent souvent être utilisées, ce qui peut augmenter la complexité de la résolution du problème.

Exemple :

Il y a :**n**tâchesetceproblème$P_m|\ \ |C_{max}$

Le nombre de solution possibles est :$m \cdot n!$

Donc, la complexité de ce problème est :**Exponentièlle.**

6. Méthode utilisée

6.1.Recherche locale

La méthode utilisée dans cette étude est la recherche locale, également connue sous le nom de recherche locale itérative. Cette approche repose sur l'exploration progressive de l'espace de recherche en se déplaçant d'une solution à une autre, généralement en utilisant des règles de voisinage pour générer de nouvelles solutions à partir des solutions actuelles. Contrairement aux méthodes globales qui examinent tout l'espace de recherche simultanément la recherche locale se concentre sur une petite région de l'espace de recherche à la fois. Cela permet une exploration plus efficace des solutions potentielles, en particulier dans les espaces de recherche de grande dimension. Dans cette étude, nous appliquons la recherche locale pour résoudre le problème spécifique de« problème d'ordonnancement stochastique » utilisant des algorithmestels que **«** Algorithme de recherche locale basique, Recherche tabou adaptative**»**

❖ l'ordonnancement stochastique sur deux machines parallèles, la recherche locale peut être utilisée pour optimiser l'affectation des tâches aux machines afin de minimiser le temps total de traitement. Voici comment cela pourrait fonctionner :

1_Initialisation : Sélectionnez une solution initiale qui affecte les tâches aux machines de manière arbitraire.

2_Évaluation : Calculez le temps total de traitement pour la solution initiale.

3_Voisinage : Définissez un voisinage pour la solution actuelle en déplaçant une tâche d'une machine à une autre ou en permutant les positions de deux tâches sur la même machine.

4_Exploration : Parcourez les solutions voisines pour trouver celle qui offre la meilleure amélioration locale du temps total de traitement.

5_Amélioration : Parcourez les solutions voisines pour trouver celle qui offre la meilleure amélioration locale du temps total de traitement.

6_Critère d'arrêt : Répétez les étapes 3 à 5 jusqu'à ce qu'aucune amélioration ne soit possible ou jusqu'à ce qu'un critère d'arrêt prédéfini soit atteint (comme un nombre maximum d'itérations).

7_Solution finale : La solution obtenue après la convergence de l'algorithme est considérée comme la solution optimale ou une approximation de celle-ci.

Exemple :

Supposons que nous avons trois tâches à ordonnancer : T1, T2, T3, avec les temps d'exécution de la (Tableau 3-1)suivants sur chaque machine :

Tâche	**T1**	**T2**	**T3**
Temps d'exécution m1	2	3	4
Temps d'exécution m2	3	4	2

Tableau 3- 1 : Planifier destâches.

Supposons que notre solution initiale soit la(Tableau 3-2) suivante :

Machine 1	**Machine 2**
T1 **T3**	**T2**

Tableau 3- 2: Solution 1.

Dans cette configuration, le (C_{max}) est de 3+4=7 unités de temps

Maintenant, explorons le voisinage de cette solution. Un voisinage potentiel pourrait être d'échanger les tâches **T2**et **T3** entre les machines, ce qui donne la (Tableau 3-3)suivante :

Machine 1	**Machine 2**
T1 **T2**	**T3**

Tableau 3- 3: Solution 2.

Dans cette nouvelle configuration, le (C_{max}) est de 4+3=7 unités de temps.

Cependant, si nous continuons à explorer le voisinage, nous pourrions découvrir une solution meilleure. Par exemple, si nous échangeons les tâches **T1**et **T2** entre les machines, nous obtenons la configuration suivante :

6.2 Méthodes des intervalles

Un projet est un ensemble de tâches qui doivent être réalisées dans le respect d'un certain ordre donné par les relations de précédence. Ces relations définissent quelles tâches doivent être complètement terminées avant que celles qui sui vent puissent être à leur tour réalisées, on parlera dans se cas de réalisation séquentielle ; des tâches pourront être réalisées en même temps, on parlera dans ce cas de réalisation en parallèle. Cet ensemble de relations série et parallèle peut être représenté par un réseau ou les nœuds représentent les tâches et les arcs reliant les différents nœuds représentent les relations de précédence. Cette représentation s'appelle « Project Evaluation and Review Technique » (PERT). Le but de cette méthode est donc de trouver le chemin critique du réseau, c'est-à-dire l'ensemble des tâches critiques. Une tâche est critique lorsque tout retard pris sur cette tâche retarde d'autant la fin du projet. La méthode CPM permettra donc d'identifier les tâches critiques, mais aussi de calculer les marges des autres tâches, c'est-à-dire la différence entre la date de début au plus tôt de la tâche et la date de début au plus tard de la tâche. La marge représente le temps en plus disponible pour réaliser la tâche, on peut l'utiliser pour commencer la tâche plus tard ou l'utiliser au cas où la tâche pourrait prendre plus de temps. La méthode CPM se déroule en deux temps : premièrement la phase qu'on appellera ascendante, puis deuxièmement la phase descendante. Lors de la phase ascendante on recherche la date de début au plus tôt pour chaque tâche, celle-ci correspond au maximum des dates de début au plus tôt des prédécesseurs auxquelles on ajoute leur durée.

L'opération est effectuée de tâche en tâche dans l'ordre topologique, c'est-à-dire qu'une tâche donnée doit être traitée avant l'un de ses successeurs. Pour pouvoir effectuer ce calcul il faut assigner la date de début au plus tôt de la première tâche à zéro. Puis lors de la phase descendante on calcule la date de début au plus tard, celle-ci correspond au minimum de la date de début au plus tard de l'ensemble des tâches précédentes auxquelles on retire la durée de la tâche en question. Cette opération est effectuée dans l'ordre inverse du tri topologique.

Pour pouvoir effectuer ce calcul il faut assigner à la dernière tâche une date de début au plus tard identique à sa date de début au plus tôt. En résumé :

Si $\boldsymbol{d_i}$ représente la durée de la tâche $\boldsymbol{i}$, li représente sa date de début au plus tôt et$\boldsymbol{T_i}$sa date de début au plus tard, on obtient :

$$t_i = \max_{j \in pred(i)} (t_j + d_j) \textbf{Équation 1}$$

$$T_i = \min_{j \in succ(i)} (T_j - d_j) \textbf{Équation 2}$$

Ainsi la méthode CPM nous permet de définir la date de début au plus tôt, la date de début au plus tard ainsi que la marge de chaque tâche et permet de définir la date de fin au plus tôt du projet [14].

Dans méthode des intervalles nous avons choisi de traiter de l'imperfection de l'information concernant la durée des tâches. Les ressources seront considérées sans incertitude, c'est-à-dire disponible tout le temps et en quantité voulue, les relations de précédence elles aussi seront considérées fixes, c'est-à-dire qu'une tâche ne changera pas de prédécesseur(s) ni de successeur(s) au cours de la réalisation du projet. La modélisation de l'imperfection de l'information choisie pour cette méthode est celle de l'intervalle, modèle utilisé lorsque l'information est incomplète ou imprécise, comme par exemple lors du lancement d'un projet totalement nouveau ou utilisant des technologies encore inutilisées. Ce type de modélisation est simple et sera facile à exploiter pour des experts devant définir les bornes de cet intervalle. Les relations de précédence étant fixes, les ressources étant connues et parfaitement disponibles, il est alors possible la méthode CPM comme mode de représentation du problème. Cependant les durées des tâches étant définies par des intervalles, le chemin critique évolue et il est donc impossible d'appliquer la méthode CPM de façon classique et celle-ci nécessite alors d'être modifiée [14].

6.2.1. Revue de littérature appliquée à la méthode

Ce sont les auteurs travaillant sur l'ordonnancement flou qui ont été les premiers confrontés à ce problème d'adaptation de la méthode du CPM à des situations où les durées des activités sont modélisées sur des intervalles. En effet, lors de la réalisation de la phase descendante de la méthode CPM en nombres flous, Dubois, Fargier et Galvagnon (2003) furent confrontés à des problèmes pour le calcul de la date de début au plus tard et de la marge. Le problème vient du fait que lorsque l'on commence la phase descendante on égalise la date de début au plus tard et la date de début au plus tôt de la dernière tâche. Ils ont donc eu l'idée pour simplifier leur problème de faire les calculs d'abord avec des intervalles plutôt qu'avec des nombres flous (Chanas, Dubois et Zielifiski, 2002), c'est-à-dire en affectant à chaque tâche une durée minimale et une durée maximale[11].

