République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Modèle Intelligent et Décisions (M.I.D)*

Thème

Optimisation des découpes des feuilles 2D

Réalisé par :

- **Bouarfa Amal Yassmina**

Présenté le 29 septembre 2022 devant le jury composé de MM.

- *Mr.HADJILA Fathallah.* *(Président)*
- *Mr.BELABED Amine* *(Encadreur)*
- *Mr.SMAHI Ismail* *(Examinateur)*

Année universitaire : 2021-2022

Dédicaces

Je dedie ce travail à :

Mon très cher PAPA.

Aucune dédicace ne saurait exprimer mes respects, ma reconnaissance et mon profond amour. Puisse Dieu vous préserver et vous procurer santé et bonheur .

Je vous souhaite un prompt rétablissement Inchallah

Remerciements

Tout d'abord je tiens à remercier **ALLAH** le tout puissant de m'avoir donné la santé, la volonté, le courage et la patience pour mener à terme ma formation et pouvoir réaliser ce travail.

Qu'il me soit permis, Monsieur **BELABED AMINE**, de vous exprimer mes remerciements, ma gratitude, et tout le respect pour votre assistance continue.

J'exprime aussi ma gratitude à messieurs **SMAHI ISMAIL** et **HADJILA FETHALLAH** qui ont eu l'amabilité d'évaluer ce travail.

J'adresse mes sincères remerciements à tous les professeurs, et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions.

Table des matières

Liste des figures

Acronymes

CSP =Cutting Stock Problem

2DBPP =2D Bin Packing Problem

HTML =Hyper Text Markup Language

NPM = Node Package Manager

CSS = Cascading Style Sheets

UML = Unified Modeling Language

Introduction Générale

[] De nombreux matériaux utilisés dans l'industrie et la construction se présentent initialement sous la forme d'unités entières de taille standard (feuilles de verre, bobines d'étain, contreplaqué, tôle, etc.). Afin d'utiliser ces matériaux directement ou d'obtenir des produits semi-finis, il est nécessaire de les diviser en parties ayant les dimensions requises. Ce faisant, des déchets se forment généralement et le matériau réellement utilisé ne représente qu'un pourcentage de la quantité totale, tandis que le reste est jeté. Il est vrai que dans certains cas une certaine application peut être trouvée pour les restes, mais leur utilisation soit nécessite des dépenses supplémentaires (le matériau résiduel doit être refondu, soudé, etc.) et est donc associée à des pertes, soit il est utilisé comme un produit final avec beaucoup moins de valeur que les produits originaux obtenus (les restes de bois de construction sont utilisés comme combustible, par exemple). Par conséquent, la minimisation des vestiges est clairement un problème important et réaliste, car elle permet de réduire la perte de matériaux précieux.

Le problème découpe a fait l'objet de nombreuses études et a abouti à des centaines de solutions et de résultats. Les raisons pour cela sont multiples. D'abord ,il existe de nombreuses variantes de ces problèmes qui génèrent des modèles et différentes solutions de contournement. Ensuite, tous les intérêts industriels et économiques derrière ces questions sont importantes, conduisant à des recherches permanentes sur ces sujets connus au cours des années. En fin de compte, les préoccupations environnementales ont été le moteur initial pour notre étude.

Dans ce travail nous nous sommes intéressé à la modélisation et la résolution des problèmes de découpe en 2D .Notre solution est basée sur l'utilisation de l'algorithme génitique, ce dernier est exprimées en réduisant le probléme de découpe à un problème de Bin paking 2D . Le travail a été réalisé dans le cadre de l'obtention du diplôme de Master en Informatique Option : Modèle Intelligent et Décisions (M.I.D)

L'application consiste donc à optimiser et générer le stock notamment en matiére premiére et de chutes. Le but étant donc de générer une liste des découpes optimisée . Notre travail tente de répondre à des problèmatiques d'optimisation des coûts de production.

Comment découper un nombre donné d'éléments de manière à utiliser une quantité minimale des feuilles, et une quantité maximale de chute ?

Plan du mémoire

En plus de ce chapitre introductif, ce document se compose de trois chapitres, suivis d'une conclusions générale.

Le premier chapitre introduit le cadre théorique de notre travail, tel que les problèmes d'optimisation ; les algorithmes et les exemples.

Le deuxième chapitre introduit spécifiquement le problème de découpage, y compris la classification et la formalisation d'un problème de découpe.

Le chapitre 3 décrit, d'une part, la conception et la mise en œuvre de notre solution. Il montre également des études fonctionnelles et non fonctionnelles telles que le diagrammes de cas d'utilisation, et le diagrammes de séquence de système ; et d'autre part ,l'étude technique et les technologies utilisée, ainsi qu'une présentation de l'application en décrivant l'architecture de système et l'algorithme de découpage utilisé.

Chapitre 1

Optimisation, Notions de Base

1.1 Introduction

L'optimisation combinatoire définit un cadre formel pour de nombreux problèmes de de l'industrie, de la finance ou de la vie quotidienne. Les problèmes d'optimisation combinatoire sont habituellement définis comme une problématique de choix d'une meilleure alternative dans un ensemble très grand mais fini d'alternatives. En raison du très grand nombre d'alternatives pour ces problèmes, l'ensemble des alternatives, dit aussi ensemble de solutions réalisables, est défini en compréhension, en d'autres termes les solutions réalisables se distinguent par un ensemble de propriétés ou de conditions, dites aussi contraintes, qu'elles doivent toutes remplir. Une évaluation est associée à toute solution réalisable à l'aide d'une fonction dite fonction objectif. Un certain nombre de techniques peuvent être employées pour résoudre les problèmes d'optimisation combinatoire, notamment l'énumération, la programmation dynamique, la programmation linéaire et les techniques d'intelligence artificielle. Les problèmes d'optimisation combinatoire font souvent appel aux techniques d'intelligence artificielle. La principale fonction des techniques de résolution est de déterminer l'ordre de recherche. La technique de solution est basée sur la création de sommets et l'évaluation ultérieure de l'évolution du système au fur et à mesure que l'arborescence se développe. Les solutions locales sont améliorées en comparant la solution actuelle avec les limites inférieures et supérieures d'une fonction objective au fur et à mesure que le système se développe. L'évaluation est basée sur la comparaison de la solution actuelle avec les limites inférieures et supérieures. Nous nous intéressons principalement dans ce chapitre au problèmes d'optimisation et ses différentes variantes ,dont on va énumérer les problèmes d'optimisation ses types ainsi les algorithmes proposée [1]

1.2 Définition des problèmes d'optimisation

Un problème d'optimisation, noté P, est caractérisé par un ensemble réalisable ou admissible X non-vide et une fonction objectif f qui associe un scalaire dans R à chaque élément x de l'ensemble X. Les éléments de X sont dits solutions réalisables. En résolvant le problème P on trouve parmi les valeurs possibles, une solution qui minimise ou maximise f, c'est-à-dire, dans le cas du problème de minimisation , trouver une solution $x \in X$ telle que $f(x) \geq f(x)$ pour tout élément x dans X. Une telle solution est dite optimale et sera notée $X(X, f)$

On se contentera de la présentation des problèmes de minimisation,les problèmes de maximisation suivent les mêmes règles, à quelques variations près. Nous ne considérons ici que les problèmes qui admettent au moins une solution optimale. Un problème d'optimisation peut avoir plusieurs fonctions objectifs, c'est donc un problème d'optimisation multicritère ou multiobjectif.

L'ensemble réalisable X est habituellement défini comme partie de solutions réalisables peuvent alors être représentées comme des vecteurs dont les n composantes sont les variables du problème. L'ensemble X est délimité couramment par un système d'inégalités appelées contraintes du problème. Les contraintes sont construites à l'aide de combinaisons des variables,et permettent de caractériser les propriétés communes aux solutions de X afin de les distinguer parmi toutes les solutions de R n. La description de l'ensemble X est donc implicite. Les programmes linéaires sont sans doute les problèmes d'optimisation les plus connus. La fonction objectif et les contraintes de ces problèmes sont linéaires. [2]

1.3 Types de problèmes d'optimisation

Dans cette section nous allons citer queleques type de problèmes d'optimisation

1.3.1 Problèmes d'optimisation discrète

Les Problèmes d'optimisation discrets, à l'opposé de l'optimisation continue, un type de problème d'optimisation spécialement étudié. Tout ou partie des variables de ce type appartiennent à l'ensemble des entiers, soit

$X \subseteq \ \mathbb{Z}^m \ \mathbb{R}^{n^- m}$ avec $0 \leq \ m \leq \ n$

Lorsque $m = n, P(X, f)$ est considéré comme un problème entier, sinon c'est l'un des problèmes d'optimisation mixte en nombres entier.Bien que l'ensemble réalisab soit plus restreint à l'optimisation discrète, ces problèmes sont souvent plus difficiles dans leurs versions continues. Les ensembles réalisable de problèmes d'optimisation et de problèmes possibles sont infinis, et les problèmes en nombres entiers sont au plus dénombrables. Nous présentons dans la partie suivante les problèmes qui nous intéressent, à savoir les problèmes d'optimisation combinatoire.

1.3.2 Optimisation combinatoire

La combinatoire en mathématiques est une branche dont le but est de dénombrer les arrangements possibles qui peuvent être formés à l'aide des éléments d'un ensemble fini.

La définition de l'optimisation combinatoire est directement liée à la recherche de problèmes. Le but de la résolution d'un tel problème est de trouver l'élément optimal à partir d'un ensemble dénombrable fini. Cet objet peut être une structure entière, sous-jacente ou graphique. Un problème d'optimisation combinatoire consiste à trouver le meilleur dans un ensemble discret appelé l'ensemble des solutions possibles. En général, cet ensemble est fini mais comporte un grand nombre d'éléments et il est décrit de cette manière, c'est-à-dire par une liste relativement courte de contraintes satisfaisant les solutions possibles. Pour définir la notion de meilleure solution, une fonction, dite fonction objectif, est introduite. Pour chaque solution, elle renvoie un réel et la meilleure solution est celle qui minimise ou maximise la fonction objectif. Évidemment, un problème d'optimisation combinatoire peut avoir plusieurs solutions.Généralement l'ensemble admissible X d'un problème d'optimisation combinatoire est défini comme un sous-ensemble de l'ensemble des parties d'un ensemble fini d'éléments E i.e $E = \{e_1, ..., e_n\}$ et $X \subseteq\ 2^E$.

