

Rapport d’optimisation

Master 2 M.I.A.S.H.S.

Mention : I.M.A.

Spécialité ou Parcours : Ingénierie de la décision et data sciences

Sous la direction de PINSON Eric

Session : Janvier 2023

LAMBERT Clara et JADEAU Guillaume

Faculté : Sciences

Institut Mathématiques Appliquées

Année universitaire 2022-2023



**CHARTE DE NON PLAGIAT**

**Protection de la propriété intellectuelle**

Tout travail universitaire doit être réalisé dans le respect intégral de la propriété intellectuelle d’autrui. Pour tout travail personnel, ou collectif, pour lequel le candidat est autorisé à utiliser des documents (textes, images, musiques, films etc.), celui-ci devra très précisément signaler le crédit (référence complète du texte cité, de l’image ou de la bande-son utilisés, sources internet incluses) à la fois dans le corps du texte et dans la bibliographie. Il est précisé que l’UCO dispose d’un logiciel anti-plagiat dans lms.uco.fr, aussi est-il demandé à tout étudiant de remettre à ses enseignants un double de ses travaux lourds sur support informatique.

*Cf. « Prévention des fraudes à l’attention des étudiants »*

Nous soussignons LAMBERT Clara et JADEAU Guillaume étudiants en master 2 ingénierie de la décision et data sciences s’engageons à respecter cette charte.

Fait à Angers, le 19 septembre

Signature :

LAMBERT Clara et JADEAU Guillaume

Sommaire

Table des matières

[1.Modélisation 2](#_Toc125901119)

[PARTIE 1 – Formalisation PLNE de la problématique 2](#_Toc125901120)

[Métadonnées : 2](#_Toc125901121)

[Variables de décision : 2](#_Toc125901122)

[Fonction objective : 2](#_Toc125901123)

[Contraintes : 3](#_Toc125901124)

[Partie 2 - définition d’heuristiques et métaheuristiques 4](#_Toc125901125)

[1) Heuristiques : 4](#_Toc125901126)

[2) Méta-heuristiques : 9](#_Toc125901127)

[Partie 3 – calculs de bornes supérieurs 13](#_Toc125901128)

[Borne supérieur trivial : 13](#_Toc125901129)

[Borne supérieur : 13](#_Toc125901130)

[Borne inférieur trivial : 13](#_Toc125901131)

[Borne inférieur : 13](#_Toc125901132)

[Relaxation Lagrangienne : 14](#_Toc125901133)

[Partie 4 – Implémentation du projet 17](#_Toc125901134)

[1) Lecture des données 17](#_Toc125901135)

[2) Implémentation de l’heuristique 17](#_Toc125901136)

[3) Implémentation de la métaheuristique 19](#_Toc125901137)

[4) Implémentation de la borne triviale et de la borne Lagrangienne 20](#_Toc125901138)

# 1.Modélisation

## PARTIE 1 – Formalisation PLNE de la problématique

### Métadonnées :

* : Nombre d’articles numéro disponibles
* : Prix de vente de l’article numéro
* : Indice de vente de l’article numéro
* : Capacité de travail du poste
* : Nombre d’exemplaires minimum par lot
* : Nombre d’exemplaires maximum par lot
* : Nombre de types de lots différents maximum
* : Coût de conditionnement d’un T-shirt
* : Indice commercial minimum par lot

### Variables de décision :

représente le numéro du T-shirt,

représente le type de lot

k représente le numéro du poste de conditionnement

* si le type de lot existe, 0 sinon
* le nombre de lots effectués avec le T-shirt du type de lot sur le poste de conditionnement
* le nombre d’articles présents dans le type de lot

### Fonction objective :

### Contraintes :

1. Les articles dans un lot sont tous distincts :
2. Le nombre minimal et maximal de T-shirts dans un lot :
3. L’indice commercial d’un lot est supérieur ou égal au seuil requis :
4. La capacité de travail des postes est respectée (lie les variables et en même temps) :
5. Respect du nombre maximum de types de lot :
6. Articles en nombre limité :
7. Lier les variables et  :
8. Lier les variables et  :

## Partie 2 - définition d’heuristiques et métaheuristiques

### Heuristiques :

#### Sur-mesure :

Cette heuristique est créée de toute pièce pour le problème. Elle se base sur l’utilisation d’un knapsack auquel nous rajoutons quelques étapes.

