Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement

2020-2021

Etude d'un modèle de chaîne logistique multi-acteurs

Entreprise
Polytech

Tuteur entreprise
Jean-Charles BILLAUT

Étudiant
Yohan DERENNE (DI5)

Tuteurs académiques
Jean-Charles BILLAUT
Yannick KERGOSIEN

3 avril 2021

Liste des intervenants

Entreprise

Polytech
64 avenue Jean Portalis
37200 Tours, France
polytech.univ-tours.fr

Nom	Email	Qualité
Yohan DERENNE	yohan.derenne@etu.univ-tours.fr	Étudiant DI5
Jean-Charles BILLAUT	jean-charles.billaut@univ-tours.fr	Tuteur académique, Département Informatique
Yannick KERGOSIEN	yannick.kergosien@univ-tours.fr	Tuteur académique, Département Informatique
Jean-Charles BILLAUT	jean-charles.billaut@univ-tours.fr	Tuteur entreprise

Avertissement

Ce document a été rédigé par Yohan DERENNE susnommé l'auteur.

L'entreprise Polytech est représentée par Jean-Charles BILLAUT susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Jean-Charles BILLAUT et Yannick KERGOSIEN susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Yohan DERENNE, *Etude d'un modèle de chaîne logistique multi-acteurs:* , Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2020-2021.

```
@mastersthesis{
    author={DERENNE, Yohan},
    title={Etude d'un modèle de chaîne logistique multi-acteurs: },
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université de Tours},
    address={Tours, France},
    year={2020-2021}
}
```

Document mis en forme sous LaTeX

grâce à la classe de document `polytech.cls` V2.5.2 (2019/08/07)

Table des matières

Table des figures

1 Introduction

Dans ce chapitre, nous étudions le contexte et le périmètre de notre projet. Nous verrons aussi les hypothèses ainsi que les bases méthodologiques utilisées pour la gestion du projet.

1 Contexte

Dans cette section nous verrons une description du problème d'ordonnancement à résoudre puis nous verrons dans quels cas réels ce type de problème est susceptible de ressortir.

1.1 Description du problème d'ordonnancement

Nous cherchons à résoudre un problème d'ordonnancement qui concerne la production et la distribution des commandes à des clients. Les clients effectuent leurs commandes auprès du producteur. Le producteur va alors préparer cette commande puis il va ensuite sous-traiter la livraison par une entreprise de logistique tierce (distributeur).

Des pénalités sur les retards peuvent également survenir. En effet, si les délais de livraison sont dépassés, le distributeur devra payer une pénalité au producteur qui peut être sous forme de factures dans un cas réel. Côté producteur, les pénalités de retard seront dues aux clients et peuvent être sous forme de bons d'achats dans un cas réel.

Concernant les véhicules, le producteur se charge des frais liés au nombre de véhicules utilisés pour la livraison. Ces frais seront donc perçus par le distributeur.

Voici le schéma reprenant les principaux échanges entre les différents acteurs :

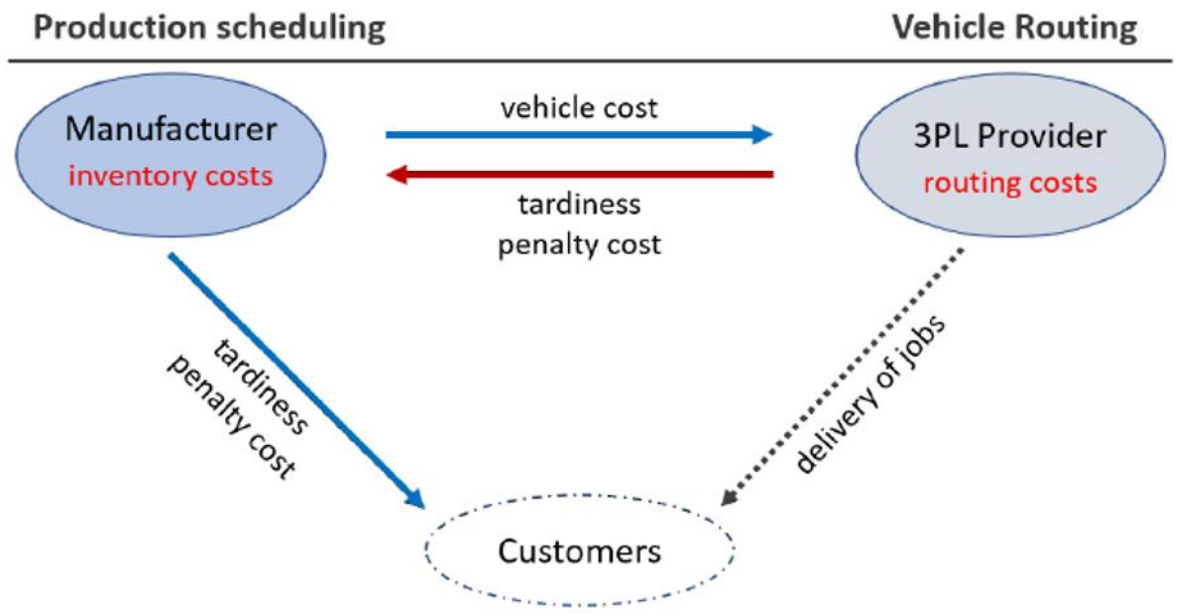


Figure 1.1 – *Acteurs du système*

Les commandes à préparer sont modélisées par des tâches (jobs) et les véhicules sont destinés à livrer un certain nombre de commandes et sont donc modélisés par des lots (batches). Les lots sont donc un ensemble de commandes prêtes à être livrées.

Les principaux coûts à prendre en compte sont les coûts d'inventaire. En effet, il existe les coûts de stockage des tâches en production ainsi que des coûts d'inventaire des tâches prêtes à être livrées.

Il est alors intéressant de pouvoir optimiser cette production de façon à minimiser les coûts d'inventaire lors des productions ainsi que les délais de livraison en fonction du nombre de véhicules disponibles pour la distribution sachant que des pénalités sont ajoutées lors des retards. Dans ce PRD, on s'intéresse plus particulièrement à la partie de l'optimisation de la production en fonction de la demande en distribution.

Ce projet s'inscrit aussi dans le cadre de la poursuite d'une étude réalisée par Jean-Charles Billaut, Sonja Rohmer et Hugo Chevroton intitulé "A framework for production and outbound distribution : manufacturer dominates" [1].

1.2 Enjeux

Ce problème d'ordonnancement est courant et peut être rencontré par de nombreuses entreprises. Par exemple, l'entreprise pharmaceutique Sanofi est une multinationale française, leader mondial dans le domaine de la santé. Sanofi est propriétaire du site d'Amilly qui est une grande plateforme logistique. Ce site reçoit des palettes de toutes sortes de médicaments et doit constituer des commandes pour de nombreux clients. Les clients peuvent être des hôpitaux, des pharmacies ou encore des grossistes. Ces commandes sont réparties dans différents camions chargés d'effectuer la distribution à travers la France. Sanofi doit alors optimiser la production de médicaments ainsi que leur distribution de façon à minimiser les coûts ainsi que les délais. Cette entreprise est donc touchée par ce problème d'ordonnancement multi-acteur.

De même, toujours dans le domaine médical, les délais de livraison peuvent avoir un grand impact, c'est pourquoi il est impératif de pouvoir optimiser ces livraisons. Comme le cas au CHRU de Tours, la production de chimiothérapies doit être effectuée en fonction de la demande et doit être livrée au fur et à mesure aux patients. De plus, ce sont des produits très périssables, ils ne peuvent donc pas être stockés très longtemps. C'est pour cela que le CHRU de Tours cherche à minimiser ce temps de stockage et le temps d'attente des patients. De plus, la production de chimiothérapies nécessite de nombreux produits toxiques et leur production est assez délicate et coûteuse. Il est donc important que les délais de livraison et de production soient respectés tout en minimisant les coûts, les pertes et les durées de stockage. Le CHRU rencontre donc le même problème où la distribution doit être coordonnée avec la production.

2 Objectifs

Ce problème de chaîne logistique multi-acteurs est un problème assez courant dans le domaine de la production et de la distribution. Dans ce PRD, on se concentre seulement sur la partie production. En effet, le problème d'optimisation d'itinéraire rencontré par le distributeur ne sera pas étudié ici car celui-ci peut-être simplement résolu avec des méthodes optimales classiques de type "voyageur de commerce".

C'est pourquoi l'objectif de ce PRD est de pouvoir trouver une solution au problème de production. Cependant, ce type de problème ne peut pas être résolu rapidement et de façon optimale si le nombre de commandes à réaliser est trop important. Il est donc important de mettre au point des méthodes heuristiques (Heuristique/Métaheuristique) pour trouver la meilleure solution possible. C'est pourquoi notre étude consiste à trouver le meilleur algorithme permettant de trouver une solution la plus proche possible de la solution optimale en un temps raisonnable.

Ce projet est principalement encadré par M. Jean-Charles Billaut qui est l'un des chercheurs ayant déjà réalisé des études sur ce projet. De plus, M. Yannick Kergosien rejoint aussi le tutorat de ce projet.

3 Hypothèses

Il existe trois différents types de négociations possibles entre les acteurs à prendre en compte lors de la résolution du problème :

- Le distributeur est dominant et impose son nombre de véhicules disponibles au producteur. Dans ce cas, le producteur devra optimiser sa production en minimisant ses coûts pour s'adapter aux conditions imposées par le distributeur.
- Le producteur est dominant et choisit à quel moment les commandes seront disponibles pour la livraison (il choisit aussi le nombre de véhicules) et le distributeur devra s'adapter.
- Le producteur et le distributeur coopèrent (appartiennent à la même entreprise).

Nous nous intéresserons seulement au cas où le producteur est dominant. C'est donc le producteur qui impose le nombre de véhicules, les dates de départ des véhicules ainsi que la composition des lots (donc les destinations à desservir).

Le problème de production sera considéré comme un problème de type "flow-shop" avec un minimum de deux machines. Les machines sont considérées comme des étapes de la production. Chaque tâche doit passer dans chaque machine. A ce problème seront ajoutés des coûts de stockage ainsi que des dates de départ de véhicules (départ des lots).

Une autre hypothèse à prendre en compte dans ce problème est le transport des commandes depuis le site du producteur vers le site du distributeur. Cette distance est considérée comme courte et donc le temps lié à ce transport est négligé dans ce système. Il en est de même pour les événements extérieurs comme les accidents, ou les conditions météo (temps ou catastrophe naturelle). Ces derniers seront également négligés.

Sachant que nous nous concentrons seulement sur le problème d'optimisation du producteur, nous ne pouvons pas prévoir quels seront les coûts liés au retard des livraisons qui sont liés au résultat d'optimisation côté distributeur. Ces pénalités devront donc être estimées.

4 Illustration du problème à optimiser

Comme vu dans les hypothèses, le problème d'ordonnancement côté producteur est assimilé à un problème de type "flow shop". Ce type de problème est assez courant dans le domaine de l'ordonnancement. Il est caractérisé par différentes tâches et plusieurs machines. Dans ce problème, deux contraintes principales sont à respecter :

- Chaque tâche doit passer par toutes les machines
- Une machine ne peut traiter qu'une seule tâche à la fois.

Il existe aussi deux états possibles pour une tâche dans notre cas :

- "en cours de production"
- "en attente de livraison"

Une fois qu'une tâche commence à passer sur la première machine, son processus passe à l'état "en cours de production". Dès que la tâche a fini de passer sur toutes les machines, son processus passe à l'état "en attente de livraison" et reste dans cet état jusqu'à ce que la commande (la tâche) soit récupérée par un véhicule.

L'objectif est de pouvoir minimiser le temps passé dans ces deux états de façon à minimiser les coûts d'inventaires. En effet, ces deux états possèdent des coûts (plus ou moins élevés par rapport à l'autre).

Nous allons voir différents exemples qui permettent d'illustrer et de différencier ce qui est attendu et ce qui est à éviter.

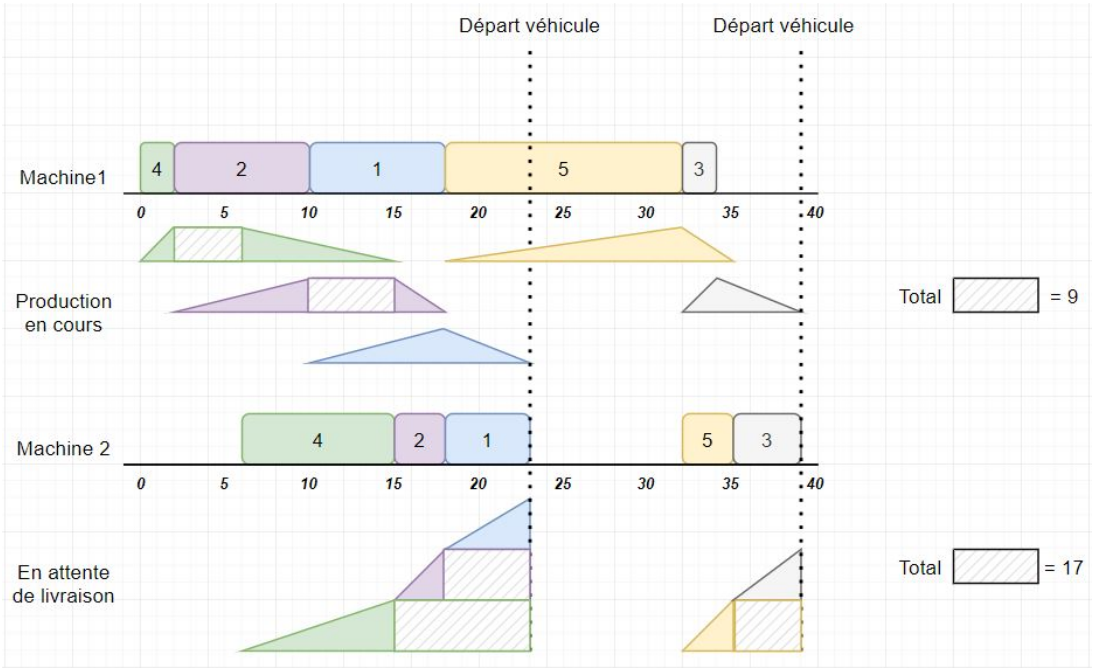


Figure 1.2 – *Illustration d'un ordonnancement intéressant*

Dans cette figure ci-dessus, les zones hachurées correspondent aux zones modifiables à minimiser. Ces zones correspondent aux coûts d'inventaire. Ici les coûts de stockage sont assez minimes (que ce soit pour les productions en cours ou en attente, on arrive à un total de 26 unités de temps). Cela peut donc être un cas intéressant.

Nous allons voir maintenant un cas moins intéressant avec la figure ci-dessous :

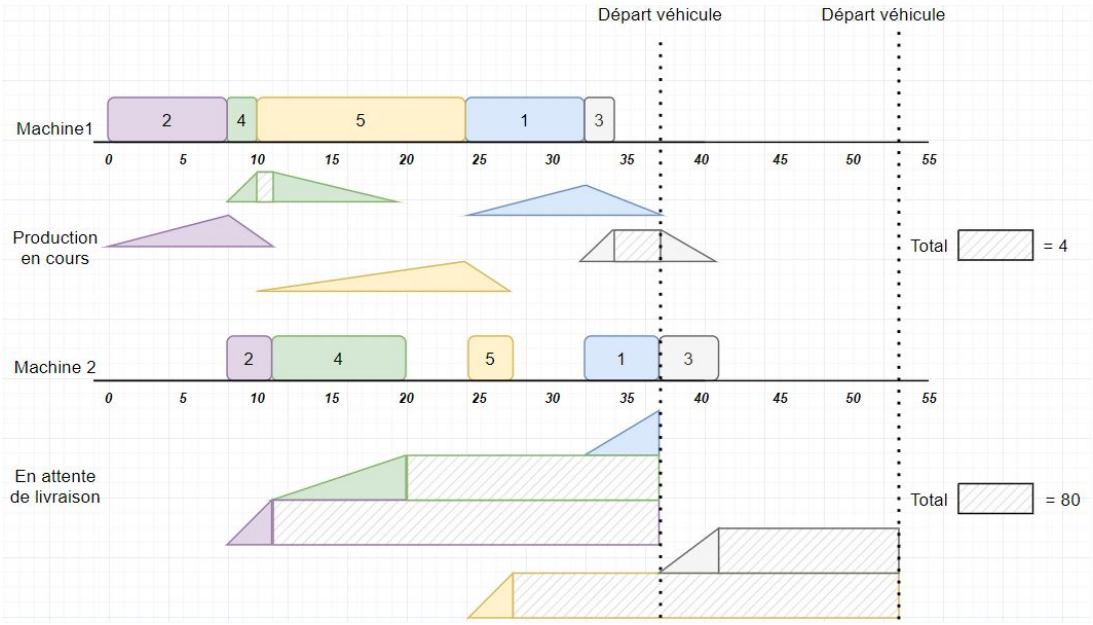


Figure 1.3 – *Illustration d'un ordonnancement moins intéressant*

Dans cette figure, avec les mêmes tâches, on s'aperçoit bien qu'avec ces conditions, les zones hachurées sont beaucoup plus grandes (84 unités de temps au total) et donc les coûts d'inventaire vont être beaucoup plus importants. C'est donc une solution moins intéressante que la précédente.