Un problème d'optimisation combinatoire peut aussi être formalisé comme un problème d'optimisation en variables binaires. En associant une variable x_i à tout élément ei de E, tout sous-ensemble de E peut être représenté par un vecteur $x = (x_1, ..., x_n)$, avec $x_i = 1$ si ei appartient au sous-ensemble et $x_i = 0$ sinon. Ce qui fait l'élégance des problèmes d'optimisation combinatoire, c'est qu'en plus des formulations en programmes mathématiques, il est souvent possible de les formuler comme des problèmes de la théorie des graphes

1.3.3 Mono-objectif

Les problèmes à objectif unique (mono-objectif) sont définis par une fonction unique

Un problème d'optimisation est à objectif unique lorsqu'un seul objectif (critère) est donné. Dans ce cas, la solution optimale est bien définie, elle a le coût optimal (min, max). Formellement, à chaque instance d'un tel problème est associé un ensemble de solutions potentielles W respectant certaines contraintes et une fonc-

tion objectif W∈ Y associée à chaque solution acceptable s de W une ou plusieurs valeurs. Une instance de résolution d'un problème d'optimisation (W) consiste à trouver la solution optimale $s*$ de W qui optimise (minimise ou maximise) la valeur de la fonction objectif dans le cas de la minimisation : l'objectif est de trouver $S*$ de W tel que

$$(S*) \in \ (S)$$

pour tout élément s de W. Le problème de maximisation peut être défini de manière similaire. [3]

1.3.4 Multi-objectif

Le problème d'optimisation combinatoire multi-objectifs peut être écrit sous la forme d'un problème, où p représente le nombre de cibles X qui est un ensemble combinatoire.

$minz(x) = (z_1(x), ..., z_p(x)) \ / \ x \in \ \mathbb{X}$

Afin de ne pas confondre les solutions réalisables et leurs images dans $\mathbb{R}^{\prime}$, il est d'usage en optimisation multiobjectif de distinguer l'espace des décisions X , qui contient l'ensemble admissible X, et l'espace des objectifs $\mathbb{Z} \subseteq \ \mathbb{R}^{\prime}$, , qui lui contient l'image de l'ensemble admissible X dans $\mathbb{R}^{\prime}$.

À toute solution réalisable x est associé un vecteur critère $z(x) = (z_1(x), ..., z_p(x))$ qui correspond à son image dans $\mathbb{Z}$, telle que z_k est une fonction objectif pour tout k dans $\{1, ., p\}$.

L'ensemble des points correspondant aux images des solutions admissibles dans Z est noté $\mathbb{Z} = z(x) : x \in \ X$ En raison du caractère contradictoire des objectifs, il n'y a généralement pas une solution réalisable qui minimise simultanément les objectifs.L'optimalité dans un contexte multi-objectif repose sur la dominance et l'efficacité au sens de Paretodont nous reprenons le formalisme dans les définitions 1 et 2

Définition 1 Soient $\mathcal{Z}$, $\mathcal{Z}\prime \in \ \mathbb{R}^{\prime}$ deux vecteurs.

$\mathcal{Z}$ domine largement $\mathcal{Z}\prime$, noté $\mathcal{Z} \leq \ \mathcal{Z}\prime, si \ \mathcal{Z}_k \leq \ \mathcal{Z}\prime_k$ pour tout k dans {1,.., p}

$\mathcal{Z}$ domine $\mathcal{Z}\prime$, noté $\mathcal{Z} \leq \ \mathcal{Z}\prime, si \ \mathcal{Z}_k \leq \ \mathcal{Z}\prime_k$ pour tout k dans {1,.., p} et$\mathcal{Z} \neq \ \mathcal{Z}\prime$

$\mathcal{Z}$ domine strictement $\mathcal{Z}\prime$, noté $\mathcal{Z} < \mathcal{Z}\prime, si \ \mathcal{Z}_k < \mathcal{Z}\prime_k$ pour tout k dans {1,.., p}

Définition 2

Un point $z$dans $\mathcal{Z}$ est (faiblement) non-dominé s'il n'existe pas un autre point $z\prime$ dans $\mathcal{Z}$ tel que $z\prime$ domine (strictement) z

Une solution x dont l'image z(x) dans $\mathcal{Z}$ est (faiblement) non-dominé est dite (faiblement) efficace. On notera Z_N et Z_{WN} les ensembles de points non-dominés et faiblement non dominés. Il est facile de voir que $Z_N \subset \mathrm{Z}_{WN}$. Les ensembles des solutions efficaces et faiblement-efficaces seront notés respectivement X_E et X_{WE}. [4]

1.4 Les compléxites des problèmes d'optimisation

Les problèmes peuvent être classé selon les propriétés de l'ensemble des solutions :

1. Un problème de décision : C'est un problème dont la réponse est tout simplement **oui** ou **non**.

2. Un problème polynomial réductible : On a L1 et L2 deux problèmes de décision, on dit L1 est polynomial réductible a L2 s'il existe un algorithme polynomial qui convertit chaque instance d'entrée de L1 a une autre instance de L2.

Un problème de la classe P : Un problème est dit polynomial s'il existe un algorithme de complexité polynomiale qui permet de répondre à la question posée par ce problème, quelles que soient les données du problème. La classe P est l'ensemble de tous les problèmes de reconnaissances polynomiaux.

Un problème de la classe NP : Un problème est de classe NP si, pour toute instance du problème, une solution suggérée ou supposée peut être vérifiée en temps polynomial par rapport à la taille de l'instance pour confirmer l'existence d'une instance pour le problème.

Un problème de la classe NP-hard : On dit qu'un problème appartient à la classe NP-difficile si tous les autres problèmes de NP sont polynomialement réductibles dans cette dernière.

Un problème de la classe NP-complet : S'il appartient à NP et que tous les autres problèmes de NP sont polynomialement réductibles dans ce dernier. Il s'agit d'un problème de décision et NP-difficile à la recherche d'une solution optimale.

1.5 Exemples des Problèmes d'optimisation

On qualifie de problème combinatoire, un problème dont la résolution fait face à un grand nombre de combinaisons à découvrir.

1.5.1 Problème de sac à dos(knapsack)

1.5.1.1 Introdcution

Supposons que nous planifions une journée à la plage. Nous devons alors remplir nos sacs à dos avec des éléments nécessaires pour le voyage. Étant donné que la capacité du sac à dos est limitée (un poids ou un volume maximum) nous devons prendre une décision pour pouvoir le remplir Chaque type d'élément est caractérisé par son poids(ou un volume) et sa valeur (niveau d'importance)

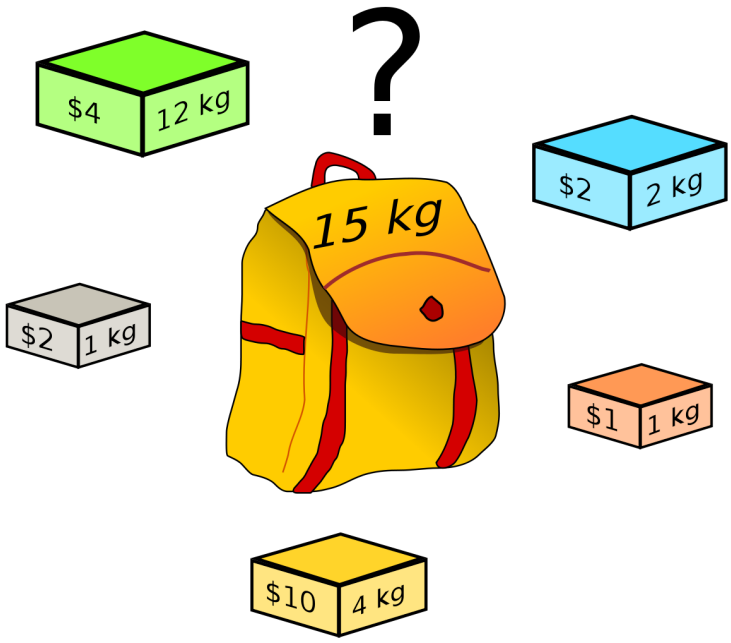


FIGURE 1.1 – Problème de sac-à-dos

1.5.1.2 Formalisme

Nous avons un sac à dos dont le contenu ne doit pas dépasser une capacité c. Considérons un ensemble de n objets, numérotés avec un indice de $1n$, chacun pondéré avec une valeur de p_i.(profit en anglais). Le problème du sac à dos consiste

à trouver le sous-ensemble chargé dans le sac de capacité c pour maximiser le profit total

1.5.1.3 Instances

Il existe de nombreux cas de problème de sac à dos : problème de sac à dos avec variables binaires, problème de sac à dos avec variables entières, problème de sac à dos multidimensionnel, problème de sac à dos multi-objectifs, etc. le problème du sac à dos permet de modéliser de nombreux problèmes tels que : problème de gestion ,de la charge , etc.[5]

1.5.1.3.1 Problème de sac à dos à variables binaires Cet exemple, également connu sous le nom de sac à dos fractionnaire, utilise un programme linéaire binaire pour modéliser le problème du sac à dos. Cette modélisation se présente comme suit :

1. Variables de décision : Les variables sont définies sur l'ensemble 0,1 : La variable xi du i-ème élément,$xi = 1$ si l'élément i est mis dans le sac,$xi = 0$ s'il est laissé de côté.