L’objectif est de créer des types de lots au fur et à mesure que la méthode va tourner.

Tout d’abord nous allons trier par ordre décroissant les t-shirts par leur prix de vente.

Ensuite nous allons faire tourner l’algorithme suivant :

Algorithme :

Nous allons effectuer un knaspsack avec les données suivantes.

* Nous travaillerons avec chaque type de t-shirt mais nous ignorons leur nombre et le remplacerons par 1 (en effet chaque type de t-shirt ne peut être présent qu’une seule fois par type de lot)
* La valeur à maximiser sera le prix des t-shirts
* Chaque t-shirt aura un poids de 1
* La capacité du sac sera de

Le résultat qui ressort nous donne le meilleur type de lot. Mais nous devons vérifier si les contraintes de type de lots sont respectées. Ainsi nous vérifions que le poids est supérieur à  et que l’indiçage est supérieur à .

Ensuite nous maximisons la production de ce type de lot. Ce qui en ressort qu’au moins un type de t-shirt n’est plus disponible. Ainsi nous le retirons de la liste des t-shirt disponible.

Puis nous réitérons cet algorithme autant de fois que le nombre de type de lots différents que nous avons c’est-à-dire  ou si nous n’avons plus de t-shirt.

Les limites de l’algorithme :

* Si nous avons un grand nombre de type de t-shirt et un nombre restreint de type de lot alors l’algorithme va privilégier les t-shirt ayant un prix élevé mais potentiellement peu nombreux car déjà présent dans un autre type de lot plutôt que des t-shirt à bas prix mais qui sont disponible en grande quantité. Ce qui entrainera potentiellement des types de lots en quelques exemplaire uniquement alors qu’il aurait pu y avoir des lots présent en des milliers d’exemplaires. Ce qui entraine un manque à gagner pouvant être grand en l’absence d’utilisation de tous les t-shirts et donc aussi des stocks important à gérer derrières.

Avantages de l’algorithme :

* L’algorithme est facile à implémenter
* L’algorithme est rapide à faire tourner
* L’algorithme donne une borne inférieur qui permet d’être améliorée par une méta-heuristique

#### Gloutonne

#### Pour utiliser la méthode Gloutonne nous utiliserons la formalisation du corrigé numéro 5.

Construction incrémentale d’une solution de e dans E :

Une image contenant texte

Description générée automatiquement

Pour résoudre un nouveau problème selon ce principe il faut définir :

* Une solution construisant un « début de solution »
* Une fonction donnante/ mettant à jour l’ensemble des candidats
* Une fonction heuristique évaluant la qualité de chaque candidat

**Fonction de début de solution**

Une solution réalisable, mais l’une des pires solutions en termes de profit est de ne produire qu’un seul exemplaire de chaque type de lot et de ne mettre que le stricte nécessaire dans chaque type de lot. C’est-à-dire le nombre minimale de t-shirt, ne pas dépasser le nombre maximal et respecter l’indiçage nécessaire dans chaque type de lot.

Pour ce faire nous appliquerons un bin packing dans lequel nous ne disposerons que d’un seul t-shirt de chaque type au lieu de di. Il faudra qu’il vérifie quand même les restriction mentionné en amont.

Ensuite nous produisons au maximum chaque type de lot en partant du premier défini jusqu’au dernier. Nous obtenons donc des t-shirt non utilisés potentiellement très nombreux. Si il ne nous en reste pas nous avons atteint un optimum pour vider nos stock mais pas forcément un optimum de profit.

**Fonction mettant à jour l’ensemble des candidats**

Nous devons évaluer les t-shirt restants pouvant être affecté à un type de lot, cela nous permet d’obtenir une solution améliorable. Pour cela nous appliquons les contraintes citées lors de la modélisation. Auquel nous ajoutons une contrainte vérifiant qu’un nombre de t-shirt disponible est au moins aussi grand que la production d’un type de lot auquel nous voulons qu’il appartienne.

