UNIVERSITE D'ABOMEY CALAVI (BENIN)

THE ABDUS SALAM INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS (ITALY)

INSTITUT DE MATHEMATIQUES ET DE SCIENCES PHYSIQUES

Centre d'Excellence d'Afrique en Sciences Mathématiques, Informatique et Applications
Adresse mail : secretariat@imsp-uac.org

MEMOIRE DE MASTER 2

FILIÈRE : RECHERCHE OPÉRATIONNELLE ET AIDE À LA DÉCISION

Thème

Optimisation des coûts dans le Beer Game

Rédigé par :

METSANOU TSAFACK DEXTER STEPHANE
...
Adresse mail : dexter.metsanou@imsp-uac.org

Superviseur :

PROFESSEUR MAMADOU KABA TRAORE
(IMS Lab, Université de Bordeaux, France)
Adresse mail : mamadou-kaba.traore@u-bordeaux.fr

Membres du Jury:

PROF . GUY DEGLA
DR . FRANCK HOUENOU

Soutenu le 31/08/2023

B.P. : 613 PORTO-NOVO

REPUBLIQUE DU BENIN

Dédicaces

Je dédie ce travail :

À toute ma famille, source d'espoir et de motivation.

Remerciements

Je tiens à remercier mon superviseur, le Professeur Mamadou Kaba TRAORE pour ses conseils et son appui le long de la rédaction de ce projet. Sa disponibilité, ses grandes connaissances ainsi que son expérience m'ont été précieux tout au long de la rédaction de ce mémoire. Je remercie également le responsable de la formation recherche opérationnelle de l'IMSP le Professeur Guy DEGLA, pour son soutien tout au long de ma formation. Une mention particulière au personnel du corps enseignant, en particulier au directeur de l'IMSP le Professeur Carlos OGOUYANDJOU pour leur précieuse contribution à ma formation. Je remercie aussi le Centre d'Excellence d'Afrique en Science Mathématiques, Informatique et Application (CEA-SMIA) pour son soutien financier au cours de ce master. J'aimerais aussi exprimer mes gratitudes envers mes frères, sœurs ainsi qu'à mes amis.

Résumé

Ce travail se concentre sur l'optimisation des coûts dans le Beer Game, un modèle pédagogique représentant une chaîne d'approvisionnement. L'objectif principal est de développer une solution mathématique basée sur une approche de recherche opérationnelle pour réduire les coûts opérationnels et améliorer la performance globale du système. Pour cela, nous combinons la modélisation mathématique, les méthodes d'optimisation et les simulations numériques du Beer Game. Nous explorons les recherches antérieures sur l'optimisation des coûts dans les chaînes d'approvisionnement ainsi que les études liées au Beer Game. En concevant un modèle mathématique précis à la chaîne d'approvisionnement du Beer Game, nous définissons les paramètres de coûts essentiels, tels que les coûts de stockage et de rupture de stock. Nous étudions ensuite différentes méthodes d'optimisation et sélectionnons celle qui est la mieux adaptée à notre modèle. Par le biais de simulations, nous menons des expérimentations pour évaluer les performances de chaque stratégie d'optimisation en termes de coûts opérationnels. Cette étude contribue ainsi à une meilleure compréhension des défis de gestion de la chaîne d'approvisionnement et fournit des pistes concrètes pour les professionnels souhaitant améliorer l'efficacité de leur chaîne logistique.

Mots-clés : Beer Game, Chaîne d'approvisionnement, Modélisation mathématique, Optimisation, Simulation.

Abstract

This dissertation focuses on cost optimization in the Beer Game, an educational model representing a supply chain. The main objective is to develop a mathematical solution based on an operations research approach to reduce operational costs and improve overall system performance. To achieve this, we combine mathematical modeling, optimization methods and numerical simulations of the Beer Game. We explore previous research on cost optimization in supply chains, as well as studies related to the Beer Game. By designing an accurate mathematical model of the Beer Game supply chain, we define key cost parameters, such as storage and out-of-stock costs. We then study different optimization methods and select the one best suited to our model. Using simulations, we conduct experiments to assess the performance of each optimization strategy in terms of operating costs. This study thus contributes to a better understanding of supply chain management challenges, and provides concrete avenues for professionals wishing to improve the efficiency of their supply chain.

Keywords : Beer Game, Supply chain, Mathematical modeling, Optimization, Simulation.

Table des matières

Liste des tableaux

Table des figures

Introduction

La gestion efficace des chaînes d'approvisionnement est un enjeu crucial pour les entreprises opérant dans des environnements concurrentiels et complexes. L'interaction dynamique entre les différents acteurs de la chaîne, tels que les fabricants, les distributeurs, les grossistes, les détaillants et les clients finaux, peut entraîner des fluctuations de la demande, des coûts opérationnels élevés et des ruptures de stock. Le Beer Game, également connu sous le nom de jeu de la bière, est un modèle pédagogique emblématique utilisé pour illustrer ces défis et sensibiliser les acteurs de la chaîne d'approvisionnement à l'importance de la coordination et de la gestion efficiente des stocks. L'objectif central de notre mémoire de recherche est d'étudier l'optimisation des coûts dans le Beer Game. Face aux complexités de la chaîne d'approvisionnement, notre recherche vise à développer des stratégies permettant de réduire les coûts opérationnels tout en améliorant la performance globale du système. Pour atteindre cet objectif, nous adoptons une approche rigoureuse et multidisciplinaire combinant la modélisation mathématique, les méthodes d'optimisation et les simulations du jeu de la bière. Dans un premier temps, nous présenterons en détail le jeu de la bière, en mettant en exergue son contexte, ses enjeux et ses principales caractéristiques. Ensuite, nous explorerons la littérature portant sur l'optimisation des coûts dans les chaînes d'approvisionnement. La mise en œuvre de notre étude se concrétisera par la conception d'un modèle mathématique, qui reflète fidèlement la dynamique complexe de la chaîne d'approvisionnement du Beer Game. Nous identifierons et définirons les paramètres clés relatifs aux coûts, notamment les coûts de stockage et de rupture de stock. Notre recherche se concentrera ensuite sur l'identification et l'adaptation des méthodes d'optimisation les plus adaptées à notre modèle, parmi une gamme variée d'approches telles que les heuristiques, les algorithmes génétiques et la programmation non linéaire. La phase d'expérimentation sera cruciale pour évaluer la performance des différentes stratégies d'optimisation. Après simulation, nous analyserons les résultats obtenus et évaluerons l'impact de chaque stratégie sur les coûts opérationnels de la chaîne d'approvisionnement.

CHAPITRE 1

REVUE DE LITTÉRATURE

Introduction

La logistique joue un rôle fondamental en entreprise, car elle englobe toutes les activités visant à assurer la disponibilité des biens et services là où la demande existe, tout en optimisant la gestion des quantités, des délais et des coûts. Cette fonction ne se limite pas uniquement à l'organisation des transports, des matières premières et des marchandises, mais elle englobe également l'ensemble des techniques de contrôle et de gestion des flux de matières premières et de produits, depuis leurs sources d'approvisionnement jusqu'au point de consommation. L'évolution de la logistique, ainsi que l'implication de ses divers acteurs, ont donné naissance au concept de la chaîne logistique (supply chain). Cette dernière joue un rôle crucial dans le fonctionnement global de l'entreprise, s'étendant du fournisseur du fournisseur jusqu'au client du client, en englobant la fabrication et le stockage des produits en amont et en aval. Dans ce chapitre, nous allons tout d'abord définir les notions de base de logistique, ensuite nous allons présenter le Beer Game et enfin nous allons présenter quelques stratégies d'optimisation de coûts dans une chaine logistique.

1 Chaine logistique

La Supply Chain (chaîne logistique), est le processus qui est généré lorsqu'un client passe une commande jusqu'à ce que le produit ou le service soit livré et payé. Par conséquent, la Supply Chain comprend la planification, l'exécution et le contrôle de toutes les activités liées aux flux de matériaux et d'information, à l'achat de matières premières, à la transformation intermédiaire du produit ainsi qu'à sa la livraison au client final.

Définition 1.1. ***La Supply Chain*** *est l'ensemble des étapes et des réseaux utilisés par un produit, dès sa fabrication jusqu'à son arrivée au client final. Elle se compose de plusieurs fournisseurs et entreprises qui alimentent chaque maillon de la chaîne[2].*

De nos jours, les Supply Chains sont devenues très complexes en raison de l'internationalisation, de l'augmentation des types de flux et de l'évolution des modes de consommation mondiaux.

La Supply Chain se divise en trois étapes principales :

— **L'approvisionnement** : il s'agit de savoir comment, où et quand les matières premières sont obtenues et fournies pour la fabrication des produits.

— **La production** : elle comprend la fabrication des produits finis à l'aide de matières premières.

— **La distribution** : cette phase concerne les activités réalisées afin que les produits atteignent leur destination finale. Elle s'effectue par le biais d'un réseau de grossistes, d'entrepôts, de magasins physiques ou de plateformes en ligne (pour les entreprises e-commerce).

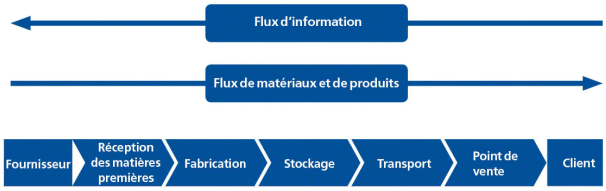

FIGURE 1.1 – Supply Chain [2]

Définition 1.2. ***La logistique*** *est le secteur d'activité responsable d'acheminer le bon produit à la destination indiquée, dans les délais impartis, dans les quantités et coût requis, et en bon état. Elle vise à respecter les conditions convenues au préalable avec le client[2].*

1.1 Différences entre la logistique et la Supply Chain

La différence entre la logistique et la chaine logistique est consignée selon les axes ci-dessous :

Objectifs

— Supply Chain : Atteindre un taux de compétitivité maximal sur le marché et optimiser les bénéfices.

— Logistique : Satisfaire le client grâce à une gestion parfaite des commandes.

Entreprises impliquées

— Supply Chain : Plusieurs organisations sont généralement impliquées dans une même Supply Chain.

— Logistique : Peut être entièrement gérée par une même société.

Relation entre chaque domaine

— Supply Chain : Désigne l'écosystème des processus qui aboutissent à un produit.

— Logistique : Englobe une partie des opérations de la Supply Chain.

Départements d'entreprise concernés

— Supply Chain : Elle comptabilise plus de départements que la gestion logistique : le développement des produits, le contrôle qualité, le service client, les activités logistiques, etc.

— Logistique : Elle regroupe principalement le stockage, le transport et la gestion des stocks.

2 Le Beer Game et de ses enjeux :

Dans le contexte de la gestion de la chaîne d'approvisionnement et de la logistique, où les défis peuvent être particulièrement complexes, une approche immersive s'avère essentielle pour une compréhension approfondie. C'est précisément dans cette optique que le Beer Game, également désigné sous le nom de jeu de la bière a été mis sur pieds.

2.1 Le Beer Game

2.1.1 Les acteurs

Le jeu de la bière, également connu sous le nom de Beer Game, est un modèle pédagogique développé par le professeur Jay Forrester du MIT en 1961 [1]. Il s'agit d'un jeu de rôle basé sur une chaîne d'approvisionnement simplifiée, comprenant quatre acteurs clés (**acteurs internes du système**) : **le fabricant (F), le distributeur (D), le grossiste (W),le détaillant (R)** et un **acteur externe au système : le client final**.

- **Le fabricant :** Responsable de la production de la bière. Il doit décider du volume de production en fonction des commandes reçues du **distributeur**. Le fabricant doit également gérer ses stocks pour répondre à la demande du **distributeur**.
- **Le distributeur :** Reçoit la bière du **fabricant** et est responsable de la distribution au **grossiste**. Il prend des décisions concernant les quantités de bière à **commander au fabricant en fonction de la demande de grossiste.**
- **Le grossiste :** Reçoit la bière du **distributeur** et est responsable de la distribution au **détaillant**. Il prend des décisions concernant les quantités de bière à **commander au distributeur en fonction de la demande de détaillant.**
- **Le détaillant :** Reçoit la bière du **grossiste** et vend le produit au **client final**. **Le détaillant doit décider du volume à commander auprès du grossiste en fonction de la demande du client final.**
- **Le client final** : Il représente la demande réelle pour la bière. Sa demande est transmise de manière séquentielle le long de la chaîne, passant du détaillant au grossiste, puis au distributeur, puis au fabricant.

2.1.2 Le fonctionnement

Chaque acteur (secteur) de la chaîne d'approvisionnement prend des décisions de réapprovisionnement périodiquement, en fonction de la demande qu'il reçoit[5]. A chaque tournée, chacun d'eux effectue les taches suivantes :

- Vérifier le stock de bière reçu de la part de son distributeur.
- Vérifier les commandes reçues de la part de son client.
- Livrer son client (dans la limite de stock disponible).
- Passer une commande à son fournisseur.

Les acteurs n'ont pas de visibilité complète sur la demande réelle et ne peuvent baser leurs décisions que sur les informations transmises par le maillon précédent. Cela crée un effet de coup de fouet (phénomène de chaine d'approvisionnement qui induit une amplification de la variabilité de la demande [8]), où de petites variations de la demande au niveau du client final sont amplifiées à mesure que l'on remonte la chaîne d'approvisionnement.