La première constatation fut que les différentes valeurs recherchées, c'est-à-dire la date de début au plus tôt, la date de début au plus tard et la marge, pour une tâche donnée, se trouvent dans les configurations extrêmes. Ainsi pour atteindre les va leurs extrêmes des intervalles des

différentes valeurs cherchées il faudra assigner aux différentes tâches du réseau : soit leur durée maximale, soit leur durée minimale, ce qui donne un nombre de configuration possible à 2''ombre de lâche s, c'est-à-dire que le problème est «N P-hard ». Le calcul de la date de début au plus tôt à l'aide d'intervalles ne pose pas de problèmes particuliers et ce calcul est de complexité linéaire. Il suffit d'assigner la durée minimale à toutes les tâches précèdent une tâche donnée pour obtenir sa date de début au plus tôt minimale, et inversement les durées maximales pour obtenir la date de début au plus tard maximale. Par contre, calculer les marges des tâches et définir leur criticité est chose bien moins facile, et calculer la marge demeure un problème NP-dur (Chanas et Zelinski, 2003). Récemment des algorithmes polynomiaux capable de trouver des dates de début au plus tôt ont été mis au point (Zelinski, 2005). Ce n'est que récemment qu'un groupe d'auteurs (Chanas, Dubois et Zielifiski, 2002) ont développé des configurations, c'est-à-dire l'affectation des durées des différentes tâches du réseau, permettant de trouver les bornes des intervalles des marges. Les configurations définissant les différentes valeurs pour une tâche donnée sont les sui vantes :

- Pour le calcul de la date de début au plus tôt minimale :

  Assigner la durée de l'ensemble des tâches à leurs valeurs minimales.

- Pour le calcul de la date de début au plus tôt maximale :

  Assigner la durée de l'ensemble des tâches à leurs va leurs maximales.

- Pour le calcul de la date de début au plus tard minimale :

  Assigner la durée de toutes tâches à leur durée minimale sauf sur un chemin allant de la tâche étudiée jusqu'à la fin du réseau où là la durée assignée sera maximale.

- Pour le calcul de la date de début au plus tard maximale :

  Assigner la durée des tâches à leurs valeurs minimales sauf sur un chemin allant du début du réseau jusqu'à la fin.

- Calcul de la marge minimale :

  Assigner la durée de toutes les tâches à leurs valeurs minimales sauf sur un chemin allant du début à la fin du réseau et passant par la tâche étudiée.

- Calcul de la marge maximale :

  Assigner la durée des tâches à leurs valeurs minimal es sauf sur un chemin allant du début à la fin du réseau.

L'ensemble de ces résultats permet donc de calculer les différents intervalles des valeurs recherchées avec des algorithmes polynomiaux. En effet, il suffit maintenant d'énumérer les différents chemins possibles afin de trouver les bornes des différentes valeurs cherchées.

Les calculs sont donc possibles quelque soit la taille du réseau. Néanmoins les infoll11ations que l'on tire de ces calculs ne sont pas forcément très significatives pour un gestionnaire de projet. En effet les informations fourni es par ces calculs ne sont que des intervalles. Ensuite, au niveau de la criticité des tâches, il n'y a que peu d'information : soit elles seront toujours critiques, c'est-à-dire marge minimale égale à la marge maximale égale à zéro, soit elles ne sont jamais critiques, c'est-à-dire marge minimale différente de zéro et finalement possiblement critique, c'est-à-dire parfois critique parfois non suivant les configurations. Le problème principal reste toujours le même : définir la criticité des tâches exactement comme dans une méthode CPM "classique". La notion de criticité est indispensable au gestionnaire de projet pour savoir où porter le plus d'attention et savoir où se trouvent les dangers en cas de prise de retard. Il faut donc essayer de traiter le problème différemment afin d'essayer de fournir un peu plus d'information au gestionnaire de projet[14].

6.3 Traitement du problème

Pour traiter ce problème, nous avons vu qu'il fallait travailler avec des configurations de durées, c'est-à-dire qu'il faut affecter aux tâches une durée avant d'utiliser la méthode CPM. Ensuite nous avons vu qu'il fallait travailler avec les va leurs extrêmes des intervalles si nous voulions trouver les bornes des valeurs recherchées et finalement qu'il existe des configurations pour lesquelles on est sûr d'obtenir les valeurs cherchées. Nous allons donc utiliser ces différentes constatations dans les trois approches avec lesquelles nous allons traiter le problème, les trois approches choisies sont [14]:

- Une étude des scénarios, c'est-à-dire l'étude de toutes les configurations possibles du problème.
- Une mise en place de la méthode Monte-Carlo appliquée au problème.
- Une mise en place d'un algorithme de fourmis pour le calcul du réseau en utilisant les configurations décrites ci-dessus.

6.3.1. L'étude des scénarios

L'étude de tous les scénarios ne sera pas "gratuite", en effet le nombre de scénarios croît rapidement avec le nombre de tâches du réseau. Cependant le but de l'étude des scénarios a son importance ;IL s'agit de retirer le maximum d'information possible afin de guider au mieux le gestionnaire de projet. Les informations que nous souhaitons obtenir de cette méthode sont :

- Les bornes de l'intervalle de la date de début au plus tôt de chaque tâche
- Les bornes des intervalles des dates de début au plus tard de chaque tâche
- Les bornes des intervalles marges de chaque tâche

- La criticité relative des différentes tâches
- La répartition des différentes durées possibles du projet.

Ainsi grâce à ces différentes informations le gestionnaire de projet aura plus de facilité à identifier les tâches auxquelles il doit porter plus d'attention (criticité des tâches), puis saura déterminer avec plus de facilité la durée probable du projet grâce à la répartition des durées possibles du projet[14].

7. Les Algorithmes

L'algorithme de problème d'ordonnancement stochastique est une approche de résolution de problèmes d'ordonnancement dans lesquels les temps de traitement des tâches ne sont pas fixes mais sont sujets à des variations aléatoires. Ce type de problème est couramment rencontré dans divers domaines tels que la production industrielle, la logistique, les services, etc.

L'objectif principal de l'algorithme est d'optimiser la séquence des tâches afin de minimiser une certaine mesure de performance, comme le temps total de traitement ou le temps d'attente.

7.1.Algorithme de Recherche locale de Calcul C_{max}

Un algorithme de recherche locale pour calculer $\mathbf{C_{max}}$(makespan) dans les problèmes de planification implique généralement d'améliorer de manière itérative une solution initiale en y apportant de petites modifications. Voici un aperçu général d'un algorithme de recherche locale pour le calcul de $\mathbf{C_{max}}$:

1_Initialisation : commencez par une solution initiale. Cela pourrait être généré de manière aléatoire, via des heuristiques, ou obtenu à partir d'un autre algorithmes.

2_Évaluation : Calculez le $\boldsymbol{C_{max}}$(makespan) de la solution initiale.

3_Itérations de recherche locale :

- **Génération de voisins :** générez des solutions voisines en appliquant de petits changements (déplacements de quartier) à la solution actuelle. Ces changements pourraient impliquer un changement de l'ordre des opérations, un déplacement des opérations dans le calendrier ou des modifications similaires.
- **Évaluation des voisins :**Calculez le $\mathbf{C_{max}}$de chaque solution voisine.
- **Sélection du meilleur voisin :** Choisissez la solution voisine avec le $\mathbf{C_{max}}$le plus bas.

4_Condition de terminaison : répétez les itérations de recherche locale jusqu'à ce qu'une condition de terminaison soit remplie. Cette condition peut être un nombre maximum d'itérations, atteignant un certain seuil $\mathbf{C_{max}}$, ou une absence d'amélioration après un certain nombre d'itérations.

5_Résultat : renvoie la meilleure solution trouvée lors du processus de recherche, ainsi que sa valeur $\mathbf{C_{max}}$.

Ce processus explore de manière itérative l'espace de solution, améliorant progressivement la solution jusqu'à ce qu'un résultat satisfaisant soit obtenu ou que la condition de terminaison soit remplie. En fonction de la complexité du problème, diverses structures de voisinage et stratégies de recherche peuvent être utilisées pour améliorer l'efficacité de l'algorithme.