Pour illustrer, à chaque itération, il deviendra impossible d’affecter certains t-shirt, en effet au fur et à mesure que nous affectons des t-shirt, les types de lots vont atteindre leur nombre maximal de t-shirt. Cette contrainte est modélisée par :

**Fonction heuristique évaluant la qualité de chaque candidat**

Dans notre cas, les candidats sont les affectations d’un t-shirt de type i à une type de lot k auquel il n’appartient pas (Sk représente les t-shirt présent dans le type de lot k),

Ainsi, nous regardons pour un t-shirt disponible présente en assez grand nombre, nous calculons l’intérêt pour le t-shirt d’être présente dans le type de lot k. C’est-à-dire que nous multiplions le nombre de fois que le lot k est produit auquel nous multiplions le prix pour voir le gain.  
 **Limite de la méthode gloutonne**

Cette méthode gloutonne est hautement calculatoire si nous disposons de nombreux t-shirt pour un nombre restreints de type de lot. De plus il se peut qu’il nous reste quand même de nombreux t-shirt dans nos stock à la fin.

**Formalisation de la méthode gloutonne**

e ← i∈V , x(i,k) = *solution initiale du bin packing*

Cand ← Ensemble des t-shirt pouvant être ajoutés à e respectant le nombre de disponibilités et vérifier que pour le type de lot k emax ne soit pas atteint.

Tant que Cand ≠ Ø faire

Application du calcul des scores à l’affectation d’un t-shirt i à un type de lot k

Choisir le meilleur composant de Cand par trie décroissant des scores et l’ajouter à e.

emax -= 1

Mise à jour de l’ensemble Cand pouvant être ajouté à e.

Retourner e

1. Recuit simulé

Le recuit simulé, du a sa simplicité est très souvent utilisé dans l’optimisation. Il s’appuie sur l’algorithme de Metropolis-Hastings, qui permet de décrire l’évolution d’un système thermodynamique. On chercher, en refroidissant le matériau lentement, le maximum global. La fonction à minimiser est l’énergie E du système et on introduit un paramètre fictif, la température T du système.

Cet algorithme part d’un seul point dans l’espace des solutions (un ordonnancement heuristique S) et essaie d’étudier l’espace des solutions afin de trouver un meilleur horaire S’.

Dans le cadre d’une fonction objective de maximisation de profit, si le nouvel horaire S’ a une valeur plus grande que le meilleur horaire S précédent alors on sélectionnera S’ comme meilleur horaire ( S <- S’). Si le nouvel horaire S’ n’est pas meilleur que le précédent horaire S, alors S sera accepté avec la probabilité P(S,S’,T), avec T la Température.

Après chaque étape, le système se refroidit en diminuant la ‘température’ et la probabilité P(S,S’,T), qui est déterminée pour l’acceptation d’un nouvel horaire.

Dans notre cas :

1. On choisit une température de départ T=T0 (généralement choisit très grand au départ) et une solution initiale S=S0 (qui peut être générée grâce à une heuristique classique, comme l’algorithme glouton par exemple)
2. On génère une solution aléatoire S’ = dans le voisinage de la solution actuelle.
3. On compare nos solutions S et S’ selon le critère de Metropolis
4. On répète les étapes 2 et 3 jusqu’à ce que l’équilibre (le meilleur horaire) soit atteint
5. On décroit la température et on répète jusqu’à ce que le système soit gelé(T = 0)
   * Décroissance continue :

On fait baisser la température d’une façon continue, par exemple en utilisant cette loi :

Attention à ne pas choisir un trop grand sinon la température baissera trop vite, et inversement avec un trop petit la température baissera trop lentement.

* + Décroissance par paliers :

Pour chaque valeur de la température, on itère l’algorithme de Metropolis jusqu’à atteindre un équilibre statistique, puis on diminue la température.