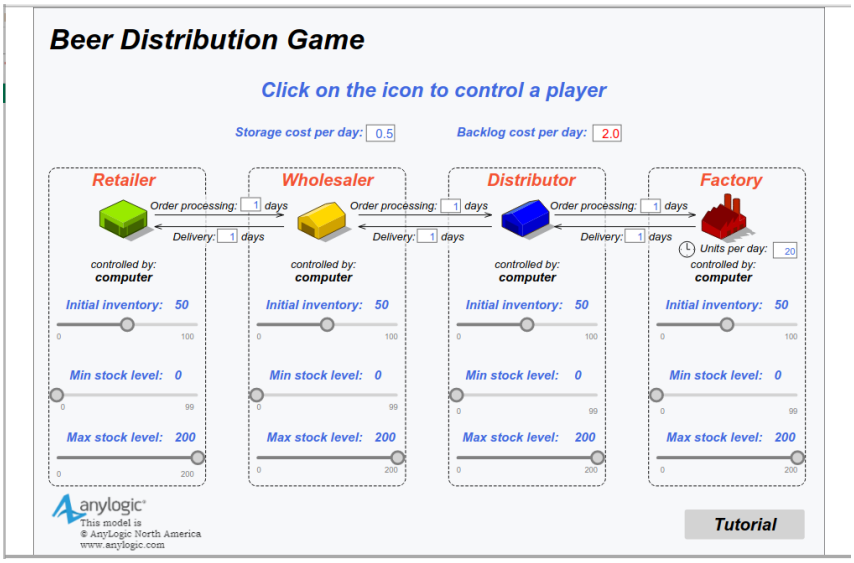

FIGURE 1.2 – Beer Game[14]

2.2 Enjeux du Beer Game

Les enjeux du Beer Game résident dans les problématiques de coordination et de gestion de la chaîne d'approvisionnement. En raison des retards de communication, des variations de la demande et de l'effet de coup de fouet (bullwhip effect[8]), de petites fluctuations de la demande peuvent être amplifiées et se traduire par des ruptures de stock, des sur-stocks et des coûts élevés tout au long de la chaîne .

Les décisions prises par chaque acteur peuvent être caractérisées par un comportement appelé "effet de coup de fouet". Ce phénomène désigne une amplification des variations de la demande au fur et à mesure que l'on remonte la chaîne d'approvisionnement, ce qui conduit souvent à des distorsions importantes entre la demande réelle et les commandes passées [8].

Les conséquences de cet effet de coup de fouet sont multiples :

— Des niveaux de stock inutilement élevés.

— Des ruptures de stock fréquentes dans la chaîne d'approvisionnement.

— Des coûts supplémentaires pour les acteurs impliqués, et une diminution globale de la performance du système.

En somme, le jeu de la bière permet de mettre en évidence les défis de coordination et de gestion de la chaîne d'approvisionnement, ainsi que les conséquences de l'effet de coup de fouet sur les coûts et la performance globale du système.

3 Optimisation des coûts dans les chaînes d'approvisionnement

L'optimisation des coûts dans les chaînes d'approvisionnement est un sujet d'intérêt majeur pour les chercheurs et les professionnels en logistique et gestion de la chaîne d'approvisionnement. De nombreuses études ont été menées pour développer des approches et des stratégies visant à minimiser les coûts opérationnels tout en maintenant ou améliorant la performance globale de la chaîne. Nous pouvons citer Khadija Eddoug et Saâd Lissane El Haq qui ont travaillees sur l'optimisation conjointe des coûts de transport et de stock dans une chaîne logistique de distribution multi niveaux[9], Bidyut Biman Sarkara, Nabendu Chaki dont les travaux portaient sur le jeu de la bière au détail distribué pour un système d'aide à la décision [4] et Afshin Oroojlooy, MohammadReza Nazari, Lawrence Snyder, Martin Takáč qui ont utilises le machine learning pour résoudre un problème d'optimisation d'inventaire [3]. Afin d'optimiser ces coûts, nous devons définir des stratégies d'approvisionnement et de gestion de stock.

3.1 Stratégies

3.1.1 La stratégie de gestion des stocks en flux tendu

Le flux tendu, également connu sous le nom de juste à temps, est une méthode de gestion et d'organisation des stocks où l'entreprise attend les commandes de ses clients avant de procéder à l'approvisionnement. Ainsi, elle passe commande pour les matières premières et les éléments à monter uniquement lorsqu'ils sont nécessaires. L'objectif du flux tendu est de réduire, voire éliminer, les stocks intermédiaires afin de minimiser les coûts de stockage tout en évitant le gaspillage dû à la péremption des produits.

En raison de l'importance de la gestion des stocks, il est crucial que l'entreprise, ses clients et ses fournisseurs soient en accord sur la méthode de réapprovisionnement. Pour optimiser une gestion de stock en flux tendu, l'entreprise doit respecter certaines règles essentielles :

— Définir avec précision ses besoins en production.

— Privilégier les approvisionnements locaux pour réduire les coûts de transport.

— Assurer une logistique efficace pour le transport et la livraison des matériaux et produits finis.

— Mettre en place des horaires flexibles pour s'adapter aux fluctuations de la demande.

— Avoir une gestion optimale des stocks et des commandes pour maintenir l'équilibre entre l'offre et la demande

En adoptant le flux tendu de manière rigoureuse et en respectant ces règles, l'entreprise peut bénéficier d'une meilleure efficacité opérationnelle, d'une réduction des coûts liés aux stocks et d'une plus grande réactivité pour répondre à la demande du marché. Cependant, il est important que l'entreprise prenne en compte les risques potentiels tels que les retards de livraison ou les fluctuations de la demande, et qu'elle développe des plans d'urgence pour y faire face de manière proactive.

3.1.2 La stratégie de gestion des stocks en partage d'information

En partageant des informations en temps réel sur les niveaux de stock, les prévisions de demande et les délais de livraison, les acteurs de la chaîne peuvent mieux anticiper les variations de la demande et réagir de manière proactive. Cela permet de réduire les incertitudes et de minimiser les coûts liés aux ruptures de stock et aux sur-stocks. Cette stratégie est caractérisée de la manière suivante :

— Transparence et collaboration : Les différents acteurs partagent volontairement leurs données de stock et d'autres informations importantes. Cela favorise une meilleure compréhension des besoins de chaque maillon de la chaîne et permet une prise de décision collective et éclairée.

— Prise de décision réactive : En partageant des informations en temps réel, les acteurs peuvent réagir rapidement aux fluctuations de la demande, aux retards de livraison ou aux changements dans l'environnement commercial.

— Réduction des incertitudes : Le partage d'informations permet de réduire les incertitudes et les imprévus liés aux variations de la demande ou aux délais de livraison, ce qui contribue à une meilleure planification des stocks.

— Amélioration de la précision des prévisions : Grâce aux données partagées, les prévisions de vente et de demande peuvent être affinées, ce qui réduit les risques de surstockage ou de rupture de stock.

— Optimisation des niveaux de stock : En ayant une vue globale de la chaîne d'approvisionnement, les acteurs peuvent ajuster plus précisément leurs niveaux de stock pour éviter les excédents ou les pénuries.

— Gestion des aléas : Le partage d'informations facilite la mise en place de plans d'urgence et de stratégies d'atténuation des risques en cas de perturbations dans la chaîne d'approvisionnement.

Il est important de noter que le partage d'informations nécessite une confiance mutuelle entre les partenaires commerciaux et peut être sensible en termes de confidentialité des données. Des accords contractuels et des mécanismes de sécurisation des données doivent être mis en place pour assurer la protection des informations stratégiques.

3.1.3 La stratégie d'approvisionnement collaboratif

En mettant en place une approche collaborative entre les différents maillons de la chaîne, les acteurs peuvent coopérer pour rationaliser les commandes, optimiser les livraisons groupées et partager les coûts logistiques. Cette approche peut entraîner des économies d'échelle et réduire les coûts totaux de la chaîne. Cette stratégie est caractérisée de la maniere suivante :

— Partenariat à long terme : La stratégie d'approvisionnement collaboratif encourage les relations à long terme entre les fournisseurs et les clients. Plutôt que de se concentrer uniquement sur les transactions à court terme, cette approche vise à établir des partenariats solides et mutuellement bénéfiques.

— Partage d'informations : Le partage transparent d'informations est au cœur de la collaboration. Les fournisseurs et les clients partagent des données telles que les prévisions de la demande, les niveaux de stock, les coûts, les contraintes de production, les innovations techniques, etc. Cela permet une meilleure compréhension des besoins et des contraintes de chaque partie et facilite une prise de décision éclairée.

— Co-développement et innovation : Les partenariats collaboratifs favorisent la co-création de produits et services innovants. Les fournisseurs et les clients travaillent ensemble pour développer de nouvelles solutions, améliorer les produits existants et innover dans les processus d'approvisionnement.

— Gestion des risques : La collaboration permet de mieux anticiper et gérer les risques dans la chaîne d'approvisionnement. Les partenaires travaillent ensemble pour identifier les vulnérabilités et développer des plans d'atténuation des risques.

— Flexibilité et agilité : La stratégie d'approvisionnement collaboratif offre une plus grande flexibilité pour s'adapter aux fluctuations de la demande, aux changements

de marché et aux imprévus. Les partenaires peuvent ajuster rapidement leurs processus et capacités pour répondre aux besoins changeants.

— Performance globale : En encourageant la collaboration, cette approche vise à améliorer la performance globale de la chaîne d'approvisionnement, en optimisant les coûts, en réduisant les délais, en améliorant la qualité des produits et services, et en augmentant la satisfaction des clients.

La stratégie d'approvisionnement collaboratif peut être bénéfique tant pour les fournisseurs que pour les clients, car elle favorise une approche gagnant-gagnant. Cependant, elle nécessite une confiance mutuelle, un engagement à long terme et une communication ouverte entre les partenaires. Des accords contractuels clairs et des mécanismes de résolution des conflits peuvent être mis en place pour soutenir la collaboration et garantir le respect des engagements pris par chacune des parties prenantes.

3.1.4 Les stratégies d'optimisation basées sur la modélisation mathématiques et des algorithmes :

Ces stratégies sont des approches de résolution de problèmes qui utilisent des techniques mathématiques et informatiques pour trouver les meilleures solutions possibles dans un espace de recherche donné. Ces algorithmes cherchent à optimiser une fonction objectif en explorant différentes combinaisons de variables ou de paramètres pour atteindre un résultat optimal, c'est-à-dire maximiser ou minimiser la fonction objectif en fonction des contraintes spécifiées.

Il existe plusieurs types d'algorithmes d'optimisation, chacun avec ses propres caractéristiques et domaines d'application. Quelques-unes des stratégies d'optimisation basées sur des algorithmes couramment utilisées sont :

— Algorithmes génétiques (AG) : Les algorithmes génétiques sont inspirés du processus d'évolution naturelle et utilisent des opérations de sélection, de croisement et de mutation pour explorer l'espace des solutions possibles. Ils commencent par une population d'individus représentant des solutions potentielles et évoluent au fil des générations pour trouver une solution optimale.

— Recherche tabou (RT) : La recherche tabou est une méthode d'optimisation qui consiste à explorer l'espace des solutions en évitant les mouvements répétitifs qui pourraient

conduire à des cycles. Elle maintient une liste de solutions interdites (liste tabou) pour éviter de revisiter des solutions précédemment explorées.

— Recuit simulé (RS) : Le recuit simulé est inspiré du processus de refroidissement des métaux, où un matériau est chauffé puis refroidi lentement pour obtenir une structure cristalline stable. L'algorithme simule ce processus en explorant l'espace des solutions en fonction d'une température qui diminue progressivement.

— Optimisation par essaims particulaires (PSO) : Le PSO s'inspire du comportement social des essaims d'oiseaux ou d'insectes. Les particules représentant des solutions potentielles se déplacent dans l'espace de recherche en fonction de leur meilleure solution trouvée localement et de la meilleure solution globale du groupe.

— Colonies de fourmis (ACO) : L'ACO est inspiré du comportement des fourmis cherchant de la nourriture. Les fourmis communiquent en déposant des phéromones sur le chemin vers la nourriture, ce qui crée des pistes phéromonales. L'algorithme simule ce processus pour trouver un chemin optimal vers une solution.

— Programmation linéaire (PL)/Programmation non linéaire(PNL) : La PL/PNL est une méthode d'optimisation utilisée pour résoudre des problèmes avec une fonction objectif linéaire/non linéaire et des contraintes linéaires/non linéaires. Elle est couramment utilisée dans les problèmes de gestion des stocks, de planification de la production et de routage.

Une des approches courantes pour l'optimisation des coûts dans les chaînes d'approvisionnement est l'utilisation de modèles mathématiques et d'outils d'optimisation. Ces modèles sont conçus pour prendre en compte différentes variables telles que les coûts de stockage, les coûts de transport, les coûts de production et les contraintes spécifiques à chaque chaîne. Ils permettent d'identifier les politiques d'approvisionnement, de production et de distribution qui minimisent les coûts tout en respectant les contraintes opérationnelles.

Ces différentes stratégies ont été largement explorées dans des jeux similaires au Beer Game tels que le Pizza Game qui simule la chaîne d'approvisionnement d'une pizzeria. Les participants incarnent différents rôles, de la production des ingrédients à la livraison des pizzas [6], le Risk Pool Game qui se concentre sur la collaboration entre fournisseurs et distributeurs pour gérer les variations de la demande [7], permettant ainsi de mieux comprendre leur efficacité et leur applicabilité dans la gestion réelle des chaînes d'approvisionnement.