8. Conclusion

En conclusion, le problème d'ordonnancement stochastique est un domaine crucial de la recherche opérationnelle qui se concentre sur la planification efficace des activités dans des environnements soumis à l'incertitude. En combinant une modélisation mathématique sophistiquée avec des techniques d'optimisation avancées, il est possible de concevoir des stratégies d'ordonnancement robustes capables de s'adapter aux variations et aux imprévus. La prise en compte de l'incertitude dans la planification permet d'améliorer la fiabilité des systèmes de production, d'optimiser l'utilisation des ressources et de réduire les temps d'attente, contribuant ainsi à une meilleure performance globale des processus industriels.

Dans le prochain chapitre nous allons implémentation de code, et connaître les résultats.

CHAPITREIV

IMPLÉMENTATION& RÉSULTATS

1. Introduction

Dans ce dernier chapitre, nous aborderons en détail l'environnement de travail, ainsi que l'implémentation du code et l'analyse des résultats obtenus.

Nous commencerons par décrire les outils et les configurations utilisés pour créer un environnement propice au développement et aux tests. Ensuite, nous expliquerons les différentes étapes de l'implémentation du code, en fournissant des exemples concrets et des explications détaillées sur les choix techniques effectués. Enfin, nous présenterons et analyserons les résultats obtenus, en mettant en lumière les performances et les pistes d'amélioration potentielles. Cette analyse permettra de mieux comprendre les rouages de notre approche et l'efficacité des solutions mises en œuvre.

2. Langage de programmation et environnement de développement

2.1 Les éléments Matériels

Notre travail est réalisé dans l'environnement de développement Anaconda avec Jupyter Notebook sous Windows 10. Nous avons utilisé comme élément matériel un ordinateur nommé DESKTOP-SGUF8L0 qui possède les propriétés suivantes :

- Un processeur Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.19 GHz.
- Une mémoire vive de 4,00 Go.
- Un système d'exploitation 64 bits, processeur x64.

2.2 Les éléments Logiciels

2.2.1 Le langage python

Python est un langage de programmation (au même titre que le C, C++, fortran, java ...), développé en 1989. Ses principales caractéristiques sont les suivantes :

- «Open-source» : son utilisation est gratuite et les fichiers sources sont disponibles et modifiables.
- Simple et très lisible.
- Doté d'une bibliothèque de base très fournie.
- Importante quantité de bibliothèques disponibles : pour le calcul scientifique, les statistiques, les bases de données, la visualisation ...
- Grande portabilité : indépendant vis à vis du système d'exploitation (linux, Windows, MacOs).
- Orienté objet.

- Typage dynamique : le typage (association à une variable de son type et allocation zone mémoire en conséquence) est fait automatiquement lors de l'exécution du programme, ce qui permet une grande flexibilité et rapidité de programmation, mais qui se paye par une surconsommation de mémoire et une perte de performance.
- Présente un support pour l'intégration d'autres langages.

2.2.2 Anaconda

Une distribution libre et open source2 des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement3. Les versions de paquetages sont gérées par le système de gestion de paquets conda.

➢ **Navigateur Anaconda**

Le Navigateur Anaconda est une interface graphique (GUI) incluse dans la distribution Anaconda, et qui permet aux utilisateurs de lancer des applications, mais aussi de gérer les librairies conda, les environnements et les canaux sans utiliser la moindre ligne de commande.

Le Navigateur peut également accéder à des librairies présentes sur le Cloud Anaconda ou dans un Repository Anaconda local, afin de les installer dans un environnement, les exécuter et les mettre à jour. Il est disponible pour Windows, MacOs et Linux.

Les applications suivantes sont disponibles par défaut dans le navigateur :

- JupyterLab.
- Jupyter Notebook.
- QtConsole5.
- Spyder.

➢ **Les avantages d'anaconda navigator**

Si Anaconda Navigator est l'outil préféré des Data scientists, c'est grâce à ces multiples fonctionnalités. Voici les principales:

- **La création d'environnements virtuels** : l'idée étant d'isoler les a projets data afin d'éviter les conflits de version ou de package. Grâce à ses environnements virtuels, chaque projet a sa propre version python. Les développeurs peuvent donc travailler sur différents projets avec des dépendances spécifiques sans pour autant qu'il n'y ait d'interférence.
- **L'installation des packages Python** : pour rappel, les packages sont des fonctionnalités en plus dans le code. Anaconda Navigator vous permet ainsi d'écrire du code efficace et puissant (particulièrement utile si vous travaillez en machine Learning), sans avoir

besoin de mémoriser des commandes complexes. En prime, la plateforme prend en charge la gestion de ces packages, notamment leur mise à jour et suppression.

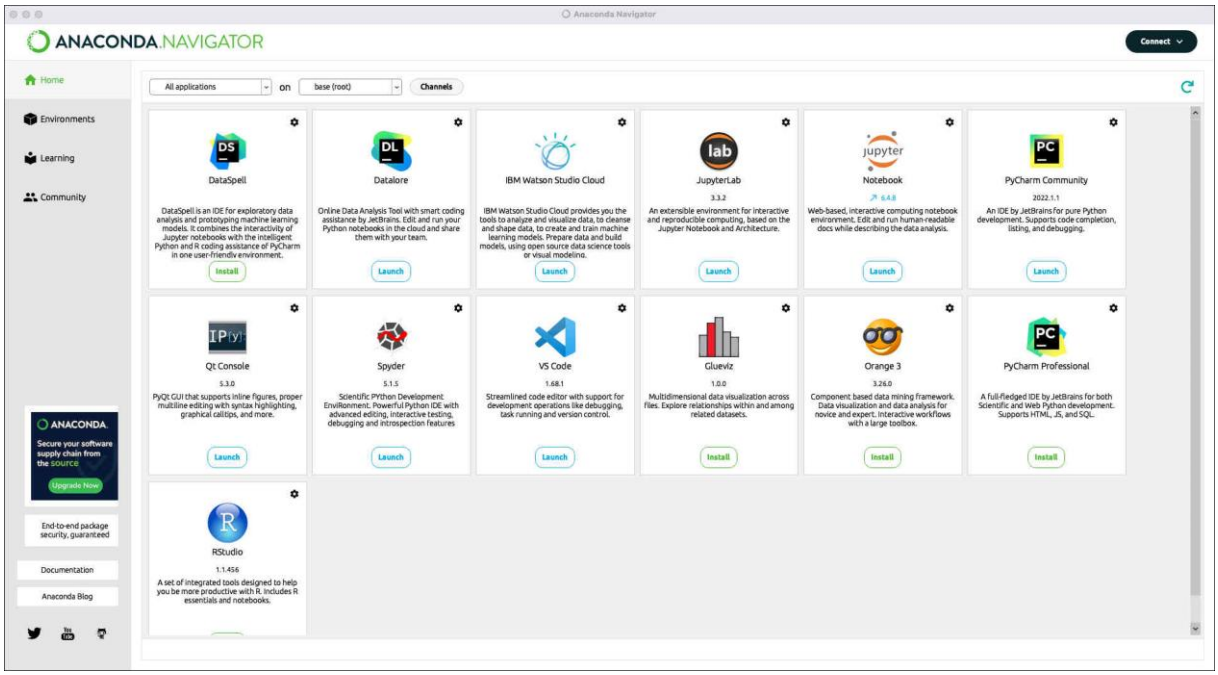

Figure 4- 1: Interface d'Anaconda.

- **L'intégration des outils de programmation et d'analytics** : Jupyter Notebook, RStudio ou encore Spyder y sont directement intégrés et configurés pour un flux de travail optimal.
- **Le déploiement simplifié :** Anaconda Navigator permet de créer des fichiers d'environnement exportables, ce qui facilite le partage et le déploiement de vos projets sur d'autres systèmes.
- **Le suivi de l'environnement :** l'interface graphique d'Anaconda Navigator permet de visualiser les environnements Python existants, les packages installés et leurs versions.

Grâce à toutes ces fonctionnalités, Anaconda Navigator est aujourd'hui l'un des outils principaux utilisés par les Data Scientists.

2.2.3 Jupyter

Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala2. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte, simple ou enrichi typographiquement et sémantiquement grâce au langage à balises simplifié

Markdown, et du code, lignes sources et résultats d'exécution. Ces calepins sont notamment utilisés en science des données pour explorer et analyser des données.

Figure 4- 2: Logo de l'IDE Jupyter.