Amélioration de l’algorithme :

* On peut améliorer l’algorithme du recuit simulé en ajoutant une variable qui mémorise la meilleure valeur rencontrée jusqu’à présent. En effet sans ça, il est possible que l’algorithme puisse converger vers une certaine solution (qui n’est pas la meilleure).

Avantages :

* Il est facile à implémenter
* Possède un bon ensemble de solutions
* Peut être utilisé dans la majorité des problèmes d’optimisation
* Lorsque le nombre d’itérations tend vers l’infini, il converge vers l’optimum global.

Inconvénients :

* L’algorithme peut se piéger. C’est-à-dire qu’une fois à basse température dans un minimum local, il ne peut pas en sortir.
* Il est compliqué de choisir une température de départ (trop basse : recherche peu qualitative, trop élevée : gros temps de calcul)
* On ne peut pas savoir si la solution trouvée est optimale.

### Méta-heuristiques :

#### Améliorer l’heuristique 1

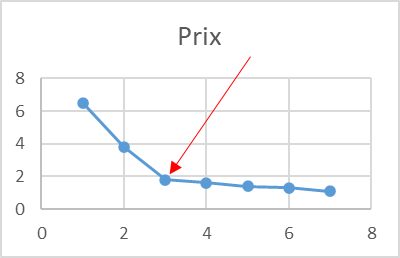
A partir de la première heuristique « Sur-mesure » nous pouvons voir plusieurs améliorations de son résultat. Les méthodes proposées ci-après n’assurent pas d’obtenir un optimum mais assure en revanche d’améliorer le résultat précédemment calculée.

##### Algorithme génétique

Nous disposons d’une solution composée des meilleurs performances individuelles pour chaque type de lot mais ne tenant pas compte du nombre produit de chaque type de lot. L’objectif va être de faire tourner l’algorithme de l’heuristique plus de fois et de regarder des solutions intéressante diminuant le nombre de stock restant et augmentant le profit net. Avec ces nouvelles solutions nous allons dépasser la valeur maximale du nombre de type de lot Lmax. Mais nous allons essayer d’interchanger des ensembles de t-shirt présent dans différent type de lot et voir si la solution est améliorée. Tout en gardant à l’esprit qu’il faut satisfaire les contraintes de type de lot. Donc il ne faut pas qu’un t-shirt soit présent plusieurs fois dans le lot, que le nombre de t-shirt dans le lot soit compris entre Emin et Emax et que la valeur de l’indiçage du lot dépasse Rmin.

Algorithme :

e ← solutions après avoir fait tourner l’algorithme de ‘Sur-mesure’ jusqu’à ce qu’il ne puisse plus resté de possibilité de faire des types de lot



Coude

Tant que Card(e) > Lmax faire :

Prendre a un sous ensemble de t-shirt du type de lot A, A étant le dernier type de lot de la liste e avec a les x t-shirts apportant le plus de profit dans A, x étant déterminé par une représentation graphique des prix des t-shirts et en cherchant le coude du graphe (cf ci-contre)

a

Pour chaque B dans e :

Interchanger avec un sous-ensemble de la solution B de telle manière que le nombre de t-shirt dans b ne dépasse pas Emax et soit supérieur à Emin tout en gardant son indiçage supérieur à Rmin

Si la solution est meilleure alors :

Nous gardons cette nouvelle solution

Nous retirons A de e

Fin Si

Fin Pour

Si A est dans e alors :

Nous retirons A de e

Fin Si

Retourner e

Limites de l’algorithme :

* L’algorithme est de complexité O( (Card(e)-Lmax) x (Card(e)-1) ) soit en O(n2), il serait donc être long à faire tourner si Lmax est élevé
* L’algorithme peut ne pas atteindre l’optimum mais converge vers une solution meilleure que l’originel
* L’algorithme est assez complexe à mettre en place

Avantages de l’algorithme :

* L’algorithme dépend de Lmax et dans notre exemple Lmax est réduit donc l’algorithme est rapide à faire tourner

##### Destruction, reconstruction

L’objectif de cette méta-heuristique est de détruire la production d’un des types de lots présent dans la solution. En se faisant nous libérerons des t-shirts et nous essaierons de voir si il est possible d’augmenter la production d’autres types de lot présent dans la solution. Une fois cette production augmentée, nous allons réappliquer l’algorithme de l’heuristique ‘Sur-mesure’ pour rajouter un nouveau type de lot que nous allons produire au maximum lui aussi. Ainsi si la solution apporte un meilleur profit nous gardons le nouveau résultat sinon nous gardons l’ancien et nous répèterons ce processus autant de fois qu’il y a de Type de lot.

