Optimisation de l'approvisionnement d'une chaîne logistique multi-échelon dans un cas d'approvisionnement multiple

Agathe Métaireau[1,2], Clarisse Dhaenens[1], Nadarajen Veerapen[1], Alexandre Gerussi[2]

[1] Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL F-59000 Lille
[2] Vekia, 143 rue d'Athènes, 59000, Lille

Mots-clés : *Gestion des stocks, Approvisionnement multi-échelon, Métaheuristiques*

1 Introduction

Dans les problèmes de gestion des stocks d'une chaîne logistique, on cherche à calculer des quantités de produits à commander afin de satisfaire au maximum une demande extérieure incertaine tout en minimisant le nombre de produits restants en stock après satisfaction de cette demande. Une façon classique de résoudre ce problème est de calculer les quantités à commander indépendamment pour chacune des entités qui composent la chaîne logistique. Cette approche, qu'on appelle simple-échelon, est assez efficace car elle s'intéresse à un système dynamique simple. Cependant, une approche simple-échelon ne permet pas de traiter les dépendances entre les entités d'une même chaîne logistique (entrepôt commun par exemple). Afin de traiter les dépendances entre les entités et les contraintes qu'elles ont en commun, on peut s'intéresser au calcul des quantités à commander pour la chaîne logistique dans son ensemble. C'est ce qu'on appelle l'approche multi-échelon [2]. Dans nos travaux, nous cherchons à développer une méthode d'optimisation multi-échelon efficace pour répondre à des problèmes réalistes de gestion des stocks. Cette méthode sera guidée par un cas d'étude réel.

2 Modèle

On s'intéresse au problème de gestion des stocks multi-échelon pour une chaîne logistique avec un approvisionnement multiple. Cela implique que les magasins peuvent s'approvisionner auprès de plusieurs sources différentes (entrepôts ou fournisseurs externes par exemple). Afin de résoudre ce problème, nous avons mis en place un modèle mathématique qui décrit la dynamique d'une chaîne logistique. Ce modèle se veut générique et adaptable à un grand nombre de cas d'étude. Afin de mettre en place un modèle réaliste, nous avons ajouté les contraintes et coûts suivants. Premièrement, nous ajoutons la contrainte de calendrier, qui tient compte des jours de fermeture des entités, où elles ne peuvent pas commander. Nous tenons aussi compte des contraintes de capacité de stockage des entités. Enfin, nous ajoutons un coût fixe de commande et un coût de Franco (on paye un pénalité si on commande en dessous d'un certain seuil). Toutes ces contraintes, exceptée la contrainte de calendrier, sont des contraintes qui s'appliquent sur des groupes de produits. Par exemple, les coûts fixe et de Franco s'appliquent sur une commande complète. Par conséquent, le problème ne peut être décomposé pour chaque produit séparément. De plus, traiter correctement ces coûts implique de trouver un compromis entre commander plus pour éviter de multiplier les coûts fixes ou de payer un coût de Franco, mais stocker les produits plus longtemps, que qui implique aussi un coût. Par conséquent, une vision multi-période du problème est pertinente. Nous avons donc mis en place un modèle multi-échelon, multi-produit et multi-période qui permet de résoudre des problèmes d'approvisionnement au sein d'une chaîne logistique.

3 Premiers résultats

Afin de valider le modèle, nous l'avons appliqué à un cas pratique rencontré en entreprise. Ce cas concerne un réseau logistique de type *"One Warehouse N Retailers"*. C'est donc un réseau logistique à deux niveaux contenant un entrepôt qui dessert un ensemble de magasins. De plus, nous sommes dans un cas d'approvisionnement multiple, donc les magasins peuvent s'approvisionner auprès de l'entrepôt et auprès des fournisseurs extérieurs. Ce cas d'étude contient 905 produits, 163 magasins, un entrepôt et 43 fournisseurs. Le modèle a été implémenté comme un programme linéaire mixte (MILP). Nous avons extrait des instances du jeu de données en sélectionnant des sous-ensemble d'items, et utilisé le MILP pour les résoudre. Nous avons fixé un horizon de 7 jours. Cela nous a permis de valider le modèle et d'évaluer les temps d'exécution pour une résolution exacte. Ces temps d'exécution avec le solver CBC pour une limite d'écart à l'optimal de 10% sont disponibles dans le tableau 1. Les expériences ont été menées sur un ordinateur Dell Latitude 5480 avec un processeur Intel Core i5, 2.60 GHz de CPU et 16 Go de RAM.

Nombre d'items	1	2	5	10	>15
Temps d'exécution	5.20	5.28	6.13	275.70	>7200

TAB. 1 – Temps de résolution en secondes du solver MILP pour les instances du cas d'étude

Ces premières expérimentations nous permettent de voir que les temps d'exécution du programme linéaire en nombres entiers sont déjà très élevés pour les instances réduites. Nous souhaitons résoudre des instances à plusieurs centaines d'items. Nous souhaitons donc travailler sur une métaheuristique qui permettrait d'obtenir des résultats de bonne qualité en temps raisonnable. Pour ce faire, nous avons commencé à implémenter une recherche locale en utilisant le framework jMetalPy ([1]). La recherche locale que nous avons mise en place est fonctionnelle, et nous travaillons actuellement sur la définition d'une solution initiale de bonne qualité et d'opérateurs de voisinage spécifiques. Ces résultats seront présentés lors du congrès ROADEF.

4 Conclusions et perspectives

Les expériences que nous avons menées nous permettent de valider que le modèle permet d'optimiser la gestion des stocks multi-échelon. Nous avons aussi comparé les coûts obtenus avec le modèle multi-échelon avec le moteur simple-échelon déjà en place. Les premiers résultats multi-échelon semblent plus intéressants en termes de coûts que les résultats simple-échelon. A l'avenir, nous aimerions démontrer l'intérêt du modèle multi-échelon de façon plus rigoureuse en mettant en place un protocole de comparaison plus détaillé, du même type que ce lui présenté par Métaireau et al. [3]. De plus, nous allons continuer à travailler sur la recherche locale afin d'obtenir des résultats concluants pour l'optimisation des grandes instances.

Références

[1] Antonio Benítez-Hidalgo, Antonio J Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. jmetalpy : A python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51 :100598, 2019.

[2] Ton de Kok, Christopher Grob, Marco Laumanns, Stefan Minner, Jörg Rambau, and Konrad Schade. A typology and literature review on stochastic multi-echelon inventory models. *European Journal of Operational Research*, 269(3) :955–983, 2018.

[3] Agathe Métaireau, Rabin Sahu, Simon Delecourt, Alexandre Gerussi, and Manuel Davy. Multi-periodic joint replenishment planning method for various all-unit discounts. In *ICORES 2022*, 2022.