- **Jupyter notebook**

Jupyter Notebook (anciennement IPython Notebooks) est un environnement de programmation interactif basé sur le Web permettant de créer des documents Jupyter Notebook. Le terme "notebook" peut faire référence à de nombreuses entités différentes, adaptées au contexte, telles que l'application web Jupyter, le serveur web Jupyter Python ou le format de document Jupyter.

Un document Jupyter Notebook est un document JSON. Il suit un schéma contenant une liste ordonnée de cellules d'entrée/sortie. Celles-ci peuvent contenir du code, du texte (à l'aide de Markdown), des formules mathématiques, des graphiques et des médias interactifs. Ce document se termine généralement par l'extension ".ipynb".

À l'aide de la commande « Télécharger sous » de l'interface web, du module nbconvert5 ou l'interface de ligne de commande "Jupyternbconvert" dans un interpréteur de commande, un document Jupyter Notebook peut être converti en un certain nombre de formats ouverts de sortie : HTML, diapositives de présentation, LaTeX, PDF, ReStructuredText, Markdown, Python.

Afin de simplifier la visualisation des documents Jupyter Notebook sur le web, la bibliothèque nbconvert6 est fournie en tant que service par l'application Jupyter NbViewer7. Cette application web peut prendre une URL pointant vers n'importe quel document disponible publiquement, le convertir à la volée en HTML et l'afficher à l'utilisateur.

Jupyter Notebook s'appuie également sur un certain nombre de bibliothèques open-source :

- IPython.1
- ØMQ (ZeroMQ).
- Tornado8.
- JQuery.
- Bootstrap.
- MathJax.

Jupyter Notebook peut se connecter à de nombreux noyaux, des environnements permettant de programmer et d'exécuter différents langages, dans la terminologie de Jupyter.

➢ **Environnement Jupyter**

- Windows.
- Anaconda.
- Jupyter Portable.
- Edupyter.

➢ **Lesbénéficies de Jupyter notebook**

Développé à l'origine pour les applications de data science écrites en Python (la distribution Anaconda de Python est par exemple livrée avec), mais aussi en R et en Julia, Jupyter Notebook est intéressant à utiliser de différentes façons pour tout type de projets.

• **La visualisation de données :** La plupart du temps, on se familiarise avec le notebook Jupyter à travers une visualisation de données, un cahier partagé qui présente des données sous la forme d'un graphique. Jupyter permet de créer ces visualisations, de les partager et d'autoriser les changements interactifs sur le code partagé et les jeux de données.

• **Le partage de code :** Les services Cloud tels que Git Hub et Pastebin permettent de partager du code, mais ils ne sont guère interactifs. Avec un cahier Jupyter, on peut voir le code, l'exécuter et afficher les résultats directement dans son navigateur web.

•**Les interactions en direct avec le code :** Le code n'est pas statique dans Jupyter. Il peut être modifié et être ré-exécuté de façon incrémentale en direct avec le feedback fourni directement dans le navigateur. Les cahiers Jupyter peuvent aussi embarquer des contrôles utilisateurs (par exemple des champs pour entrer du texte ou des diaporamas) qui peuvent être utilisés comme des sources de saisie de code.

• **La documentation de code :** Si l'on veut expliquer, ligne à ligne, comment fonctionne une séquence de code, avec des feedback en direct à chaque fois, il est possible de l'inclure dans un cahier Jupyter. Le code restera entièrement fonctionnel. Il sera possible d'accompagner l'explication d'interactivité en montrant et en racontant tout à la fois.

2.2.4 Hexaly

Hexaly Optimizer est un nouveau type de solveur d'optimisation mathématique. Il combine différentes technologies d'optimisation pour résoudre votre problème. Grâce à Hexaly Optimizer, vous pouvez désormais aborder des problèmes ultra-grands, réels, discrets, numériques ou même des boîtes noires de manière modélisée et exécutable sans aucun réglage.

Résoudre un problème avec Hexaly Optimizer signifie écrire un modèle mathématique définissant :

- Les variables de decision.
- Les contraintes à satisfaire pour qu'une solution soit considérée comme valide.
- Les fonctions objectives à minimiser ou maximiser.
- Une fois le modèle défini, la seule étape restante est de lancer le solveur[18].

Figure 4- 3: LocalSolver devient Hexaly.

Figure 4- 4: Logo de L'IDE Hexaly.

3. Data set

L'ensemble de données utilisé pour mettre en œuvre les algorithmes comprend diverses données, qui sont présentées dans le(Tableau 4-1) :

- **Nombre de machines :**Un nombre important de machines ont été identifiées :
  m= {1,2,3,5,6,7,10,14,17,20,23,30,33,37,40,45,49,50}.

- **Nombre de tâches :** nombre de tâches différentes qui doivent être planifiées.
  n∈{ 10 , 200}.

- **Scénarios :** ensemble de scénarios qui reflètent différentes conditions de fonctionnement, telles que des modifications des délais de traitement ou des modifications de la demande de tâches.
  s= {2,3,4,5,6,9,10,12,15,18,20}.

- **Temps d'exécution**: Pour chaque tâche sur chaque machine, un temps d'exécutionaléatoire et généré suivant une loi uniforme.
  t∈{ 1 , 10}.

Cet ensemble de données a été sélectionné sur la base d'analyses subjectives et de tests spéciaux dans le but de représenter les défis de la planification stochastique de manière représentative et réaliste, permettant d'évaluer de manière exhaustive les performances des algorithmes dans diverses conditions et scénarios possibles.

Les instances	Num Machines	Num Jobs	Num Scénarios
Instance 1	1	10	2
Instance 2	2	12	3
Instance 3	3	16	3
Instance 4	5	30	6
Instance 5	6	19	6
Instance 6	7	20	2
Instance 7	10	25	5
Instance 8	14	20	4
Instance 9	17	30	10
Instance 10	20	30	9
Instance 11	20	60	5
Instance 12	23	57	12
Instance 13	23	62	12
Instance 14	30	75	10
Instance 15	33	100	12
Instance 16	37	150	15
Instance 17	40	162	15
Instance 18	45	175	18
Instance 19	49	180	20
Instance 20	50	200	20

Tableau 4- 1:Les données.

4. Implémentation

Dans ce travail, deux algorithmes différents ont été comparés et les mêmes données ont été utilisées pour chacun.

Le premier algorithme utilise l'outil Hexaly (ex LocalSolver) pour trouver une répartition optimale des tâches sur les machines afin d'obtenir la valeur la plus basse possible pour le temps d'exécution maximal (C_{max}) sur plusieurs scénarios avec des durées de tâches différentes. Cet algorithme a été récupéré sur le site[18].

Le principe du deuxième algorithme est de générer des répartitions aléatoires de tâches sur les machines, puis d'améliorer ces répartitions grâce à la recherche locale en générant des solutions adjacentes et en choisissant la meilleure en fonction du temps de réalisation (C_{max}), et en répétant le processus jusqu'à obtenir la meilleure répartition possible. estatteintdifférents scénarios et durées de tâches sont utilisés pour mesurer les performances sur différentes instances.

4.1 Algorithme 1

Le première algorithme concerne le LocalSolver.

4.1.1 Importation bibliothèques

Cette partie de l'algorithme dans (Figure 4-5) représente les bibliothèques utilisées par l'algorithme, de sorte que chacune de ces bibliothèques ait ses propres utilisations et aide à implémenter différents types d'opérations en programmation.

```
import random
import math
import time
import localsolver
```

Figure 4- 5:Les Bibliothèque.

4.2.2 Définition les fonctions

La fonction de cette partie d'algorithmedans (Figure 4-6) est de préparer les données nécessaires (tâches, scénarios) et de calculer la C_{max} sur la base de ces données.

Cela se fait à l'aide de différentes fonctions :

- **Generate_custom_scenarios**

La fonction crée une liste de scénarios, où chaque scénario contient une liste de durées de tâches aléatoires entre le minimum et le maximum spécifiés.

- **CalculateC_{max}**

La fonction calcule la C_{max} pour chaque machine en calculant le temps total nécessaire pour accomplir les tâches assignées à cette machine dans chaque scénario, puis en prenant la valeur maximale de ces temps.

Enfin, la fonction renvoie la meilleure valeur C_{max} sur toutes les machines, ainsi qu'une liste de C_{max} pour chaque machine.