5 Bases méthodologiques

Concernant la gestion de projet, nous avons utilisé le modèle en “cascade”. Ce modèle permet d’inscrire différentes étapes à réaliser et à valider au cours du projet. Chaque étape a pour objectif de réaliser une tâche finale particulière qui devra être validée pour passer à l’étape suivante.

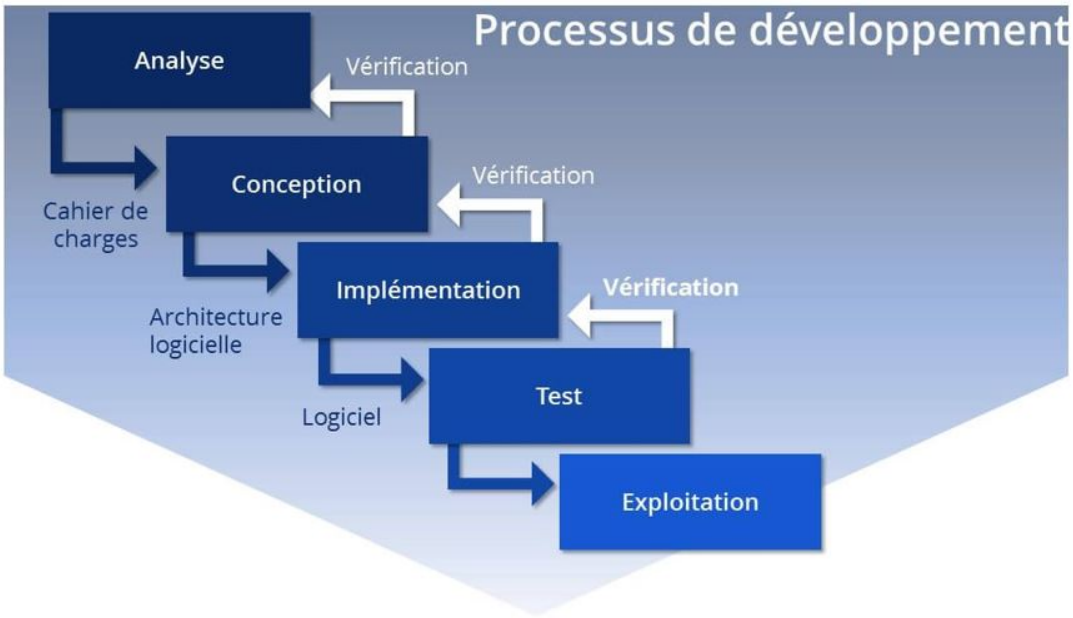


Figure 1.4 – *Cycle en cascade*

Des réunions avec le tuteur principal (M. Billaut) ont été mises en place de façon régulière (deux réunions par mois) afin d’assurer le bon déroulement du projet. Celles-ci se déroulaient selon le plan suivant :

- Revue de ce qui a été fait
- Revue de ce qu’il y a à faire
- Revue du planning
- Questions

Plus d’informations concernant la gestion du projet sont disponible en annexe dans la partie Planification, gestion de projet

2 Description générale

1 Environnement du projet

Notre outil s'appuiera sur des instances de données déjà connues qui ont été générées aléatoirement. Ces instances ne changeront pas et nous utiliserons toujours les mêmes afin d'établir des comparaisons de performances entre les différents résultats obtenus. Ces instances pourront aussi être utilisées par d'autres personnes dans la poursuite de cette recherche. Ainsi, les résultats obtenus par ces personnes pourront être comparés à nos résultats afin de vérifier un avancement.

2 Caractéristiques des utilisateurs

Les utilisateurs de ce logiciel seront des chercheurs en informatique fondamentale et appliquée. Leurs connaissances et leur expérience sur l'utilisation des outils informatiques sont donc notables. Il ne sera donc pas nécessaire de développer une interface graphique "User Friendly" pour ce type d'utilisateurs. Une documentation détaillée sera donc suffisante pour l'utilisation de ce logiciel. Les utilisateurs seront donc des experts en informatique et pourront avoir l'accès au code afin de poursuivre leur recherche.

3 Fonctionnalités du système

Nous rappelons dans ce projet qu'il existe trois différents types de négociations :

- Le producteur domine les négociations et le distributeur s'adapte
- Le distributeur domine les négociations et le producteur s'adapte
- Le producteur et le distributeur coopèrent

Dans ce PRD, nous nous concentrons essentiellement dans le cas où le producteur domine les négociations. Cependant, il est important de prendre en compte les autres cas en vue de la poursuite de l'étude.

Voici le diagramme des cas d'utilisation de notre système :

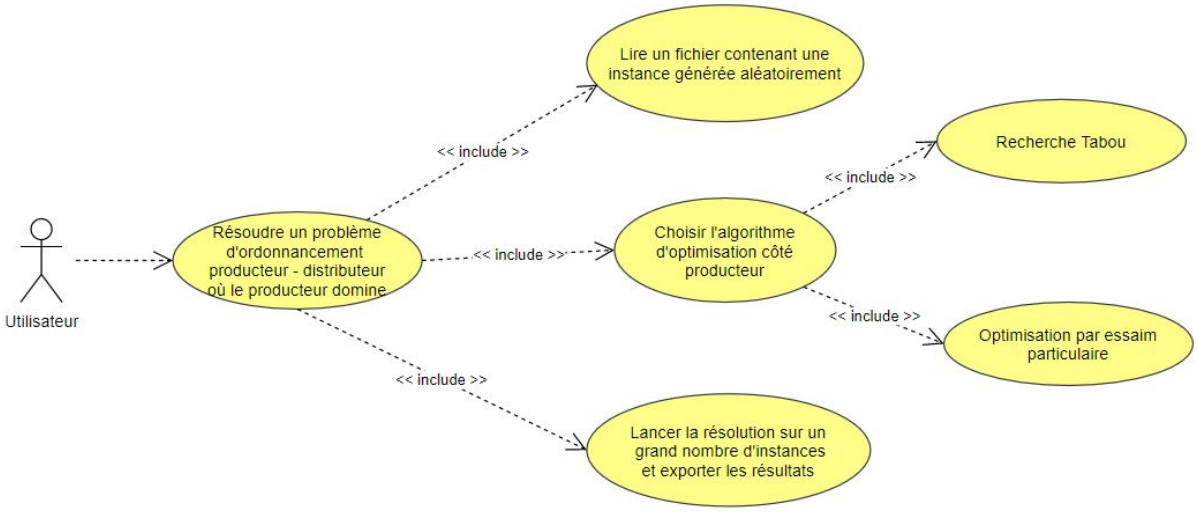


Figure 2.1 – *Digramme des cas d'utilisation*

4 Structure générale du système

Le système sera constitué en trois parties :

- Données en entrée (instances)
- Algorithme de résolution (recherche Tabou ou optimisation par essaims particulaires)
- Données en sortie (résultats)

4.1 Données en entrée

Les données entrées devront être répertoriées dans un dossier. Ce dossier sera constitué de plusieurs fichiers ayant le même format. Un fichier correspond à une instance générée afin d'étudier les performances des algorithmes. Une instance est constituée de plusieurs paramètres. Ces paramètres seront dans le même ordre pour chaque instance (fichier) et seront séparés par un saut de ligne.

Exemple :

```
7    5    4000  5
0
88   58   70    13    32
2    3    4     6     7
105  6    12
172  41
...
```

Figure 2.2 – *Exemple d'un fichier en entrée*

Interprétation :

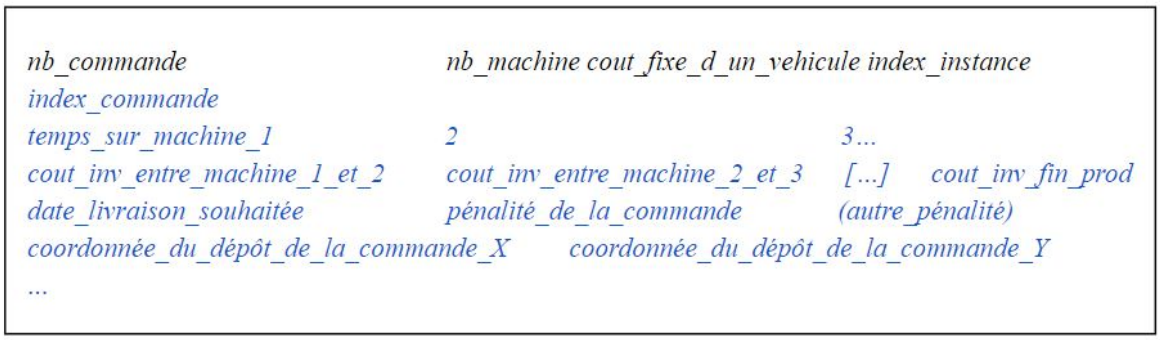


Figure 2.3 – *Interprétation d'un fichier en entrée*

Les éléments en bleu constituent une seule commande. Cela signifie que ces éléments sont répétés pour toutes les commandes (l'exemple décrit une seule commande sur les sept).

4.2 Données en sortie

Les données en sortie seront contenues dans un seul fichier texte (.txt). Ce fichier sera constitué d'une liste de résultats représentant les meilleurs coûts obtenus pour chaque instance en entrée. Chaque coût sera suivi du temps qu'il a fallu à l'algorithme pour trouver une solution ainsi que des paramètres utilisés pour la résolution.

Chaque ligne dans le fichier résultat sera sous cette forme :

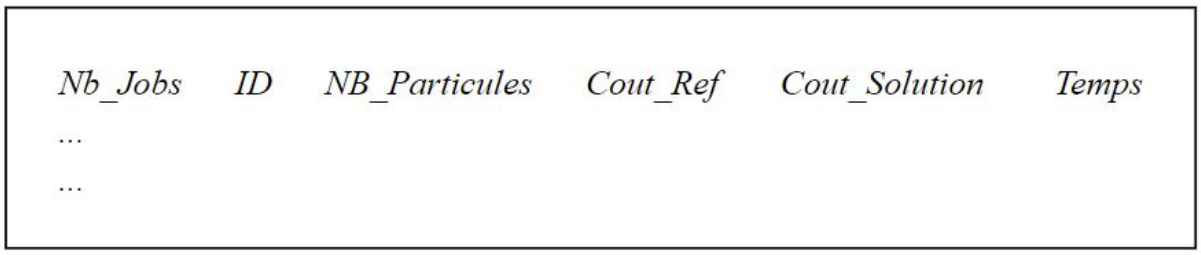


Figure 2.4 – *Interprétation d'un fichier en sortie*

Des paramètres concernant la résolution de l'algorithme pourront aussi être ajoutés.

Les résultats pourront être exploités sur l'outil Excel.

4.3 Système

Voici un schéma récapitulatif qui reprend l'architecture générale du système :

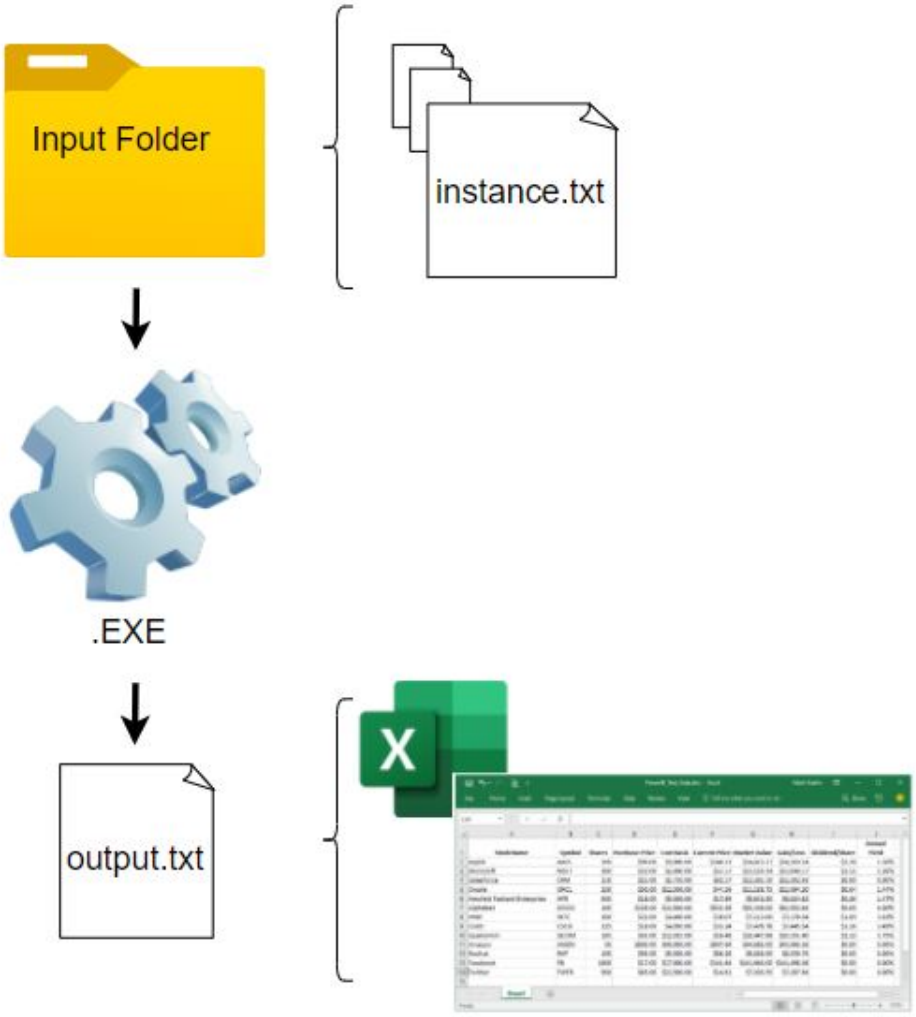


Figure 2.5 – *Architecture globale du système*

Deux exécutables devront être créés. L'un utilisera un algorithme de recherche Tabou et l'autre utilisera un algorithme d'optimisation par essaims particulaires. Cependant mis à part la méthode de résolution, les architectures des deux logiciels seront identiques.