CHAPITRE 2

MÉTHODES D'OPTIMISATION DES COÛTS ET MODÉLISATION DU SYSTÈME BEER GAME

Introduction

Dans un monde où l'efficacité et la compréhension des systèmes sont essentielles, les méthodes d'optimisation et la modélisation jouent un rôle crucial. Les méthodes d'optimisation des coûts visent à rationaliser les dépenses tout en maximisant la valeur, offrant ainsi un avantage concurrentiel précieux. D'un autre côté, la modélisation du système nous invite à explorer les complexités dans des secteurs tels que les chaînes d'approvisionnement et les processus logistiques en simulant la dynamique des flux de matériaux. Dans cette partie, nous allons dans un premier temps présenter quelques méthodes d'optimisation ensuite la modélisation du système Beer Game suivant notre politique stratégique optimale de commande.

1 Méthodes d'optimisation des coûts

L'optimisation est un domaine essentiel dans de nombreux secteurs, où l'objectif est de trouver la meilleure solution possible à un problème donné. Pour ce faire, différentes approches d'optimisation sont utilisées, chacune avec ses caractéristiques et ses applications spécifiques.

1.1 Les métaheuristiques

Les métaheuristiques sont des algorithmes d'optimisation conçus pour résoudre des problèmes d'optimisation difficiles, pour lesquels les méthodes classiques ne sont pas suffisamment efficaces. Ces approches itératives et stochastiques visent à converger vers un optimum

global. Agissant comme des algorithmes de recherche, les métaheuristiques apprennent progressivement les caractéristiques d'un problème afin de trouver une approximation de la meilleure solution possible.

Il existe une variété importante de métaheuristiques, allant de méthodes simples de recherche locale à des algorithmes complexes de recherche globale. Leur niveau d'abstraction élevé leur permet d'être adaptées à un large éventail de problèmes, sans nécessiter de changements majeurs dans leur structure.

Soit S un ensemble de solutions à un problème d'optimisation et soit f une fonction qui mesure la qualité $f(s)$ de chaque solution $s \in S$. Le but des métaheuristiques est de déterminer une solution $s \in S$ de valeur $f(s)$ minimale. En d'autres termes, le problème est de déterminer la valeur suivante :

$$\min_{s \in S} f(s)$$

Un voisinage est une fonction N qui associe un sous-ensemble de S à tout solution s. Une solution $s^{'} \in N(s)$ est dite voisine de s. Une solution $s \in S$ est un minimum local relativement à N si $f(s) \leq f(s^{'})$ pour tout $s^{'} \in N(s)$, alors qu'il s'agit d'un minimum global si $f(s) \leq f(s^{'})$ pour tout $s \in S$.

1.1.1 Algorithme de descente

Elle se décrit comme suit :

1. choisir une solution sS .
2. déterminer une solution $s^{'}$ qui minimise f dans $N(s)$.
3. si $f(s^{'}) < f(s)$ alors poser $s \longleftarrow s^{'}$ et retourner à 2., sinon STOP.

Le principal défaut de cette méthode est qu'elle s'arrête au premier minimum local rencontré.

1.1.2 Recuit Simulé

Elle a été proposée par Kirkpatrick, Gelatt et Vecchi en 1983. Elle se décrit comme suit :

1. Choisir une solution $s \in S$ ainsi qu'une température initiale T.
2. Tant que aucun critère d'arrêt n'est satisfait faire.
   Choisir aléatoirement une solution $s^{'} \in N(s)$.
   Générer un nombre réel aléatoire $r \in [0, 1]$.

Si $r < p(T,s,s)$ alors poser $s \longleftarrow s^{'}$.

Mettre à jour T .

3. Fin du tant que

En général, on définit

$$p(T,s,s') = e^{\frac{f(s)-f(s')}{T}}$$

où p représente la distribution de Boltzmann. Ainsi :

— Si $f(s') \leq f(s)$, alors $p(T,s,s') \geq 1 > r$, ce qui signifie qu'on accepte s'.

— Si T a une très grande valeur, alors $p(T,s,s') \approx 1$ et s' est donc très probablement acceptée.

— Si $T \approx 0$ et $f(s) < f(s')$, alors $p(T,s,s') \approx 0$ et s' est donc très probablement refusée.

La température initiale doit être élevée afin de faciliter les déplacements dans S au début de la recherche. Puis, T est graduellement réduite jusqu'à atteindre une valeur proche de 0.

La méthode du Recuit Simulé est une méthode sans mémoire. On peut facilement l'améliorer en rajoutant une mémoire à long terme qui stocke la meilleure solution rencontrée. En effet, l'algorithme du Recuit Simulé décrit ci-dessus peut converger vers une solution s^* en ayant visité auparavant une solution s de valeur $f(s) < f(s^*)$. Il est donc logique de mémoriser la meilleure solution rencontrée durant le processus de recherche. Le critère d'arrêt peut être une limite sur le temps ou sur le nombre d'itérations sans modification de la solution courante.

L'un des principaux défauts de la méthode du Recuit Simulé est le choix aléatoire du voisin $s^{'} \in N(s)$. On peut donc être proche de l'optimum et passer juste à côté sans le voir.

1.1.3 Recherche Tabou

Elle a été proposée par Glover en 1986. Son principe est de choisir à chaque itération la meilleure solution $s^{'} \in N(s)$, même si $f(s^{'}) > f(s)$. Lorsqu'on atteint un minimum local s par rapport au voisinage N , la Recherche Tabou se déplace donc vers une solution $s^{'}$ de valeur $f(s^{'}) \geq f(s)$. Le danger est alors de revenir à s puisque, en général, $s \in N(s^{'})$ et $f(s) \leq f(s^{'})$. Pour éviter de tourner en rond, on crée une liste T qui mémorise les dernières solutions visitées et interdit tout déplacement vers une solution de cette liste.T est la liste Tabou et solutions ne demeurent dans elle que pour un nombre limité d'itérations faisant d'elle une liste avec mémoire à court terme. Elle se décrit comme suit :

1. Choisir une solution $s \in S$, poser $T \longleftarrow$ et $s^* \longleftarrow s$.
2. Tant que aucun critère d'arrêt n'est satisfait faire.

   Déterminer une solution $s^{'}$ qui minimise f dans $N_T(S)$.

   Si $f(s^{'}) < f(s^*)$ alors poser $s^* \longleftarrow s^{'}$.

   Poser $s \longleftarrow s^{'}$ et mettre à jour T.
3. Fin du tant que.

Le critère d'arrêt peut etre un nombre maximum d'itérations sans amélioration de s^*, ou un temps limite après lequel la recherche doit s'arrêter.

L'algorithme de descente, le Recuit Simulé et la Recherche Tabou sont probablement les méthodes de Recherche les plus connues et les plus utilisées. Il existe cependant d'autres méthodes comme la colonies de fourmis, l'algorithme évolutionnaire, les réseaux de neurones comme le renseigne la figure(2.1) ci-dessous :

Algorithme	Acronyme	Type de problème	Extensions
Recuit simulé	SA	Discret (chemin)	Continu
Recherche tabou	TS	Discret (affectation)	
Essaims particulaires	PSO	Continu	Discret
Colonies de fourmis	ACO	Discret (chemin)	
Algorithme évolutionnaire	GA	Discret – Continu	
Adaptation de covariance	CMA	Continu	
Affine shaker	AS	Continu	Discret
Réseaux de neurones	NN	Identification	

FIGURE 2.1 – Métaheuristiques[13]

1.2 La programmation non linéaire

1.2.1 Formulation générale des problèmes d'optimisation non linéaire

La forme générale d'un problème d'optimisation est la suivante :

$$\begin{cases} \min\limits_{x \in \mathbb{R}^n} f(x) \\ s.c \\ g(x) \leq 0 \\ h(x) = 0 \end{cases} \tag{2.1}$$

où les fonctions f , g et h sont non-linéaires.

L'équation $g(x) \leq 0$ désigne les contraintes d'inégalité et l'équation $h(x) = 0$ désigne les contraintes d'égalité.

1.2.2 Définitions

Définition 1.1. *Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, on dit que f est continue au point $a \in \mathbb{R}^n$ si pour tout réel $\epsilon > 0$ il existe $\sigma > 0$ tel que*

$$||x-a|| < \sigma \Longrightarrow ||f(x)-f(a)|| < \epsilon$$

Définition 1.2. *Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ une fonction continue en $a \in \mathbb{R}^n$. On dit que f est différentiable en a s'il existe une application linéaire, notée $f^{'}(a)$, telle que pour tout $h \in \mathbb{R}^n$ on ait*

$$f(a+h) = f(a) + f^{'}(a)h + h\epsilon(h)$$

où $\epsilon(.)$ est une fonction continue en 0 vérifiant $\lim\limits_{h \longrightarrow 0} \epsilon(h) = 0$.

Définition 1.3. *Un ensemble $K \in \mathbb{R}^n$ est dit convexe si pour tout couple $(x, y) \in K^2$ et $\forall \lambda \in [0,1]$ on a $\lambda x + (1-\lambda)y \in K$.*

Cette définition peut s'interpréter en disant que le segment reliant x et y doit être dans K

Définition 1.4. *On dit qu'une fonction $f : K \longrightarrow \mathbb{R}^n$, définie sur un ensemble convexe K, est convexe si elle vérifie :*

$\forall (x, y) \in K^2, \quad \forall \lambda \in [0,1], \quad f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$

Définition 1.5. *On dira que f est strictement convexe si :*

$\forall (x, y) \in K^2, \quad \forall \lambda \in [0,1], \quad f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y).$

Définition 1.6. *Un ensemble $K \in \mathbb{R}^n$ est dit compact si, de toute suite x_k, où $x_k \in K$, $\forall k$, on peut extraire une sous-suite convergente.*

1.2.3 Existence de solution

Notre problème d'optimisation (2.1) ci-dessus est réécrire en mettant les contraintes sous la forme $x \in K \subset \mathbb{R}^n$ de la manière suivante :

$$\min_{x \in K} f(x) \tag{2.2}$$

- Si $f : K \in \mathbb{R}^n \longrightarrow \mathbb{R}^n$ est continue et si de plus K est un ensemble compact, alors le problème (2.2) admet une solution optimale $x^* \in K$[12], qui vérifie donc

$$f(x^*) \leq f(x) \forall x \in K$$

.

- Soit $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ une fonction continue sur $\mathbb{R}^n$. Si

$$\lim_{||x|| \longrightarrow \infty} f(x) = \infty$$

alors (2.2) admet une solution optimale x^*[12].

1.2.4 Unicité

Soit $f : K \in \mathbb{R}^n \longrightarrow \mathbb{R}^n$ une fonction convexe sur K convexe. Le minimum de f sur K, s'il existe, est unique[12].

2 Modélisation du système Beer Game

Nous nous concentrons particulièrement sur la modélisation des différentes étapes de la chaîne d'approvisionnement et la définition d'une stratégie d'approvisionnement pour évaluer son impact sur les performances globales.

2.1 Fluctuations de la Demande

Pour capturer la réalité de la variabilité de la demande, notre modèle intègre des fluctuations dans les commandes du détaillant. Ces fluctuations peuvent être générées à partir de données historiques ou de scénarios hypothétiques, permettant ainsi de simuler diverses situations de demande.

2.2 Définition de la Stratégie d'approvisionnement

Un aspect central de notre étude consiste à définir et à implémenter une stratégie d'approvisionnement pour chaque étape de la chaîne d'approvisionnement.

Nous avons choisi une approche pour illustrer l'impact de la coordination et de la prévision :

L'approche Prévisionnelle : Ici, les commandes sont basées sur des prévisions de la demande future, des niveaux d'inventaire et des commandes en cours. Cette approche vise à réduire les effets négatifs de la réactivité excessive en coordonnant les décisions entre les étapes.

2.3 Modélisation des Étapes de la Chaîne d'approvisionnement

La modélisation du système Beer Game repose sur la représentation des quatre principales étapes de la chaîne d'approvisionnement : le détaillant, le grossiste, le distributeur et l'usine. Chaque étape est caractérisée par son propre niveau d'inventaire et ses commandes en cours, qui évoluent au fil du temps en réponse aux variations de la demande. Ainsi, à chaque tournée et pour chaque parties prenantes, nous avons la mise en équations ci-dessous.

2.3.1 Modélisation au niveau du détaillant (R)

- **Inventaire (INV)**

$$RINV_t = \begin{cases} RINV_{t-1} + RIS_{t-1} - RBL_{t-1} - RIO_{t-1} & si \;\; RINV_{t-1} + RIS_{t-1} \geq RBL_{t-1} - RIO_{t-1} \\ 0 & sinon \end{cases}$$

- **Backlog (BL)**

$$RBL_t = \begin{cases} RBL_{t-1} + RIO_{t-1} - RINV_{t-1} - RIS_{t-1} & si \;\; RBL_{t-1} + RIO_{t-1} \geq RINV_{t-1} + RIS_{t-1} \\ 0 & sinon \end{cases}$$

- **Commande entrante espérée (ED)**

$$RED_t = \theta.RIO_{t-1} + (1-\theta).RED_{t-1} \;\; avec \;\; \theta \in [0,1]$$

- **Ajustement de stock (AS)**

$$RAS_t = \alpha_s(RDINV - RINV_t + RBL_t)$$

α_s est la paramètre d'ajustement de stock compris dans l'interval $[0,1]$ et $DINV$ le niveau d'inventaire optimal.

- **Ajustement de ligne d'approvisionnement (ASL)**

$$RASL_t = \alpha_{sl}(ROP_{t-1})$$

α_{sl} est la paramètre d'ajustement de ligne d'approvisionnement compris dans l'interval $[0,1]$.