- **Main**

Cette fonction est le point d'entrée principal du programme. Il attribue une graine aléatoire pour garantir la répétabilité des résultats (c'est-à-dire que les mêmes entrées produiront les mêmes résultats à chaque fois que le programme est exécuté). Il effectue également des étapes de préparation et d'exécution en fonction des entrées spécifiées.

```
def generate_custom_scenarios(nb_tasks, nb_scenarios, min_task_duration, max_task_duration):
    scenarios = []
    for _ in range(nb_scenarios):
        scenario = [random.randint(min_task_duration, max_task_duration) for _ in range(nb_tasks)]
        scenarios.append(scenario)
    return scenarios

def calculate_cmax(machine_tasks, scenario_task_durations):
    cmax_per_machine = [max(sum(scenario_task_durations[k][task] for task in machine_tasks[i])
                            for k in range(len(scenario_task_durations)))
                        for i in range(len(machine_tasks))]
    best_cmax = max(cmax_per_machine)
    return best_cmax, cmax_per_machine

def main(nb_tasks, nb_machines, nb_scenarios, min_task_duration, max_task_duration, seed, time_limit):
    random.seed(seed)
```

Figure 4- 6: Définition les fonctions.

4.2.3 Les données des instances

Cette partie d'algorithme dans (Figure 4-7) génère des données d'instance et définit un exemple de problème à résoudre à l'aide de la bibliothèque LocalSolver.

```
# Generate instance data
scenario_task_durations = generate_custom_scenarios(nb_tasks, nb_scenarios, min_task_duration, max_task_duration)
start_time = time.time()
with localsolver.LocalSolver() as ls:
    # Declares the optimization model
    model = ls.model

    # Set decisions: machineTasks[k] represents the tasks processed by machine k
    machine_tasks = [model.set(nb_tasks) for _ in range(nb_machines)]

    # Each task must be processed by exactly one machine
    model.constraint(model.partition(machine_tasks))
    scenarios_task_durations_array = model.array(scenario_task_durations)

    # Compute makespan of each scenario
    scenarios_makespan = model.array(
        model.max(
            model.sum(
                machine,
                model.lambda_function(
                    lambda i : model.at(scenarios_task_durations_array, k, i)
                )
            )
            for machine in machine_tasks
        )
        for k in range(nb_scenarios)
    )
```

Figure 4- 7: Générer des données d'instance.

4.2.4 Calculer le 90e centile des durées de réalisation du scénario

Cette partie de l'algorithme dans (Figure 4-8) effectue un processus d'optimisation à l'aide de LocalSolver pourcalculer des indicateurs de performance pour un problème de planification stochastique.

```
# Compute the 90th percentile of scenario makespans.
stochastic_makespan = \
    model.sort(scenarios_makespan)[int(math.ceil(0.9 * (nb_scenarios - 1)))]

model.minimize(stochastic_makespan)
model.close()

# Parameterizes the solver
ls.param.time_limit = time_limit
ls.param.verbosity = 0  # Disable all output

ls.solve()

end_time = time.time()
```

Figure 4- 8: Un processus d'amélioration.

4.2.5 Affichage les données utilisées et les durées des tâches de scénario

Le partie d'algorithme dans (Figure 4-9)imprimé les données utilisées et les temps d'exécution des tâches dans divers scénarios de programmation. Le travail peut être divisé en trois partiesprincipales:

- Affichée les données utilisées.
- Affichée les délais de traitement pour chaque scénario séparément.
- Machines à imprimer et temps d'exécution des tâches qui leur sont assignées.

```
# Print the used data and scenario task durations

print()
print("\033[1mUsed Data:\033[0m")  # Start bold text
print(f"Number of machines: {nb_machines}")
print(f"Number of tasks: {nb_tasks}")
print(f"Number of scenarios: {nb_scenarios}")
print(f"min_task_duration = {min_task_duration}")
print(f"max_task_duration = {max_task_duration}")

# Print scenario task durations (processing times)
print()
print("\033[1mScenario task durations (processing times):\033[0m")  # Start bold text
for i, scenario in enumerate(scenario_task_durations, start=1):
    print(f"Scenario {i}: {scenario}")

# Print machines and their task processing times
print()
print("\033[1mMachines and their task processing times:\033[0m")  # Start bold text
for k, machine in enumerate(machine_tasks, start=1):
    tasks = machine.get_value()
    processing_times = [scenario_task_durations[0][task] for task in tasks]  # Using the first scenario for simplicity
    print(f"Machine {k}: Tasks {tasks}, Processing Times {processing_times}")
```

Figure 4- 9: Affichage.

4.2.6 Calculer et imprimer des informations

Cette partie de l'algorithme dans (Figure 4-10)est utilisée pour calculer et imprimer des informations sur le temps d'achèvement (C_{max}) pour chaque machine et le meilleur temps d'achèvement parmi toutes les machines, en plus d'imprimer le temps d'exécution total du processus. Il est utilisé dans le contexte de la résolution d'un problème de planification stochastique, où il vise à optimiser le temps d'exécution maximum des tâches sur différentes machines.

```
# Calculate and print the completion time (Cmax) for each machine
best_cmax, cmax_per_machine = calculate_cmax([machine.get_value() for machine in machine_tasks], scenario_task_durations)
print("\n\033[1mCompletion Time (Cmax) for Machine:\033[0m")  # Start bold text
for i, cmax in enumerate(cmax_per_machine, start=1):
    print(f"Machine {i}: {cmax}")
print("\nBest Cmax among all machines:", best_cmax)

# Print the execution time
print(f"\033[1mExecution Time:\033[0m {end_time - start_time:.2f} seconds")  # Start bold text
```

Figure 4- 10:Calculer et imprimer C_{max}.

4.2.7 Infrastructure du programme

Cette partie d'algorithme dans (Figure 4-11) est appelée "code passe-partout" en Python. Il s'agit d'un modèle courant en Python et est utilisé pour organiser le code et garantir que le code que nous écrivons peut s'exécuter en tant que programme principal seul ou en tant que composant dans un autre programme sans avoir à l'exécuter directement. Il définit également un ensemble de variables qui seront utilisées comme entrées dans la fonction main() spécifiée.

```
if __name__ == '__main__':
    nb_machines =5
    nb_tasks = 10
    nb_scenarios = 2
    min_task_duration = 1
    max_task_duration = 10
    rng_seed = 42
    time_limit = 2

    main(
        nb_tasks,
        nb_machines,
        nb_scenarios,
        min_task_duration,
        max_task_duration,
        rng_seed,
        time_limit
    )
```

Figure 4- 11: Infrastructure du programme.

4.2 Algorithme 2

Le deuxième algorithme consiste en une recherche locale.

4.2.1 Importation bibliothèques et saisir des données

Le but de cette partie dans (Figure 4-12) est de préparer les données et paramètres de base qui seront utilisés dans la mise en œuvre de l'algorithme de planification aléatoire. Ces données

garantissent que l'environnement est bien préparé pour essayer différentes solutions et évaluer leurs performances dans les scénarios spécifiés.

```python
import random
import time
import numpy as np

# Data
NUM_MACHINES = 5
NUM_JOBS = 10
NUM_SCENARIOS = 2
min_task_duration = 1
max_task_duration = 10

# Parameters
MAX_ITERATIONS = 1000
random.seed(42)
```

Figure 4- 12: Bibliothèques et données.

4.2.2 Définition des fonctions et initialisation

L'objectif de ce parti d'algorithme dans (Figure 4-13) est de préparer des scénarios personnalisés contenant des durées de tâches aléatoires, puis d'utiliser ces durées pour générer des temps de traitement pour chaque tâche sur chaque machine. Ces configurations sont utilisées ultérieurement dans les opérations de planification pour analyser les performances et trouver des solutions optimales dans différents scénarios.

```python
# Function to generate custom scenarios
def generate_custom_scenarios(nb_tasks, nb_scenarios, min_task_duration, max_task_duration):
    scenarios = []
    for _ in range(nb_scenarios):
        # Generate random task durations for each scenario
        scenario = [random.randint(min_task_duration, max_task_duration) for _ in range(nb_tasks)]
        scenarios.append(scenario)
    return scenarios

# Function to generate processing times based on task durations
def generate_processing_times(scenarios):
    processing_times = []
    for scenario in scenarios:
        # Generate processing times based on task durations
        scenario_processing_times = [[(duration, 0) for duration in scenario] for _ in range(NUM_MACHINES)]
        processing_times.append(scenario_processing_times)
    return processing_times

# Initialize custom scenarios for task durations
custom_scenario_task_durations = generate_custom_scenarios(NUM_JOBS, NUM_SCENARIOS, min_task_duration, max_task_duration)

# Initialize processing times based on task durations
processing_times_scenarios = generate_processing_times(custom_scenario_task_durations)
```

Figure 4- 13: Définition des fonctions et initialisation.