2. Contraintes : Le problème sac-à-dos est limité par deux contraintes. En plus de la contrainte d'intégrité $x_i \in \ 0, 1$, le sac-à-dos ne doit pas dépasser sa capacité c (volume ou poids) $\sum_{i=1}^{n} w_i x_i \leq$ c

3. Critère :La fonction objectif du sac-à-dos est définie comme la maximisation du profit des objets placés dans le sac :

$$f(x) = \sum_{i=1}^{n} p_i x_i$$

où p_i est le profit de l'objet i.

Résoudre le problème du sac à dos revient à trouver le vecteur solution $\bar{x} =$

$$(\bar{x_1}, .., \bar{x_n})$$

qui **maximise** la fonction objectif définie par l'équation

Mathématiquement, un problème de sac à dos est représenté comme suit :

Soient n nombre d'objets pi profit de l'objet i, wiN poids de l'objet $ietx = (x1, .., xn)$ vecteur de décision.

Maximiser

$$f(x) = \sum_{i=1}^{n} p_i x_i$$

S.c.

$$x_i \in \{0, 1\}, i \in \{1, .., n\}$$

$$\sum_{i=1}^{n} p_i x_i \leq c$$

1.5.2 Problème de voyageur de commerce (TSP : Travelling Salesman Problem)

1.5.2.1 Introduction

Un agent commercial doit voyager entre une liste de villes, en visitant chaque ville une fois et une seule. L'itinéraire le plus efficace entre les villes peut être calculé en termes de distance, de temps ou de consommation de carburant.

L'agent doit se conformer à certaines restrictions lorsqu'il est sur la route. On connaît la distance entre chaque paire de villes. Chaque ville ne doit être visitée qu'une seule fois. [6]

FIGURE 1.2 – – Exemple d'itinéraire d'un voyageur de commerce entre les villes : Annaba - Constantine - Souk Ahras et El Kala

1.5.2.2 Formalisme

Supposons que nous voulions déterminer l'itinéraire optimal entre un ensemble donné de villes. La distance entre deux villes est spécifiée par la fonction de coût c.

Nous pouvons représenter un graphe complet à n sommets par G=(V, E, c), où V est l'ensemble des sommets (c'est-à-dire les villes), E est l'ensemble des arêtes et c est une fonction de coût qui représente la distance entre les villes. Le problème du voyageur de commerce (TSP) consiste à trouver le plus court chemin cyclique qui visite chaque ville une et une seule fois. Un tel chemin est appelé un hamiltonien (ou, dans certains textes, une tournée). Il y a n ! chemins possibles pour un ensemble de n villes. La combinaison est la clé de la difficulté de ce problème car le nombre de chemins potentiels augmente avec le nombre de villes à visiter.

1.5.2.3 Instances

Un graphe dirigé ou non dirigé est utilisé pour représenter le problème du voyageur de commerce. Si les distances entre les villes ne sont pas égales à l'aller et au retour ($c_j \neq c_i$), le problème est qualifié d'asymétrique. Un problème de voyageur de commerce avec un chemin asymétrique est un problème dans lequel le chemin n'est pas symétrique. Un problème de voyageur de commerce symétrique est un problème dans lequel le chemin n'est pas dirigé (c'est-à-dire $c_{ij} = c_{ji}$). la figure ci dessus presente le graphe complet non orienté de l'exemple de la figure 1.1

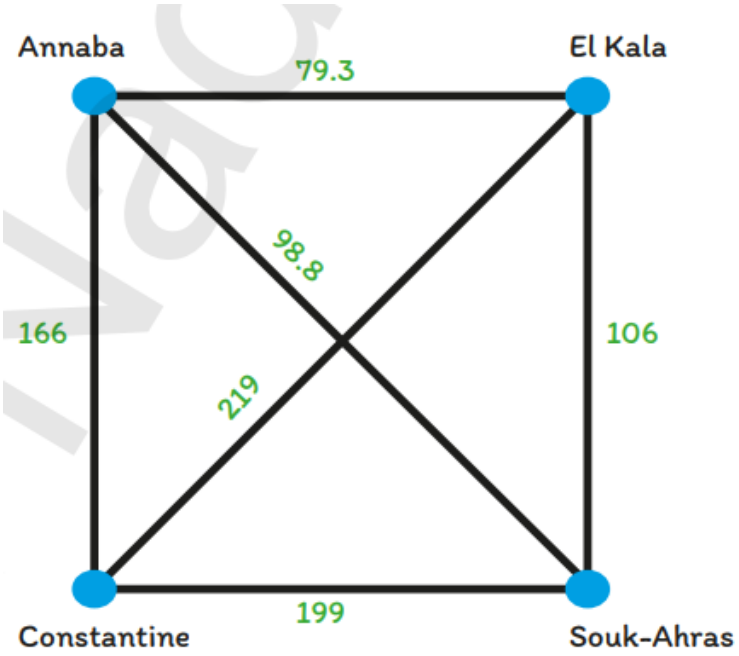


FIGURE 1.3 – Graphe complet non-orienté valué

1.Un graphe est complet si chaque sommet du graphe est relié directement à tous les autres sommets.

2. Un cycle (circuit) dans un graphe est un chemin fermé.

1.5.2.4 Instance Problème de voyage de commerce symétrique

Un programme linéaire binaire ou entier peut être utilisé pour modéliser le problème symétrique du voyageur de commerce. La modélisation binaire se présente comme suit :

1. Variables de décision : "Les variables sont définies sur l'ensemble 0, 1 : La variable x_{ij} correspond à la présence de l'arc i_j dans le cycle hamiltonien du voyageur".

On note $x_{ij} = 1$ si le voyageur prend le chemin existant entre les villes i et j, sinon $x_{ij} = 0$.

2. Contraintes : Le problème du voyageur de commerce est limité par un certain nombre de contraintes :

(a) Contrainte d'intégrité : x_{ij} ϵ 0, 1

(b) Le voyageur doit entrer une seule fois dans une ville :

$$\sum_{i=1}^{n} x_{ij} = 1$$

(c) Le voyageur doit sortir une seule fois d'une ville :

$$\sum_{j=1}^{n} x_{ij} = 1$$

3. Critère : La fonction objectif du problème du voyageur de commerce est définie comme la longueur minimale du trajet oùc_{ij} est la distance entre les villes

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

La résolution du problème du voyageur de commerce revient à une matrice qui minimise la fonction objectif définie dans l'équation :

$$\bar{x} = \begin{pmatrix} \bar{x_{1,1}} & \cdots & \bar{x_{1,n}} \\ \vdots & \ddots & \vdots \\ \bar{x_{n,1}} & \cdots & \bar{x_{n,n}} \end{pmatrix}$$

Un problème de voyageur (TSP) est Mathématiquement représenté comme suit :

Soient n nombre de villes, $c_{ij} \in *$ distance entre les villes i et j

$$x = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{pmatrix}$$

Minimiser

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

s.c.

$$x_i \in \{0, 1\} \qquad i \in \{1, .., n\}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad i \in 1, .., n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i \in 1, .., n$$

1.5.3 Problème d'ordonnancement de tâches

1.5.3.1 Introduction

La production en atelier est la création par l'utilisation et la transformation des ressources.Pour un fonctionnement normal de ce système, plusieurs processus sont effectués en tenant compte du temps et du coût de production.

En général, le problème d'ordonnancement est lié aux points suivants :

- Un ensemble de tâches ou de Jobs qui doivent être effectuées.

- Un ensemble de ressources ou machines à utiliser par ces jobs.

- Un programme à identifier, pour allouer les ressources aux taches.

Les problèmes d'ordonnancement sont des problèmes d'optimisation définis en terme de tâches et ressources [7]

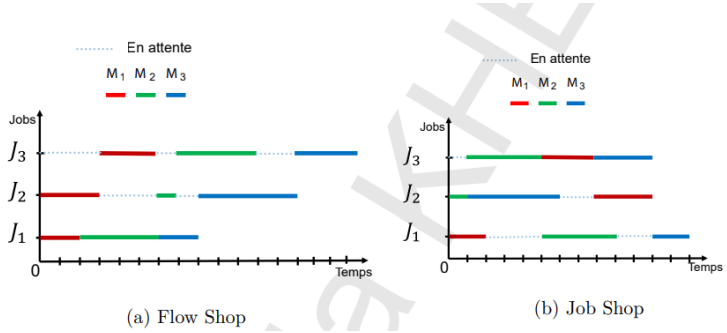
1.5.3.2 Formalisme

Une tâche (job) i est l'unité de travail de base à effectuer commençant à une date t_i et terminant à une date c_i .Elle utilise les ressources pendant un temps $p_i = c_i t_i$ (processing time). Un job peut être interrompu, c'est-à-dire il est composé de plusieurs opérations. Dans le cas où le travail ne peut être interrompu, il est considéré comme une tache unique à effectuer

Une ressource est tout ce qu'on utilise pour effectuer un job.Les ressources sont limitées en temps et en quantités. Elles peuvent être des machines, ouvriers, équipements, etc.

Soit $J = \{1, .., n\}$ l'ensemble des jobs et $M = \{1, .., m\}$ l'ensemble des machines. Chaque tâche $j \in \mathbb{J}$ est composée d'au plus m opérations d'ordre $O_{j1}, .., O_{jm}$. Chaque opération doit être effectuée sur une machine $k \in \mathbb{M}$ particulière dans un temps précis .