- **Commande optimal (OP^*)**

$$ROP_t^* = RED_t + RAS_t - \alpha_s(RASL_t)$$

- **Politique de commande (OP)**

$$ROP_t = max(0, ROP_t^*)$$

- **Fonction de coût (CR)**

$$CR_t = cs * RINV_t + cb * RBL_t$$

cs est le coût de stockage et cb le coût de rupture de stock.

2.3.2 Modélisation au niveau du grossiste (W)

- **Inventaire**

$$WINV_t = \begin{cases} WINV_{t-1} + WIS_{t-1} - WBL_{t-1} - WIO_{t-1} & si \;\; WINV_{t-1} + WIS_{t-1} \geq WBL_{t-1} - WIO_{t-1} \\ 0 & sinon \end{cases}$$

- **Backlog**

$$WBL_t = \begin{cases} WBL_{t-1} + WIO_{t-1} - WINV_{t-1} - WIS_{t-1} & si \;\; WBL_{t-1} + WIO_{t-1} \geq WINV_{t-1} + WIS_{t-1} \\ 0 & sinon \end{cases}$$

- **Commande entrante espérée**

$$WED_t = \theta.WIO_{t-1} + (1-\theta).WED_{t-1} \;\; avec \;\; \theta \in [0,1]$$

- **Ajustement de stock**

$$WAS_t = \alpha_s(WDINV - WINV_t + WBL_t)$$

α_s est la paramètre d'ajustement de stock compris dans l'interval $[0,1]$ et $WDINV$ le niveau d'inventaire optimal.

- **Ajustement de ligne d'approvisionnement**

$$WASL_t = \alpha_{sl}(WOP_{t-1})$$

α_{sl} est la paramètre d'ajustement de ligne d'approvisionnement compris dans l'interval $[0,1]$.

- **Commande optimal**

$$WOP_t^* = WED_t + WAS_t - \alpha_s(WASL_t)$$

- **Politique de commande**

$$WOP_t = max(0, WOP_t^*)$$

- **Fonction de coût (CW)**

$$CW_t = cs * WINV_t + cb * WBL_t$$

cs est le coût de stockage et cb le coût de rupture de stock.

2.3.3 Modélisation au niveau du distributeur (D)

- **Inventaire**

$$DINV_t = \begin{cases} DINV_{t-1} + DIS_{t-1} - DBL_{t-1} - DIO_{t-1} & si \;\; DINV_{t-1} + DIS_{t-1} \geq DBL_{t-1} - DIO_{t-1} \\ 0 & sinon \end{cases}$$

- **Backlog**

$$DBL_t = \begin{cases} DBL_{t-1} + DIO_{t-1} - DINV_{t-1} - DIS_{t-1} & si \;\; DBL_{t-1} + DIO_{t-1} \geq DINV_{t-1} + DIS_{t-1} \\ 0 & sinon \end{cases}$$

- **Commande entrante espérée**

$$DED_t = \theta.DIO_{t-1} + (1-\theta).DED_{t-1} \;\; avec \;\; \theta \in [0,1]$$

- **Ajustement de stock**

$$DAS_t = \alpha_s(DDINV - DINV_t + DBL_t)$$

α_s est la paramètre d'ajustement de stock compris dans l'interval $[0,1]$ et $DDINV$ le niveau d'inventaire optimal.

- **Ajustement de ligne d'approvisionnement**

$$DASL_t = \alpha_{sl}(DOP_{t-1})$$

α_{sl} est la paramètre d'ajustement de ligne d'approvisionnement compris dans l'interval $[0,1]$.

- **Commande optimal**

$$DOP_t^* = DED_t + DAS_t - \alpha_s(DASL_t)$$

- **Politique de commande**

$$DOP_t = max(0, DOP_t^*)$$

- **Fonction de coût (CD)**

$$CD_t = cs * DINV_t + cb * DBL_t$$

cs est le coût de stockage et cb le coût de rupture de stock.

2.3.4 Modélisation au niveau de du fabriquant (F)

- **Inventaire**

$$FINV_t = \begin{cases} FINV_{t-1} + FIS_{t-1} - FBL_{t-1} - FIO_{t-1} & si \;\; FINV_{t-1} + FIS_{t-1} \geq FBL_{t-1} - FIO_{t-1} \\ 0 & sinon \end{cases}$$

- **Backlog**

$$FBL_t = \begin{cases} FBL_{t-1} + FIO_{t-1} - FINV_{t-1} - FIS_{t-1} & si \;\; FBL_{t-1} + FIO_{t-1} \geq FINV_{t-1} + FIS_{t-1} \\ 0 & sinon \end{cases}$$

- **Commande entrante espérée**

$$FED_t = \theta.FIO_{t-1} + (1-\theta).FED_{t-1} \;\; avec \;\; \theta \in [0,1]$$

- **Ajustement de stock**

$$FAS_t = \alpha_s(FDINV - FINV_t + FBL_t)$$

α_s est la paramètre d'ajustement de stock compris dans l'interval $[0,1]$ et $FDINV$ le niveau d'inventaire optimal.

- **Ajustement de ligne d'approvisionnement**

$$FASL_t = \alpha_{sl}(FOP_{t-1})$$

α_{sl} est la paramètre d'ajustement de ligne d'approvisionnement compris dans l'interval $[0, 1]$.

- **Commande optimal**

$$FOP_t^* = FED_t + FAS_t - \alpha_s(FASL_t)$$

- **Politique de commande**

$$FOP_t = max(0, FOP_t^*)$$

- **Fonction de coùt (CF)**

$$CF_t = cs * FINV_t + cb * FBL_t$$

cs est le coût de stockage et cb le coût de rupture de stock.

2.4 Objectif

L'objectif fondamental étant de minimiser les coûts c'est-à-dire de trouver un équilibre délicat entre la gestion efficace des stocks et la réponse adéquate à la demande. Nous avons donc le problème d'optimisation 2.3 ci-dessous :

$$\begin{cases} \min \sum_{t=1}^{n} (CR_t + CW_t + CD_t + CF_t) \\ s.c \\ \alpha_{s_i}, \alpha_{sl_i} \in [0, 1] \quad \text{i}\in [|1, 4|] \end{cases} \tag{2.3}$$

2.5 Implémentation du modèle d'optimisation

Notre implémentation se fera sur deux différentes stratégies

- **Stratégie 1** : Ici Nous considérons que les quatre secteurs ont des politiques de commande identiques c'est-à-dire que pour les quatre acteurs, les paramètres d'ajustement de stock (α_s) et les paramètres d'ajustement de ligne d'approvisionnement(α_{sl}) sont identiques. Nous avons donc le problème d'optimisation 2.4 ci-dessous :

$$\begin{cases} \min \sum_{t=1}^{n} (CR_t + CW_t + CD_t + CF_t) \\ s.c \\ \alpha_s, \alpha_{sl} \in [0,1] \end{cases} \tag{2.4}$$

- **Stratégie 2** : Ici nous considérons que les quatre secteurs ont des politiques de commande différentes c'est-à-dire que pour les quatre acteurs, les paramètres d'ajustement de stock (α_s) et les paramètres d'ajustement de ligne d'approvisionnement(α_{sl}) sont différents. Nous avons donc le problème d'optimisation 2.5 ci-dessous :

$$\begin{cases} \min \sum_{t=1}^{n} (CR_t + CW_t + CD_t + CF_t) \\ s.c \\ \alpha_{s_1}, \alpha_{sl_1}, \alpha_{s_2}, \alpha_{sl_2}, \alpha_{s_3}, \alpha_{sl_3}, \alpha_{s_4}, \alpha_{sl_4} \in [0,1] \end{cases} \tag{2.5}$$

CHAPITRE 3

SIMULATION ET RÉSULTATS ET DISCUSSION

Introduction

La gestion efficace de la chaîne d'approvisionnement est un défi crucial pour les entreprises cherchant à minimiser les coûts opérationnels tout en satisfaisant la demande du marché. Le Beer game, un exercice pédagogique, offre une opportunité d'explorer ces défis au travers d'une simulation réaliste d'une chaîne d'approvisionnement de bière. Dans cette section, nous présentons les résultats de l'implémentation d'un modèle d'optimisation et de stratégies visant à minimiser les coûts de stockage tout en assurant une gestion fluide des stocks le long de la chaîne.

1 Description du scénario d'expérimentation avec les paramètres choisis

Dans le cadre de notre recherche, nous avons conçu un scénario d'expérimentation spécifique au sein du Beer Game. Cette étape cruciale de notre démarche méthodologique vise à fournir un cadre réaliste et maîtrisé pour évaluer l'impact des stratégies d'optimisation des coûts au sein d'une chaîne d'approvisionnement interconnectée.

1.1 Paramètres Sélectionnés

Les paramètres que nous avons choisis pour notre scénario d'expérimentation sont déterminants pour engendrer une dynamique authentique et signifiante au sein de la chaîne d'approvisionnement simulée. Ces paramètres, en interaction les uns avec les autres, ont été ajustés pour créer un environnement réaliste et refléter les défis typiques rencontrés dans la gestion des chaînes d'approvisionnement réelles. Voici les paramètres clés que nous avons sélectionnés :

- Les acteurs de la Chaîne d'approvisionnement : Notre scénario implique quatre acteurs interdépendants, représentant le fabricant (l'usine(F)) , le distributeur(D), le grossiste(W) et le détaillant(R). Chacun de ces acteurs prend des décisions de commande en fonction de la demande prévue et de la disponibilité des stocks.
- Coûts de Stockage (cs) et de Rupture de Stock (Cb) : Les coûts associés au stockage des produits ainsi qu'aux ruptures de stock ont été définis avec les valeurs respectives 0.5 et 2 par unité (1000) et par tournée. Ces coûts jouent un rôle majeur dans la prise de décision des acteurs de la chaîne et influencent leur volume de commande.
- Délais de Livraison : Les délais nécessaires pour que les commandes soient traitées et livrées entre les différents maillons de la chaîne ont été spécifiés. Ces délais ont un impact direct sur la disponibilité des produits et sur la gestion des stocks.
- Taux de Demande : Les taux de demande, représentant les besoins en produits des clients, ont été déterminés de manière à introduire des fluctuations réalistes et des variations saisonnières.
- Période d'expérimentation : Notre scénario s'étend sur une période définie, comprenant plusieurs itérations pour refléter le fonctionnement dynamique d'une chaîne d'approvisionnement au fil du temps.

1.2 Objectifs de l'expérimentation

L'objectif principal de notre expérimentation est d'évaluer comment différentes stratégies d'optimisation des coûts influencent les performances de la chaîne d'approvisionnement simulée dans le Beer Game. Nous cherchons à identifier la stratégie la plus efficace pour réduire les coûts opérationnels tout en maintenant un niveau de service satisfaisant, en minimisant les ruptures de stock et en optimisant les niveaux de stock.

Suivant chacune de nos deux stratégies, nous avons effectué différente implémentation (Annexe, code python) selon le cas.

Nous avons utilisé la fonction minimize de la bibliothèque SciPy pour résoudre les problèmes d'optimisation 2.4 et 2.5.

2 Résultats et Interprétations

Plusieurs exécutions ont été réalisées pour les deux scénarios examinés. Toutes les exécutions ont été menées avec 100 demandes de client finale, un nombre maximum d'itérations fixe à 100, sur une période de 100 jours avec un délai de livraison de deux jours.

La séquence ci-dessous représente les demandes du client final.

$[16, 11, 7, 6, 16, 7, 16, 9, 16, 10, 9, 15, 8, 10, 12, 15, 10, 11, 10, 10, 9, 7, 7, 7, 10, 8, 9, 11, 13, 11, 11,]$
$[4, 11, 13, 8, 15, 9, 9, 8, 16, 10, 9, 7, 11, 12, 13, 10, 10, 14, 10, 10, 12, 8, 14, 10, 15, 5, 11, 12, 12, 4, 9,]$
$[10, 11, 13, 5, 12, 8, 13, 8, 9, 12, 10, 12, 7, 11, 8, 9, 15, 7, 5, 6, 14, 9, 17, 7, 14, 12, 8, 12, 16, 16, 6, 5,]$
$[12, 14, 4, 15, 7, 10]$.

- **Stratégie 1**

  Suivant cette stratégie, nous avons obtenus, pour chaque secteur, le paramètre d'ajustement de stock (α_s) et le paramètre d'ajustement de ligne d'approvisionnement(α_{sl}) suivant :

  — Détaillant : $\alpha_s = 0.011$ et $\alpha_{sl} = 0.0098$

  — Grossiste : $\alpha_s = 0.011$ et $\alpha_{sl} = 0.0098$

  — Distributeur : $\alpha_s = 0.011$ et $\alpha_{sl} = 0.0098$

  — Fabriquant(l'usine) : $\alpha_s = 0.011$ et $\alpha_{sl} = 0.0098$

  Sous ces valeurs optimales, nous avons observé l'évolution de taux de commande, du niveau de stock, et du coût de chaque secteur durant chaque semaine tels que représentés sur les figures ci-dessous.

  Le tableau 3.1 resume les coûts opérationnels du système.

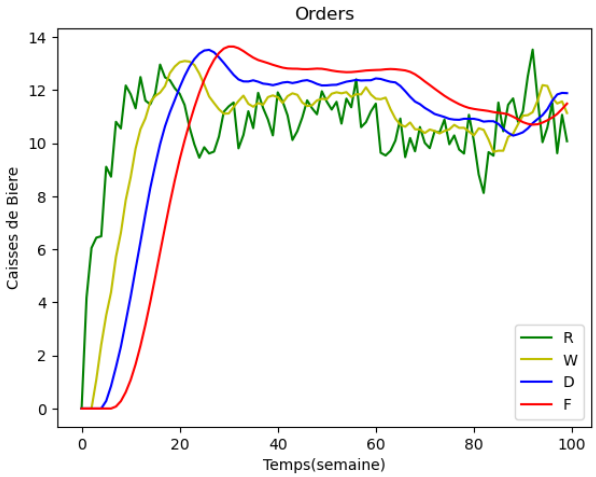

FIGURE 3.1 – Stratégie 1 : Commandes passées par le détaillant, le grossiste, le distributeur et l'usine.