Algorithme :

e ← solutions après avoir fait tourner l’algorithme de ‘Sur-mesure’

Pour chaque A dans e faire :

Détruire le type de lot A et incrémenter le nombre de t-shirt disponible utilisé dans A

Pour chaque B dans e B A faire :

Augmenter sa production au maximum

Fin Pour

Appliquer l’algorithme de ‘Sur-mesure’ autant de fois qu’il reste de place pour atteindre Lmax

Si la solution est meilleure alors :

A est retiré de e

Nous gardons ces nouvelles productions et ajoutons les nouveaux types de lots à e

Fin Si

Fin Pour

Limites de l’algorithme :

* Cet algorithme est de complexité O(n2) ce qui fait que pour un Lmax important il serait long à faire tourner
* L’algorithme pourrait ne pas atteindre l’optimum si sa solution de départ est mal construite. Il dépend donc de la solution d’origine

Avantages de l’algorithme :

* L’algorithme est simple à modéliser et à faire tourner
* Comme le nombre de type de lot de l’exemple est réduit il est rapide à faire tourner
* L’algorithme est convergeant et sa solution tend vers l’optimum

#### Recherche-local/ Tabou (pistes de réflexion)

Nous utilisons une fonction objectif augmentée avec des pénalités (alpha, beta et gamma) sur les contraintes.

Alpha 🡪 pénalité sur la fonction objectif primaire

Beta 🡪 pénalité sur l’indiçage minimum à respecter

Gamma 🡪 pénalité sur l’excès de capacité des type de lots (Emax)

Nous obtenons ainsi une fonction objectif augmentée correspondant à la somme des contraintes multipliées par leur pénalité respective.

Nous procédons ensuite à des voisinages. Nous prenons donc notre stratégie précédente puis Nous utilisons notre recherche locale :

* Nous pouvons par exemple que l’indiçage est réduit
* Nous pouvons aussi augmenter Lmax

## Partie 3 – calculs de bornes supérieurs

Nous utiliserons les modélisations et notations des formalisations 3, 4, 5 et 6 du corrigé.

### Borne supérieur trivial :

Une borne supérieur trivial (BST) est de vendre tous les t-shirts sans tenir compte des contraintes des types de lots. Ce qui nous donnerai :

### Borne supérieur :

Une borne supérieur (BS) serait de vendre tous les t-shirts sans tenir compte de certaines contraintes comme celle de l’indiçage (r(i)) ou ne pas restreindre le nombre de type de lot (Lmax). Ce qui nous donnerai un nouveau modèle à résoudre plus simple et BS serait la fonction objective :

### Borne inférieur trivial :

Une borne inférieur trivial (BIT) est de ne vendre aucun t-shirt donc pas de gain mais pas de coûts de conditionnement non plus. Ce qui nous donnerai un résultat de :

### Borne inférieur :

Une borne inférieur (BI) pourrait être la modification de certaines contraintes comme la diminution du nombre de type de lot (Lmax) ou la diminution du nombre de t-shirt présent dans un lot (emax). Ce qui nos donnerai un nouveau problème à résoudre. Cela pourrait également être la transformation des coûts de conditionnement en coût fixe représentant le conditionnement de l’ensemble des t-shirt même les invendus. Ce qui nous donnerai la fonction objective privée des coûts de conditionnement :

### Relaxation Lagrangienne :

Nous voulons calculer une borne supérieure LB de l’optimum. Celle-ci est basée sur la dualisation Lagrangienne des contraintes [2] et [3] de notre modélisation. Nous noterons λ, ρ et ϕ les multiplicateurs de Lagrange associés.