Un ordre consiste à allouer des opérations à des intervalles sur toutes les machines. Un problème de planification de travaux implique de trouver un moyen réalisable de minimiser Cmax le temps nécessaire pour terminer un travail. Une représentation possible du problème de planification des tâches est un diagramme de Gantt. Chaque opération est associée à une barre transversale dont la longueur est proportionnelle au temps de l'opération



(a) Flow Shop (b) Job Shop

FIGURE 1.4 – Représentation de Gantt d'un ordonnancement de 3 tâches sur 3 machines, cas du Flow shop et Job shop

1.5.3.3 Instances

Il existe plusieurs cas de problèmes d'ordonnancement en fonction du nombre de machines et de la nature du travail. On trouve les ordonnancements sur une machine unique ou sur des machines parallèles. Il existe aussi un ordonnancement linéaire ou multiple

- Machine unique : Toutes les tâches sont effectuées par une seule machine. Nous recherchons l'ordre des tâches sur cette machine

- Machines parallèles : ensemble de tâches exécutées par des machines identiques. Les tâches ne sont pas toutes effectuées sur des machines

-Atelier à parcours unique (Flow Shop) : : c'est-à-dire que le processus est linéaire. Les travaux sont divisés en tâches qui doivent s'exécuter sur toutes les machines.

- Atelier à parcours multiples (Job Shop) : Ce processus n'est pas linéaire.(Figure 1.4a). Le travail est décomposé en travaux qui doivent s'exécuter sur toutes les machines

1.5.4 Problème des découpes

Les problèmes de découpe ont été largement étudiés et ont produit un grand nombre de résultats. Les raisons a cela sont multiples. Tout d'abord, l'industrie contribue à toutes ces questions importantes, conduisant à une recherche permanente sur ces sujets connus depuis de nombreuses années. [8]

Il existe de nombreux types de ces problèmes, créant différentes approches et solutions de modélisation qui peuvent être liés à différents domaines, voici quelques exemples :

^ Découpe réaliste : Le but est d'utiliser le minimum de matière : barre, surface pour créer un certain ensemble d'éléments.

^ Placement des tâches dans le temps : il s'agit de créer un planning possible,par exemple, l'organisation de la semaine d'un artisan. Les taches ont des valeurs et des durées différentes.

^ L'utilisation des surfaces : des palettes de tailles géométriques différentes sont posées sur une aire.Le but est alors de limiter les surfaces inutilisées. Les caracteristiques de ces problèmes sont resumées dans la figure 1.5

Type de problème	Dim-ensior	Critères	Domaines d'application
Découpes proprement dite	$\Re 2$ $\Re 1$	*Minimiser* . - les chutes - la matière utilisée	Découpes de pièces dans - l'aéronavale - la confection - l'ébénisterie - la maroquinerie - la papeterie - la sidérurgie - la verrerie...
Placement de tâches dans le temps	$\Re 1$	*Maximiser :* - la rentabilité, *Optimiser :* - l'utilisation des ressources.	Emploi du temps - d'un artisan - d'une école - d'un atelier...
Utilisation de surfaces	$\Re 2$ $\Re 1$	*Minimiser :* - les surfaces inutilisées	- Étalages en grande surface - Machines outils dans un atelier - Puces sur une tranche de silicium - Composants électroniques sur une plaque époxite - Palettes ou étagères dans un entrepôt - Palettes sur une machine automatique...
Remplissage de volumes	$\Re 3$ $\Re 2$ $\Re 1$	*Minimiser :* - l'espace inutilisé - les déplacements pour atteindre les colis	- Objets dans un entrepôt - Caisses dans un wagon, un camion, un cargo - Colis dans une caisse - Pièces dans un four en sidérurgie...
Maximisation du poids transporté	$\Re 1$	*Maximiser :* - le poids embarqué	Remplissage de cargos ou camions par des objets lourds (le critère volume est secondaire)
Remplissage de rayons	$\Re 1$	*Maximiser :* - l'utilisation des rayons	- Grandes surfaces, - Bibliothèques - Entrepôts...
Gestion de mémoire par zones	$\Re 1$	*Minimiser :* - les zones inaccessibles	- Informatique

FIGURE 1.5 – Les types des problèmes de découpe

1.6 Exemples d'algorithmes d'optimisation

Les algorithmes d'optimisation cherchent à déterminer l'ensemble d'entrée d'une fonction qui fournit la valeur minimale ou pour cette fonction. Par exemple, nous chercherons la découpe optimale d'une feuille pour produire le plus grand nombre de boîtes possible.Cette optimisation peut se faire sans contrainte ou sous contrainte. Un algorithme d'optimisation est une procédure mathématique qui permet d'obtenir les minimums(ou maximums) d'une fonction réele f (que l'on appelle fonction objective) $min(x)n \in \mathbb{R}f(x)$ En général la solution est un sous-espace $A \in \ n \in \mathbb{R}$ qui est soumis à un ensemble de contraintes (conditions sur les variables), qui sont exprimées comme un système d'équations et inéquations. Les éléments de A sont appelés solutions admissibles et souvent ont des bornes supérieures et inferieures $x_l \leq \ x \leq \ x_u \in \ \mathbb{A}$

1.6.1 Recherche Locale

Dans les algorithmes, la recherche locale est une méthode générale pour résoudre des problèmes d'optimisation, c'est-à-dire des problèmes qui trouvent la meilleure solution dans un ensemble de solutions. La recherche locale consiste à passer d'une solution à l'autre dans l'espace des solutions candidates jusqu'à trouver une solution considérée comme optimale, ou jusqu'à ce que le temps imparti soit dépassé.[8]

1.6.1.1 Description

1.6.1.1.1 Notion de voisinage

L'algorithme de recherche locale part d'une solution candidate et la déplace itérativement vers une solution voisine. Cette méthode n'est applicable que lorsqu'un concept est défini sur l'espace de recherche. Par exemple, le voisinage de l'arbre recouvrant est un autre arbre qui ne diffère que d'un nœud.

Le même problème peut avoir plusieurs définitions différentes de voisinage ; L'optimisation locale avec des voisinages qui limitent les modifications de la composante k de la solution est souvent appelée k-optimale.

Habituellement, chaque solution candidate a plus d'une solution voisine choisir vers laquelle aller en utilisant uniquement des informations sur les solutions adjacentes à la solution actuelle, d'où le terme recherche locale.

1.6.1.1.2 Critère d'arrêt

Le critère d'arrêt de la recherche locale peut être une limite en durée. Une autre possibilité est de s'arrêter lorsque le meilleur trouvé par l'algorithme ne s'améliore pas pendant un certain nombre d'itérations.

Un algorithme de recherche locale est bien un algorithme optimal où la recherche peut être arrêtée alors que la meilleure solution de l'algorithme n'est pas la meilleure, Cette situation peut se produire même si le point d'arrêt est dû au fait que la solution actuelle ne peut pas être améliorée car la solution peut être trouvée très loin dans le voisinage des solutions passées par l'algorithme.

Les algorithmes de recherche locale sont largement utilisés dans les problèmes d'optimisation difficiles, tels que les problèmes informatiques (en particulier l'intelligence artificielle), mathématiques, de recherche opérationnelle, d'ingénierie et de bio-informatique ,Les problèmes suivants se prêtent bien à l'utilisation de la recherche locale :

1-le problème de couverture de sommets dans lequel une solution est un arbre recouvrant un graphe simple et l'objectif est de trouver la solution avec le nombre de nœuds minimum ;

2-le problème du voyageur de commerce où une solution est un cycle contenant tous les nœuds du graphe et l'objectif est de minimiser la longueur totale du cycle ;

3-le problème SAT où une solution est une association et l'objectif est de maximiser le nombre de clauses vérifiées par l'association ; dans ce cas, la solution finale est celle qui satisfait toutes les clauses.

Beaucoup de problèmes peuvent être formulés de plusieurs manières en termes d'espace de recherche et d'objectif. Par exemple, pour le problème du voyageur de commerce une solution peut être un cycle et le critère à maximiser la combinaison du nombre de nœuds et de la longueur du cycle. Mais une solution peut aussi être un chemin, et le transformer en cycle peut faire partie de l'objectif.

1.6.2 Méta-Heuristiques

Une heuristique est une stratégie conventionnelle pour se déplacer intelligemment dans l'espace des solutions, pour obtenir la meilleure approximation possible, dans un temps donné

Deux types d'heuristiques sont principalement utilisés : les heuristiques qui construisent itérativement une solution et les heuristiques natives, à partir desquelles on cherche une méthode localement optimale.

Des heuristiques plus poussées ont été mises au point et ont données naissance à une nouvelle famille d'algorithmes : les méta-heuristiques.

"Le mot méta heuristique est dérivé de la composition de deux mots grecs : - heuristique qui vient du verbe heuriskein et qui signifie 'trouver'. - méta qui est un suffixe signifiant 'au -delà', 'dans un niveau supérieur'.

La méta-heuristique est définie comme une méthode algorithmique qui permet de guider et d'orienter le processus de recherche d'une solution vers des régions riches en solutions optimales, dans le but de trouver des solutions, qui peuvent être sous-optimales en tout cas très proches de l'optimum, dans un délai raisonnable . Les méta-heuristiques peuvent tirer parti de l'expérience acquise lors de l'optimisation de la recherche, pour mieux guider le reste de la recherche. Le but de la métaheuristique est de réussir à trouver l'optimum global. Pour cela, l'idée est à la fois de parcourir la recherche et d'explorer des domaines qui s'annoncent prometteurs mais sans être " piégé " par un optimum local.

Les méta-heuristiques s'inspirent souvent de processus naturels et sont de plus en plus hybridées avec d'autres méthodes opérationnelles.

Les principales méta-heuristiques utilisées pour résoudre des problèmes discrets sont :

-le recuit simulé : formalisée en 1986 par F. Glover , l'exploration de l'espace de recherche en acceptant de réduire sa solution pour s'affranchir des optimaux locaux.Tout au long du processus, le recuit acceptera de plus en plus ces décompositions, ce qui le fera converger vers l'optimum global.

-La recherche avec tabous : contrairement à l'incubation, est définie et conceptuelle sur la mémoire.Le choix de la meilleure solution voisine d'une solution pousse l'algorithme à trouver des optima locaux ;et lorsque l'exploration de l'espace de recherche est effectuée en limitant le voisinage de la solution en montrant "" certains mouvements, l'algorithme devrait théoriquement atteindre l'optimum global.