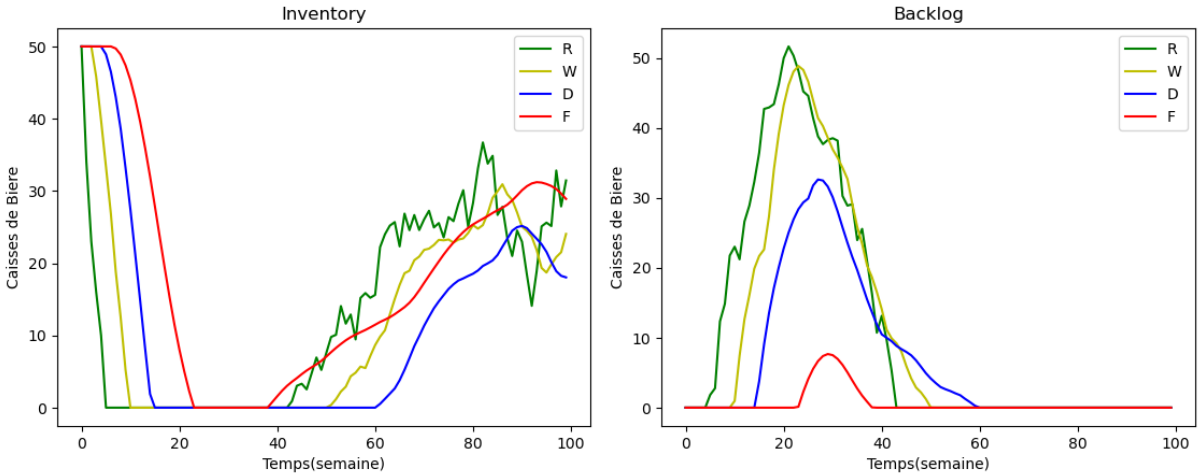

FIGURE 3.2 – Stratégie 1 : Niveau de stock et de rupture de stock du détaillant, du grossiste, du distributeur et de l'usine.

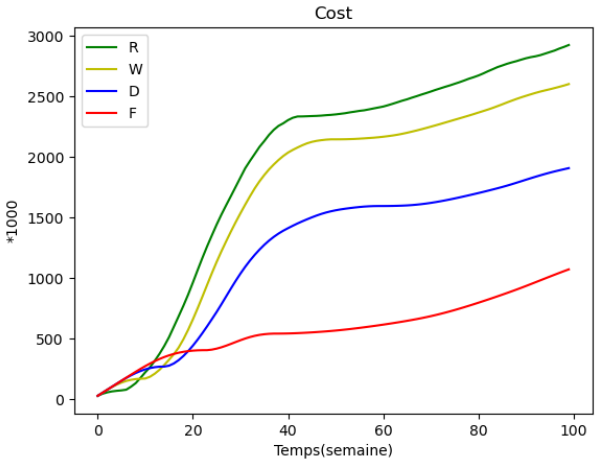

FIGURE 3.3 – Stratégie 1 : Évolution du coût du détaillant, du grossiste, du distributeur et de l'usine.

Coût	Inventaire	Backlog	Total
Détaillant	657 298	2267 198	2 924 496
Grossiste	622 988	1978 640	2 601 629
Distributeur	580 039	1 326 876	1 906 915
Fabriquant	932 896	137 165	1 070 062
Acteurs	2 793 222	5 709 880	8 503 103

TABLE 3.1 – Stratégie 1 : coûts opérationnels du système (en Fcfa)

- **Stratégie 2**

  Suivant cette stratégie, nous avons obtenus, pour chaque secteur, le paramètre d'ajustement de stock (α_s) et le paramètre d'ajustement de ligne d'approvisionnement(α_{sl}) suivant :

  — Détaillant : $\alpha_s = 0.011$ et $\alpha_{sl} = 0.810$

  — Grossiste : $\alpha_s = 0.096$ et $\alpha_{sl} = 0.99$

  — Distributeur : $\alpha_s = 0.25$ et $\alpha_{sl} = 1$

  — Fabriquant(l'usine) : $\alpha_s = 0.70$ et $\alpha_{sl} = 1$

  Sous ces valeurs optimales, nous avons observé l'évolution de taux de commande, du niveau de stock, et du coût de chaque secteur durant chaque semaine tels que

représentés sur les figures ci-dessous.

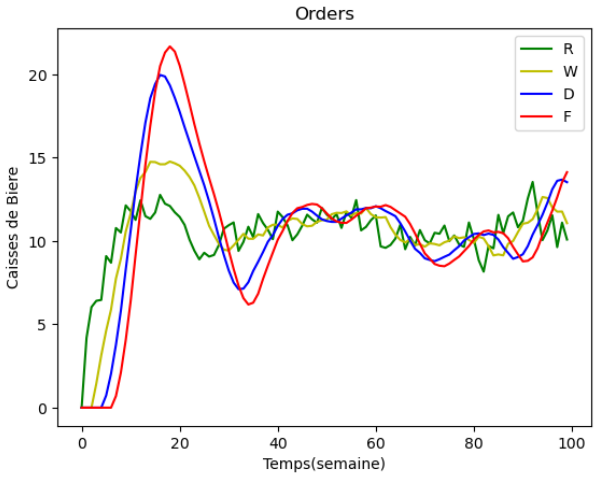

FIGURE 3.4 – Stratégie 2 : Commandes passées par le détaillant , le grossiste, le distributeur et l'usine.

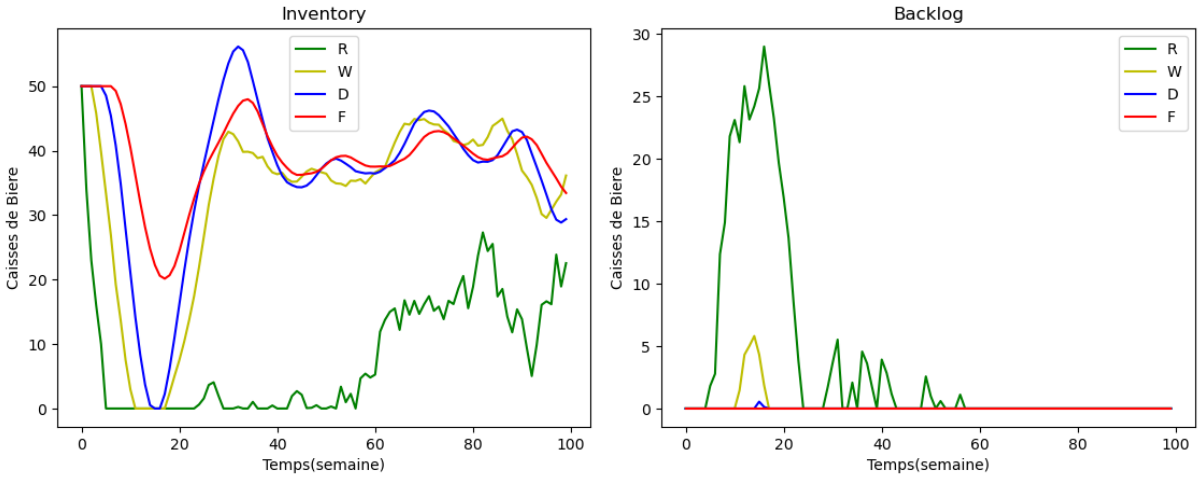

FIGURE 3.5 – Stratégie 2 : Niveau de stock et de rupture de stock du détaillant, du grossiste, du distributeur et de l'usine.

Le tableau 3.2 resume les coûts opérationnels du système.

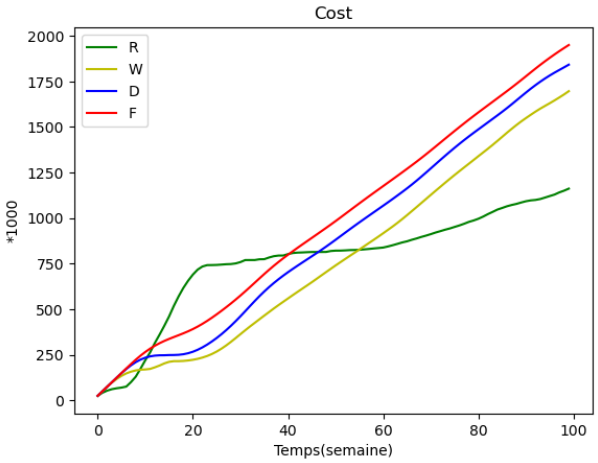

FIGURE 3.6 – Stratégie 2 : Évolution du coût du détaillant, du grossiste, du distributeur et de l'usine.

Coût	Inventaire	Backlog	Total
Détaillant	413 898	747 907	1 161 806
Grossiste	1 650 915	45 579	1 696 530
Distributeur	1 840 488	1 345	1 841 834
Fabriquant	1 949 772	0	1 949 772
Acteurs	5 855 111	794 832	6 649 943

TABLE 3.2 – Stratégie 2 : coûts opérationnels du système (en Fcfa)

Dans les deux scénarios, les niveaux de stocks initiaux décroissent puis connaissent une série d'oscillations croissantes depuis le détaillant jusqu'à l'usine. Cette dynamique est engendrée par une modification du taux de commande des clients au fil des semaines. Cette transition provoque une hausse des commandes au niveau du détaillant, qui se propage en vagues à travers toute la chaîne d'approvisionnement jusqu'à atteindre l'usine. Cela entraîne un épuisement des stocks et des augmentations successives des taux de commande pour répondre aux commandes en attente. Une fois ces commandes satisfaites, les niveaux de stocks augmentent en raison des taux de commande accrus, en particulier dans les zones proches de l'usine.

Dans le premier cas, le coût total du système s'élève à 8 503 103 Fcfa, alors que dans le deuxième cas, il est de 6 649 943 Fcfa. Dans le contexte du premier scénario, le secteur

de l'usine de fabrication enregistre le score le plus bas de 1 070 062 Fcfa, tandis que le détaillant, le grossiste et le distributeur obtiennent respectivement des scores de 2 924 496 Fcfa, 2 601 629 Fcfa et 1 906 915 Fcfa. En revanche, dans le cadre du deuxième scénario, le secteur affichant le score le plus bas est le secteur du détaillant qui enregistre le score de 1 161 806 Fcfa, tandis que le grossiste, le distributeur et l'usine obtiennent respectivement des scores de 1 696 530 Fcfa, 1 841 832 Fcfa et 1 949 772 Fcfa. Cela s'explique par le fait que tous les parties prenantes parviennent à contenir leurs stocks lorsque la commande du client augmente. Autrement dit, il n'y a pas de gagnant clair comme dans le premier scénario, mais tous les secteurs atteignent l'objectif de maintenir des stocks faibles et d'éviter les arriérés.

Les résultats montrent que les meilleures performances sont obtenues lorsque les secteurs appliquent la stratégie 2 plutôt la stratégie 1(figure 3.7). Ces résultats ne sont pas surprenants car, dans le premier cas, nous disposons de plus de variables et donc de plus de possibilités de trouver de bonnes solutions.

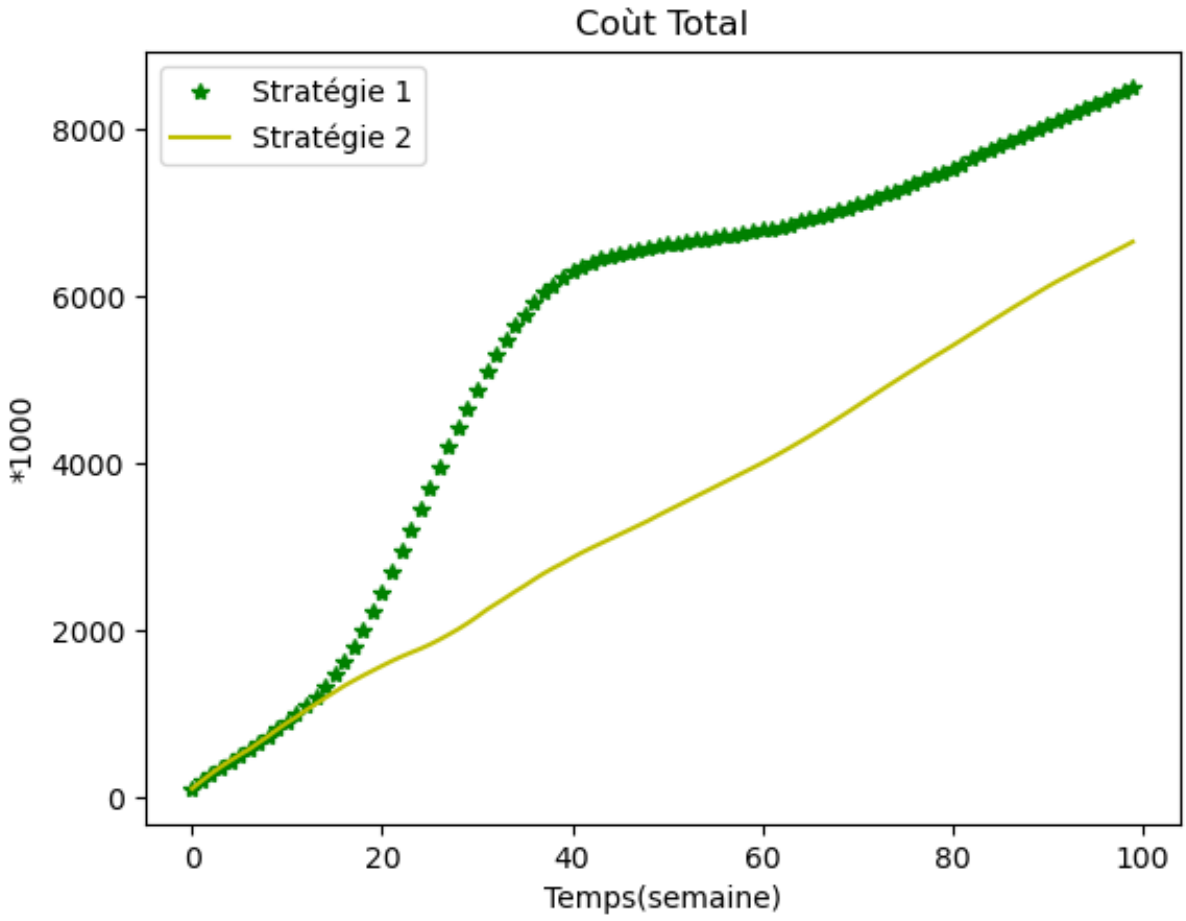

FIGURE 3.7 – Coût total du système

3 Évaluation des Résultats

3.1 Simulation dans AnyLogic

Les expériences de simulations sur le logiciel AnyLogic ont aboutis aux résultats présentés à la figures (3.8) ci-dessous.