Concernant l'architecture du programme, celui-ci se caractérise sous forme d'objets avec un pattern Model/Controler. Le diagramme de classe se situe à la page suivante :

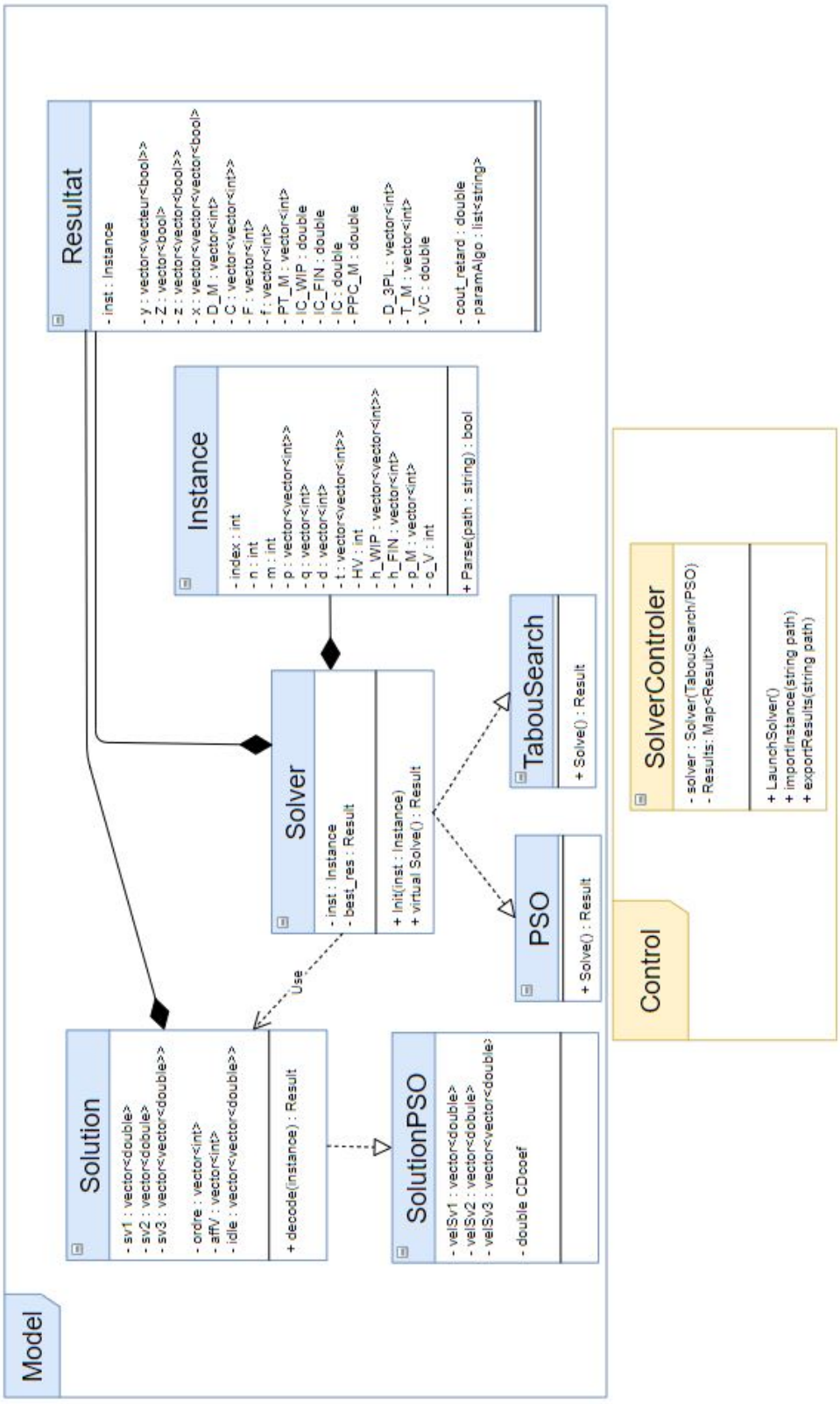


Figure 2.6 – *Diagramme de classes*

3 État de l'art

Le problème d'ordonnancement producteur-distributeur que nous étudions est encore un problème assez récent et très spécifique. Ce problème dépend grandement de la structure à laquelle on souhaite l'intégrer. C'est pour cela qu'aujourd'hui nous ne trouvons pas d'articles traitant exactement du même problème avec les mêmes paramètres et objectifs. Cependant, notre problème de production s'axe principalement sur un problème de type flow-shop et nous pouvons retrouver de nombreux articles à ce sujet aujourd'hui. Dans cette section, nous allons donc faire l'état de l'art à partir de l'étude déjà réalisée sur le sujet par M. Chevroton, Mme. Rohmer et M. Billaut. Nous étudierons aussi d'autres articles se rapprochant le plus possible de notre problème.

1 Point de départ

Dans la poursuite de l'étude menée par Hugo Chevroton, Sonja Rohmer et Jean-Charles Billaut [1], nous pouvons examiner les recherches réalisées à partir des articles et thèses publiés. Ces derniers permettent de relever que les méthodes de résolution exacte ne peuvent pas résoudre le problème de façon optimale si le nombre de commandes est supérieur à huit à cause de la complexité du problème. C'est pour cela que des méthodes approchées ont été utilisées telles qu'un algorithme GRASP (Greedy randomized adaptive search procedure) et un algorithme génétique. L'analyse des résultats obtenus par ces algorithmes démontre que l'algorithme génétique est plus performant que l'algorithme GRASP. En effet, le temps utilisé pour trouver une solution proche de l'optimal est très court. De plus, la meilleure solution obtenue par l'algorithme génétique dévie très peu de la solution optimale. Une approche a aussi été entreprise au cours de cette étude avec une méthode de recherche Tabou. Cependant les résultats ne furent pas aussi compétitifs que ceux obtenus par le GRASP et l'algorithme génétique. Malgré des résultats peu convaincants, cette voie reste à approfondir ou à redévelopper.

2 La montée des heuristiques et métaheuristiques

Une revue de l'université de Parana en 2014 qui étudiait les publications sur un problème flow-shop spécifique [4]. Cet article permet d'identifier quelles sont les méthodes majoritairement abordées pour résoudre ce type de problème. Il en résulte que ce problème a suscité un intérêt croissant au sein des chercheurs car il y a en moyenne de plus en plus d'articles sur le sujet. En

effet, les problèmes flow-shop sont très vite identifiables dans le domaine de l'industrie avec les usines dans les cas réels. C'est donc un problème qui touche de plus en plus les entreprises pour pouvoir optimiser leurs productions. Cet article identifie aussi les méthodes de résolution adoptées par les chercheurs et la majorité des publications porte sur des algorithmes métaheuristiques. Les algorithmes métaheuristiques sont donc beaucoup plus privilégiés que les heuristiques simples et les méthodes de résolution exactes. En effet, les métaheuristiques sont des versions complexes et améliorées d'algorithmes heuristiques simples. Les méthodes de résolution exacte sont quant à elles beaucoup trop gourmandes en temps et ne peuvent donc souvent pas résoudre le problème rapidement. Les métaheuristiques sont donc privilégiées par rapport à leur efficacité en matière de temps mais aussi par rapport à leur très faible déviation à la solution optimale. C'est pour cela que ces dernières sont très étudiées par les chercheurs.

Deux heuristiques se sont révélées intéressantes pour résoudre le problème d'optimisation avec un coût d'inventaire lorsque les tâches sont en cours de production (Work In Progress Costs). Cette étude à était réalisée par Jaehwan Yang et Marc E. Posner dans l'article [8]. La première heuristique s'appuie sur la règle qu'une machine ne doit pas avoir de temps d'inactivité et que les tâches dont le temps total est le plus court sont ordonnées en premier (Heuristique GS).

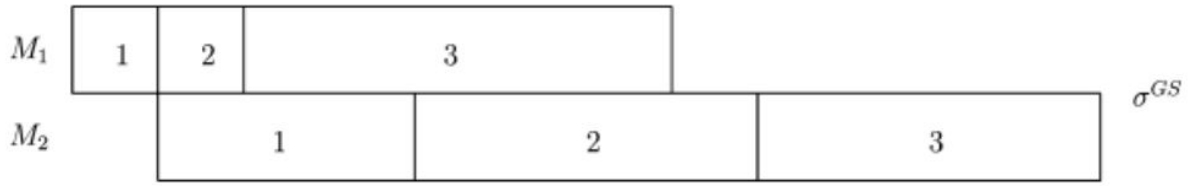


Figure 3.1 – *Exemple heuristique GS*

La deuxième heuristique impose qu'il ne doit y avoir aucun temps d'attente lorsqu'une tâche doit passer d'une machine i à une machine i+1 et que les tâches dont le temps total est le plus court sont ordonnées en premier (Heuristique NW).

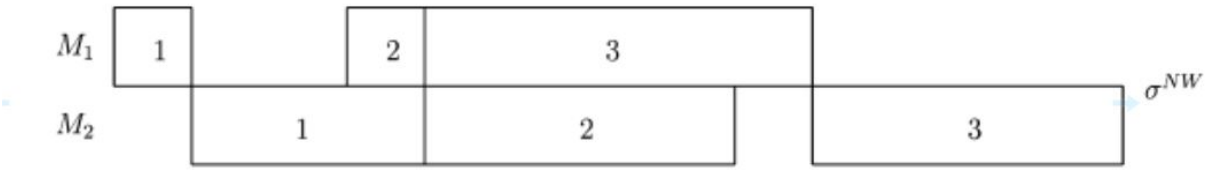


Figure 3.2 – *Exemple heuristique NW*

Le premier algorithme est satisfaisant en termes de temps de production et le second est satisfaisant en matière de coût d'inventaire. Dans notre cas, où le producteur est dominant, le second algorithme peut être très utile pour minimiser les coûts d'inventaire en cours de production. Cependant, cet algorithme ne prend pas en compte le coût d'inventaire de fin de production (avec des lots et date de départs des véhicules) et ne peut donc pas forcément être efficace dans le cas où les coûts d'inventaire de fin de production sont plus chers que les coûts d'inventaire en cours de production.

L'article de Luh, Zhou et de Tomastik [3] est orienté aussi vers la minimisation des coûts de stockage en cours de production. Ils utilisent une relaxation lagrangienne dans leur problème. C'est-à-dire qu'ils suppriment une ou plusieurs contraintes qui rendent le problème NP-difficile. Cela a aussi pour effet de modifier la fonction objectif qui intègre la ou les contraintes enlevées. Les tests de Zhou démontrent qu'il peut générer des ordonnancements avec cette méthode et cela avec de bons résultats au niveau des délais de production.

3 Les performances des algorithmes de recherche Tabou

Un premier article fait référence à un algorithme de recherche Tabou. En effet, l'article de Marc Reimann, Karl Doerner et Richard F. Hartl [5], ressort la performance d'un algorithme Tabou face à celui d'un algorithme décomposé de colonies de fourmis. Les conclusions de cet article indiquent que l'algorithme de colonies de fourmis est très compétitif par rapport à la recherche Tabou. Malgré le fait que ces algorithmes sont utilisés pour résoudre un problème différent du nôtre, les analyses de ces algorithmes permettent de mettre en évidence leur efficacité et leur adaptabilité à un grand nombre de problèmes. De plus, le problème étudié dans cet article est très similaire au problème de calcul d'itinéraire avec plusieurs véhicules côté distributeur dans notre cas. Un autre élément très important qui ressort dans cet article est l'analyse des résultats obtenus. En effet, il n'y a pas que le facteur solution qui est important car si une heuristique arrive à trouver une solution à peine moins bonne qu'une autre heuristique mais avec un temps de calcul mille fois inférieur, alors elle sera privilégiée. Cependant, les utilisations commerciales ne dépendent pas que du facteur temps et qualité de la solution, mais elles dépendent aussi de la flexibilité et de la simplicité des approches. Il est mentionné que la plupart des métaheuristiques sont peu convaincants en matière de flexibilité et de simplicité. Ces trois facteurs (solutions, temps et flexibilité) sont donc des éléments importants à prendre en compte lors de nos futures analyses.

Le problème étudié par Christian Viergutz et Sigrid Knust dans l'article [7] est multi-acteurs et comprend un producteur et un distributeur. Le problème traité par ces auteurs est donc relativement similaire au nôtre. Leurs recherches et analyses s'appuient sur un algorithme par séparation et évaluation (Branch and Bound). Les auteurs mettent aussi en avant les performances d'un algorithme de recherche Tabou avec de bons résultats obtenues sur des grandes instances avec des temps de calculs compétitifs.

4 Les performances des algorithmes d'optimisation par essaims particulaires

D'après l'étude de Chao-Tang Tseng [6] en 2006, l'optimisation par essaims particulaires peut être très efficace pour la résolution de problèmes de type flow-shop avec des flux de lots. En effet, M. Tseng a utilisé une version discrète de cet algorithme, c'est donc une DPSO (Discrete Particular Sworm Optimisation). Ses conclusions sont très positives par rapport à cette méthode car celle-ci trouve une solution très proche de l'optimal en un temps très court. En effet, les résultats de Chao-Tang démontrent que cet algorithme est largement plus rapide qu'une résolution d'un modèle de programmation linéaire par contrainte. De plus, son algorithme a aussi été comparé à un algorithme génétique (GA) et un algorithme génétique hybride (HGA).

Les comparaisons entre les résultats des différents algorithmes démontrent que l'algorithme DPSO surpasse les algorithmes génétiques avec des problèmes de petites et grandes tailles (la taille du problème dépend du nombre de machines et de tâches à ordonnancer). Cependant cet algorithme est traité sur un problème flow-shop qui ne gère que des flux de lots donc les coûts d'inventaire ne sont pas vraiment pris en compte dans ce cas. Malgré le fait que cet algorithme soit testé sur un problème légèrement différent du nôtre, cela reste une piste très intéressante à approfondir dans notre étude.

Enfin, un dernier article ayant des objectifs très similaires à notre sujet se révèle intéressant. Les auteurs Jianyu Long, Panos M. Pardalose et Chuan Li [2] démontrent les performances d'un algorithme d'optimisation par essaims particulaires orienté multi-objectif (LMPSO). Leur étude se base sur plusieurs objectifs (optimisation multi-objectif) distincts des uns des autres qui sont au nombre de trois. Les objectifs sont de minimiser les coûts d'inventaire, minimiser les délais de livraison et minimiser les coûts de retard. Aucun des trois objectifs ne doit être privilégié par rapport aux autres, c'est pour cela que les auteurs ont pris en compte un optimum de Pareto. L'analyse de leurs résultats obtenus sur 110 instances démontre la performance de ce type d'algorithmes avec de tels objectifs. Nous pouvons prendre en compte une nouvelle fois ce type d'algorithmes (PSO/OEP) malgré le fait que la production étudiée dans cet article ne soit pas de type flow shop.

4 Analyse et conception

Comme vu dans la partie de l'illustration, nous avons un problème de type "flow-shop". Nous allons identifier les différents paramètres, variables et contraintes qui sont propres au cas que nous étudions. Cette partie décrit la modélisation mathématiques du problème.

1 Paramètres

Voici une liste des paramètres du problème.

Paramètres	Description
n	nombre de tâches
m	nombre de machines
$p_{i,j}$	temps de production de la tâche J_j sur la machine Mi, j∈[1,n], i∈[1,m]
d_j	date de livraison de la tâche J_j, j∈[1,n]
$t_{s1,s2}$	temps de trajet entre le site s1 et le site s2, s1,s2∈[0,1,...,n+1]
HV	très grande valeur arbitraire
h_{ij}^{WIP}	coût stockage de la tâche J_j en cours de production sur la machine M_i, j∈[1,n], i∈[1,m]
h_j^{FIN}	coût stockage de la tâche J_j en fin de production, j∈[1,n]
π_j^M	coût de pénalité de la tâche J_j si livré retard , j∈[1,n]
c^V	coût par véhicule

Figure 4.1 – *Paramètres*

Pour pouvoir tester les algorithmes, différents paramètres seront générés aléatoirement. Ces paramètres devront être cohérents avec les cas de la vie réelle, c'est pourquoi une partie sera consacrée pour expliquer la génération de ces paramètres.

2 Variables

Il existe deux types de variables pour le cas de la production. En effet, certaines ne peuvent être connues qu'après avoir résolu le problème de distribution. Par exemple, il est impossible de connaître le coût total des pénalités liées au retard si nous ne savons pas à quel moment les commandes sont distribuées. C'est aussi pour cela qu'une variable pour estimer les dates de livraison a été mise en place.

Variables	Description
$y_{j1,j2}$	= 1 si la tâche J_{j1} est ordonnancée avant la tâche J_{j2}, 0 sinon, $1 \leq j1,j2 \leq n$
Z_k	= 1 si le véhicule k est utilisé, 0 sinon, $1 \leq k \leq n$
$z_{j,k}$	= 1 si la tâche J_j est transportée par le véhicule k, 0 sinon $1 \leq j \leq n, 1 \leq k \leq n$
$x_{j1,j2,k}$	= 1 si la tâche J_{j1} et la tâche J_{j2} sont transportées par le véhicule k et que J_{j1} est livré avant la tâche J_{j2} , 0 sinon, $1 \leq j1,j2 \leq n, 1 \leq k \leq n$
D_j^M	estimation de la date de livraison de la tâche J_j , $1 \leq j \leq n$
$C_{i,j}$	date de fin de production de la tâche J_j sur la machine M_i, $1 \leq j \leq n, 1 \leq i \leq n$
F_k	date de départ du véhicule k , $1 \leq k \leq n$
f_j	date de départ de la tâche J_j, $1 \leq j \leq n$
$PT_j^M >= 0$	estimation du retard de la tâche J_j pour le producteur, $1 \leq j \leq n$
IC_WIP	Coût d'inventaire total en cours de production
IC_FIN	Coût d'inventaire total en attente de livraison
IC	Coût d'inventaire total
PPC^M	pseudo coût de retard du producteur
V	Nombre de véhicule requis par le producteur

Figure 4.2 – *Variables*

Variables connues après avoir résolu le problème du distributeur tiers :

Variables	Description
D_j^{3PL}	date de livraison de la tâche J_j
T_j^M	retard de la tâche J_j
VC	coût final des véhicules

Figure 4.3 – *Variables estimées*

3 Contraintes et expressions

Notre problème est constitué de nombreuses contraintes et expressions. Nous allons voir dans cette partie comment les contraintes sont formées avec les paramètres et les variables présentés dans les parties précédentes.