-les algorithmes évolutionnaires : dérivés de la théorie de l'évolution de Darwin, manipulent plusieurs solutions à la fois et en les combinant forment de nouvelles solutions. [8]

1.6.3 Programmation linéaire

En mathématiques, les problèmes de programmation linéaire sont des problèmes d'optimisation où la fonction objectif et les contraintes sont linéaires.

La programmation linéaire traite également de la résolution des problèmes PL.

La programmation linéaire est un domaine central de l'optimisation, car les problèmes de PL sont les problèmes d'optimisation les plus faciles - toutes les contraintes y étant linéaires. Beaucoup de problèmes réels de recherche opérationnelle peuvent être exprimés comme un problème de PL. Pour cette raison un grand nombre d'algorithmes pour la résolution d'autres problèmes d'optimisation sont fondés sur la résolution de problèmes linéaires. La programmation linéaire est essentiellement appliquée pour résoudr des

problèmes d'optimisation à moyen et long terme (problèmes stratégiques et tactiques, dans le vocabulaire de la recherche opérationnelle). Les domaines d'application de ces problèmes sont très nombreux aussi bien dans la nature des problèmes abordés (planification et contrôle de la production, distribution dans des réseaux) que dans les secteurs d'industrie : industrie manufacturière, énergie (pétrole, gaz, électricité, nucléaire), transports (aériens, routiers et ferroviaires), télécommunications, industrie forestière, finance.

En mathématiques, les problèmes de programmation linéaire (PL) sont des problèmes d'optimisation où la fonction objectif et les contraintes sont toutes linéaires. Néanmoins, la plupart des résultats présentés ici sont également vrais si l'objectif est une fonction monotone croissante de chaque variable considérée. La programmation linéaire désigne également la manière de résoudre les problèmes de PL.[6]

Le terme programmation linéaire suppose que les solutions à trouver doivent être représentées en variables réelles. S'il est nécessaire d'utiliser des variables discrètes dans la modélisation du problème, on parle alors de programmation linéaire en nombres entiers (PLNE). Il est important de savoir que ces derniers sont nettement plus difficiles à résoudre que les PL à variables continues.

1.6.4 Programmation entiere

La programmation linéaire entière (ILP) ou programmation linéaire entière (ILP) est un domaine des mathématiques et de l'informatique théorique dans lequel on considère des problèmes d'optimisation d'une forme spécifique. Ces problèmes sont décrits par des fonctions de coût et des contraintes linéaires et des variables entières.

Des contraintes d'intégrité sur les variables (qui distinguent l'OLNE de l'optimisation linéaire classique) sont nécessaires pour modéliser certains problèmes, notamment algorithmiques. Mais cette contrainte supplémentaire complique le problème et nécessite des techniques particulières.

Un problème d'optimisation est un problème mathématique où, étant donné un ensemble de variables et de contraintes sur ces variables, on doit trouver une affectation qui maximise (ou minimise) une fonction de coût. Lorsque les contraintes et les fonctions de coût sont des combinaisons linéaires de variables, on parle de problèmes linéaires, et si ces variables ne sont autorisées à prendre des valeurs que dans l'ensemble des entiers, alors le problème est des entiers.

Les contraintes qui obligent les variables à prendre des valeurs entières sont appelées contraintes d'intégrité. Lorsque l'on supprime cette contrainte, on parle d'un problème de relaxation relaxée ou continue, puis on a affaire à un problème d'optimisation linéaire.[9]

De nombreux problèmes de recherche opérationnelle et d'algorithmes peuvent être transformés en problèmes OLNE. Par exemple, le problème de couverture d'ensemble est le suivant. Étant donné un ensemble A, on dit que l'élément e est couvert par A si e appartient à A ; pour un ensemble U et une famille S de sous-ensembles de U, le problème est de recouvrir tous les éléments U avec la plus petite sous-famille possible de S Ce problème se traduit naturellement sous la forme suivante :

minimiser :$\sum_{S \in \mathcal{S}} x_S$ Minimiser le nombre de sous-ensembles)

tel que :$\sum_{S:\ e \in S} x_S 1 \forall e \in \mathcal{U}$ (Tous les éléments sont couverts)

$x_S \in \{0,1\} \forall S \in \mathcal{S}$ (Chaque sous-ensemble est, ou bien dans la couverture, ou bien pas)

1.7 Conclusion

Nous avons présenté dans ce chapitre quelques exemples les plus connues des problèmes d'optimisation . Nous avons essayé de donner une image globale sur le monde d'optimisation , une introduction, un bref historique et une description de certains problèmes d'optimisation et des Algorithmes d'optimisation avec des exemples

Chapitre 2

Problème de découpes

2.1 Introduction

Nous commençons dans ce chapitre, par la présentation de quelques définitions préliminaires et propriétés sur lesquelles se basent les modélisations et résolutions des problémes de découpe non contraint, contraint et double contraint à deux dimensions. Ensuite, nous faisons une interprétation du probléme de sac à dos à une et à deux dimensions

2.1.1 Definition

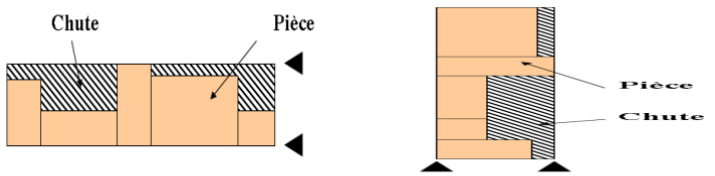
Une bonne gestion des processus industriels entraîne des problèmes d'optimisation. Certains d'entre eux sont NP-hard et ont besoin d'algorithmes spéciaux à résoudre. L'un de ces problèmes est le problème du stock de découpe (CSP). Le découpage précis et rapide avec moins de déchets possibles est un élément très important du processus de travail. L'objectif est de découper les articles 2D du stock rectangulaire, en minimisant les déchets. Le problème est très difficile et la plupart des auteurs résolvent la version simplifiée du problème lorsque les éléments sont rectangulaires. Le temps de calcul augmente exponentiellement lorsque le nombre d'articles augmente. Trouver la solution optimale pour les problèmes de grande taille dans un délai raisonnable est impossible. Par conséquent, les algorithmes exacts et les méthodes numériques traditionnelles ne peuvent être appliqués qu'à de très petits problèmes, moins de 100 éléments.[9]

"Définissons tout d'abord les problèmes de découpe : comment, dans un « matériau » de dimensions données, découper (ou disposer) un nombre maximal d'éléments de dimensions plus petites ? Ou encore, comment découper (ou disposer) un nombre donné d'éléments de manière à utiliser une quantité minimale du « matériau » de base ?"

Un arrangement ou un placement optimal de certains éléments connus dans une hiérarchie de domaines, avec certains autres éléments donnés, est l'objectif d'un problème d'optimisation combinatoire appelé problème de découpage. L'arrangement optimise la fonction objectif, qui est la fonction étudiée, compte tenu des contraintes caractérisant le problème. Il est connu sous le nom de problème de Bin Packing (Bin = boîte, à Pack = emballeur). [8]

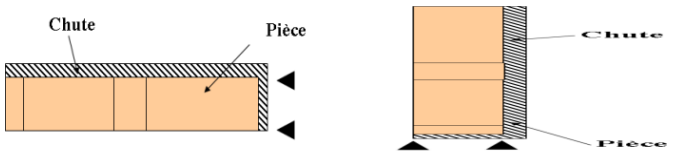
comme mentionné précédemment ,Pour l'instant, nous supposons que nous avons un $n$rectangle sur lequel nous voulons effectuer une analyse pour produire un ensemble de fragments appartenant à S.

Dans cette section, nous présentons une vue générale des motifs de bandes et découpes



(a). Bande générale horizontale. (b). Bande générale verticale.

FIGURE 2.1 – Bandes générales horizontale et verticale



(a). Bande uniforme horizontale. (b). Bande uniforme verticale.

FIGURE 2.2 – Bandes générales horizontale et verticale

2.1.1.1 Les principaux modéles de bandes

1. Une bande générale horizontale (figure 1.(a)) (resp. verticale (figure 1.(b)) est un ensemble de fragments rectangulaires adjacents tel qu'il existe une ligne horizontale avec les propriétés suivantes :

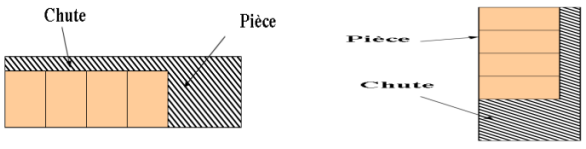
-elle touche toutes les piéces sur leur longueur .

-l'un des demi-plans définis par celle-ci contient toutes les pièces.

2. Une bande uniforme est une bande pour laquelle existe exactement deux lignes horizontales(resp. verticales)possèdent les propriétés ci-dessus.Dans ce cas, tous les fragments de même hauteurr (resp. longueur) (figure 1.1.(a) et 1.1.(b)).

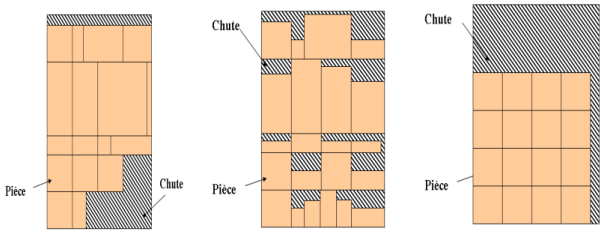
3. Une bande homogéne horizontale (resp. verticale)est une bande uniforme où il n'y a qu'un type de pièces participant (figure 1.2.(a) et 1.2.(b)).

4. Une bande optimale de longueur l et de hauteur w (resp. de hauteur w et de longueur l) est la bande commune occupant la zone de bande maximale (l, w) (resp. (w, l)).