Costs	Inventory	Backlog	Total
Retailer	1,171.5	322	1,493.5
Wholesaler	1,821	52	1,873
Distributor	2,315	245	2,560
Factory	3,499.5	449	3,948.5
All players	8,807	1,068	9,875

FIGURE 3.8 – AnyLogic : Niveau de de stock et coûts (échelle 1/1000) opérationnels du système.

3.2 Comparaison des différents résultats de simulations

Le tableau 3.3 ci-dessous récapitule les résultats des différentes simulations effectuées.

Nous constatons que les coûts totaux opérationnels de tout le système en suivants les stratégies 1 et 2 sont inférieurs au coût total sur AnyLogic. Cette différence traduit une

Coût	AnyLogic	Stratégie 1	Stratégie 2
Détaillant	1 493 500	2 924 496	1 161 806
Grossiste	1 873 000	2 601 629	1 696 530
Distributeur	2 560 000	1 906 915	1 841 832
Fabriquant	3 948 500	1 070 062	1 949 772
Total	9 875 000	8 503 103	6 649 943

TABLE 3.3 – Coût de différentes simulations (en Fcfa)

amélioration global des performance du système suivant la politique de commande que nous avons définit.

Conclusion

Dans ce mémoire, nous avons abordé les enjeux cruciaux liés à la gestion efficace des chaînes d'approvisionnement, en mettant en évidence les défis résultant de l'interaction dynamique entre les acteurs, les fluctuations de la demande et les coûts opérationnels. Notre recherche a débuté par une analyse approfondie du jeu de la bière lui-même. Nous avons examiné en détail son contexte et ses caractéristiques clés, afin de mieux comprendre les dynamiques sous-jacentes qui engendrent des déséquilibres et des inefficacités au sein de la chaîne d'approvisionnement. En se basant sur les notions de modélisation mathématique, nous avons entrepris la conception d'un modèle précis qui représente la chaîne d'approvisionnement du Beer Game. L'expérimentation approfondie à travers la programmation non linéaire et les simulations numériques a permis d'évaluer l'efficacité des stratégies d'optimisation proposées. Ces simulations ont générées des résultats concrets qui offrent un aperçu des améliorations potentielles et de l'impact réel sur les coûts opérationnels de la chaîne d'approvisionnement. En somme, notre mémoire a pour vocation de contribuer à l'amélioration continue des pratiques de gestion des chaînes d'approvisionnement, en mettant en lumière les avantages stratégiques découlant d'une optimisation efficace des coûts. Au terme de cette recherche, nous prenons conscience des limites inhérentes à notre modèle. Cependant, notre travail jette les bases d'une meilleure compréhension de l'optimisation des coûts dans les chaînes d'approvisionnement, tout en ouvrant la voie à de futures investigations tel que l'utilisation de réseau de neurones pour apprendre puis déterminer la bonne décision à chaque étape.

Bibliographie

[1] Forrester, J. W. (1961). Industrial dynamics. MIT Press.

[2] F. Hémici, M. Bounab.(2016) , Techniques de gestion. 4 édition ; Dunod 11, rue Paul-Bert, 92240 Malakof.

[3] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence V. Snyder, Martin Takáč (2016). A Deep Q-Network for the Beer Game : Using Machine Learning to Solve Inventory Optimization Problems,COR@L Technical Report 18T-005

[4] Bidyut Biman Sarkara, Nabendu Chakib (2012). A Distributed Retail Beer Game for Decision Support System. Procedia - Social and Behavioral Sciences 65 278 – 284

[5] Sterman, J. D. (1989). Modeling managerial behavior : misperceptions of feedback in a dynamic decision making experiment. Management Science, 35(3), 321-339.

[6] Forza, C., Salvador, F. (2002). Teaching the bullwhip effect using the beer distribution game and the pizza game.

[7] Federgruen, A., Liu, K. J. R. (1998). Coordinating a Supply Chain with an Unknown Demand Distribution and a General Cost Structure.

[8] Lee, H. L., Padmanabhan, V., Whang, S. (1997). The bullwhip effect in supply chains (2020). Sloan Management Review, 38(3), 93-102

[9] Khadija Eddoug, Saâd Lissane.(2015)optimisation conjointe des coûts de transport et de stock dans une chaîne logistique de distribution multi niveaux : Une approche basée sur la simulation. Conférence Internationale : Conception et Production Intégrées, Dec 2015, Tanger, Maroc. ⟨hal-01260802⟩

[10] M. Minoux. Programmation mathématique ()

[11] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard (2003). Métaheuristiques pour l'optimisation difficile, EYROLLES, pp.356, Algorithmes.

[12] S.Mottelet. (2003).RO04/TI07 - Optimisation Non-linéaire

[13] http ://www.metaheuristics.org

[14] AnyLogic

Annexes

A.1 Stratégie 1

Stratégie 1

September 8, 2023

[ ]:
```
import numpy as np
from copy import deepcopy
import matplotlib.pyplot as plt
from scipy.optimize import *
nombre = 100
```

[ ]:
```
chaine_sequence = "16 11 7 6 16 7 16 9 16 10 9 15 8 10 12 15 10 11 10 10 9 7 7␣
↪7 10 8 9 11 13 11 11 4 11 13 8 15 9 9 8 16 10 9 7 11 12 13 10 10 14 10 10 12␣
↪8 14 10 15 5 11 12 12 4 9 10 11 13 5 12 8 13 8 9 12 10 12 7 11 8 9 15 7 5 6␣
↪14 9 17 7 14 12 8 12 16 16 6 5 12 14 4 15 7 10"
IO = [int(x) for x in chaine_sequence.split()]

print(IO)
```

[ ]:
```
#minization with same stategies for all player

#define of the cost function

def costFns(alpha, paramsInit, incomingOrders, thetas, DINV, Cs, Cbl, delai):

    #DICT,DINV,Cs,Cbl,thetas,alpha,paramsInit,incomingOrders= [],50,0.5,2,0.
↪25,[0,0],[0,0,50,0,0,0,0],IO
    DICT = []
    CR,CW,CD,CF = [],[],[],[]
    dict_tm1 = {}
    dict_tm1["RED"] = paramsInit[0]
    dict_tm1["ROP"] = paramsInit[1]
    dict_tm1["RINV"] = paramsInit[2]
    dict_tm1["RBL"] = paramsInit[3]
    dict_tm1["RIS"] = paramsInit[4]
    dict_tm1["ROS"] = paramsInit[5]
    dict_tm1["RASL"] = paramsInit[6]
    dict_tm1["RIO"] = incomingOrders[0]
    dict_tm1["WED"] = paramsInit[0]
    dict_tm1["WOP"] = paramsInit[1]
    dict_tm1["WINV"] = paramsInit[2]
    dict_tm1["WBL"] = paramsInit[3]
```

```
    dict_tm1["WIS"] = paramsInit[4]
    dict_tm1["WOS"] = paramsInit[5]
    dict_tm1["WASL"] = paramsInit[6]
    dict_tm1["WIO"] = paramsInit[0]
    dict_tm1["DED"] = paramsInit[0]
    dict_tm1["DOP"] = paramsInit[1]
    dict_tm1["DINV"] = paramsInit[2]
    dict_tm1["DBL"] = paramsInit[3]
    dict_tm1["DIS"] = paramsInit[4]
    dict_tm1["DOS"] = paramsInit[5]
    dict_tm1["DASL"] = paramsInit[6]
    dict_tm1["DIO"] = paramsInit[0]
    dict_tm1["FED"] = paramsInit[0]
    dict_tm1["FOP"] = paramsInit[1]
    dict_tm1["FINV"] = paramsInit[2]
    dict_tm1["FBL"] = paramsInit[3]
    dict_tm1["FIS"] = paramsInit[4]
    dict_tm1["FOS"] = paramsInit[5]
    dict_tm1["FASL"] = paramsInit[6]
    dict_tm1["FIO"] = paramsInit[0]
    costR = dict_tm1["RINV"]*Cs+dict_tm1["RBL"]*Cbl
    costW = dict_tm1["WINV"]*Cs+dict_tm1["WBL"]*Cbl
    costD = dict_tm1["DINV"]*Cs+dict_tm1["DBL"]*Cbl
    costF = dict_tm1["FINV"]*Cs+dict_tm1["FBL"]*Cbl
    CR.append(costR)
    CW.append(costW)
    CD.append(costD)
    CF.append(costF)
    costFunction=costR+costW+costD+costF
    DICT.append(dict_tm1)
    for t in range(1,nombre):
        dict_t = {}
        dict_t["RIO"] = incomingOrders[t]
        dict_t["RED"] = thetas*(dict_tm1["RIO"]) + (1-thetas)*(dict_tm1["RED"])
        dict_t["RINV"] =␣
↪max(0,dict_tm1["RINV"]+dict_tm1["RIS"]-dict_tm1["RBL"]-dict_tm1["RIO"])
        dict_t["RBL"] =␣
↪max(0,dict_tm1["RBL"]+dict_tm1["RIO"]-dict_tm1["RINV"]-dict_tm1["RIS"])
        dict_t["RIS"] = dict_tm1["WOS"]
        dict_t["ROS"] =␣
↪min(dict_tm1["RBL"]+dict_tm1["RIO"],dict_tm1["RINV"]+dict_tm1["RIS"])
        dict_t["RASL"] = dict_tm1["ROP"]
        dict_t["ROP"] =␣
↪max(0,dict_t["RED"]+(alpha[0]*(DINV-dict_t["RINV"]+dict_t["RBL"]-alpha[1]*(dict_t["RASL"]))
        dict_t["WIO"] = dict_tm1["ROP"]
        dict_t["WED"] = thetas*dict_tm1["WIO"] + dict_tm1["WED"]*(1-thetas)
```

```
        dict_t["WINV"] =␣
↪max(0,dict_tm1["WINV"]+dict_tm1["WIS"]-dict_tm1["WBL"]-dict_tm1["WIO"])
        dict_t["WBL"] =␣
↪max(0,dict_tm1["WBL"]+dict_tm1["WIO"]-dict_tm1["WINV"]-dict_tm1["WIS"])
        dict_t["WIS"] = dict_tm1["DOS"]
        dict_t["WOS"] =␣
↪min(dict_tm1["WBL"]+dict_tm1["WIO"],dict_tm1["WINV"]+dict_tm1["WIS"])
        dict_t["WASL"] = dict_t["WBL"]+dict_tm1["WOP"]
        dict_t["WOP"] =␣
↪max(0,dict_t["WED"]+alpha[0]*(DINV-dict_t["WINV"]+dict_t["WBL"]-(alpha[1]*dict_t["WASL"])))
        dict_t["DIO"] = dict_tm1["WOP"]
        dict_t["DED"] = thetas*dict_tm1["DIO"] + dict_tm1["DED"]*(1-thetas)
        dict_t["DINV"] =␣
↪max(0,dict_tm1["DINV"]+dict_tm1["DIS"]-dict_tm1["DBL"]-dict_tm1["DIO"])
        dict_t["DBL"] =␣
↪max(0,dict_tm1["DBL"]+dict_tm1["DIO"]-dict_tm1["DINV"]-dict_tm1["DIS"])
        dict_t["DIS"] = dict_tm1["FOS"]
        dict_t["DOS"] =␣
↪min(dict_tm1["DBL"]+dict_tm1["DIO"],dict_tm1["DINV"]+dict_tm1["DIS"])
        dict_t["DASL"] = dict_t["DBL"]+dict_tm1["DOP"]
        dict_t["DOP"] =␣
↪max(0,dict_t["DED"]+alpha[0]*(DINV-dict_t["DINV"]+dict_t["DBL"]-(alpha[1]*dict_t["DASL"])))
        dict_t["FIO"] = dict_tm1["DOP"]
        dict_t["FED"] = thetas*dict_tm1["FIO"] + dict_tm1["FED"]*(1-thetas)
        dict_t["FINV"] =␣
↪max(0,dict_tm1["FINV"]+dict_tm1["FIS"]-dict_tm1["FBL"]-dict_tm1["FIO"])
        dict_t["FBL"] =␣
↪max(0,dict_tm1["FBL"]+dict_tm1["FIO"]-dict_tm1["FINV"]-dict_tm1["FIS"])
        dict_t["FIS"] = min (dict_tm1["FOP"],delai*50)
        dict_t["FOS"] =␣
↪min(dict_tm1["FBL"]+dict_tm1["FIO"],dict_tm1["FINV"]+dict_tm1["FIS"])
        dict_t["FASL"] = dict_t["FBL"]+dict_tm1["FOP"]
        dict_t["FOP"] =␣
↪max(0,dict_t["FED"]+alpha[0]*(DINV-dict_t["FINV"]+dict_t["FBL"]-(alpha[1]*dict_t["FASL"])))

        DICT.append(dict_t)

        #cost
        costR += dict_t["RINV"]*Cs+dict_t["RBL"]*Cbl
        costW += dict_t["WINV"]*Cs+dict_t["WBL"]*Cbl
        costD += dict_t["DINV"]*Cs+dict_t["DBL"]*Cbl
        costF += dict_t["FINV"]*Cs+dict_t["FBL"]*Cbl
        CR.append(costR)
        CW.append(costW)
        CD.append(costD)
        CF.append(costF)
```

```
        costR + costW + costD + costF
        dict_tm1 = deepcopy(dict_t)

    return costR + costW + costD + costF
```