Le coût total représente l'ensemble des coûts du système. Cela inclut le coût d'inventaire en cours de production et en fin de production de chaque commande. Il est calculé avec cette expression :

$$IC = \sum_{j=1}^{n} \sum_{i=1}^{m-1} (C_{i+1,j} - p_{i+1,j} - C_{i,j})\, h_{i,j}^{WIP} + \sum_{j=1}^{n} (f_j - C_{m,j})\, h_j^{FIN} \tag{1}$$

Les coûts de retards côté producteur sont calculés de la façon suivante :

$$PC^M = \sum_{j=1}^{n} \pi_j^M T_j^M \tag{2}$$

Cependant, T_j^M n'est pas connu tant que nous n'avons pas résolu le problème du côté distributeur (3PL). Le retard doit donc être estimé (PT_j^M) pour pouvoir calculer une estimation du coût (PPC^M).

$$PT_j^M = \max\left(0, D_j^M - d_j\right) \tag{3}$$

$$PPC^M = \sum_{j=1}^{n} \pi_j^M PT_j^M \tag{4}$$

Les coûts représentés par les véhicules sont les suivants :

$$VC = c^V V \tag{5}$$

Une contrainte relative à l'ordre des tâches (une tâche ne peut pas être produit à la fois avant et après une autre) :

$$y_{j1,j2} + y_{j2,j1} = 1 \quad (\forall j1, j2 \in \{1, \ldots, n\}, j1 \leq j2) \tag{6}$$

Contrainte pour dire qu'une tâche ne peut pas entrer en production sur une machine avant que celle-ci ne soit libre :

$$C_{i,j2} \geq C_{i,j1} + p_{i,j2} - HV\,(1 - y_{j1,j2}) \tag{7}$$

Contrainte indiquant que chaque job doit passer dans chaque machine dans l'ordre :

$$C_{i,j} \geq C_{i-1,j} + p_{i,j} \tag{8}$$

Chaque commande doit être transportée dans un véhicule :

$$\sum_{k=1}^{n} z_{j,k} = 1 \tag{9}$$

Chaque véhicule ne peut partir que si les commandes qu'il doit transporter sont produites :

$$F_k \geq C_{m,j} - HV\,(1 - z_{j,k}) \tag{10}$$

Une commande ne peut pas être livrée tant qu'elle n'est pas produite :

$$f_j \geq F_k - HV\,(1 - z_{j,k}) \tag{11}$$

Un véhicule k est utilisé à partir du moment où il doit transporter au moins une commande :

$$HV \times Z_k \geq \sum_{j=1}^{n} z_{j,k} \tag{12}$$

Dans un même lot, un job j1 a un prédécesseur j2 avec un seuil de livraison inférieur, sinon le prédécesseur est le site de production :

$$\sum_{i \in \{0\} \cup \{j/d_j \leq d_{j1}\}} x_{i,j1,k} = z_{j1,k} \tag{13}$$

Dans un même lot, un job j1 a un successeur j2 avec un seuil de livraison supérieur, sinon le successeur est le site de dépôt :

$$\sum_{i \in \{n+1\} \cup \{j/d_j \geq d_{j1}\})} x_{j1,i,k} = z_{j1,k} \tag{14}$$

Pour le routage d'un véhicule k, le site de production a un successeur seulement si le véhicule k est utilisé :

$$\sum_{j=1}^{n} x_{0,j,k} \leq Z_k \tag{15}$$

L'estimation de la date de livraison est donnée via cette contrainte :

$$D_{j2}^{M} \geq D_{j1}^{M} + t_{j1,j2} - HV\,(1 - x_{j1,j2,k}) \tag{16}$$

Contrainte donnant une borne inférieure sur la date de livraison d'une tâche j transporté par un véhicule k :

$$D_{j}^{M} \geq F_k + t_{0,j} - HV\,(1 - z_{j,k}) \tag{17}$$

Estimation du retard d'un job :

$$PT_{j}^{M} \geq D_{j}^{M} - d_j \tag{18}$$

4 Fonction objectif

L'objectif principal de notre problème est de minimiser les coûts. Pour cela, les coûts d'inventaire des commandes en production et les coûts d'inventaires des commandes en attente de livraison doivent être minimes. Il en est de même pour les coûts liés aux retards et les coûts liés aux locations de véhicules.

Ainsi, nous avons la fonction objectif suivante :

$$\text{Minimize } PTC^{M} = IC + PPC^{M} + VC \tag{19}$$

5 Génération des instances

Les instances sont des fichiers contenant des paramètres générés aléatoirement afin de tester l'efficacité des algorithmes. Cependant, le facteur aléatoire est contrôlé pour chaque paramètres afin d'avoir des données proches du domaine réel. Les instances générées et utilisées par M. Chevroton seront réutilisées dans ce PRD afin de pouvoir comparer les performances des algorithmes.

Nous disposons alors de 250 instances au total. Les instances sont regroupées par groupe. En effet, il existe plusieurs instances ayant le même nombre de commandes.

Les nombres de commandes fixés sont : {5,6,8,9,10,20,30,40,50,60,70,80,90,100}.

Chaque instance est alors différenciée par son nombre de commandes et un index.

5 Mise en oeuvre

1 Implémentation des algorithmes pour la PSO

Nous verrons dans cette partie les algorithmes développés pour pouvoir résoudre notre problème d'ordonnancement avec le principe de la méta-heuristique appelé PSO.

1.1 Algorithme PSO

L'optimisation par essaim particulaire (PSO) vise à trouver la meilleure solution avec un ensemble de particules. On peut considérer les particules dans un espace, initialisés avec une position aléatoire au départ. Celles-ci vont converger vers la meilleure solution jusqu'à que la majorité des particules se retrouvent à la même position (représenté par 3 vecteurs $sv1, sv2, sv3$) dans l'espace, soit la meilleure solution trouvée. En effet, les particules vont se déplacer en fonction des solutions trouvées par l'ensemble des particules à un instant t. Les particules communiquent donc entre elles pour déterminer leurs déplacements (vélocité pour chaque vecteur $velSv1, velSv2, velSv3$). Ainsi, les particules ayant obtenu les meilleurs résultats possèdent un coefficient d'attraction supérieur aux autres particules.

Nous avons donc implémenté le raisonnement de la PSO dans notre algorithme. Le principe de la PSO a donc dû être adapté à notre problème. Pour cela, nous avons implémenté un décodage pour transformer les positions d'une particule en un ordonnancement avec des coûts. Nous verrons ce décodage ainsi que le voisinage dans les parties suivantes.

Voici alors l'algorithme principal de notre PSO :

Data : Ns (Nombre de particules) et D (durée maximum pour la résolution)
Result : Meilleure solution trouvée
1 Initialiser un essaim S avec Ns particules, où les particules ont leurs propres positions Xp et vélocité Vp;
2 Décoder toutes les solutions de chaque particule initialisée;
3 Améliorer la qualité des particules en se basant sur des propriétés dérivées voisines;
4 Garder la meilleure solution;
5 **while** *La condition d'arrêt n'est pas atteinte* **do**
6 Decoder les solutions de chaque particule et calculer la Crowding Distance (CD);
7 **foreach** *Particule Xp* **do**
8 Sélectionner deux particules aléatoirement avec le poids CD;
9 Mise à jour de la vélocité Vp puis de la position Xp de la particule ;
10 Décoder la solution complète de la particule;
11 Améliorer la qualité de la particule basée sur des propriétés dérivées voisines;
12 Garder la solution trouvée si celle-ci est la meilleure jamais trouvée;
13 **end**
14 **end**

Algorithme 1 : Algorithme principal

Dans cet algorithme principal, nous calculons des Crowding Distance (CD) afin de pouvoir sélectionner des particules aléatoires (ligne 8). Le calcul des CD se fait de la manière suivante :

1 **foreach** *Particule Xp* **do**
2 Xp[i].CD=0;
3 **end**
4 Trier l'ensemble des particules par ordre croissant de leurs résultats trouvé.;
5 $fmin$=Xp[0].resultat_cout;
6 $fmax$=Xp[n-1].resultat_cout;
7 Xp[0].CD=1;
8 **for** *i=1 à n-1* **do**
9 Xp[i].CD=(Xp[i+1].resultat_couts - Xp[i-1].resultat_couts)/($fmax - fmin$)
10 **end**

Algorithme 2 : Algorithme de calcul des CD

La mise à jour des vélocités puis des positions se fait de la manière suivante :

$$p.vel[i] = \lambda 1 * p.vel[i] + \lambda 2(pa.sv[i] - p.sv[i]) + \lambda 3(pb.sv[i] - p.sv[i])$$

$$p.sv[i] = p.sv[i] + p.vel[i]$$

Avec $\lambda 1$, $\lambda 2$ et $\lambda 3$ des valeurs générés aléatoirement entre 0 et 1. pa et pb sont les particules aléatoires sélectionnés avec les CD à la ligne 8 de l'algorithme principal.

1.2 Décodage d'une solution

Chaque particule doit être décodée pour transformer la position d'une particule en un résultat avec des coûts. Ce décodage est appelé à chaque fois que nous modifions la position d'une particule (c'est à dire les vecteurs $sv1$, $sv2$ ou $sv3$).

Le vecteur $sv1$ (de taille n) est le vecteur qui permet de déchiffrer l'ordonnancement des jobs. Le décodage du vecteur $sv1[j]$ utilise la règle SPV (Smallest Position Value) qui consiste à ordonnancer les jobs j. Plus la valeur de $sv1[j]$ est petite par rapport aux autres, plus l'ordonnancement du job j sera tôt.

Le vecteur $sv2$ (de taille n) est utilisé pour connaître l'affectation des jobs aux véhicules. Pour le décodage de $sv2[j]$, on utilise une règle de l'arrondi (RA). Les valeurs arrondies du vecteur vont donner le numéro du véhicule auquel le job j va être affecté. Ces valeurs sont comprises entre 0 et le nombre total de jobs.

Le vecteur $sv3$ (de taille $m * n$) est utilisé pour connaître les temps morts. En effet, un élément $sv3[i][j]$ va donner un temps mort à un job j avant de pouvoir commencer sa production sur la machine i. Ce vecteur n'a pas besoin d'être transformé car il contiendra de base des valeurs exploitables. Seules les valeurs négatives sont incohérentes et sont alors remplacées par des 0.

Une fois tous les vecteurs décodés, il est alors possible de déterminer chaque variable du système. Ainsi nous pouvons calculer l'ensemble des coûts générés par cette solution.

Ci-dessous l'algorithme de décodage d'une particule :

Data : S (Solution non décodée)
Result : Solution décodée
1 Décoder le vecteur $sv1$ pour récupérer l'ordonnancement avec la règle SPV (Smallest Position Value);
2 Décoder le vecteur $sv2$ pour récupérer l'affectation aux véhicules avec une règle de l'arrondi (RA);
3 Mettre à 0 les éléments négatifs du vecteur $sv3$;
4 Calculer les dates de fin de production Cij;
5 Calculer les dates de départ des véhicules;
6 Calculer les dates de départ des jobs;
7 Estimer les dates de livraisons des jobs (méthode du plus proche voisin);
8 Calculer l'ensemble des coûts;

Algorithme 3 : Algorithme de décodage de particule

Cet exemple ci-dessous permet d'illustrer le principe du décodage. (Les durées de fabrication sur les machines ainsi que les distances sont des données fixes d'une instance)

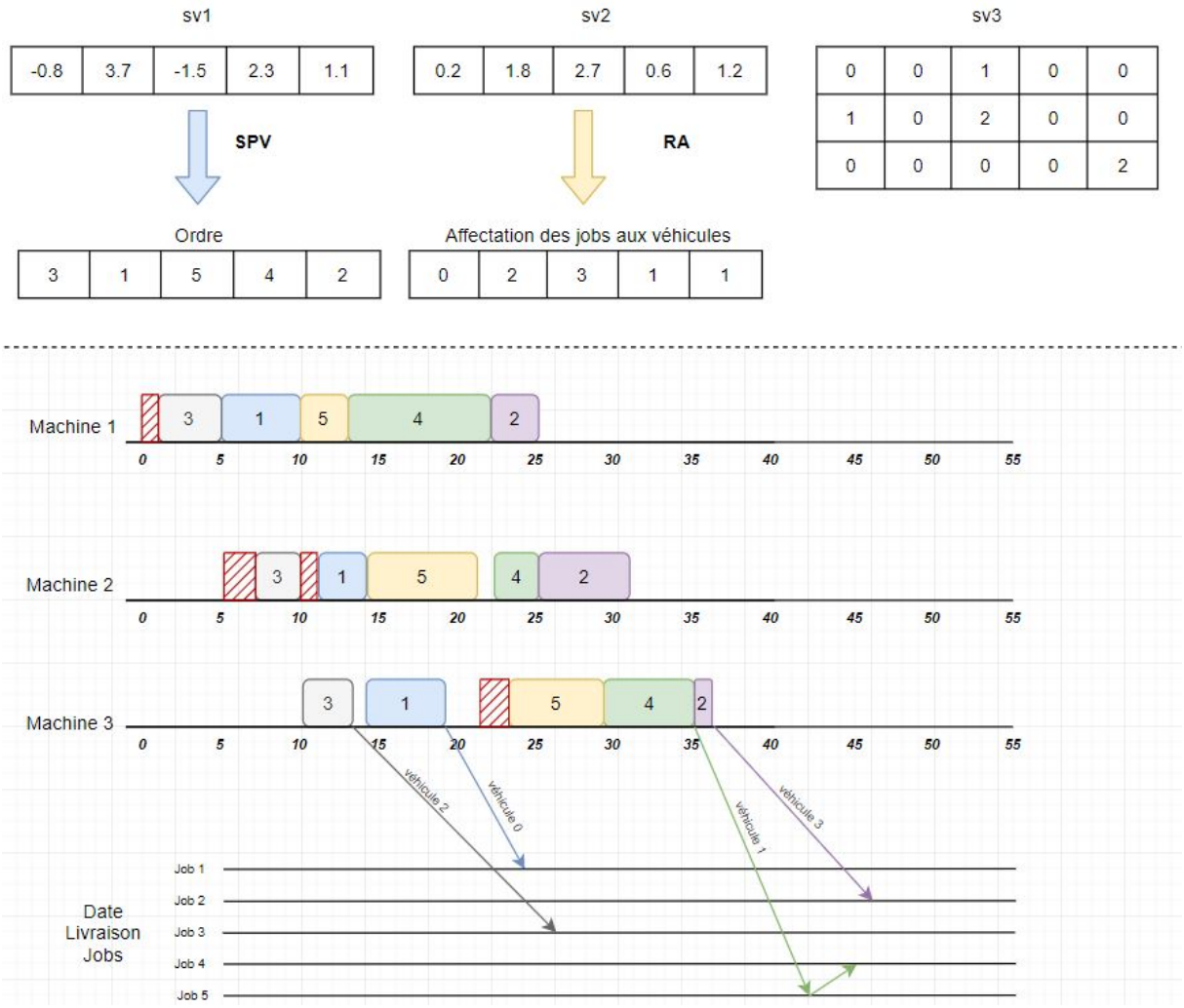


Figure 5.1 – *Illustration du décodage*

1.3 Initialisation des particules

Avant de pouvoir enclencher l'engrenage de l'algorithme principal de la PSO, il est nécessaire d'initialiser les particules. Il faut pour cela initialiser les vecteurs $sv1, sv2, sv3$ de chaque particule. Il en est de même pour la vélocité de chacun des vecteurs.

Le vecteur $sv1$ est initialisé par une valeur aléatoire entre 0 et -1.