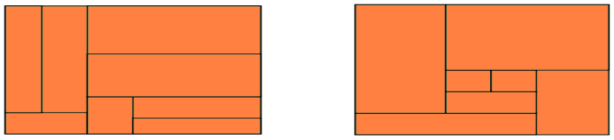
(a). Bande homogène horizontale. (b). Bande homogène verticale.

FIGURE 2.3 – Bandes homogénes horizontale et verticale.



(a). Uniforme. (b). Général. (c). Homogène.

FIGURE 2.4 – Modéles de découpe uniforme, générale et homogéne..



(a). Modèle de découpe guillotine. (b). Modèle de découpe non guillotine.

FIGURE 2.5 – Modéles de découpe guillotine et non guillotine.

2.1.1.2 Les principaux modéles de découpes

1. Un modéle de découpe uniforme (ou dissection uniforme)est un motif formé par une combinaison de bandes verticales et horizontales.(figure 1.3.(a)).

2. Un modéle de découpe générale (ou dissection générale) est un motif composé d'une combinaison de bandes verticales et horizontales. (figure 1.3.(b)).

3. Un modéle de découpe homogéne est une analyse unifiée dont la solution est due à la répétition d'un seul type de fragment de S (figure 1.3.(c)).

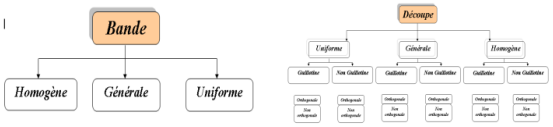
2.1.1.3 Types de problémes de découpe

En pratique, les utilisateurs sont souvent contraints par la découpe disponible, la manière de procéder avec le plan de découpe est différente. [10]

Les outils couramment utilisés sont :

1. La découpe du type guillotine : réaliser la production d'un ensemble de sous-entités au sein de l'entité préexistante.Si nous supposons que l'entité est une plaque rectangulaire, le découpage effectué par la décomposition horizontale à son côté opposé parallèle aux deux autres (figure 1.4)

2. La découpe du type non-guillotine : En général cette découpe offre une meilleure solution que la découpe guillotine.



(a). Modèles de bandes. (b). Modèle de découpe.

FIGURE 2.6 – – Classification des modèles de découpe et de bandes

En effet, ce détourage reprend le même usage que dans le style détourage et de plus il se fait en marquant les arréts avant d'atteindre le côté opposé du rectangle à recadrer

3. La découpe non-orthogonale : cette découpe ne prend pas en considération l'orientation des piéces.Cette fois, les pièces peuvent être tournées et translatées(on effectue des rotations sur les piéces, donc elles ne sont pas fixées).

2.2 Problème de découpes

La question de la réduction de la quantité de matériaux utilisés dans un processus de production a fait l'objet de nombreuses études et recherches. De nombreuses solutions et résultats ont résulté de ces recherches, fournissant une grande quantité de solutions et de résultats. Les enjeux de la réduction de la quantité de matière utilisée dans les processus de production sont importants pour plusieurs raisons. Tout d'abord, les enjeux industriels sont considérables, il y a donc un programme de recherche permanent sur des questions connues depuis longtemps. Tous les problèmes de découpe sont des problèmes d'optimisation, dans lesquels les composants sont liés géométriquement les uns aux autres. Par conséquent, il n'y a pas de connexion explicite entre les composants.Tous les composants partagent le même ensemble d'attributs. Étant donné que l'objectif et les contraintes sont toujours décrits comme des fonctions mathématiques et sont définis globalement, tous les attributs des composants sont concernés. L'objectif de chaque problème est compacite : soit le nombre de composants à placer à l'intérieur du paquet doit être maximisé, soit le nombre d'articles qui tombent doit être minimisé. [11]

Les problèmes de découpe représentent une application classique de la programmation en nombres entiers (PNE). Ils impliquent la découpe d'objets volumineux, tels que des feuilles, des rouleaux ou des planches, en objets de plus petite taille pour répondre à une demande.

Nous présentons dans cette section les différentes variantes du découpe ainsi qu'un aperçu plus général des autres qui sont encore difficiles à résoudre, dans le sens de la recherche et du développement d'algorithmes précis de resolution :

2.2.1 Probléme de découpe non contraint

2.2.1.1 Probléme non pondéré

Une instance du probléme de découpe non contraint non pondéré à deux dimensions est définie par le couple

$$(R, S).R = (L, W)$$

est l'entité disponible représentée par le rectangle initial de longueur L et de hauteur W.

L'ensemble des sous-entités correspond à un ensemble de piéces rectangulaires représenté par$S$ tel que

$$|S| = n$$

.Chaque type de piéces est représenté par des petites dimensions

$$(l_i, w_i)$$

,pour

$$i = 1, ..., n,$$

où l_i (resp w_i) désigne la longueur (resp. hauteur) de la piéce i. Soient P l'ensemble fini des vecteurs représentant les différents plans de découpe réalisables,$\xi = (\xi_1, ..., \xi_n)$ avec ξ_i le nombre de répétitions de la iéme piéce dans le plan ξ et F une application [**?**]

définie par :

$F : P \to N, telque, F(_i) = L_W - \sum_{i=0}^{n} \pi_i \xi_i$

où π désigne la surface de la piéce $i = 1, ..., n$ (la surface de la piéce i est donnée par $i = l_i w_i$. Une solution du probléme consiste à trouver le couple (ξ, F) tel que $\xi \subset P$ Une solution optimale du probléme est le couple $(\xi, F(\xi))$ tel que le profit du plan de découpe réalisable ξ

soit égale à F de valeur minimum (i.e. un plan de découpe optimal qui minimise la chute sur le rectangle initial). [7]

2.2.1.2 Probléme pondéré

Une instance du probléme de découpe pondérée non contraint à deux dimensions est définie par le triplet (R, S, c). R et S ont la meme signification que dans le cas du probléme non pondéré. En revanche, ce probléme différe du précédent par l'introduction du vecteur profit$c = (c_i), i = 1, ..., n$, où à chaque piéce i on fait correspondre un profit c_i

$$(ci \neq l_i w_i)$$

, pour$i = 1, ..., n$. L'application précédente peut s'écrire cette fois sous la forme suivante :

F : P $\to N, telque,$

$$F(\xi) = \sum_{i=0}^{n} c_i \xi_i$$

Dans ce cas, le but est de trouver le couple (ξ, F) qui maximise la fonction d'utilité F.

2.2.2 Probléme de découpe pondéré contraint

Une instance du probléme de découpe contraint est définie par le quadruplet

$$(R, S, c, b)$$

R représente le rectangle initial, S l'ensemble des piéces, c le vecteur profit associé aux piéces et$b =$ $, i = 1, ..., n$ une borne supérieure pour le nombre d'occurrences de chaque type de piéces dans un plan de découpe réalisable pour plus de clarté Évidemment, ce problème peut être formulé par le programme suivant :

$$MaxF = \sum_{i=0}^{n} c_i \xi_i$$

$$\xi_i \leq b_i; \xi_i \in \mathbb{N} pour i = 1, ..n, xi_i \in \mathbb{P}$$

où ξ i est le nombre d'occurrences de la piéce du type i dans le plan de découpe réalisable$\xi ßP$ contraint par la borne supérieure b_i. De la méme maniére, une solution optimale pour ce probléme est représentée par le couple ($\xi*$, F), $telque_i * ß$ donne le meilleur plan de découpe de valeur maximum F sous les contraintes ξ_i b$_i$, pour $i = 1, ..., n$

2.2.3 Probléme de découpe pondéré double contrain

On definit une instance du probléme de découpe double contraint par (R, S, c, a, b). R représente le rectangle initial, S l'ensemble des piéces, c le vecteur profit associé aux piéces,$a = (a_i)i = 1, ..., n$ et $b = (b_i)i = 1, ..., n$ représentent respectivement une borne inférieure et une supérieure pour le nombre d'occurrences de chaque type de piéce dans un plan de découpe réalisable.

De même, ce problème peut être construit par le programme suivant :

$$MaxF = \sum_{i=0}^{n} c_i \xi_i \quad ; \xi_i \leq b_i$$

$$\xi_i \geq a_i; \xi_i \in \mathbb{N} \quad pour i = 1, ..n, xi_i \in \mathbb{P}$$

Une solution optimale pour ce probléme est représentée par le couple (ξ, F), tel que ξ donne le meilleur plan de découpe de valeur maximum F sous les contraintes $\xi_i b_i$ et $\xi_{ii} \geq a_i$ pour $i = 1, ..., n$

Lorsque la contrainte $a_i = 0, i = 1, .., n$, le probléme est équivalent au probléme contraint et que lorsque $a_i = b_i, i = 1, n$ le probléme devient le probléme de découpe avec satisfaction exacte des demandes.

2.2.4 Classification

La typologie créée par Dychoff[12] aide les différents domaines de recherche et disciplines à partager plus efficacement les informations. En général, les problèmes d'emballage sont traités en informatique combinatoire et en géométrie. Dyckhoff et Finke ont développé un système de classification des problèmes de découpage et d'arrangement qui comprend quatre catégories Ces catégories sont les suivantes :

1. le nombre de dimension de problème (1D, 2D,. . ., N Dimension)

2. Type de tâche : tous les objets et une sélection de bins, ou bien ; une sélection d'objet et plusieurs bins (conteneur).

3. Caractéristique des bins : plusieurs bins, des bins de taille identique, des bins de taille différente, un seul bins . . .

4. Caractéristique des objets : objet (pièces) de forme identique, peu d'objet de forme identique,objet de forme différente, objet de forme relativement différente. [13]

Recemment une nouvelle typologie a été proposée par Wäscher [14] et al [15] dans l'interet d'inclure les problématiques de découpe , de rangement, et d'établir une catégorisation complète de tous les problèmes connus dans le domaine en ce qui concerne le problème de bin packing en deux dimensions (BPP-2D), qui est d'ailleur un problème trés similaire au problème de découpe en 2D (CP-2D)

les deux spécificités les plus rencontrées sont les suivantes :

-l'orientation : les objets peuvent être à orientation fixe (le cas orienté) ou bien ils peuvent être tournés de 90 degrés (le cas non orienté). -la contrainte guillotine : Si elle est imposée, les objets rangés peuvent être restitués par des coupes bout à bout parallèles aux dimensions de bins.