[ ]:
```
#minization
DINV,Cs,Cbl,thetas,alphas,paramsInit,incomingOrders,delai= 50,0.5,2,0.25,[0.
↪5]*2,[0,0,50,0,0,0,0],IO,0.75
bound= [(0, 1)]*2

minimums = minimize(costFns,alphas,args=(paramsInit, incomingOrders, thetas,␣
↪DINV, Cs, Cbl, delai),bounds=bound)
print(minimums)
```

[ ]:
```
#apha opt
ass = minimums.x.reshape(2)
alpha_opts = []
for i in range (len(ass)):
    alpha_opts.append(ass[i])

alpha_opts
```

[ ]:
```
#plot Orders Inventory and cost function with the alphaopt

#optimal function

def costFnsopt(alpha, paramsInit, incomingOrders, thetas, DINV, Cs, Cbl, delai):

    DICT = []
    CR,CW,CD,CF,Coutl = [],[],[],[],[]
    dict_tm1 = {}
    dict_tm1["RED"] = paramsInit[0]
    dict_tm1["ROP"] = paramsInit[1]
    dict_tm1["RINV"] = paramsInit[2]
    dict_tm1["RBL"] = paramsInit[3]
    dict_tm1["RIS"] = paramsInit[4]
    dict_tm1["ROS"] = paramsInit[5]
    dict_tm1["RASL"] = paramsInit[6]
    dict_tm1["RIO"] = incomingOrders[0]
    dict_tm1["WED"] = paramsInit[0]
    dict_tm1["WOP"] = paramsInit[1]
    dict_tm1["WINV"] = paramsInit[2]
    dict_tm1["WBL"] = paramsInit[3]
    dict_tm1["WIS"] = paramsInit[4]
    dict_tm1["WOS"] = paramsInit[5]
    dict_tm1["WASL"] = paramsInit[6]
```

```
    dict_tm1["WIO"] = paramsInit[0]
    dict_tm1["DED"] = paramsInit[0]
    dict_tm1["DOP"] = paramsInit[1]
    dict_tm1["DINV"] = paramsInit[2]
    dict_tm1["DBL"] = paramsInit[3]
    dict_tm1["DIS"] = paramsInit[4]
    dict_tm1["DOS"] = paramsInit[5]
    dict_tm1["DASL"] = paramsInit[6]
    dict_tm1["DIO"] = paramsInit[0]
    dict_tm1["FED"] = paramsInit[0]
    dict_tm1["FOP"] = paramsInit[1]
    dict_tm1["FINV"] = paramsInit[2]
    dict_tm1["FBL"] = paramsInit[3]
    dict_tm1["FIS"] = paramsInit[4]
    dict_tm1["FOS"] = paramsInit[5]
    dict_tm1["FASL"] = paramsInit[6]
    dict_tm1["FIO"] = paramsInit[0]
    costR = dict_tm1["RINV"]*Cs+dict_tm1["RBL"]*Cbl
    costW = dict_tm1["WINV"]*Cs+dict_tm1["WBL"]*Cbl
    costD = dict_tm1["DINV"]*Cs+dict_tm1["DBL"]*Cbl
    costF = dict_tm1["FINV"]*Cs+dict_tm1["FBL"]*Cbl
    CR.append(costR)
    CW.append(costW)
    CD.append(costD)
    CF.append(costF)
    costFunction=costR+costW+costD+costF
    Coutl.append(costFunction)
    DICT.append(dict_tm1)
    for t in range(1,nombre):
        dict_t = {}
        dict_t["RIO"] = incomingOrders[t]
        dict_t["RED"] = thetas*(dict_tm1["RIO"]) + (1-thetas)*(dict_tm1["RED"])
        dict_t["RINV"] =␣
↪max(0,dict_tm1["RINV"]+dict_tm1["RIS"]-dict_tm1["RBL"]-dict_tm1["RIO"])
        dict_t["RBL"] =␣
↪max(0,dict_tm1["RBL"]+dict_tm1["RIO"]-dict_tm1["RINV"]-dict_tm1["RIS"])
        dict_t["RIS"] = dict_tm1["WOS"]
        dict_t["ROS"] =␣
↪min(dict_tm1["RBL"]+dict_tm1["RIO"],dict_tm1["RINV"]+dict_tm1["RIS"])
        dict_t["RASL"] = dict_tm1["ROP"]
        dict_t["ROP"] =␣
↪max(0,dict_t["RED"]+(alpha[0]*(DINV-dict_t["RINV"]+dict_t["RBL"]-alpha[1]*(dict_t["RASL"]))
        dict_t["WIO"] = dict_tm1["ROP"]
        dict_t["WED"] = thetas*dict_tm1["WIO"] + dict_tm1["WED"]*(1-thetas)
        dict_t["WINV"] =␣
↪max(0,dict_tm1["WINV"]+dict_tm1["WIS"]-dict_tm1["WBL"]-dict_tm1["WIO"])
```

```
        dict_t["WBL"] =␣
↪max(0,dict_tm1["WBL"]+dict_tm1["WIO"]-dict_tm1["WINV"]-dict_tm1["WIS"])
        dict_t["WIS"] = dict_tm1["DOS"]
        dict_t["WOS"] =␣
↪min(dict_tm1["WBL"]+dict_tm1["WIO"],dict_tm1["WINV"]+dict_tm1["WIS"])
        dict_t["WASL"] = dict_tm1["WOP"]
        dict_t["WOP"] =␣
↪max(0,dict_t["WED"]+alpha[0]*(DINV-dict_t["WINV"]+dict_t["WBL"]-(alpha[1]*dict_t["WASL"])))
        dict_t["DIO"] = dict_tm1["WOP"]
        dict_t["DED"] = thetas*dict_tm1["DIO"] + dict_tm1["DED"]*(1-thetas)
        dict_t["DINV"] =␣
↪max(0,dict_tm1["DINV"]+dict_tm1["DIS"]-dict_tm1["DBL"]-dict_tm1["DIO"])
        dict_t["DBL"] =␣
↪max(0,dict_tm1["DBL"]+dict_tm1["DIO"]-dict_tm1["DINV"]-dict_tm1["DIS"])
        dict_t["DIS"] = dict_tm1["FOS"]
        dict_t["DOS"] =␣
↪min(dict_tm1["DBL"]+dict_tm1["DIO"],dict_tm1["DINV"]+dict_tm1["DIS"])
        dict_t["DASL"] = dict_tm1["DOP"]
        dict_t["DOP"] =␣
↪max(0,dict_t["DED"]+alpha[0]*(DINV-dict_t["DINV"]+dict_t["DBL"]-(alpha[1]*dict_t["DASL"])))
        dict_t["FIO"] = dict_tm1["DOP"]
        dict_t["FED"] = thetas*dict_tm1["FIO"] + dict_tm1["FED"]*(1-thetas)
        dict_t["FINV"] =␣
↪max(0,dict_tm1["FINV"]+dict_tm1["FIS"]-dict_tm1["FBL"]-dict_tm1["FIO"])
        dict_t["FBL"] =␣
↪max(0,dict_tm1["FBL"]+dict_tm1["FIO"]-dict_tm1["FINV"]-dict_tm1["FIS"])
        dict_t["FIS"] = min (dict_tm1["FOP"],delai*50)
        dict_t["FOS"] =␣
↪min(dict_tm1["FBL"]+dict_tm1["FIO"],dict_tm1["FINV"]+dict_tm1["FIS"])
        dict_t["FASL"] = dict_tm1["FOP"]
        dict_t["FOP"] =␣
↪max(0,dict_t["FED"]+alpha[0]*(DINV-dict_t["FINV"]+dict_t["FBL"]-(alpha[1]*dict_t["FASL"])))

        DICT.append(dict_t)

        costR += dict_t["RINV"]*Cs+dict_t["RBL"]*Cbl
        costW += dict_t["WINV"]*Cs+dict_t["WBL"]*Cbl
        costD += dict_t["DINV"]*Cs+dict_t["DBL"]*Cbl
        costF += dict_t["FINV"]*Cs+dict_t["FBL"]*Cbl
        CR.append(costR)
        CW.append(costW)
        CD.append(costD)
        CF.append(costF)
        costR + costW + costD + costF
        Coutl.append(costR + costW + costD + costF)
        dict_tm1 = deepcopy(dict_t)
```

```
    return costR + costW + costD + costF,CR,CW,CD,CF,DICT,Coutl
```

[ ]:
```
#call of optimal function

DINV,Cs,Cbl,thetas,alphaopt,paramsInit,incomingOrders,delai= 50,0.5,2,0.
↪25,alpha_opts,[0,0,50,0,0,0,0],IO,0.75


Cout, crs, cws, cds, cfs,Dictionnaire,Coutl = costFnsopt(alpha_opts,paramsInit,␣
↪incomingOrders, thetas, DINV, Cs, Cbl, delai)
```

A.2 Stratégie 2

Stratégie 2

September 8, 2023

[ ]:
```
import numpy as np
from copy import deepcopy
import matplotlib.pyplot as plt
from scipy.optimize import *
nombre = 100
```

[ ]:
```
chaine_sequence = "16 11 7 6 16 7 16 9 16 10 9 15 8 10 12 15 10 11 10 10 9 7 7␣
 ↪7 10 8 9 11 13 11 11 4 11 13 8 15 9 9 8 16 10 9 7 11 12 13 10 10 14 10 10 12␣
 ↪8 14 10 15 5 11 12 12 4 9 10 11 13 5 12 8 13 8 9 12 10 12 7 11 8 9 15 7 5 6␣
 ↪14 9 17 7 14 12 8 12 16 16 6 5 12 14 4 15 7 10"
IO = [int(x) for x in chaine_sequence.split()]

print(IO)
```