Le vecteur $sv2$ est initialisé par une valeur aléatoire entre $-0,5$ et $m - 0,5$.

Le vecteur $sv3$ est initialisé par une valeur aléatoire entre 0 et 50.

Les vecteurs contenant les vélocités sont tous initialisés à 0.

1.4 Solutions de référence

Afin d'avoir une base, nous avons réalisé des solutions de référence. Nous avons réalisé 6 solutions de référence en tant que particule avec les paramètres suivants :

- $sv1[j] = dj$ croissant, $sv2[j] = (j+1)/4$, $sv3[i][j] = 0$.
- $sv1[j] = dj$ croissant, $sv2[j] = (j+1)/3$, $sv3[i][j] = 0$.
- $sv1[j] = dj$ croissant, $sv2[j] = (j+1)/5$, $sv3[i][j] = 0$.
- $sv1[j] =$ somme $(i = 0$ à $n)$ des P[i][j] croissant, $sv2[j] = (j+1)/4$, $sv3[i][j] = 0$.
- $sv1[j] =$ somme $(i = 0$ à $n)$ des P[i][j] croissant, $sv2[j] = (j+1)/3$, $sv3[i][j] = 0$.
- $sv1[j] =$ somme $(i = 0$ à $n)$ des P[i][j] croissant, $sv2[j] = (j+1)/5$, $sv3[i][j] = 0$.

Ces références seront parmi notre ensemble de particules lors de l'initialisation. La PSO se déroule alors toujours avec un minimum de 6 particules.

1.5 Voisinage

Afin d'améliorer la qualité des particules, nous avons mis en place la recherche de meilleures solutions en fonction du voisinage d'une particule à améliorer.

Voici les différents voisinages que nous avons réalisés :

- $V1$: Permuter les valeurs entre deux éléments du vecteur $sv1$.
- $V2$: Permuter les valeurs entre deux éléments du vecteur $sv2$.
- $V3$: Permuter les valeurs entre deux blocs d'éléments du vecteur $sv1$.
- $V4$: Permuter les valeurs entre deux blocs d'éléments du vecteur $sv2$.
- $V5$: Modifier les valeurs d'un élément aléatoire du vecteur $sv3$.
- $V6$: Modifier les valeurs d'un élément du vecteur $sv3$ en fonction de ses coûts WIP et FIN.
- $V7$: Mélanger aléatoirement les éléments du vecteur $sv2$.

2 Analyse des résultats et performances de la PSO

Nous verrons dans cette section une étude des résultats ainsi que des performances de notre PSO. Nous avons utilisé Excel pour cette étude.

2.1 Études des résultats

Afin d'avoir une base sur laquelle nous pouvons comparer nos résultats, nous allons reprendre les résultats qu'a obtenus M. Hugo Chevroton avec des algorithmes différents mais réalisés sur les mêmes instances.

Dans un premier temps, nous avons comparé les résultats obtenus entre la PSO et un algorithme génétique (GA) :

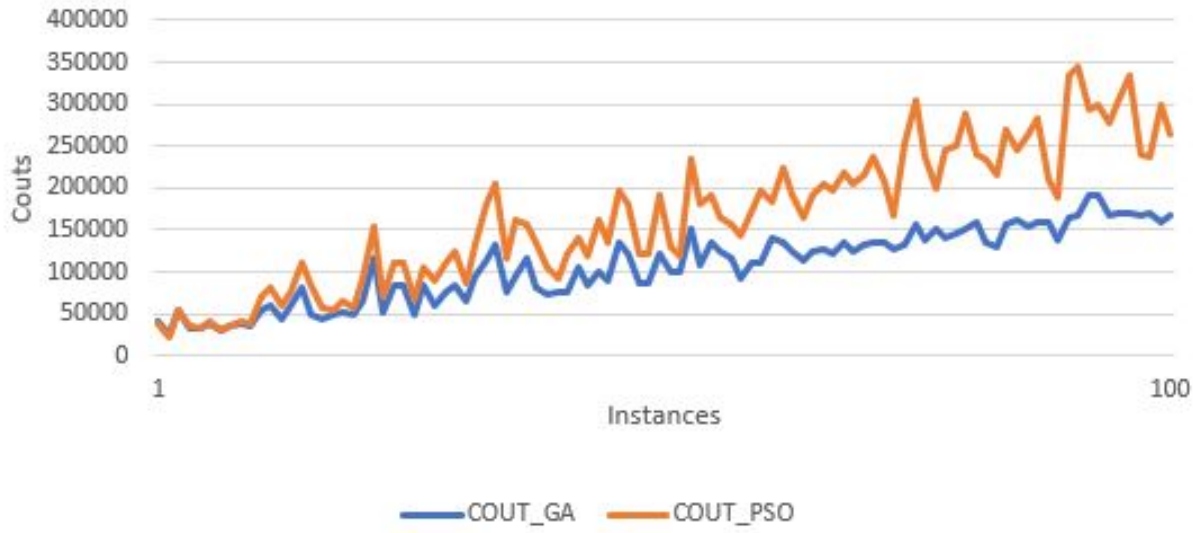


Figure 5.2 – *Comparaisons des résultats entre PSO et GA*

On se rend compte ici que le l'algorithme génétique ressort de meilleurs résultats que la PSO. En effet, on s'aperçoit que les résultats sont plutôt concurrents jusqu'à $n = 30$, mais après cela, plus on augmente n, plus l'écart se creuse.

On remarque le même phénomène lorsque nous comparons la PSO avec un algorithme GRASP (Greedy randomized adaptive search procedure) :

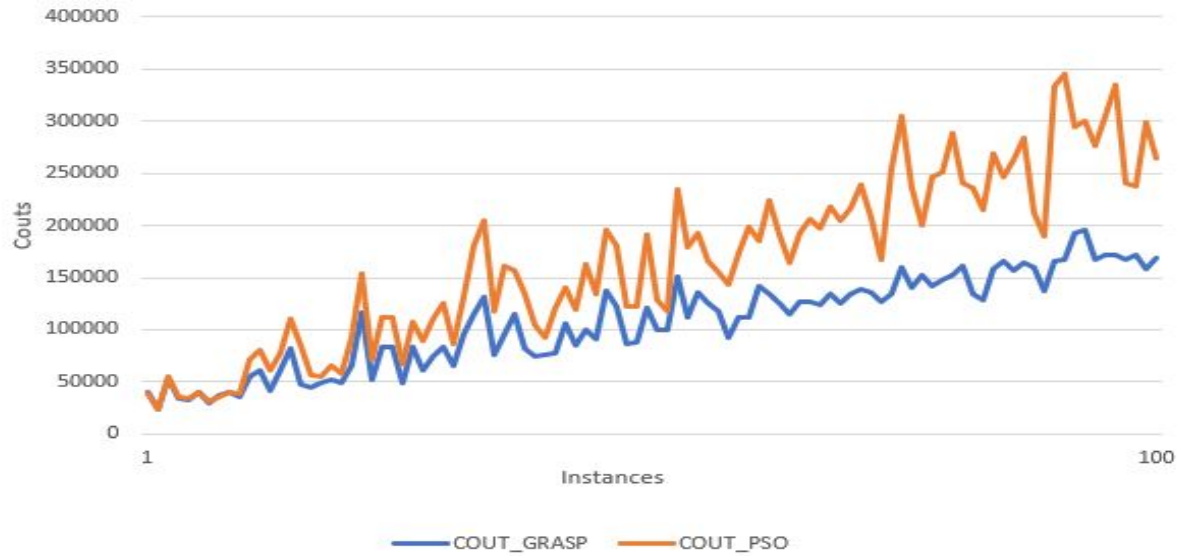


Figure 5.3 – *Comparaisons des résultats entre PSO et GRASP*

Nous pouvons conclure qu'avec un nombre de commandes croissant, notre PSO perd de plus en plus son efficacité. La cause de cela peut venir du fait que les particules n'explorent pas assez largement l'espace des solutions au cours des itérations. Il faudrait donc envisager de modifier la manière dont nous déplaçons les particules, c'est à dire la manière dont nous modifions les vecteurs de vélocité.

2.2 Études des performances

Nous avons réalisé une étude de performance de l'algorithme en matière de temps de résolution. Nous allons ici comparer les durées nécessaires entre les différents algorithmes pour trouver leur meilleure solution. Les relevées se baseront sur 10 instances contenant chacune 10 jobs à ordonnancer sur 5 machines.

Voici les différences entre les différents algorithmes :

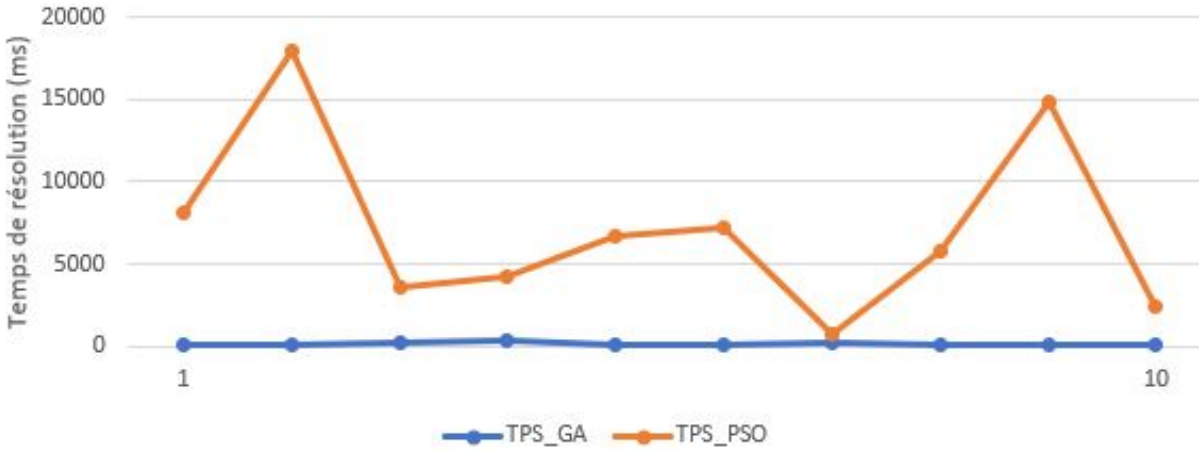


Figure 5.4 – *Comparaisons des temps de calcul entre PSO et GA*

Figure 5.5 – *Comparaisons des temps de calcul entre PSO et GRASP*

Au vu de ces derniers graphiques, nous pouvons voir que les temps de calculs pour la PSO sont beaucoup plus élevés que GRASP et GA développés par M. Chevroton. Cependant il faut prendre en compte que les temps de calcul obtenus par M. Chevroton avec GRASP et GA ont été réalisés avec un PC possédant un processeur Intel Core i7-7820HQ (CPU haut de gamme de 2017), tandis que les résultats obtenus avec la PSO ont été obtenus avec un PC possédant un processeur Intel Core i3-3120M (CPU bas de gamme de 2012). La différence des processeurs peut donc être un facteur de cette différence. Cependant, du temps pourrait être gagné si nous venons à paralléliser le calcul sur plusieurs threads.

3 Qualité de mise en oeuvre

Afin d'obtenir un projet de qualité nous avons mis en place plusieurs éléments. Tout d'abord, nous avons suivi le diagramme de classes que nous avons établi au cours des spécifications.

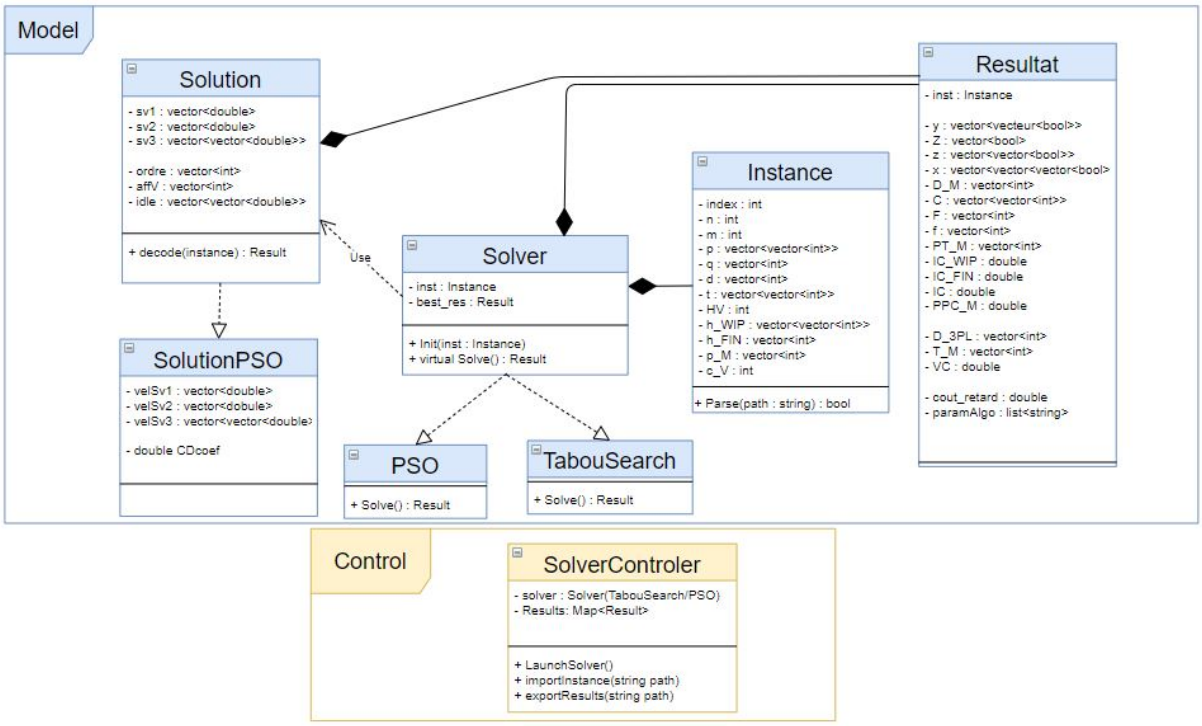


Figure 5.6 – *Diagramme de classes*

Le but était aussi de retranscrire les paramètres, les variables et les contraintes théoriques de la manière la plus transparente possible. Un second point est qu'avec une telle structure, nous pouvons intégrer facilement d'autres méta-heuristiques, tout en gardant les éléments communs à chacun comme la gestion des instances ou la manière de sortir les résultats.

Nous avons également intégré au code une documentation complète des éléments de chaque classe. Le format de la documentation de code avec "intellisense" de Visual Studio est constitué de balises XML. Ces commentaires permettent de mieux se repérer pendant le développement.

De plus, nous avons généré une documentation au format HTML grâce à l'outil Doxygen. Les paramètres de générations ont été sauvegardé dans un fichier Doxyfile afin de pouvoir regénérer la documentation à tout moment et rapidement juste en cliquant sur un simple bouton.

Figure 5.7 – *Documentation technique*

Toujours dans le cadre de la démarche qualité, des tests unitaires automatisés ont été implémentés. Ces derniers permettent de s'assurer que les différents éléments du programme fonctionnent. Nous avons pour cela créé un projet de tests avec Visual Studio que nous avons couplé à notre projet de base.