2.2.4.1 Uni-dimensionnel, 1D

Les objets sont caractérisés par une seule mesure la hauteur, la largeur ou le poids.

Exemple : le rangement de fichiers sur un support informatique, la découpe de câbles, le remplissage de camions ou de containers avec comme **seule** contrainte le poids **où** le volume des articles.

Le problème de l'emballage des bacs Uni-dimensionnels consiste à minimiser le nombre de bacs Uni-dimensionnels nécessaires pour stocker une liste caractérisée par leur longueur. Bien evidement qu'il est un NP-complet, Une instance de BPP-1D, notée (I, w, W) ,comprend un ensemble $I = 1, 2, \ldots, n$ de n objets et une fonction W qui associe à chaque objet i une valeur w_i qui correspond à sa longueur, etW une valeur positive représentant la taille du bin. [16]

Exemple : Placez un ensemble de boîtes de divers longueurs dans le moins de bacs de longueur fixe possible n.

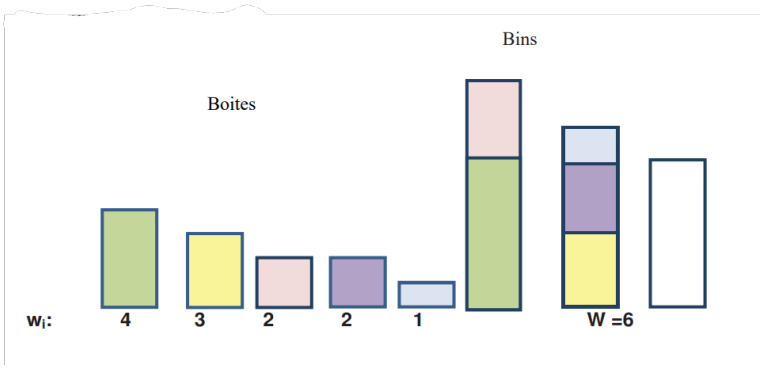


FIGURE 2.7 – – Illustration de l'exemple CSP 1D

Données N : ensemble de n boîtes avec wi longueur de la boîte $S(i \in \mathbb{N})$ et W longueur des bins

Objectif : minimiser le nombre de bins utilisés **Contraintes :** ne pas dépasser la capacité d'un bin.

2.2.4.2 Bi-dimensionnel, 2D

Les objets sont stockés sur une surface limitée comme couper des matières premières, placer des boîtes sur des palettes ou les placer dans un journal.

" Le problème de CSP bi-dimensionnel est une généralisation naturelle de CSP-1D.Il s'agit de minimiser le nombre de grands rectangles pour ranger une liste d'objets rectangulaires. Les objets doivent être rangés de telle manière que les côtés des rectangles soient parallèles à ceux des bacs.. On note (I ,w ,h ,W,H) une instance de CSP-2D"

I=1,2,...,n est la liste des objets à stocker, et une fonction w associe à chaque objet i une valeur correspondant à sa longueur, W et H étant la longueur et la largeur du bac.

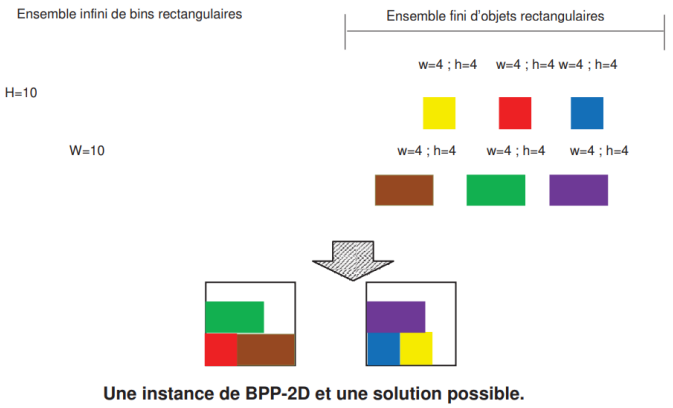
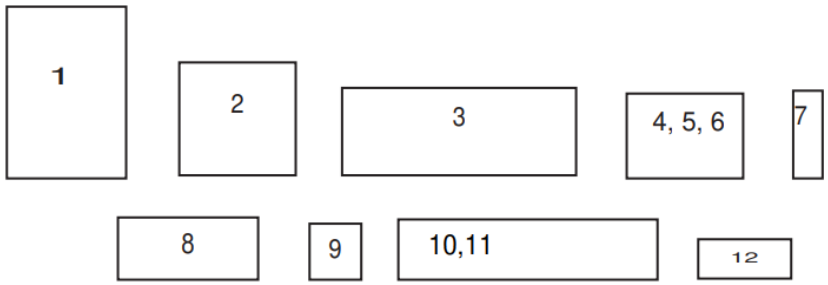


FIGURE 2.8 – – illustration de l'exemple CSP 2D

La différence entre CSP-1D et CSP-2D est que, étant donné un ensemble d'objets et de bacs, détermine s'il y a une place pour ces objets dans le groupe. Le problème est petit en ce sens qu'il est unidirectionnel et bidirectionnel

Deux types de problèmes imbriqués sont considérés : un problème d'affectation et un certain nombre de problèmes de faisabilité. Question sur l'orientation de l'objet est posée à CSP-2D Exemple : considérons l'instance CSP-2D suivante :

I= (A,B) = A= a1=(4,8), a2= (5,4), a3=(3,10), a4= (3,5), a5=(3,5), a6=(5,3), a7=(3,1),a8 = (7, 2), a9 = (2, 2), a10 = (11, 2), a11 = (2, 11), a12 = (3, 1), , B = (12,10).



Solution avec orientation

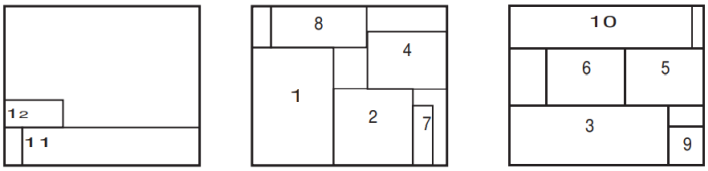


FIGURE 2.9 – CSP–Le cas orienté.

Solution sans orientation

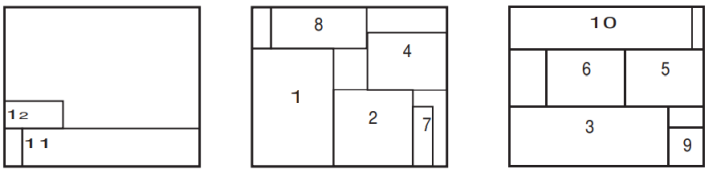


FIGURE 2.10 – CSP-Le cas non orienté.

2.2.4.3 Tri-dimensionnel, 3D

où les objets sont stockés dans un espace fixe, comme le stockage d'objets dans un entrepôt, etc. Le problème tridimensionnel de découpe est aussi connu sous le nom de problème de chargement de conteneurs, qui est une combinaison d'où découlent de nombreuses applications industrielles

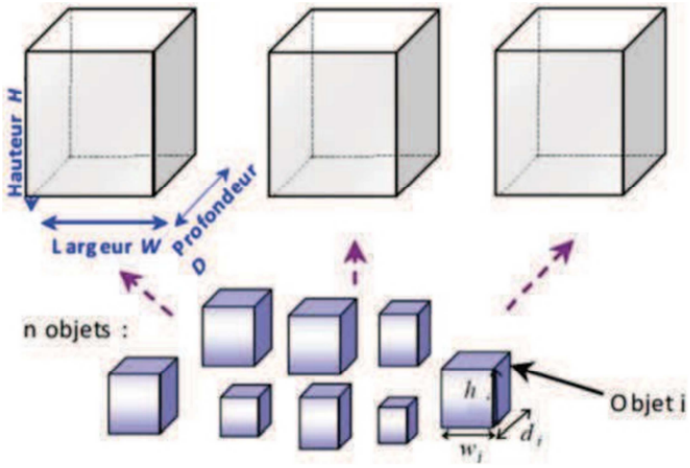


FIGURE 2.11 – Illustration graphique du CSP 3D.

Ce problème consiste à placer un ensemble S d'objets de taille,$i = 1\ldots, n$ dans un nombre minimum de conteneurs identiques $R_j, j = 1, \ldots, m$, dont la taille W est la largeur, H est la hauteur et D est la profondeur en respectant certaines contraintes.

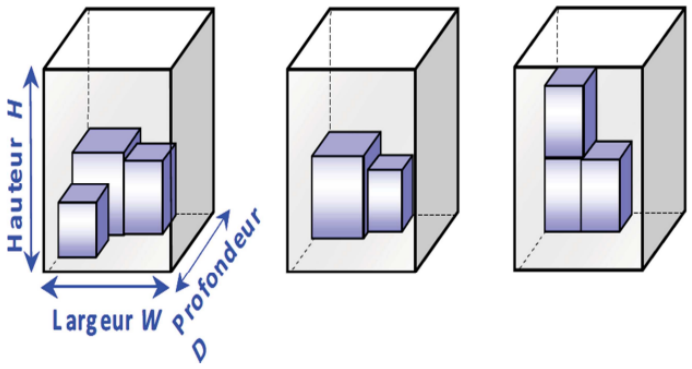


FIGURE 2.12 – llustration d'une solution réalisable pour CSP-3D

Exemple : Une instance est caractérisée par un ensemble de n objets et un ensemble de supports.

"1.éviter le débordement,

2. respecter le tonnage (volume intérieur) du support,

3. définir des précédences et des préférences de regroupement entre certains objets."

Donc, le problème fondamental est de placer tous les objets tels que :

- les objets ne se chevauchent pas.