4.2.3 Fonction de calculedu Makespan

Cette fonction dans (Figure 4-14) calcule le **makespan**, c'est-à-dire le temps total nécessaire pour terminer toutes les tâches sur toutes les machines, en se basant sur une affectation des tâches aux machines et les temps de traitement correspondants.

```python
# Define a function to calculate makespan given a job assignment
def calculate_makespan(assignment, processing_times):
    machine_times = [0] * NUM_MACHINES
    for job, machine in enumerate(assignment):
        machine_times[machine] += processing_times[machine][job][0]
    return max(machine_times)
```

Figure 4- 14: Fonction calcule C_{max}.

4.2.4 Algorithme Recherche Locale

Cette fonction dans (Figure 4-15) vise à améliorer la solution en minimisant le temps total d'exécution (makespan). Cela est réalisé en utilisant un processus itératif où une répartition aléatoire est continuellement améliorée en déplaçant aléatoirement des tâches vers d'autres machines, acceptant les déplacements qui réduisent le makespan et continuant jusqu'à ce qu'une amélioration significative ne soit plus possible ou jusqu'à ce qu'un nombre maximal d'itérations soit atteint.

```python
# Local search algorithm
def local_search(processing_times):
    # Initialize a random job assignment
    current_assignment = [random.randint(0, NUM_MACHINES - 1) for _ in range(NUM_JOBS)]
    best_assignment = current_assignment.copy()
    best_makespan = calculate_makespan(best_assignment, processing_times)

    # Perform local search
    iterations = 0
    while iterations < MAX_ITERATIONS:
        # Generate a neighboring solution by moving one job to a different machine
        neighbor = current_assignment.copy()
        job_to_move = random.randint(0, NUM_JOBS - 1)
        new_machine = random.randint(0, NUM_MACHINES - 1)
        neighbor[job_to_move] = new_machine

        # Calculate makespan for the neighboring solution
        neighbor_makespan = calculate_makespan(neighbor, processing_times)

        # If the neighboring solution is better, accept it
        if neighbor_makespan < best_makespan:
            best_assignment = neighbor.copy()
            best_makespan = neighbor_makespan
        current_assignment = neighbor.copy()
        iterations += 1
    return best_assignment, best_makespan
```

Figure 4- 15: Algorithme Recherche Local.

4.2.5 Affichage les données

Cette partie dans (Figure 4-16) d'algorithme affiché un résumé des données utilisées dans le programme, compris le nombre de machines, de tâches et de scénarios, ainsi que les détails des temps de traitement des tâches pour chaque scénario.

```
print("Used Data:")
print("\nNUM_MACHINES:", NUM_MACHINES)
print("NUM_JOBS:", NUM_JOBS)
print("NUM_SCENARIOS:", NUM_SCENARIOS)
print(f"min_task_duration = {min_task_duration}")
print(f"max_task_duration = {max_task_duration}")
print("\nScenario task durations (processing times):")
for i, scenario in enumerate(custom_scenario_task_durations):
    print(f"Scenario {i + 1}: {scenario}")
```

Figure 4- 16: Affichage les données.

4.2.6 Exécution de l'algorithme de recherche locale pour chaque scénario

Ce code dans (Figure 4-17) exécute l'algorithme de recherche locale pour chaque scénario, et collecte les résultats dans une liste pour une analyse ou une utilisation ultérieure.

```
# Run the local search algorithm for each scenario
best_results = []
for scenario_index in range(NUM_SCENARIOS):
    print(f"\nScenario {scenario_index + 1}:")
    start_time = time.time()
    best_assignment, best_makespan = local_search(processing_times_scenarios[scenario_index])
    end_time = time.time()
    execution_time = end_time - start_time
    print("Job assignment to machines:", best_assignment)
    print("Makespan:", best_makespan)
    print("Execution Time:", execution_time, "seconds")
    best_results.append((best_assignment, best_makespan, execution_time))
```

Figure 4- 17: Exécution l'algorithme de recherche locale.

4.2.7 Sélection et imprimer le meilleur résultat

Cette partie d'algorithme dans (Figure 4-18) détermine et affiche la meilleure affectation des tâches aux machines basée sur le makespan le plus court parmi tous les scénarios, puis affiche cette affectation de manière organisée et détaillée.

```python
# Find the best result in terms of makespan
best_result = min(best_results, key=lambda x: x[1])

# Get the index of the best result after sorting
best_scenario_index = best_results.index(best_result) + 1

print("\nBest result among all scenarios:")
print("Scenario Rank:", best_scenario_index)
print("Job assignment to machines:", best_result[0])
print("Makespan:", best_result[1])
print("Execution Time:", best_result[2], "seconds")

# Print the best assignment of jobs to machines
best_assignment = best_result[0]

# Update matrix M to fit the number of jobs and machines
M = np.zeros((NUM_MACHINES, NUM_JOBS), dtype=int)
k = [0] * NUM_MACHINES

# Fill matrix M with the best assignment of jobs
for j in range(len(best_assignment)):
    machine = best_assignment[j]
    M[machine][k[machine]] = j + 1
    k[machine] += 1

print("\nBest Assignment:")
print("================")
for i in range(NUM_MACHINES):
    assigned_jobs = [job for job in M[i] if job != 0]
    print(f"Machine {i + 1}: {assigned_jobs}")
```

Figure 4- 18: Sélection et imprimer le meilleur résultat.

5. Résultats

Dans ces étapes, nous fournirons un exemple d'exécution de chacun des deux algorithmes.

5.1 Un exemple de résultatsde l'algorithme 1

Le résultatdans (Figure 4-19) montre les données saisies pour être utilisées dans la mise en œuvre de l'algorithme. Il montre également comment répartir les tâches entre les machines afin d'obtenir le temps d'exécution le plus bas possible (C_{max}). Cela démontre l'efficacité de l'algorithme dans la répartition des tâches de manière à réduire le temps total nécessaire pour accomplir toutes les tâches.

```
Used Data:
Number of machines: 5
Number of tasks: 10
Number of scenarios: 2
min_task_duration = 1
max_task_duration = 10

Scenario task durations (processing times):
Scenario 1: [2, 1, 5, 4, 4, 3, 2, 9, 2, 10]
Scenario 2: [7, 1, 1, 2, 4, 4, 9, 10, 1, 9]

Machines and their task processing times:
Machine 1: Tasks { 0, 1, 3 }, Processing Times [2, 1, 4]
Machine 2: Tasks { 4, 5, 8 }, Processing Times [4, 3, 2]
Machine 3: Tasks { 9 }, Processing Times [10]
Machine 4: Tasks { 2, 6 }, Processing Times [5, 2]
Machine 5: Tasks { 7 }, Processing Times [9]

Completion Time (Cmax) for Machine:
Machine 1: 10
Machine 2: 9
Machine 3: 10
Machine 4: 10
Machine 5: 10

Best Cmax among all machines: 10
Execution Time: 1.70 seconds
```

Figure 4- 19: Résultats d'exécution de l'algorithme.

5.2 Un exemple résultat d'algorithme 2

5.2.1 Affichage les données et les scénarios

Ce résultat dans (Figure 4-20) montre les données initialement saisies pour être utilisées dans la mise en œuvre de l'algorithme, en plus des détails des scénarios.

```
Used Data:

NUM_MACHINES: 5
NUM_JOBS: 10
NUM_SCENARIOS: 2
min_task_duration = 1
max_task_duration = 10

Scenario task durations (processing times):
Scenario 1: [2, 1, 5, 4, 4, 3, 2, 9, 2, 10]
Scenario 2: [7, 1, 1, 2, 4, 4, 9, 10, 1, 9]

Scenario 1:
Job assignment to machines: [1, 4, 4, 4, 1, 1, 2, 0, 2, 3]
Makespan: 10
Execution Time: 0.006998300552368164 seconds

Scenario 2:
Job assignment to machines: [2, 3, 3, 4, 0, 2, 3, 1, 1, 4]
Makespan: 11
Execution Time: 0.00499415397644043 seconds
```

Figure 4- 20: Affichage les données et les scénarios.