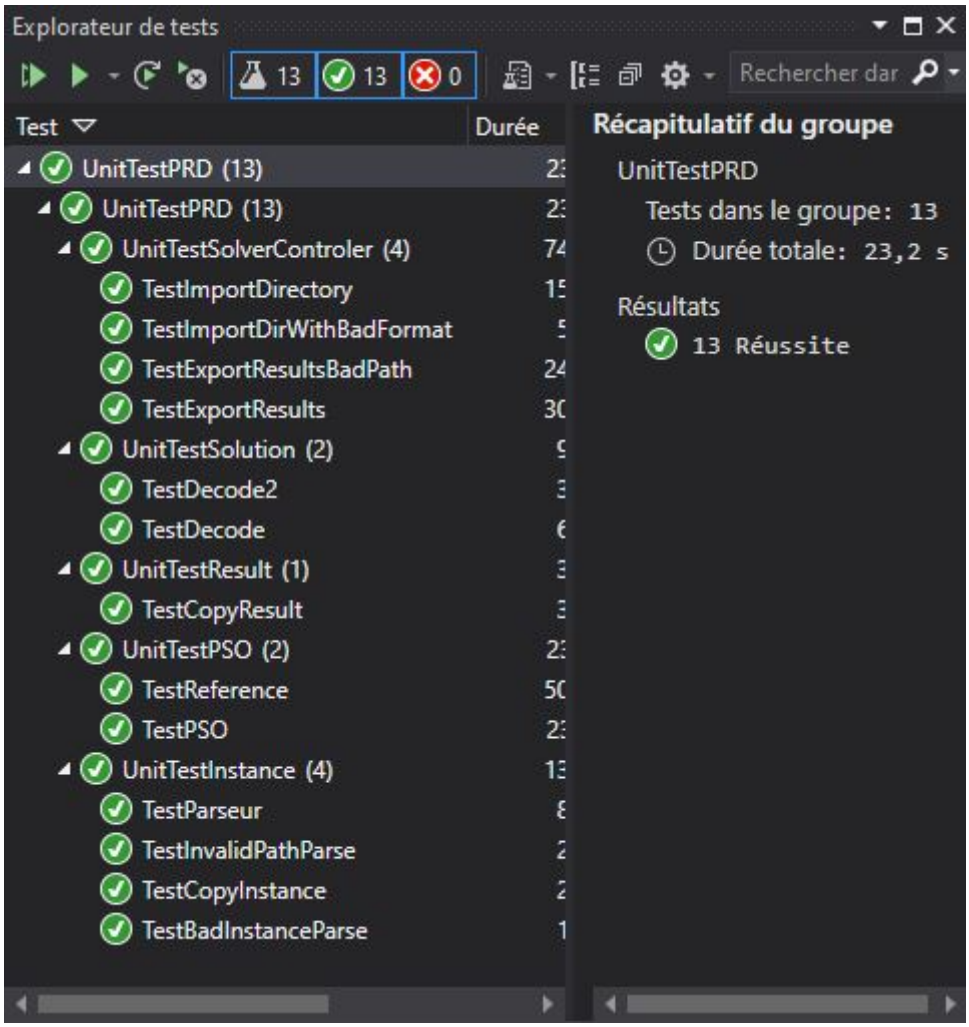


Figure 5.8 – *Tests unitaires automatisés*

Afin de gérer les versions au cours du développement, un dépôt git héberge le code source du projet avec différentes branches. Dans l'éventualité où le projet serait repris, le dépôt git possède tous les éléments pour reprendre le projet dans de bonnes conditions :

- Le code source
- La documentation
- Les ressources de travail (articles, rapports...)
- Un exécutable avec les instances
- Un Readme détaillé pour guider

https ://github.com/YohanDerenne/PRD-Operational-Research

Ci-dessous le guide d'utilisation pour la reprise (Readme git) :

Etude d'un modèle de chaîne logistique multi-acteurs - PRD - Operation Research

Projet recherche et développement 2020/2021 - Yohan DERENNE
Recherche opérationnelle - Polytech Tours DI5
Sujet : Etude d'un modèle de chaîne logistique multi-acteurs

Executer le programme

- Décompresser le dossier "release_x64.zip"
- Lancer PSO.bat (modifier les paramètres de PSO dans le .bat si besoin)

Editer le code source

- Telecharger Visual Studio
- Aller dans le dossier "src"
- Lancer le projet avec "PRD.sln"

Executer les tests sous Visual-Studio

- Parametrer le lancement en Debug x86 ou Release x86
- Aller dans "Test" -> "Explorateur de tests"
- Lancer les tests à partir de la nouvelle fenetre

Documentation technique du code

- Aller dans le dossier "doc"
- Ouvrir "Documentation.html" avec un navigateur

Pour regénérer la documentation technique:

- Telecharger le générateur de documentation Doxygen
- Dans le dossier "doc\generate", ouvrir "Doxyfile" avec Doxygen (ce fichier contient les paramètres pour la génération)
- Aller dans l'onglet "Run" puis appuyer sur le bouton "Run doxygen"

Travail PRD

L'ensemble du travail et ressources pour ce projet (rapports, état de l'art...) se trouvent dans le dossier "ressources".

Figure 5.9 – *Guide utilisateur*

6 Bilan et conclusion

1 Bilan du semestre 9

Le travail réalisé au cours du semestre 9 s'appuie essentiellement sur la partie recherche du projet. En effet, la compréhension de l'ensemble du périmètre du projet fut une première étape par le biais des articles de M. Chevroton, Mme Rohmer et M.Billaut. Nous avons pu reprendre le modèle mathématique qui comprend les paramètres, les variables, les contraintes et la fonction objectif. Ensuite nous avons fait une revue de la littérature avec un état de l'art pour identifier quelles méthodes de résolution peuvent convenir pour notre problème. Nous avons conclu par cette étude qu'il serait judicieux de créer un algorithme de recherche Tabou et un algorithme d'optimisation par essaims particulaires afin de résoudre notre problème d'ordonnancement. Un cahier des spécifications a alors été rédigé pour pouvoir réaliser le logiciel qui permettra de résoudre le problème. Avec ces éléments en main, nous pouvons commencer la partie développement dans de bonnes conditions pour le semestre 10.

2 Bilan du semestre 10

Le semestre 10 était principalement dédié pour le développement du logiciel. En effet, deux algorithmes étaient attendus pour résoudre le problème. Cependant, nous nous sommes mis d'accord avec le client/tuteur de réaliser seulement la PSO. En effet, la PSO était une priorité pour M. Billaut. De plus, lors des recherches pour l'état de l'art, la PSO ressortait souvent dans les articles traitant des problèmes flow-shop afin de démontrer son efficacité. Cela permettait aussi d'avoir un algorithme plus performant plutôt que d'avoir deux algorithmes moyens ou inachevés. En faisant ce choix nous avons réussi à développer une PSO fonctionnelle, certes moins efficaces que les algorithmes de M. Chevrotton, mais on arrive tout de même à obtenir des résultats qui respectent les contraintes. De plus le développement a été réalisé de façon à faciliter au maximum la reprise du projet afin d'obtenir d'éventuelles améliorations de l'algorithme.

Annexes

A Planification, gestion de projet

Dans cette partie, nous verrons les différents éléments qui composent la gestion de projet.

1 Méthodologies et outils

Nous rappelons que nous utilisons le modèle en cascade avec des réunions régulières au rythme de deux par mois. Certaines de ces réunions se sont déroulées à distance via Microsoft Teams.

Figure A.1 – *Logos respectifs de Microsoft Teams et Gantt Project*

Les plannings ont été réalisés avec le logiciel Gantt Project. Ces plannings sont partagés régulièrement avec le tuteur afin qu'il puisse avoir un visuel sur l'avancement du projet.

Figure A.2 – *Logo de Trello*

Pour ce qui est de l'organisation personnelle, un Trello a été mis en place afin de déterminer les tâches à réaliser (plus détaillées que pour le planning). Les tâches peuvent se retrouvées sous 5 catégories différentes :

- A faire
- En cours
- Fait
- En attente de validation
- Validé

Cela permet facilement d'avoir un visuel sur l'avancement des différentes tâches.

Un tableau répertoriant les risques a été créé dans le but de les anticiper. Des solutions ont donc été mises en place afin de pallier ces risques. De même, les rapports de réunions sont stockés dans un tableau afin de garder une trace de celles-ci.

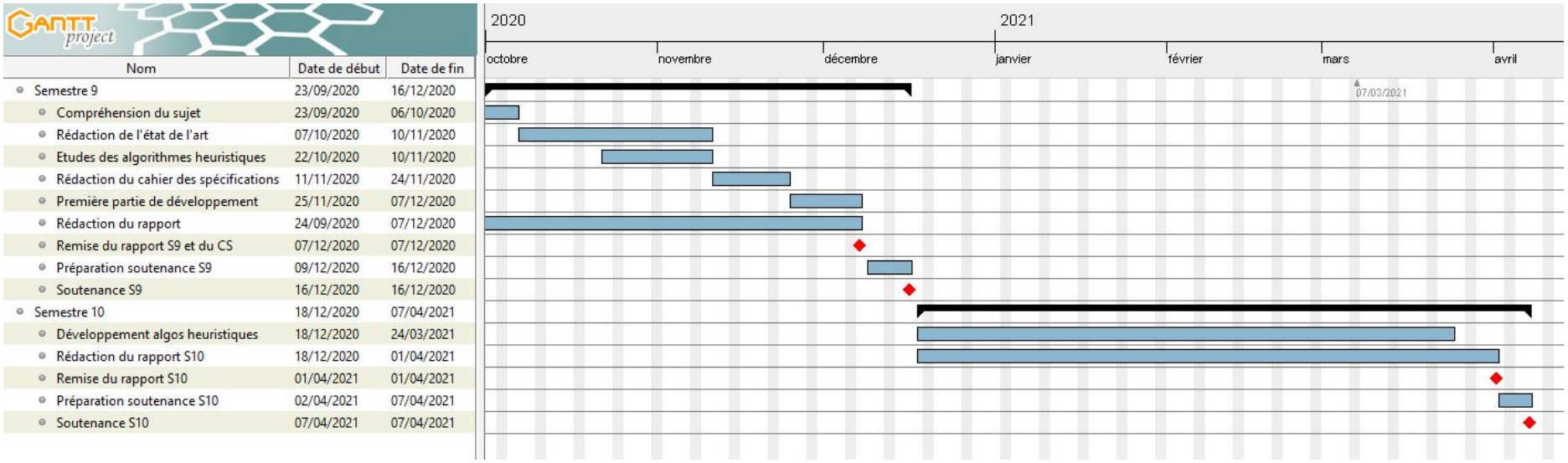
Figure A.3 – *Logos respectifs de Google Drive et GitHub*

Les données et documents en lien avec le projet de recherche et développement sont stockés sur un Cloud sur internet. Cela permettra de garder les données en cas de panne sur une machine. De même, en cas de coupure internet, des sauvegardes régulières en local sont réalisées. Les outils de stockage utilisés sont Google Drive pour les documents et Github pour le code.

2 Plannings

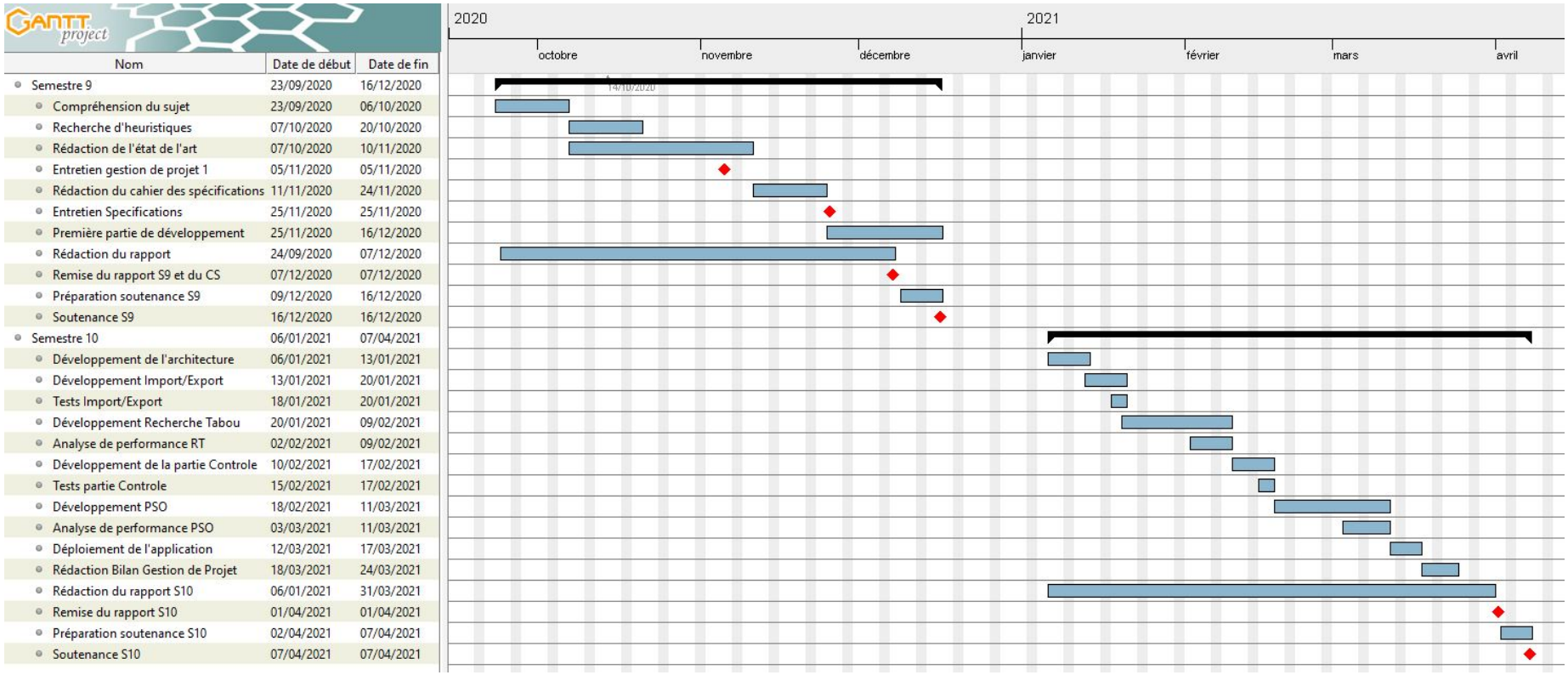
Concernant l'élaboration du planning prévisionnel, celui-ci a été réalisé avec un diagramme de Gantt. En effet, un diagramme de Gantt permet d'obtenir une vision globale du planning de manière simple et efficace. C'est pourquoi ils seront utilisés pour l'élaboration des plannings de ce projet. Celui-ci a été validé et mis à jour constamment afin que le tuteur principal puisse situer l'état d'avancement du projet au cours de chaque réunion.

Le planning du projet a évolué au cours de l'avancement du projet. Nous pouvons voir ci-dessous que le planning initial ainsi que le planning final.



Nom	Date de début	Date de fin
Semestre 9	23/09/2020	16/12/2020
Compréhension du sujet	23/09/2020	06/10/2020
Rédaction de l'état de l'art	07/10/2020	10/11/2020
Etudes des algorithmes heuristiques	22/10/2020	10/11/2020
Rédaction du cahier des spécifications	11/11/2020	24/11/2020
Première partie de développement	25/11/2020	07/12/2020
Rédaction du rapport	24/09/2020	07/12/2020
Remise du rapport S9 et du CS	07/12/2020	07/12/2020
Préparation soutenance S9	09/12/2020	16/12/2020
Soutenance S9	16/12/2020	16/12/2020
Semestre 10	18/12/2020	07/04/2021
Développement algos heuristiques	18/12/2020	24/03/2021
Rédaction du rapport S10	18/12/2020	01/04/2021
Remise du rapport S10	01/04/2021	01/04/2021
Préparation soutenance S10	02/04/2021	07/04/2021
Soutenance S10	07/04/2021	07/04/2021

Figure A.4 – *Planning initial du S9*



Nom	Date de début	Date de fin
Semestre 9	23/09/2020	16/12/2020
Compréhension du sujet	23/09/2020	06/10/2020
Recherche d'heuristiques	07/10/2020	20/10/2020
Rédaction de l'état de l'art	07/10/2020	10/11/2020
Entretien gestion de projet 1	05/11/2020	05/11/2020
Rédaction du cahier des spécifications	11/11/2020	24/11/2020
Entretien Specifications	25/11/2020	25/11/2020
Première partie de développement	25/11/2020	16/12/2020
Rédaction du rapport	24/09/2020	07/12/2020
Remise du rapport S9 et du CS	07/12/2020	07/12/2020
Préparation soutenance S9	09/12/2020	16/12/2020
Soutenance S9	16/12/2020	16/12/2020
Semestre 10	06/01/2021	07/04/2021
Développement de l'architecture	06/01/2021	13/01/2021
Développement Import/Export	13/01/2021	20/01/2021
Tests Import/Export	18/01/2021	20/01/2021
Développement Recherche Tabou	20/01/2021	09/02/2021
Analyse de performance RT	02/02/2021	09/02/2021
Développement de la partie Controle	10/02/2021	17/02/2021
Tests partie Controle	15/02/2021	17/02/2021
Développement PSO	18/02/2021	11/03/2021
Analyse de performance PSO	03/03/2021	11/03/2021
Déploiement de l'application	12/03/2021	17/03/2021
Rédaction Bilan Gestion de Projet	18/03/2021	24/03/2021
Rédaction du rapport S10	06/01/2021	31/03/2021
Remise du rapport S10	01/04/2021	01/04/2021
Préparation soutenance S10	02/04/2021	07/04/2021
Soutenance S10	07/04/2021	07/04/2021