#### Fonction de Lagrange

Exprimons la fonction de Lagrange associé à cette relaxation Lagrangienne :

Ce qui nous donne comme fonction de Lagrange :

)

Notre fonction de Lagrange dépend de quatre paramètres, x notre variable de décision de notre ancienne fonction objective, λ, ρ et ϕ nos multiplicateurs de Lagrange issues respectivement de nos contraintes [2], [2] et [3]. Ensuite nous avons factorisé par nos variables de décisions pour nous permettre d’en déduire notre sous-problème Lagrangien associé.

#### Résolution du sous-problème de Lagrange

A partir de cette nouvelle fonction nous pouvons définir le sous-problème de Lagrange associé [SPL].

Sous problème de Lagrange associé

A partir de cette nouvelle fonction nous pouvons définir le sous-problème de Lagrange associé [SPL].

Sc

IN

#### Résolution du sous-problème de Lagrange

Une fois que nous disposons du sous-problème de Lagrange associé à notre problème initial, nous pouvons le résoudre.

Pour ce faire il nous faut calculer les coûts régularisés associés à nos variables de décisions. Nous en aurons donc trois à calculer que nous appellerons c̃1, c̃2 et c̃3 associé respectivement à x, y et z.

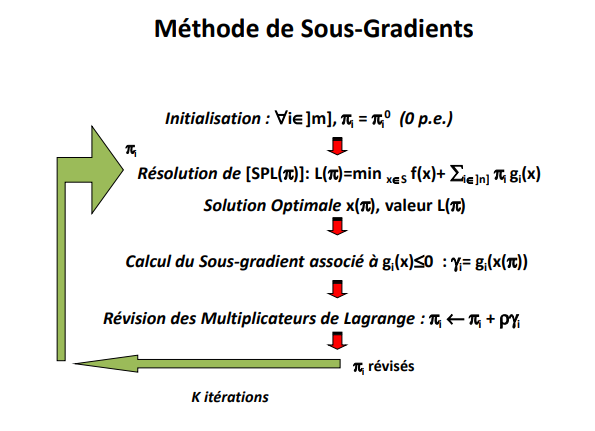
Chacun d’entre eux est égal à la factorisation devant leur variable associée. Ainsi :

c̃1 =

c̃2 =

c̃3 = -cd

Nous utiliserons ensuite un algorithme de sous-gradients tel que :



Avec **π** = (λ, ρ, ϕ)

f(x) =

et g (y) =

λ

ρ

g (y) = ( )

ϕ

g (y) =

Une fois l’algorithme terminé nous obtiendrons la solution optimale comme l’algorithme converge.

## Partie 4 – Implémentation du projet

## Lecture des données

La première étape de cette quatrième et dernière phase du projet a été d’importer et de lire les jeux de données qui nous ont été communiqués. Nous les avons importés à travers le langage python qui sera le langage de programmation de cette quatrième partie.

Une fois le programme lancé, la première possibilité qui s’offre à nous est ainsi de choisir le fichier de données que nous souhaitons utiliser. Par défaut, le fichier sélectionné est «BenchMarksVE.xls » plus précisément le premier jeu de données.

Cette étape, au-delà de l’aspect de lecture et d’importation, a été essentielle car elle nous a permis de construire chacun de nos tableaux de données, qui nous faciliterons par la suite la conception de notre heuristique et métaheuristique.

## Implémentation de l’heuristique

La deuxième étape de cette phase 4 est donc de programmer notre heuristique sous Python. Nous avons choisi d’implémenter l’heuristique ‘’sur-mesure’’ du knapsack itératif et l’heuristique gloutonne que nous avions conçue au sein de la phase 2 du projet.

Nous rappelons qu’il convient uniquement d’obtenir une solution réalisable de notre problème, solution respectant ainsi nos différentes contraintes.