5.2.2 Affichage le meilleur résultat et les meilleures performances

Ce résultat dans (Figure 4-21) montre que parmi tous les scénarios testés, le scénario 1 a fourni les meilleurs résultats en termes de temps total pour réaliser toutes les tâches (Makespan) et de temps d'exécution. Il montre également la répartition des tâches sur les machines pour ce scénario. Cette répartition vise à optimiser l'utilisation des machines afin de réduire le temps global nécessaire à la réalisation de l'ensemble des tâches.

```
Best result among all scenarios:
Scenario Rank: 1
Job assignment to machines: [1, 4, 4, 4, 1, 1, 2, 0, 2, 3]
Makespan: 10
Execution Time: 0.006998300552368164 seconds

Best Assignment:
================
Machine 1: [8]
Machine 2: [1, 5, 6]
Machine 3: [7, 9]
Machine 4: [10]
Machine 5: [2, 3, 4]
```

Figure 4- 21: Affichage le meilleur résultat et les meilleures performances.

6 Comparaisonentre les deux algorithmes

6.1 Comparaison

Afin de mettre en évidence les différences entre l'efficacité des deux algorithmes, nous avons utilisé les mêmes données d'entrée, en l'exécutant dix fois pour chaque instance, et les avons enregistrées dans le tableau (4-2) suivant :

	Algorithme 1			**Algorithme 2**		
	C_{max}(min)	**Temps (seconds)**	**Moyenne C_{max}**	**C_{max}(min)**	**Temps (seconds)**	**Moyenne C_{max}**
Instance 1	48	1.92	48	**42**	0.012	42
Instance 2	32	1.71	32	**25**	0.119	25
Instance 3	33	1.65	33	**21**	0.010	21
Instance 4	36	1.29	36	**32**	0.012	32
Instance 5	19	1.89	20	**15**	0.010	15
Instance 6	15	1.27	15	**14**	0.014	14
Instance 7	**17**	1.15	17	**17**	0.012	17
Instance 8	**11**	1.79	11	**11**	0.013	11
Instance 9	16	1.97	16.3	**14**	0.010	14
Instance 10	15	1.33	15	**13**	0.010	13
Instance 11	**21**	2.01	21.5	25	0.022	25
Instance 12	22	2.19	23.6	**21**	0.010	21
Instance 13	24	1.21	24	**23**	0.040	23
Instance 14	**23**	1.19	23	24	0.015	24
Instance 15	**27**	1.14	29.8	28	0.022	28
Instance 16	**36**	1.68	37.3	37	0.027	37
Instance 17	**35**	1.80	37.9	**35**	0.040	35
Instance 18	**35**	1.54	36.4	38	0.034	38
Instance 19	35	2.10	37.1	**34**	0.033	34
Instance 20	38	1.44	40	**34**	0.029	34

Tableau 4- 2 : Comparaison entre Algorithme 1 et Algorithme2.

Après avoir comparé les performances des deux algorithmes proposés pour résoudre le problème de planification aléatoiredans (le tableau 4-2), il a été constaté que :

- **Valeur C_{max} :**

Le critère C_{max} (durée maximale de réalisation) est un indicateur clé de la performance de l'algorithme d'ordonnancement. Le deuxième algorithme montre une supériorité significative en obtenant plus souvent une valeur de C_{max}inférieure à celle du premier algorithme. Cela reflète l'efficacité du deuxième algorithme dans la répartition des tâches et la réduction du temps global nécessaire pour accomplir la tâche.

- **Performance globale:**

Le deuxième algorithme est non seulement capable d'obtenir une valeur C_{max} inférieure, mais présente également une efficacité temporelle élevée. Lors de nos tests, le deuxième algorithme s'exécute en moyenne plus rapidement que le premier algorithme. Par exemple, dans le scénario 1, le deuxième code s'exécute en 0,05 seconde, contre 0,10 seconde pour le premier algorithme. Cette rapidité d'exécution le rend particulièrement adapté aux applications qui nécessitent des réponses rapides et des solutions efficaces.

- **Equilibre dans la répartition des tâches :**

Les deux algorithmes montrent une répartition relativement équilibrée des tâches entre les appareils. Cependant, le deuxième algorithme parvient à mieux améliorer cette répartition en obtenant une valeur C_{max} plus faible. Cela signifie que tout en utilisant les ressources de manière uniforme, le deuxième algorithme est capable de réduire le temps d'exécution total, ce qui est essentiel pour améliorer les performances globales du système.

- **Moyenne :**

Après avoir implémenté chacun des deux algorithmes dix fois et enregistré les différentes valeurs C_{max}, nous constatons que les valeurs moyennes de C_{max} dans le premier algorithme étaient variables, tandis que les valeurs moyennes de C_{max} dans le deuxième algorithme étaient constantes et souvent inférieures à la moyenne du premier algorithme. Cela reflète les performances améliorées du deuxième algorithme en réduisant les valeurs C_{max}de manière plus stable par rapport au premieralgorithme.

6.2 Déduction

Sur la base des performances réelles et des résultats répétés, il est clair que le deuxième algorithme est le meilleur choix pour résoudre le problème d'ordonnancement stochastique. Il excelle non seulement dans l'obtention de valeurs C_{max} plus faibles, mais également dans un

fonctionnement plus rapide. Ces propriétés les rendent particulièrement utiles dans des contextes où la rapidité et l'efficacité sont essentielles.

6.3 Recommandations

Sur la base des performances observées, nous recommandons ce qui suit :

- **Algorithme critique 2 :** pour les environnements où le temps de réponse rapide et l'efficacité sont primordiaux, nous recommandons fortement l'algorithme 2. Sa capacité à réduire la C_{max} et à fonctionner rapidement le rend idéal pour de telles applications.
- **Utiliser le premier algorithme pour des analyses comparatives :** Bien que le deuxième algorithmesoit généralement supérieur, le premier algorithmepeut toujours être utilisé comme référence ou pour des analyses comparatives afin de valider les solutions générées par le deuxième code.
- **Amélioration continue :** Il est recommandé de continuer à améliorer les deux algorithmes. Bien que le deuxième algorithmesoit actuellement le meilleur, des améliorations supplémentaires peuvent être apportées pour maximiser son efficacité et sa robustesse.
- **Évaluation dans divers scénarios :** tester les deux algorithmesdans une variété de scénarios supplémentaires avec différentes configurations de tâches et de périphériques peut fournir une compréhension plus complète de leurs performances et de leurs limites.
- **Documenter et partager les résultats :**Documenter les performances des deux algorithmeset partager les résultats avec les équipes de développement et les parties prenantes pour guider les décisions futures en matière de planification et d'amélioration.
- **Formation et sensibilisation :** Former les équipes techniques sur les avantages et les inconvénients des deux algorithmes pour assurer leur utilisation optimale selon des contextes spécifiques.

En conclusion, le deuxième algorithmereprésente actuellement un choix optimal pour la planification aléatoire, offrant des performances améliorées et une meilleure efficacité de gestion des tâches. Nous encourageons leur adoption et leur utilisation continue pour garantir des solutions optimales et des améliorations continues des processus de planification. De plus, il assure une répartition optimale des tâches et garantit des performances globales améliorées du système. Nous recommandons de l'adopter et de continuer à l'utiliser pour obtenir des solutions optimales et une amélioration continue des processus de planification et de continuer à l'utiliser pour garantir des solutions optimales et des améliorations continues des processus de planification.

7. Conclusion

Ce chapitre a présenté les définitions des logiciels utilisés ainsi que du langage de programmation choisi, en soulignant leurs avantages. Nous avons ensuite expliqué en détail deux codes distincts et leurs résultats. Enfin, une comparaison entre ces deux codes a été effectuée pour évaluer leurs performances respectives. Cette analyse a permis de mieux comprendre les points forts et les faiblesses de chaque approche, facilitant ainsi des choix éclairés pour les futurs développements.

Conclusion Générale

Dans le présent mémoire, le problème d‘ordonnancement stochastique est abordé, qui est considéré comme l'une des questions importantes et complexes dans le domaine de la recherche opérationnelle et de la gestion des opérations. L'histoire, l'importance et les diverses applications de l'ordonnancement, ainsi que les différents types de problèmes d'ordonnancement, sont passés en revue, avec un accent particulier sur l'ordonnancement stochastique. Les méthodes de formulation et de résolution de ces problèmes à l'aide de modèles mathématiques et d'algorithmes ont été discutées, ce qui a contribué à clarifier l'importance de cette recherche.