Figure A.5 – *Planning final du S9*

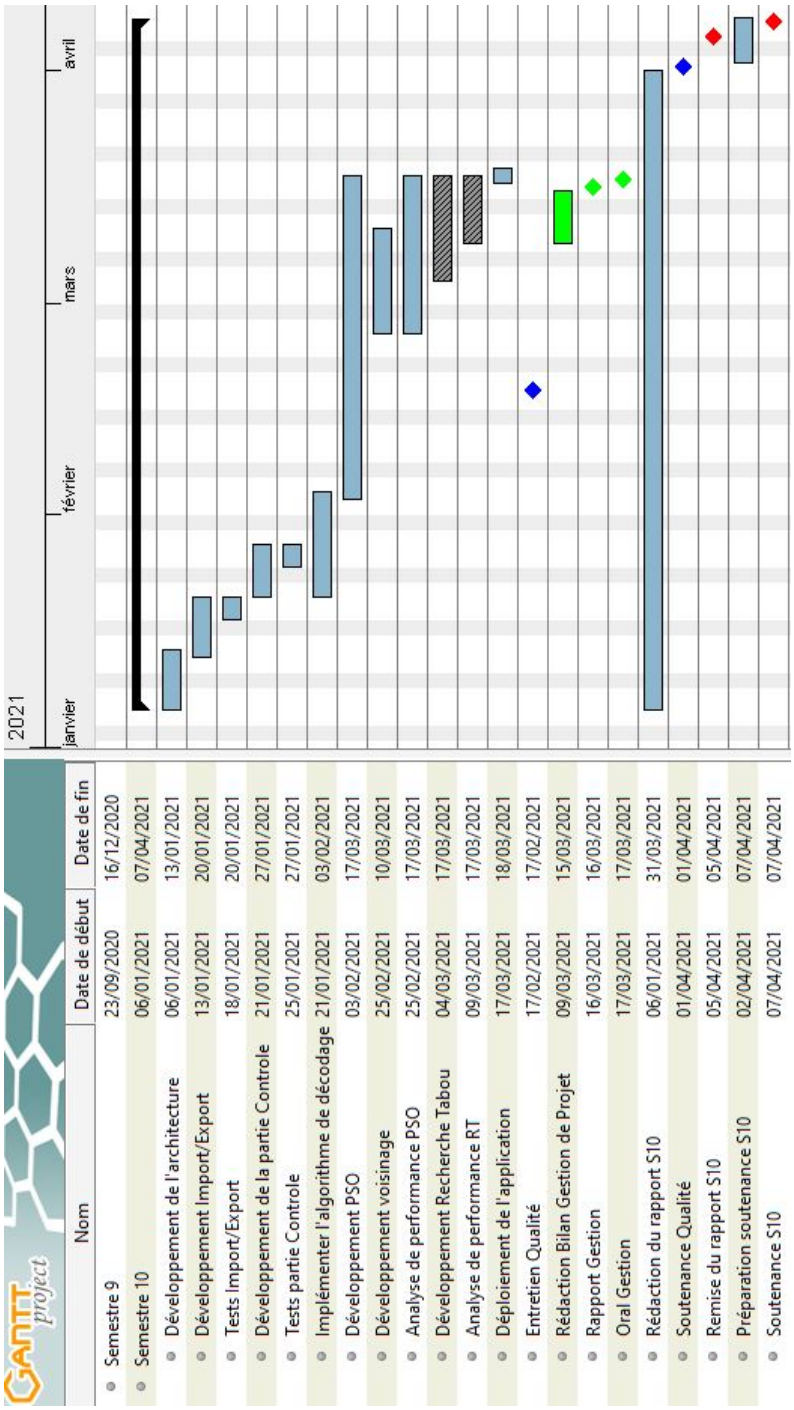


Figure A.6 – *Planning final du S10*

3 Tâches et livrables initiaux

Les livrables principaux sont sous plusieurs formes (écrit, oral, logiciel). Ainsi, les livrables sont les suivants :

- Rapport S9
- Cahier des spécifications
- Soutenance S9
- Rapport S10
- Soutenance S10
- Bilan de gestion de projet
- Logiciel de résolution avec algorithme Tabou
- Logiciel de résolution avec algorithme d'optimisation par essaims particulaires

4 Découpage des tâches

Tâche	Livrable	Estimation (jour/homme)	Deadline
Compréhension et rédaction du contexte du sujet	Contexte du sujet dans le rapport	4j/h	
Organisation du projet	Planning et plan d'évaluation des risques	1j/h	
Recherche sur des heuristiques et métaheuristiques pouvant répondre au problème étudié	Proposer des heuristiques et métaheuristiques au tuteur principal	4j/h	
Rédaction de l'état de l'art	Etat de l'art	6j/h	
Rédaction du cahier des spécifications	Cahier des spécifications	4j/h	
Rédaction du rapport S9 en continue	Rapport S9	12j/h	07/12/2020
Préparation soutenance S9	Soutenance avec slides	1j/h	16/12/2020
Mise en place du git et de l'architecture du code	Architecture du code	2j/h	
Réalisation de la fonctionnalité d'importation des instances	Fonctionnalité d'importation des instances	2j/h	
Réalisation de la fonctionnalité d'exportation des résultats	Fonctionnalité d'exportation des résultats	2j/h	

Figure A.7 – *Découpage des tâches*

Tâche	Livrable	Estimation (jour/homme)	Deadline
Tests des fonctionnalités d'import et d'export	Fonctionnalités d'import/export testés	1j/h	
Réalisation de l'algorithme de recherche Tabou	Algorithme de recherche Tabou	10j/h	
Analyse de performance de l'algorithme Tabou	Résultats et comparaisons	4j/h	
Réalisation de l'algorithme PSO	Algorithme PSO	10j/h	
Analyse de performance de l'algorithme PSO	Résultats et comparaisons	4j/h	
Réalisation du contrôle principal de l'application	Contrôle principal de l'application	4j/h	
Tests du contrôle de l'application	Contrôle testé	1j/h	
Déploiement de l'application	Programme .exe	2j/h	01/04/2021
Rédaction du rapport S10 en continue	Rapport S10	4j/h	
Rédaction du bilan de gestion de projet	Bilan de gestion de projet	1j/h	01/04/2021
Préparation soutenance S10	Soutenance avec slides	1j/h	07/04/2021

Figure A.8 – *Découpage des tâches*

5 Évaluation des risques

Voici un tableau récapitulatif des risques possibles associés à leur solution pour les anticiper :

Risque	Criticité	Probabilité	Prévention
Confinement	Moyen	Probable	Télétravail, sauvegarde des données sur un cloud. Microsoft Teams pour les réunions.
Blocage sur les outils de développement	Faible	Probable	Contacter un des tuteurs au plus vite.
Blocage lors du développement d'algorithmes	Faible	Probable	Étudier les raisons et le contour du problème puis se référer à un tuteur.
Coupure Internet	Haute	Rare	Faire des sauvegardes en local régulièrement. Utiliser une connexion 4G si possible.
Retard	Faible	Probable	Faire un historique des tâches réalisées afin d'identifier le problème. Se référer régulièrement au planning.
Perte de données locales	Haute	Rare	Sauvegarde des documents sur un Cloud (Google Drive + Overleaf (latex) + Github pour le code).

Figure A.9 – *Évaluation des risques*

6 Rapport de réunions

Voici le tableau récapitulatif des réunions :

Date	Sujet	Avec	Description	Elements validés	Tâches spécifiques à réaliser
23/09/2020	Introduction	M. Billaut	Introduction du sujet par le tuteur principal. Discussion sur la gestion du projet.		Etudier le sujet via les articles. Débuter le contexte du rapport
07/10/2020	Etat d'avancement	M. Billaut	Présentation du planning et du diagramme des cas d'utilisation. Revue du contexte, tâches à réaliser, plan du rapport. Questions sur le sujet.	Planning, diagramme des cas d'utilisations	Envoyer le rapport par mail pour validation du contexte
22/10/2020	Etat d'avancement	M. Billaut	Revue (Fait, à faire, planning, questions). Discussion sur la manière de réaliser l'état de l'art. Validation du plan global du rapport.		Envoyer le rapport par mail pour validation du contexte v2
05/11/2020	Gestion de projet	Mme Hoguet	Bilan sur la manière de gérer le PRD.	Gestion globalement correcte.	Identifier risques et solutions. Historique du planning et des réunions. Git.
05/11/2020	Etat d'avancement	M. Billaut	Revue (Fait, à faire, planning, questions).	Contexte du rapport	Rechercher des articles mêlant PSO et shop scheduling. Envoyer état de l'art pour validation
18/11/2020	Etat d'avancement	M. Billaut	Point sur l'avancement et sur le cahier des spécifications.		Envoyer le rapport pour validation état de l'art + plan état de l'art. Envoyer cahier des spés
25/11/2020	Entretien spécifications	M. Ramel	Point sur l'avancement du cahier des spécifications.	Spécifications globalement corrects.	Mettre en evidence les tests. Revoir les détails du livrables et cas d'utilisation
02/12/2020	Etat d'avancement	M. Billaut	Revue globale du rapport/spécifications. Validation de ces documents	Rapport, spécifications, planning prévisionnel S10	Demander à M. Chevronton une version de Tabou. Envoyer spé + rapport finaux.
16/12/2020	Soutenance S9	M. Billaut M.Kergosien Mme.Bidault	Soutenance évaluée du semestre 9		
21/01/2021	Etat d'avancement + études	M. Billaut	Point sur l'avancement au niveau du développement + Mise au point d'un algo de décodage + Planning = priorité PSO	algo de décodage	Implémenter l'algo de décodage en c++ et réaliser unn algo PSO
03/02/2021	Algo PSO	M. Billaut	Discussion et mise en place de la méthode de résolution de la PSO	Methode PSO	Développer l'algo PSO mis en place en C++
17/02/2021	Avancement + voisinage	M. Billaut	Point sur l'avancement + discussion sur les alogorithmes de voisinages	Algo Voisinage	Modifier le formats des fichier exportés + finir PSO en c++ + voisinage
25/02/2021	Avancement + optimisation	M. Billaut	Point sur l'avancement + analyse des performance de l'algorithme et optimisation de celui-ci		Optimiser certaines fonctions du code
12/03/2021	Avancement	M. Billaut	Point sur l'avancement et le déroulement de la fin du PRD + eventuel amélioration de l'algorithme + nouvelles solutions de référence		Inclure les solutions de références dans la PSO
17/03/2021	Oral Gestion de projet	Mme Hoguet	Discussion/entretien sur la gestion de projet pour évaluation		
01/04/2021	Oral Qualité de mise en oeuvre	M. Ramel	Discussion/entretien sur la qualité de mise en oeuvre du projet		

Figure A.10 – *Résumés de réunions*

7 Historique hebdomadaire

Semaine 1 : (23-24/9)

- Prise de connaissance du sujet via les articles et rapports
- Rendez-vous avec M. Billaut (tuteur principal) pour introduire le sujet
- Prise de connaissances de nouveaux articles suite au rendez-vous
- Prise de connaissance des modalités d'évaluation et des rapports à remettre en fin de semestre (soutenance + rapport + cahier des spécifications)
- Création d'un diagramme de cas d'utilisation du problème
- Définitions des termes techniques
- Début de rédaction du contexte pour le rapport

Semaine 2 :(30-1/9-10)

- Reprise de la rédaction du rapport
- Création du planning prévisionnel
- Création du plan du rapport
- Préparation de questions pour le prochain rendez-vous avec le tuteur (en semaine 3)
- Suite de la rédaction de la description générale

Semaine 3 :(7-8/10)

- Reprise de la rédaction de la description générale
- Réunion avec M. Billaut pour répondre aux questions concernant le sujet et valider certains aspect (planning, use case, contexte, tâche à réaliser, plan de l'état de l'art , tâches à réaliser, questions concernant le sujet) (7/10)
- Prise en compte des changements suite à la réunion + création d'un trello pour mieux organiser les tâches à réaliser
- Début de l'état de l'art (recherche d'articles avec un problème similaire)

Semaine 4 :(14-15/10)

- Correction du contexte en fonction des remarques du tuteur (correction, réorganisation du plan, ajout)
- Conférence concernant les spécifications et le déroulement du PRD avec M. Ramel et M. Ragot
- Création des illustrations (graphiques) - cas intéressant et à éviter
- Etude de l'algo de johnson
- Recherche d'articles concernant le problème (pour l'état de l'art)
- (2 conférences -> 4h -> une demi journée en moins)

Semaine 5 :(21-22/10)

- Prise de rdv pour la gestion de projet le 5/11/2020
- Organisation de la réunion du 22/10/2020
- Rédaction de l'état de l'art
- Réunion avec M.Billaut pour faire le point sur l'avancement et l'organisation, échanger des questions, revue du travail réalisé (contexte et état de l'art) et à faire. (22/10)

Semaine 6 : [confinement] (4-5/11)

- Suite Etat de l'art
- Récupération les travaux de Hugo Chevroton
- Début d'étude les travaux de Hugo Chevroton
- Correction du rapport
- Préparation entretien suivi de projet
- Entretien de suivi de projet (5/11)
- Mise à jour des méthodes d'organisation suite à l'entretien de suivi de projet
- Réunion avec M. Billaut pour l'état d'avancement de l'état de l'art / projet (5/11)

Semaine 7 :(11-12/11)

- Suite Etat de l'art
- Début de rédaction des spécifications
- Correction du rapport
- Diagramme de classes
- Organisation de la réunion du 18/11/2020

Semaine 8 : (18-19/11)

- Réunion avec M.Billaut pour faire le point sur l'avancement et les spécifications (18/11)
- Suite de la rédaction du cahier des spécifications
- Création d'un historique des plannings
- Suite du rapport + réorganisation
- Mise à jour du diagramme de classes

Semaine 9 : (25-26/11)

- Réunion avec M. Ramel pour faire un point sur les spécifications
- Mise à jour du planning en fonction des recommandations de M. Ramel (Tests)
- Mise à jour du diagramme de cas d'utilisation en fonction des recommandations de M.Ramel (modèle de négociation).
- Mise à jour du cahier des spécifications en fonction des recommandations de M. Ramel.
- Création du projet de développement sous Visual Studio
- Initialisation du dépôt git
- Finalisation du rapport et des spécifications
- Début de mise en forme LaTeX
- Conférence SOPRA (1h)
- Préparation de la réunion du 2 décembre
- Rédaction cahier de tests

Semaine 10 : (2-3/12)

- Réunion avec M. Billaut (2/12) sur les spécifications, les livrables et le rapport
- Prise en compte des remarques de M. Billaut dans le rapport et les spécifications
- Mise en forme du rapport en LaTeX
- Préparation des rendus pour le S9

Semaine 11 : (9-10/12)

- Finalisation de la rédaction du rapport
- Finalisation de la mise en forme du rapport
- Préparation de la soutenance de S9

Semaine 12 : (16-17/12)

- Soutenance S9
- Mise en place du projet de test
- Développement de l'importation des instances
- Développement des tests unitaire de l'importation des instances

Semaine 13 : (6-7/01)

- Développement de l'architecture complète (classes et héritage)
- Finalisation de l'importation des instances (import dossier)
- Tests unitaires pour l'importation validée

Semaine 14 : (13-14/01)

- Etat d'avancement avec questions par mail pour situer le projet à M.Billaut.
- Développement de l'exportation des résultats
- Développement des tests unitaires pour l'exportation
- Tests validés pour l'exportation des résultats
- Résultats exportables sur Excel

Semaine 15 : (20-21/01)

- Étude en profondeur des algos PSO
- Étude en profondeur des algos TS
- Rdv avec M.Billaut pour la PSO et l'algo de décodage

Semaine 16 : (27/01/01)

- Etude sur les algo PSO et adaptation pour notre problème
- Remodélisation de la structure des classes (refonte diagramme de classes)
- Développement de l'algo de décodage de solutions
- [Conférence informatique quantique (1h)]

Semaine 17 : (3-4/02)

- Reprise et finalisation de l'implémentation de l'algo de décodage en C++
- CM qualité et mise en oeuvre pour PRD (JYR-MD)
- Rendez-vous avec M. Billaut pour mettre en place un algorithme PSO
- Mise au propre des algorithmes pour la PSO