Au terme de cet algorithme, nous obtenons ainsi une solution réalisable du problème. Nous pouvons voir trois informations principales communiquées en sortie par Python :

* La composition des lots
* Le nombre de fois que le lots a été produit
* Le poste de conditionnement

Nous rappelons à nouveau que nos heuristiques ne nous donne qu’une solution réalisable. C’est-à-dire qu’elles respectent chacune des contraintes, cependant elle ne détiennent pas forcément le gain maximal. C’est le rôle que nous donnerons à notre métaheuristique de s’en approcher le plus possible.

**Résultats obtenus :**

Il convient désormais de notifier certaines de nos solutions obtenues, plus particulièrement nos gains :

Pour notre heuristique sur mesure la composition et la production de nos lots sur le premier jeu sont :

T-shirt numéro : (commençant par 0)

Production :

[1, 3, 6, 12] 6403

[3, 4, 6, 12] 2277

[3, 4, 12, 22] 4008

[4, 5, 12, 22] 3222

[5, 9, 12, 22] 272

[5, 9, 11, 22] 2169

[5, 9, 11, 20] 1726

[0, 9, 11, 20] 7270

[0, 9, 20, 21] 3466

[9, 10, 20, 21] 361

[10, 14, 20, 21] 997

[2, 10, 14, 21] 2517

[2, 10, 14, 16] 149

[2, 13, 14, 16] 2632

[2, 14, 16, 19] 2314

Nous obtenons un gain de 559 031 euros.

Pour notre heuristique gloutonne la composition et la production de nos lots sur le premier jeu sont :

[4, 13, 10, 21] 12688

[2, 7, 5, 23] 6403

[6, 12, 1, 15] 7389

[22, 3, 17, 19] 7341

[11, 8, 24, 16] 4024

[13, 12, 15, 17] 3494

[7, 5, 23, 10] 2277

[21, 1, 3, 8] 1132

[19, 24, 16], 4985

Nous obtenons un gain de 602118 euros.

Nous remarquons la différence de gain entre la méthode ‘’sur-mesure’’ et la méthode gloutonne donnant plus de 43 000 euros de plus pour un nombre plus restreint de type de lot.

Pour notre heuristique sur mesure la composition, la production et la distribution de nos lots sur le deuxième jeu sont :

[9, 11, 12, 15, 18, 23] 4980 Poste 1 : 29880 t-shirts

[9, 11, 14, 15, 18, 23] 1693 Poste 1 : 10158 t-shirts

[9, 13, 14, 15, 18, 23] 623 Poste 1 : 3738 t-shirts

[9, 13, 14, 15, 18, 22] 930 Poste 1 : 5580 t-shirts

[13, 14, 15, 18, 19, 22] 4369 Poste 1 : 642 et Poste 2 : 25572 t-shirts

[8, 13, 14, 15, 19, 22] 4043 Poste 2 : 24258 t-shirts

[0, 8, 13, 14, 19, 22] 324 Poste 2 : 168 et Poste 3 : 1776 t-shirts

[0, 8, 14, 17, 19, 22] 3591 Poste 3 : 21546 t-shirts

[0, 8, 10, 17, 19, 22] 3433 Poste 3 : 20598 t-shirts

[0, 8, 10, 19, 21, 22] 2130 Poste 3 : 12780 t-shirts

[0, 7, 8, 10, 19, 21] 1286 Poste 3 : 3300 et Poste 4 : 4416 t-shirts

[0, 2, 7, 8, 10, 21] 2372 Poste 4 : 14232-shirts

[0, 2, 4, 7, 8, 10] 1034 Poste 4 : 6204 t-shirts

[0, 2, 3, 4, 7, 8] 373 Poste 4 : 2238 t-shirts

[0, 2, 3, 4, 5, 7] 100 Poste 4 : 600 t-shirts

Nous obtenons un gain de 663 986 euros.

Concernant la répartition du conditionnement des lots sur les différents postes. Nous voyons très rapidement sur les jeux de données qu’il s’agit surtout là d’une aide pour la production car la contrainte est toujours respectée du fait d’une capacité supérieure au nombre total de t-shirt.