[ ]:
```
#minization with different stategies for all player

#define of the cost function

def costFn(alpha, paramsInit, incomingOrders, thetas, DINV, Cs, Cbl, delai):

    DICT = []
    CR,CW,CD,CF = [],[],[],[]
    dict_tm1 = {}
    dict_tm1["RED"] = paramsInit[0]
    dict_tm1["ROP"] = paramsInit[1]
    dict_tm1["RINV"] = paramsInit[2]
    dict_tm1["RBL"] = paramsInit[3]
    dict_tm1["RIS"] = paramsInit[4]
    dict_tm1["ROS"] = paramsInit[5]
    dict_tm1["RASL"] = paramsInit[6]
    dict_tm1["RIO"] = incomingOrders[0]
    dict_tm1["WED"] = paramsInit[0]
    dict_tm1["WOP"] = paramsInit[1]
    dict_tm1["WINV"] = paramsInit[2]
    dict_tm1["WBL"] = paramsInit[3]
    dict_tm1["WIS"] = paramsInit[4]
    dict_tm1["WOS"] = paramsInit[5]
    dict_tm1["WASL"] = paramsInit[6]
```

```
    dict_tm1["WIO"] = paramsInit[0]
    dict_tm1["DED"] = paramsInit[0]
    dict_tm1["DOP"] = paramsInit[1]
    dict_tm1["DINV"] = paramsInit[2]
    dict_tm1["DBL"] = paramsInit[3]
    dict_tm1["DIS"] = paramsInit[4]
    dict_tm1["DOS"] = paramsInit[5]
    dict_tm1["DASL"] = paramsInit[6]
    dict_tm1["DIO"] = paramsInit[0]
    dict_tm1["FED"] = paramsInit[0]
    dict_tm1["FOP"] = paramsInit[1]
    dict_tm1["FINV"] = paramsInit[2]
    dict_tm1["FBL"] = paramsInit[3]
    dict_tm1["FIS"] = paramsInit[4]
    dict_tm1["FOS"] = paramsInit[5]
    dict_tm1["FASL"] = paramsInit[6]
    dict_tm1["FIO"] = paramsInit[0]
    costR = dict_tm1["RINV"]*Cs+dict_tm1["RBL"]*Cbl
    costW = dict_tm1["WINV"]*Cs+dict_tm1["WBL"]*Cbl
    costD = dict_tm1["DINV"]*Cs+dict_tm1["DBL"]*Cbl
    costF = dict_tm1["FINV"]*Cs+dict_tm1["FBL"]*Cbl
    CR.append(costR)
    CW.append(costW)
    CD.append(costD)
    CF.append(costF)
    costFunction=costR+costW+costD+costF
    DICT.append(dict_tm1)
    for t in range(1,nombre):
        dict_t = {}
        dict_t["RIO"] = incomingOrders[t]
        dict_t["RED"] = thetas*(dict_tm1["RIO"]) + (1-thetas)*(dict_tm1["RED"])
        dict_t["RINV"] =␣
↪max(0,dict_tm1["RINV"]+dict_tm1["RIS"]-dict_tm1["RBL"]-dict_tm1["RIO"])
        dict_t["RBL"] =␣
↪max(0,dict_tm1["RBL"]+dict_tm1["RIO"]-dict_tm1["RINV"]-dict_tm1["RIS"])
        dict_t["RIS"] = dict_tm1["WOS"]
        dict_t["ROS"] =␣
↪min(dict_tm1["RBL"]+dict_tm1["RIO"],dict_tm1["RINV"]+dict_tm1["RIS"])
        dict_t["RASL"] = dict_tm1["ROP"]
        dict_t["ROP"] =␣
↪max(0,dict_t["RED"]+(alpha[0]*(DINV-dict_t["RINV"]+dict_t["RBL"]-alpha[1]*(dict_t["RASL"]))
        dict_t["WIO"] = dict_tm1["ROP"]
        dict_t["WED"] = thetas*dict_tm1["WIO"] + dict_tm1["WED"]*(1-thetas)
        dict_t["WINV"] =␣
↪max(0,dict_tm1["WINV"]+dict_tm1["WIS"]-dict_tm1["WBL"]-dict_tm1["WIO"])
        dict_t["WBL"] =␣
↪max(0,dict_tm1["WBL"]+dict_tm1["WIO"]-dict_tm1["WINV"]-dict_tm1["WIS"])
```

```
        dict_t["WIS"] = dict_tm1["DOS"]
        dict_t["WOS"] =␣
↪min(dict_tm1["WBL"]+dict_tm1["WIO"],dict_tm1["WINV"]+dict_tm1["WIS"])
        dict_t["WASL"] = dict_tm1["WOP"]
        dict_t["WOP"] =␣
↪max(0,dict_t["WED"]+alpha[2]*(DINV-dict_t["WINV"]+dict_t["WBL"]-(alpha[3]*dict_t["WASL"])))
        dict_t["DIO"] = dict_tm1["WOP"]
        dict_t["DED"] = thetas*dict_tm1["DIO"] + dict_tm1["DED"]*(1-thetas)
        dict_t["DINV"] =␣
↪max(0,dict_tm1["DINV"]+dict_tm1["DIS"]-dict_tm1["DBL"]-dict_tm1["DIO"])
        dict_t["DBL"] =␣
↪max(0,dict_tm1["DBL"]+dict_tm1["DIO"]-dict_tm1["DINV"]-dict_tm1["DIS"])
        dict_t["DIS"] = dict_tm1["FOS"]
        dict_t["DOS"] =␣
↪min(dict_tm1["DBL"]+dict_tm1["DIO"],dict_tm1["DINV"]+dict_tm1["DIS"])
        dict_t["DASL"] = dict_tm1["DOP"]
        dict_t["DOP"] =␣
↪max(0,dict_t["DED"]+alpha[4]*(DINV-dict_t["DINV"]+dict_t["DBL"]-(alpha[5]*dict_t["DASL"])))
        dict_t["FIO"] = dict_tm1["DOP"]
        dict_t["FED"] = thetas*dict_tm1["FIO"] + dict_tm1["FED"]*(1-thetas)
        dict_t["FINV"] =␣
↪max(0,dict_tm1["FINV"]+dict_tm1["FIS"]-dict_tm1["FBL"]-dict_tm1["FIO"])
        dict_t["FBL"] =␣
↪max(0,dict_tm1["FBL"]+dict_tm1["FIO"]-dict_tm1["FINV"]-dict_tm1["FIS"])
        dict_t["FIS"] = min (dict_tm1["FOP"],delai*50)
        dict_t["FOS"] =␣
↪min(dict_tm1["FBL"]+dict_tm1["FIO"],dict_tm1["FINV"]+dict_tm1["FIS"])
        dict_t["FASL"] = dict_tm1["FOP"]
        dict_t["FOP"] =␣
↪max(0,dict_t["FED"]+alpha[6]*(DINV-dict_t["FINV"]+dict_t["FBL"]-(alpha[7]*dict_t["FASL"])))

        DICT.append(dict_t)

        #cost
        costR += dict_t["RINV"]*Cs+dict_t["RBL"]*Cbl
        costW += dict_t["WINV"]*Cs+dict_t["WBL"]*Cbl
        costD += dict_t["DINV"]*Cs+dict_t["DBL"]*Cbl
        costF += dict_t["FINV"]*Cs+dict_t["FBL"]*Cbl
        CR.append(costR)
        CW.append(costW)
        CD.append(costD)
        CF.append(costF)
        costR + costW + costD + costF
        dict_tm1 = deepcopy(dict_t)

    return costR + costW + costD + costF
```

[ ]:
```
#minization
DINV,Cs,Cbl,thetas,alpha,paramsInit,incomingOrders,delai= 50,0.5,2,0.25,[0.
↪5]*8,[0,0,50,0,0,0,0],IO,0.75
bound= [(0, 1)]*8

minimum = minimize(costFn,alpha,args=(paramsInit, incomingOrders, thetas, DINV,␣
↪Cs, Cbl, delai),bounds=bound)
print(minimum)
```

[ ]:
```
#apha opt
a = minimum.x.reshape(8)
alpha_opt = []
for i in range (len(a)):
    alpha_opt.append(a[i])

alpha_opt
```

[ ]:
```
#plot Orders Inventory and cost function with the alphaopt

#optimal function

def costFnopt(alpha, paramsInit, incomingOrders, thetas, DINV, Cs, Cbl, delai):

    DICT = []
    CR,CW,CD,CF, coutl = [],[],[],[],[]
    dict_tm1 = {}
    dict_tm1["RED"] = paramsInit[0]
    dict_tm1["ROP"] = paramsInit[1]
    dict_tm1["RINV"] = paramsInit[2]
    dict_tm1["RBL"] = paramsInit[3]
    dict_tm1["RIS"] = paramsInit[4]
    dict_tm1["ROS"] = paramsInit[5]
    dict_tm1["RASL"] = paramsInit[6]
    dict_tm1["RIO"] = incomingOrders[0]
    dict_tm1["WED"] = paramsInit[0]
    dict_tm1["WOP"] = paramsInit[1]
    dict_tm1["WINV"] = paramsInit[2]
    dict_tm1["WBL"] = paramsInit[3]
    dict_tm1["WIS"] = paramsInit[4]
    dict_tm1["WOS"] = paramsInit[5]
    dict_tm1["WASL"] = paramsInit[6]
    dict_tm1["WIO"] = paramsInit[0]
    dict_tm1["DED"] = paramsInit[0]
    dict_tm1["DOP"] = paramsInit[1]
    dict_tm1["DINV"] = paramsInit[2]
    dict_tm1["DBL"] = paramsInit[3]
```

```
    dict_tm1["DIS"] = paramsInit[4]
    dict_tm1["DOS"] = paramsInit[5]
    dict_tm1["DASL"] = paramsInit[6]
    dict_tm1["DIO"] = paramsInit[0]
    dict_tm1["FED"] = paramsInit[0]
    dict_tm1["FOP"] = paramsInit[1]
    dict_tm1["FINV"] = paramsInit[2]
    dict_tm1["FBL"] = paramsInit[3]
    dict_tm1["FIS"] = paramsInit[4]
    dict_tm1["FOS"] = paramsInit[5]
    dict_tm1["FASL"] = paramsInit[6]
    dict_tm1["FIO"] = paramsInit[0]
    costR = dict_tm1["RINV"]*Cs+dict_tm1["RBL"]*Cbl
    costW = dict_tm1["WINV"]*Cs+dict_tm1["WBL"]*Cbl
    costD = dict_tm1["DINV"]*Cs+dict_tm1["DBL"]*Cbl
    costF = dict_tm1["FINV"]*Cs+dict_tm1["FBL"]*Cbl
    CR.append(costR)
    CW.append(costW)
    CD.append(costD)
    CF.append(costF)
    costFunction=costR+costW+costD+costF
    coutl.append(costFunction)
    DICT.append(dict_tm1)
    for t in range(1,nombre):
        dict_t = {}
        dict_t["RIO"] = incomingOrders[t]
        dict_t["RED"] = thetas*(dict_tm1["RIO"]) + (1-thetas)*(dict_tm1["RED"])
        dict_t["RINV"] =␣
↪max(0,dict_tm1["RINV"]+dict_tm1["RIS"]-dict_tm1["RBL"]-dict_tm1["RIO"])
        dict_t["RBL"] =␣
↪max(0,dict_tm1["RBL"]+dict_tm1["RIO"]-dict_tm1["RINV"]-dict_tm1["RIS"])
        dict_t["RIS"] = dict_tm1["WOS"]
        dict_t["ROS"] =␣
↪min(dict_tm1["RBL"]+dict_tm1["RIO"],dict_tm1["RINV"]+dict_tm1["RIS"])
        dict_t["RASL"] = dict_tm1["ROP"]
        dict_t["ROP"] =␣
↪max(0,dict_t["RED"]+(alpha[0]*(DINV-dict_t["RINV"]+dict_t["RBL"]-alpha[1]*(dict_t["RASL"]))
        dict_t["WIO"] = dict_tm1["ROP"]
        dict_t["WED"] = thetas*dict_tm1["WIO"] + dict_tm1["WED"]*(1-thetas)
        dict_t["WINV"] =␣
↪max(0,dict_tm1["WINV"]+dict_tm1["WIS"]-dict_tm1["WBL"]-dict_tm1["WIO"])
        dict_t["WBL"] =␣
↪max(0,dict_tm1["WBL"]+dict_tm1["WIO"]-dict_tm1["WINV"]-dict_tm1["WIS"])
        dict_t["WIS"] = dict_tm1["DOS"]
        dict_t["WOS"] =␣
↪min(dict_tm1["WBL"]+dict_tm1["WIO"],dict_tm1["WINV"]+dict_tm1["WIS"])
```

```
        dict_t["WASL"] = dict_tm1["WOP"]
        dict_t["WOP"] =␣
↪max(0,dict_t["WED"]+alpha[2]*(DINV-dict_t["WINV"]+dict_t["WBL"]-(alpha[3]*dict_t["WASL"])))
        dict_t["DIO"] = dict_tm1["WOP"]
        dict_t["DED"] = thetas*dict_tm1["DIO"] + dict_tm1["DED"]*(1-thetas)
        dict_t["DINV"] =␣
↪max(0,dict_tm1["DINV"]+dict_tm1["DIS"]-dict_tm1["DBL"]-dict_tm1["DIO"])
        dict_t["DBL"] =␣
↪max(0,dict_tm1["DBL"]+dict_tm1["DIO"]-dict_tm1["DINV"]-dict_tm1["DIS"])
        dict_t["DIS"] = dict_tm1["FOS"]
        dict_t["DOS"] =␣
↪min(dict_tm1["DBL"]+dict_tm1["DIO"],dict_tm1["DINV"]+dict_tm1["DIS"])
        dict_t["DASL"] = dict_tm1["DOP"]
        dict_t["DOP"] =␣
↪max(0,dict_t["DED"]+alpha[4]*(DINV-dict_t["DINV"]+dict_t["DBL"]-(alpha[5]*dict_t["DASL"])))
        dict_t["FIO"] = dict_tm1["DOP"]
        dict_t["FED"] = thetas*dict_tm1["FIO"] + dict_tm1["FED"]*(1-thetas)
        dict_t["FINV"] =␣
↪max(0,dict_tm1["FINV"]+dict_tm1["FIS"]-dict_tm1["FBL"]-dict_tm1["FIO"])
        dict_t["FBL"] =␣
↪max(0,dict_tm1["FBL"]+dict_tm1["FIO"]-dict_tm1["FINV"]-dict_tm1["FIS"])
        dict_t["FIS"] = min(dict_tm1["FOP"],delai*50)
        dict_t["FOS"] =␣
↪min(dict_tm1["FBL"]+dict_tm1["FIO"],dict_tm1["FINV"]+dict_tm1["FIS"])
        dict_t["FASL"] = dict_tm1["FOP"]
        dict_t["FOP"] =␣
↪max(0,dict_t["FED"]+alpha[6]*(DINV-dict_t["FINV"]+dict_t["FBL"]-(alpha[7]*dict_t["FASL"])))

        DICT.append(dict_t)

        #cost
        costR += dict_t["RINV"]*Cs+dict_t["RBL"]*Cbl
        costW += dict_t["WINV"]*Cs+dict_t["WBL"]*Cbl
        costD += dict_t["DINV"]*Cs+dict_t["DBL"]*Cbl
        costF += dict_t["FINV"]*Cs+dict_t["FBL"]*Cbl
        CR.append(costR)
        CW.append(costW)
        CD.append(costD)
        CF.append(costF)
        coutl.append(costR + costW + costD + costF)
        dict_tm1 = deepcopy(dict_t)
    return costR + costW + costD + costF,CR,CW,CD,CF,DICT,coutl
```

[ ]:
```
#call of optimal function
```

```
DINV,Cs,Cbl,thetas,paramsInit,incomingOrders,delai= 50,0.5,2,0.
↪25,[0,0,50,0,0,0,0],IO,0.75


cout, cr, cw, cd, cf,dictionnaire,coutl = costFnopt(alpha_opt,paramsInit,␣
↪incomingOrders, thetas, DINV, Cs, Cbl, delai)
```