Semaine 18 : (10-11/02)

- Finalisation de la mise au propre des algorithmes pour la PSO
- Envoie de ces algorithmes pour validation par le tuteur
- Correction du décodage du vecteur Sv1
- Prise de rendez-vous pour qualité de mise en oeuvre
- Début du développement de la PSO en C++
- Mise à jour du diagramme de classes

Semaine 19 : (17-18/02)

- Reprise de la rédaction du rapport
- Préparation pour la réunion d'avancement avec M. Billaut
- Préparation de la réunion de qualité de mise en œuvre avec M. Ramel
- Réunion pour faire le point sur l'avancement avec M. Billaut
- Réunion pour discuter de la qualité de mise en œuvre avec M.Ramel
- Mise à jour du format des données exportées
- Tests du bon fonctionnement de la PSO avec les autres éléments du projet

Semaine 20 : (24-25/02)

- Semaine 20 : (24-25/02)
- Analyse de résultats avec différents paramètres pour la PSO
- Développement d'un résultat simple pour référence à la PSO
- [Réunion stage (1h)]
- Fonction pour vérifier que les contraintes sont respectées
- Réunion avec M. Billaut pour discuter de l'avancement et d'amélioration

Semaine 21 : (10-11/03)

- Analyse des sorties de la PSO
- Optimisation de code
- Réunion avec M. Billaut le 12/03 pour discuter de l'avancement et les résultats
- Rédaction du rapport de gestion de projet

Semaine 22 : (17-18/03)

- Préparation de l'oral de gestion de projet
- Oral de gestion de projet le 17/03 de 9h15 à 9h30
- Mise à jour du format des résultats en sortie pour la PSO
- Développement de nouvelles solutions de référence, ajoutées en tant que particules
- Debug d'un test non passant

Semaine 23 : (24-25/03)

- Packaging et réalisation propre du repo git pour réaliser une release de la première version du logiciel.
- Rédaction du rapport S10
- Analyse des résultats de la PSO

Semaine 24 : (31-1/03-04)

- Analyse des performances de la PSO
- Publication de la release finale sur git au propre
- Soutenance qualité de mise en œuvre avec M. Ramel
- Rédaction et finalisation du rapport S10

B Description des interfaces

1 Interfaces matérielles/logicielles

Pour pouvoir utiliser le logiciel, l'utilisateur doit utiliser un ordinateur (fixe ou portable) équipé d'un processeur assez haut de gamme afin d'obtenir des résultats sur de plus courtes périodes. Aucun autre besoin matériel supplémentaire n'est nécessaire.

2 Interfaces homme/machine

Les utilisateurs étant experts dans le domaine informatique, il n'est pas nécessaire de développer une interface homme machine (IHM, GUI). Le système d'exploitation fera office d'interface afin de pouvoir utiliser les algorithmes.

3 Interfaces logiciel/logiciel

Figure B.1 – *Logos respectifs de CPLEX et Microsoft Excel*

Il peut être nécessaire d'utiliser CPLEX lors du développement des algorithmes. Il est donc recommandé d'avoir CPLEX sur les machines.

Concernant l'analyse des résultats et l'interprétation des données en sorties, il est intéressant d'utiliser le logiciel Excel.

C Cahier de Spécification

1 Spécifications Fonctionnelles

Dans cette section, nous verrons en détails les différentes fonctionnalités du programme. Ces fonctionnalités sont au nombre de cinq :

- Importations des instances
- Exportation des données
- Résolution avec un algorithme de recherche Tabou
- Résolution avec un algorithme d'optimisation par essaims particulaires
- Contrôleur

1.1 Importations des instances

Cette fonctionnalité permet de lire les instances contenues dans un fichier texte. Un chemin vers le dossier contenant les instances doit être en paramètre de cette fonction. Cette fonction retourne une liste d'objets "Instance".

1.2 Exportation des données

Cette fonctionnalité permet de sauvegarder les résultats obtenus dans un fichier texte. Un chemin vers le dossier de destination doit être en paramètre de cette fonction.

1.3 Résolution avec un algorithme de recherche Tabou

Cette fonctionnalité est un algorithme de résolution du problème d'ordonnancement basé sur la métaheuristique de recherche Tabou.

Une instance doit être passée en paramètre de cette fonction.

Cette fonction retourne le résultat obtenu sous forme d'un objet "Result".

1.4 Résolution avec un algorithme d'optimisation par essaims particulaires

Cette fonctionnalité est un algorithme de résolution du problème d'ordonnancement basé sur une métaheuristique d'optimisation par essaims particulaires.

Une instance doit être passée en paramètre de cette fonction.

Cette fonction retourne le résultat obtenu sous forme d'un objet "Result".

1.5 Contrôleur

Cette fonctionnalité est la boucle principale du programme. Elle englobe les fonctionnalités précédentes et prends en argument deux paramètres :

- Le chemin du dossier contenant les instances
- Le répertoire de destination prévu pour le fichier contenant les résultats

Cette fonctionnalité a pour but d'utiliser les algorithmes de résolutions approchées afin de trouver la meilleure solution pour chaque instance donnée en entrée du programme. Les résultats sont ensuite sauvegardés sur un fichier texte.

2 Spécifications non fonctionnelles

Dans cette section, nous verrons les éléments non fonctionnelles du logiciel.

2.1 Contraintes de développement et conception

Cette partie regroupe les éléments qui constituent des contraintes pour le développement et la conception du logiciel.

2.1.1 Performances

Toujours dans l'optique de gagner en temps de calcul, il est recommandé de développer sur une machine performante.

2.1.2 Capacités

Afin de s'assurer que les algorithmes développés sont performants, ceux-ci devront être testés avec différentes instances de tailles différentes. Ces instances sont au nombre de 250 et ont été reprises du travail de M. Chevreton lors de sa thèse. Ainsi des comparaisons de performances pourront avoir lieu.

2.1.3 Contrôlabilité

Afin de s'assurer du bon déroulement des algorithmes, des logs d'exécution pourront être affichés pour suivre l'avancement du programme. Ces derniers peuvent être plus détaillés au cours du développement (Mode Debug).

2.1.4 Sécurité

Aucune contrainte de sécurité n'est imposée. L'utilisateur étant expérimenté dans le domaine informatique, celui-ci pourra donc avoir accès au code du logiciel.

2.2 Contraintes de développement et conception

Sachant que nous travaillons sur des algorithmes lourds et nécessitant des temps de calculs courts, il est recommandé d'utiliser un langage informatique rapide et performant. C'est pour cela que le C++ s'avère être le langage adapté pour la conception des logiciels.

Concernant l'environnement de développement, aucune contrainte n'est imposée. Cependant, les travaux réalisés par M. Chevroton ont été réalisés avec Visual Studio qui est très adapté pour le développement en C++ aujourd'hui.

Les programmes doivent pouvoir être exécutés sur le système d'exploitation Windows 10.

Figure C.1 – *Logos respectifs de C++, Visual Studio et Windows 10*

D Cahier de test

L'application finale devra être soumise à différents tests afin de garantir l'intégrité du logiciel ainsi que des performances.

1 Tests d'intégration

Différents tests manuels sont nécessaires afin de vérifier le comportement de l'application :

ID	Test	Description	Résultat attendu
EXE1	Exécution avec un script batch	Exécution avec un dossier comprenant des instances avec le bon format	Aucune exception
EXE2		Exécution avec un dossier comprenant des instances avec au moins un mauvais format	Message d'erreur dans la console : "bad format"
RES	Fichier résultat	Le contenu du fichier résultat sera copié et collé dans un tableur Excel	Aucun problème avec le format des cellules

Figure D.1 – *Tests d'intégration*

2 Tests unitaires

Des tests unitaires automatisés seront à réaliser pour les composants de l'application suivant :

ID	Test	Description / Scénario	Résultat attendu
IF1	Import d'un fichier d'instance	Importer un fichier contenant une instance avec le bon format	Aucune exception
IF2		Importer un fichier contenant une instance avec un mauvais format (format différent de IF3)	Exception "bad format" attendu
IF3		Importer un fichier contenant une instance avec un mauvais format (format différent de IF2)	Exception "bad format" attendu
ID1	Import d'un dossier comprenant des fichiers d'instances	Importer un dossier contenant des instances avec le bon format	Aucune exception
ID2		Importer un fichier contenant au moins une instance avec un mauvais format (format différent de ID3)	Exception "bad format" attendu
ID3		Importer un fichier contenant au moins une instance avec un mauvais format (format différent de ID2)	Exception "bad format" attendu
EF	Export d'un fichier résultat	Exporter un fichier contenant plusieurs résultats et vérifier le format	Le format doit être correct
CT1	Respect des contraintes	Vérifications si les contraintes du modèle mathématiques sont respectés après résolution du problème avec un algorithme de recherche Tabou	Aucune erreur
CT2		Vérifications si les contraintes du modèle mathématiques sont respectés après résolution du problème avec un algorithme d'optimisation par essaim particulaire	Aucune erreur

Figure D.2 – *Tests unitaires*

3 Résultats des tests unitaires

Les différents tests unitaires ont tous été passés avec succès avec le projet de tests sous Visual Studio.

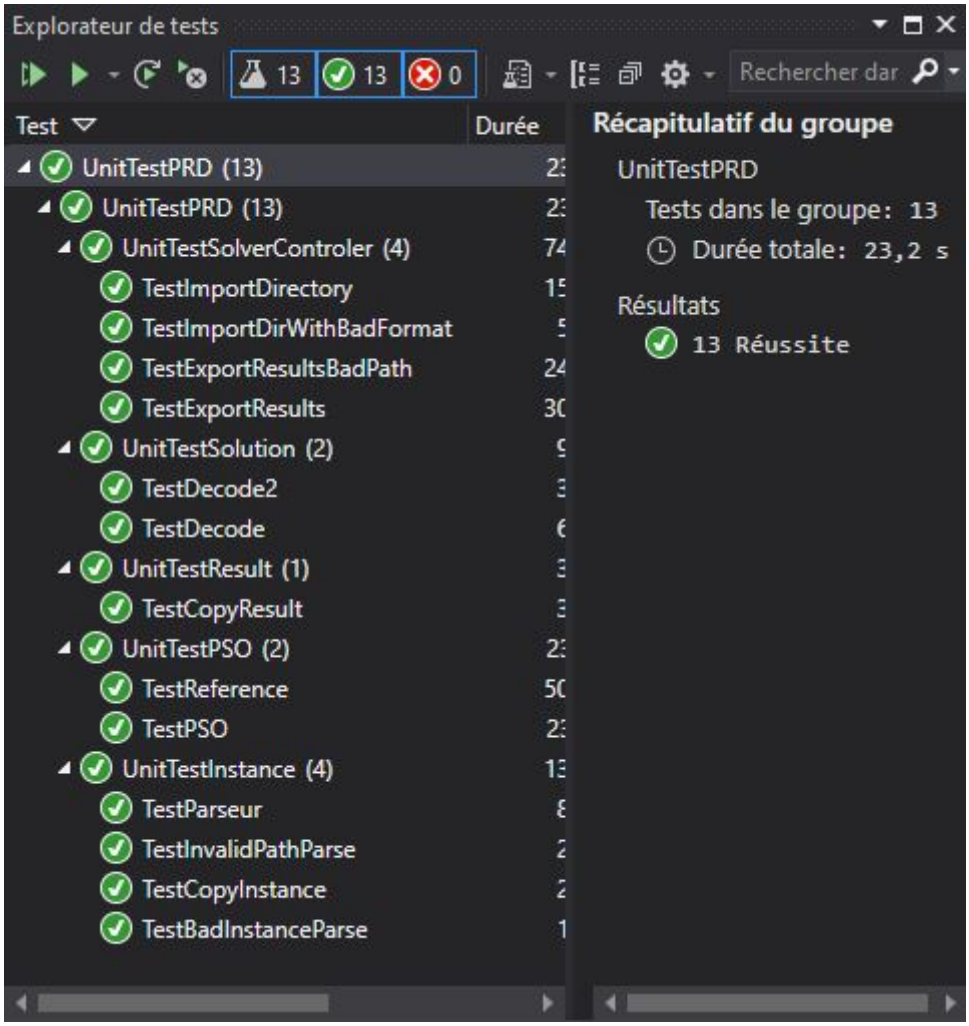


Figure D.3 – *Résultats tests unitaires*

E Bibliographie

[1] H. Chevroton, S. Rohmer et J-C Billaut. « A production and distribution framework : manufacturer dominates ». In : *Computers And Industrial Engineering CAIE* (2020).

[2] Jianyu Long, Panos M. Pardalos et Chuan Li. « Level-based multi-objective particle swarm optimizer for integrated production scheduling and vehicle routing decision with inventory holding, delivery, and tardiness costs ». In : *International Journal of Production Research* (2020).

[3] Luh, Zhou et Tomastik. « An Effective Method to Reduce Inventory in Job Shops ». In : *Transactions on robotics and automation, vol 16, no 4. 787-792* (1999).

[4] Morais, Boiko, Coelho, Rocha et Paraíso. « Multicriteria Hybrid Flow Shop Scheduling problem : literature review, analysis, and future research ». In : *Independent journal of management production, 5(4), 1004-1031* (2014).

[5] M. Reimann, K. Doerner et R. F. Hartl. « D-Ants : Savings Based Ants divide and conquer the vehicle routing problem ». In : *Computers Operations Research 31, 563–591* (2004).

[6] Tseng et Liao. « A discrete particle swarm optimization for lot-streaming flowshop scheduling problem ». In : *European Journal of Operational Research 191. 360–373* (2007).

[7] Christian Viergutz et Sigrid Knust. « Integrated production and distribution scheduling with lifespan constraints ». In : *Annals of Operations Research 213,293–318* (2012).

[8] Yang et Posner. « Flow shops with WIP and value added costs ». In : *Journal of Scheduling. 13(1). 3-16.* (2009).

F Glossaire

CPLEX Solveur commercial haute performance de programmation linéaire. 47

GUI Graphic User Interface - Interface Homme/Machine. 47

Heuristique/Métaheuristique Méthode approchée de résolution de problèmes NP-difficile. 3

IHM Interface Homme/Machine. 47

PRD Projet Recherche et Développement. 2

PSO/OEP Optimisation par Essaims Particulaires / Particle Swarm Optimization. Métaheuristique d'optimisation qui se base sur la collaboration des individus entre eux. 15

Etude d'un modèle de chaîne logistique multi-acteurs

Résumé

Ce rapport restitue l'ensemble du travail et des recherches réalisés pour le projet recherche et développement à Polytech Tours. Le sujet de ce projet est l'étude d'un modèle de chaîne logistique multi-acteurs. C'est un problème d'optimisation qui mêle un distributeur, un producteur et des clients. Le producteur réalise des commandes qui sont ensuite livrées par un distributeur tiers. L'objectif est de pouvoir ordonnancer la production de manière à minimiser l'ensemble des coûts liés aux stockages et aux retards. Ce problème d'optimisation ne pouvant pas se résoudre en un temps raisonnable avec des méthodes exactes, des méthodes approchées doivent être réalisés de façon à obtenir la meilleure solution possible en un minimum de temps. Un algorithme d'optimisation par essaims particulaires sera étudié.

Mots-clés

chaîne logistique, multi-acteurs, optimisation, ordonnancement, flow-shop, coûts d'inventaire, recherche Tabou, optimisation par essaims particulaires, GRASP, Algorithme génétique

Abstract

This report contains all the work and research done for the research and development project at Polytech Tours. The subject is a study on supply chain with multiple agents. This optimization problem integrates a manufacturer, a third-party logistic provider (3PL provider) and clients. The manufacturer produces some orders and the 3PL provider delivers them to the clients. The main objective is to schedule the production in order to minimize the whole costs related to the inventory and delays. Because this problem could not be resolved with an exact method in a reasonable time, some approximate methods should be done to find the best solution in a minimum time. A Particle Swarm Optimization algorithm will be studied.

Keywords

supply chain, multiple agents, optimization, scheduling, flow-shop, inventory costs, Tabou Search, Particle Swarm Optimization, PSO, GRASP, Genetic Algorithm

Entreprise
Polytech

POLYTECH TOURS

Tuteur entreprise
Jean-Charles BILLAUT

Étudiant
Yohan DERENNE (DI5)

Tuteurs académiques
Jean-Charles BILLAUT
Yannick KERGOSIEN