## Implémentation de la métaheuristique

Par la suite nous avons développé notre métaheuristique. Tout comme l’heuristique nous avons choisi d’en prendre une de notre partie 2. Il s’agit d’un algorithme génétique. Nous avons choisi pour celle-ci de tester l’ajout d’un t-shirt à la fois et d’en retirer un autre pour voir si le gain était accru en fonction également de la production actuel et de la nouvelle. S’il l’était alors nous inversions les 2 t-shirts et modifions la composition du lots et la quantité produite. Ainsi nous rajoutions les t-shirts retirés dans les données restantes tout en enlevant ceux intégrés.

Cette nouvelle solution est donc bien réalisable et optimise notre score.

Une fois que l’algorithme à finit de tourner nous nous sommes rendu compte que la solution n’était pas toujours nettement améliorée.

Voici le tableau de nos résultats en utilisant les résultats de nos heuristiques ‘’sur-mesure’’ :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Score initial : | 559 031 | 663 986 | 709 630 |
| Nouveau score : | 590 514 | 693 654 | 715 673 |
| Temps d’exécution : | 1 500 minutes (arrêt forcé) | 1 650 minutes (arrêt forcé) | 1630 minutes (arrêt forcé) |

Nous pouvons voir que le score initial du jeu test 1 s’améliore de 30 000 environ de même pour le test 2 mais que de 6 000 pour le test 3. Les résultats s’améliorent donc légèrement mais pour un temps de calcul assez élevé. En effet au minimum le temps de calcul arrondi était de 1 500 minutes soit 25 heures. Il faut bien prendre en compte que l’algorithme a été arrêté prématurément en raison de son temps de calcul élevé. Cette méthode basée sur l’analyse de chaque ajout possible est probablement trop longue. Il faudrait surement y incorporer de l’aléatoire pour améliorer son efficacité au niveau du temps de calcul et ressortir de meilleurs résultats.

## Implémentation de la borne triviale et de la borne Lagrangienne

À la suite de nos résultats nous avons voulu vérifier la pertinence de ceux-ci par rapport à leur borne supérieur trivial. Nous avons choisi de coder une borne trivial de la partie 3. Celle-ci ne prenant en compte aucune contraintes et donc calcul la quantité des t-shirts par leur prix en y retirant les coût de conditionnements. Voici sa formule :

Voici les résultats obtenus :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Borne sup trivial : | 682 358.07 | 796 387.72 | 841 731.5 |

Nous pouvons remarquer que pour le jeu 1 nous obtenons un résultat de plus de 86% de la borne supérieur trivial. En rappelant que cette borne supérieure ne renvoie pas forcément une solution réalisable le pourcentage obtenu peut être plus fort par rapport à l’optimum.

Nous remarquons que pour le deuxième jeu nous obtenons un résultat de plus de 87% de la borne supérieur trivial.

Enfin, nous voyons que pour le troisième jeu nous obtenons un résultat de plus de 85% de la borne supérieur trivial.

Les résultats que nous obtenons sont relativement proche de la borne supérieure triviale. Faisant de nos solutions des solutions intéressantes à proposer.

Après cela nous avons décidé de calculer notre borne Lagrangienne. Voici sa formule :

Nous avons choisi comme paramètre,  :

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 |
| Borne lagrangienne : | 606 981 | 734 336 | 769 480 |

Les résultats que nous obtenons avec nos métaheuristiques sont assez proches de nos bornes lagrangienne.

Pour conclure nous remarquons que notre heuristique et métaheuristique nous fournissent des résultats plutôt intéressants. Cependant nous ne pouvons pas être sur qu’il s’agisse de l’optimum sans une étude plus approfondi du sujet. Les temps d’exécution deviennent très importants quand les données augmentent. Nous pourrions attendre la fin d’exécution de notre métaheuristique pour parfaire nos résultats. Nous pourrions également comparer nos résultats avec ceux de d’autres heuristiques et métaheuristiques comme la génération de colonne afin de voir quels algorithmes sont les plus pertinents pour résoudre ce problème.