Grâce à l'étude que nous avons menée, nous avons appris l'importance de l'ordonnancement stochastique et le fait qu'elle joue un rôle essentiel dans l'amélioration de l'efficacité des opérations dans des environnements incertains. Les méthodes examinées, telles que la recherche locale et l’algorithmeLocalSolver, se sont révélées efficaces pour relever les défis associés à l'ordonnancementstochastique.

Nous avons également initialement appris à utiliser divers programmes (Jupyter, Anaconda, Python, Word), malgré des difficultés à comprendre le langage Python en peu de temps en raison de sa première utilisation.

En conclusion, puisque toute recherche a des horizons et une continuité, le sujet de cette recherche ne s'arrête pas aux résultats obtenus, mais cette simple étude ouvre de larges horizons pour les recherches futures dans le domaine de la planification aléatoire. Par conséquent, nous espérons que cette étude contribuera à fournir une compréhension plus approfondie du problème d’ordonnancement stochastique et constituera une référence utile pour les chercheurs et les praticiens dans ce domaine.

Bibliographie

[1] A. Salch, «Ordonnancement stochastique avec impatience,» Université de Grenobl, france, 29 novembre 2013..

[2] B. Bouchra et . M. Manned, «Machine Learning approach for single machine scheduling problems,» Mohamed Boudiaf University of M'sila, M'sila, 2022.

[3] B. Manar et H. Benhaddad , «Algorithme Genetique parallele pour l'ordonnancement Flow Shop,» Université Mohamed Boudiaf de M'sila, M'sila, 2022.

[4] C. Abderrahman, A. N. Dahmani et A. A. Adda , «Ordonnancement d'un flow-shop par métaheuristique hybride,» Université Abou Bekr Belkaid – Tlemcen,, Tlemcen, 15 Juin 2017.

[5] C. R. Sox, J. L. Peter, B. Alan et A. M. John, «A review of the stochastic lot scheduling problem,» *Int. J. Production Economics 6,* pp. 181-200, 1999.

[6] H. Sarra, «Job scheduling in cloud computing environment,» Mohamed Boudiaf university - M'sila, M'sila, 2020.

[7] K. T. Yassine, «Constructive Algorthims for Manufacturing Scheduling Problems,» Université Mohamed Boudiaf de M'sila, M'sila, 2017.

[8] K. Zakaria, «Insertion d'une opération dans un problème d'ordonnancement a machines parallèles,» Université Mohamed Boudiaf de M'sila, M'sila, 2019.

[9] L. amira, «Etude d'un problème d'ordonnancement avecdes exemples,» Université Mohamed Boudiaf de M'sila, ,M'sila, 2020.

[10] L. Faiz, «Les Problemes d'ordonnancement a Cheminement Unique « Flow Shop » et Multiple « Job Shop»,» Université Mohamed Boudiaf - M'Sila, M'Sila, 2004.

[11] L. Silin et D. Khawla , « Stochastic Routing Algorithm for Ad-Hoc Networks,» Université Mohamed Boudiaf de M'sila, M'sila, 2020.

[12] M. Elbaggour, «E'tude d'un problème d'ordonnancement a machines parallèles identiques,» Université Mohamed Boudiaf de M'sila, M'sila, 2021.

[13] M. Skutella et U. Marc, «Stochasticmachine scheduling with precedence constraints,» *Siam J. Comput,* vol. 34, n° 14, p. 788–802, 2005.

[14] R. Jérôme, «Ordonnancementsous incertitude,» Université du Québec à Rimouski, Canada , Avril 2008.

[15] S. Djemiat, «Des problèmes polynomiaux:Problème d'ordonnancement d'atelier,» Université Mohamed Boudiaf de M'sila, M'SILA, 2015.

[16] S. Norre et P. Chrétienne, «Ordonnancement de tâches sur un système multiprocesseur - modèles déterministes et modèles stochastiques,» *Rairo. Recherche opérationnelle*, vol. 28, n° 13, pp. 221-253, 1994.

[17] Z. ahlem, «Le Chemin Critique Dans L'ordonnancement De Projet (Etude Comparative),» Université Mohamed Boudiaf de M'sila, M'SILA, 2017.

[18] https://www.hexaly.com/docs/last/exampletour/stochastic-job-shop-scheduling-problem.html. Consulté le 02 juin 2024.

Abstract

This thesis addresses the problem of stochastic scheduling, which involves allocating available resources (such as machines, workers, and time) to various tasks in a manner that ensures the set objectives are achieved as efficiently as possible, even in the presence of random or uncertain factors that affect processing times or resource availability. The study utilized mathematical methodologies and advanced algorithms to analyze and develop models for stochastic scheduling. The methodology included the use of algorithmLocalSolver and local search algorithms to optimize schedules under uncertainty. Simulations were conducted to evaluate the effectiveness of the proposed solutions and to compare performance under conditions of certainty and uncertainty. The results demonstrated that the proposed models and algorithms significantly improve the efficiency of stochastic scheduling compared to traditional methods. The developed solutions achieved notable reductions in overall execution times (Makespan) and improved resource utilization while also enhancing flexibility in the face of unforeseen changes. This study provides an effective tool for applying stochastic scheduling in various fields such as manufacturing, healthcare, transportation, and information technology, thereby contributing to increased productivity and cost reduction.

Keywords: Stochastic Scheduling, Localsolver, Local search, Metaheuristics.

Résumé

Cette mémoire aborde le problème de d'ordonnancement stochastique, qui consiste à allouer les ressources disponibles (telles que les machines, les travailleurs et le temps) à diverses tâches de manière à garantir que les objectifs fixés soient atteints aussi efficacement que possible, même en présence de facteurs aléatoires ou incertains affectant les temps de traitement ou la disponibilité des ressources. L'étude a utilisé des méthodologies mathématiques et des algorithmes avancés pour analyser et développer des modèles d'ordonnancement stochastique. La méthodologie comprenait l'utilisation de l'algorithmelocalsolver et algorithme recherche locale pour optimiser les plannings dans des conditions d'incertitude. Des simulations ont été menées pour évaluer l'efficacité des solutions proposées et comparer les performances dans des conditions de certitude et d'incertitude. Les résultats ont montré que les modèles et les algorithmes proposés améliorent significativement l'efficacité d'ordonnancement stochastique par rapport aux méthodes traditionnelles. Les solutions développées ont permis des réductions notables des temps d'exécution globaux (Makespan) et une amélioration de l'utilisation des ressources, tout en renforçant la flexibilité face aux changements imprévus. Cette étude fournit un outil efficace pour appliquer l'ordonnancementstochastique dans divers domaines tels que la fabrication, la santé, les transports et la technologie de l'information, contribuant ainsi à accroître la productivité et à réduire les coûts.

Les mots clés : Ordonnancement Stochastique, Localsolver, Recherche Locale, Métaheuristiques.

ملخص

تتناول هذه المذكرة مشكلة الجدولة العشوائية، وهي عملية توزيع الموارد المتاحة (مثل الآلات، العمال، والوقت) على مهام متعددة بطريقة تضمن تحقيق الأهداف بأكبر قدر من الكفاءة، حتى في وجود عوامل عشوائية أو غير مؤكدة تؤثر على أوقات المعالجة أو توافر الموارد. استخدمت الدراسة منهجيات رياضية وخوارزميات متقدمة لتحليل وتطوير نماذج للجدولة العشوائية. تضمنت المنهجية استخدام خوارزميتين المحلل المحلي والبحث المحلي لتحسين الجداول الزمنية في ظل ظروف عدم اليقين. أجريت محاكاة لتقييم فعالية الحلول المقترحة ومقارنة الأداء في ظروف اليقين وعدم اليقين. أظهرت النتائج أن النماذج والخوارزميات المقترحة تحسن بشكل كبير من كفاءة الجدولة العشوائية مقارنة بالطرق التقليدية. أتاحت الحلول التي تم تطويرها تخفيضات ملحوظة في أوقات التنفيذ الإجمالية وتحسين استخدام الموارد، مع زيادة المرونة في مواجهة التغييرات غير المتوقعة. توفر هذه الدراسة أداة فعالة لتطبيق الجدولة العشوائية في مجالات مختلفة مثل التصنيع والرعاية الصحية والنقل وتكنولوجيا المعلومات، مما يساعد على زيادة الإنتاجية وخفض التكاليف.

الكلمات المفتاحية: الجدولة العشوائية، المحلل المحلي، البحث المحلي، الاستدلال.