- le nombre de supports utilisés est minime.

- les contraintes sont respectées

Il s'agit d'un problème lié à la charge, des contraintes extras sont appliqués à chaque charge

La classification des modèles des bandes et des découpes représentée dans figure

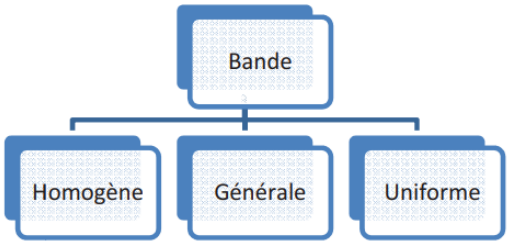


FIGURE 2.13 – -a-Modèles de Bandes

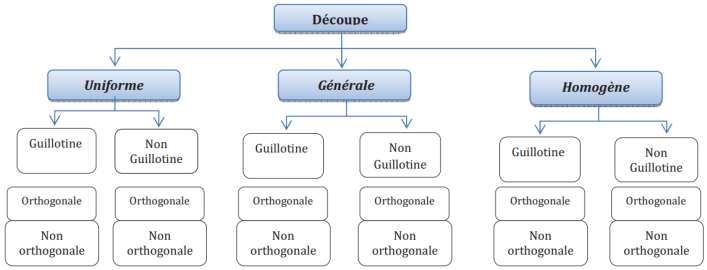


FIGURE 2.14 – -b-classification des modèles des bandes et des découpes

2.2.5 Complexite de Problème de découpe 2D

Un algorithme est dit efficace si le nombre d'opérations nécessaires pour résoudre le problème est limité par une fonction polynomiale.

La forme générale du problème de découpage (Bin Packing) ou plus précisément du problème de décision est la suivante :

Étant donné m boîtes de capacité C, peut-on emballer un ensemble de n objets dans le nombre minimum de boîtes ?

Il a été prouvé dans la théorie de la complexité que ce problème appartient à la classe des problèmes durs(difficile ,Hard) ou problèmes NP-complets.

- Notre problème repose sur :

-Disposer un ensemble de n blocs rectangulaires dans une plaque rectangulaires en utilisant le minimum de celles-ci en tenant compte de certains contraintes, està coup sûr plus difficile que le problème général de Bin packing, car nous avons aussi des contraintes spécifiques à prendre en compte.

Par conséquent, trouver la solution optimale en temps polynomial est difficile pour ce problème, car il n'existe pas encore d'algorithme puissant pour le résoudre efficacement, et tous les travaux menés Actuellement basé sur une technologie approximative «near optimum techniques ».[17]

2.2.6 Formalisation

Compte tenu de n petits articles rectangulaires, le 2DCSP doit couper tous les articles dans les grands matériaux rectangulaires dans le but de minimiser le nombre de matériaux utilisés. [16]

Soit une liste L de pièces rectangulaires $P_i(i = 1, n)$ de taille donnée et Étant donné des boîtes rectangulaires B_j et de dimensions données, satisfaisant $\forall P_i(i = 1, n) Dim(Pi) \leq Dim(B_j)$

Objectif : Notre objectif et de trier les pièces P_i dans les cases B_j en réduisant le nombre de cases utilisées

2.2.6.1 Contrainte de placements :

On définit le placement orthogonal d'une pièce P_i dans boite B_j comme étant le quadruplé $(X_{i1}, X_{i2}, Y_{i1}, Y_{i2})i$ issu de la des pièces de P_i sur les axes du plan :

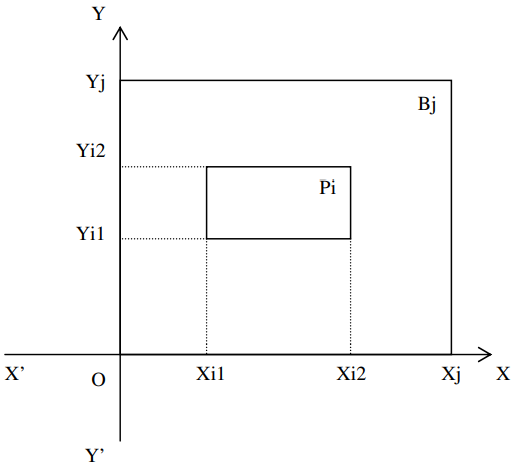


FIGURE 2.15 – le placement orthogonal

2.2.6.1.1 Contrainte de débordement :

Une pièce ne doit pas excéder le domaine $B_j X_{i1} \geq 0 X_{i2} \leq X_j Y_{i1} \geq 0 \& Y_{i2} \leq Y_j$

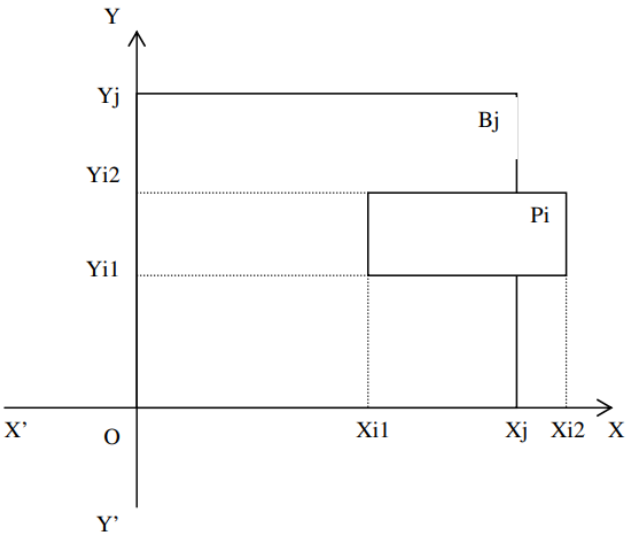


FIGURE 2.16 – le placement orthogonal-Contrainte de débordement-

2.2.6.1.2 Contrainte de chevauchement

Soit deux pièces P_i, P_k placées dans le domaine B_j (ou P_i et P_k satisfaisant la contrainte 1)

Pi s'intersecte avec P_k ssi l'une des inégalités suivantes et vérifiée :

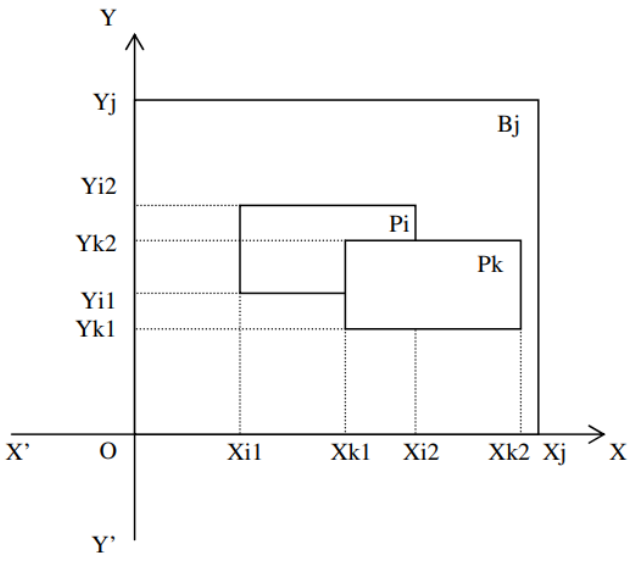


FIGURE 2.17 – le placement orthogonal-Contrainte de chevauchement-

$(\mathrm{Y}_{i1} < Y_{k2} < Y_{i2} X_{i1} < X_{k1} < X_{i2}) ou (Y_{i1} < Y_{k2} < Y_{i2} X_{i1} < X_{k2} < X_{i2})$

ou $(\mathrm{Y}_{i1} < Y_{k1} < Y_{i2} X_{i1} < X_{k1} < X_{i2}) ou (Y_{i1} < Y_{k1} < Y_{i2} X_{i1} < X_{k2} < X_{i2})$

2.3 Conclusion

En agrégeant les propriétés de différents types de modèles de coupe, nous avons montré une variété de formes de coupe.

Les problèmes de coupe peuvent être considérés comme le placement de pièces dans de gros objets. De la même manière, les problèmes de position peuvent être considérés comme des problèmes de coupe ; cela justifie le nom donné à ce type, en particulier les problèmes de découpage et de placement. D'autre part, certaines contraintes liées aux applications industrielles s'imposent en matière de découpe ou de positionnement. Citons en exemple la guillotine imposée par le découpage.Ainsi,compte tenu de la variété de ces problèmes et de leurs domaines d'application, de nombreuses variantes apparaissent sous différentes formes et sous différents noms dans la littérature.Cependant, une analyse minutieuse de ces questions révèle l'existence de certains aspects communs. En ce sens, il existe plusieurs classifications des problèmes de découpage et d'alignement dans la littérature, en particulier la classification présentée par Dyckhoff et Finke, basée sur les aspects suivants :le nombre de dimensions, le genre de tâches,l'assortiment de grands objets (plaques) et de petits objets (piéces).

Chapitre 3

Réalisation et Modélisation de notre Travail

3.1 Introduction

La phase de la réalisation est la dernière étape pour répondre à notre problématique initiale qui est l'optimiser les découpes 2D

D'abord, on va définir les différentes technologies et outils utilisés pour réaliser notre projet. Ensuite nous citerons les détailles de conception et d'implimentation de notre travail A la fin , nous présenterons les différentes fonctionnalités de notre application

3.2 Conception

La conception est l'une des étapes les plus importantes dans la construction d'un système.Dans cette section, nous analyserons les exigences fonctionnelles et non fonctionnelles et représenterons l'architecture de notee système.

3.2.1 Analyse des besoins

Les études fonctionnelles représentent la première phase du cycle de développement de l'application. Il doit clairement décrire l'application à développer, et il comprend la recherche et la description des fonctionnalités fournies par notre produit pour répondre aux besoins des utilisateurs.

3.2.1.1 Les exigences fonctionnelles

Le système à développer doit répondre aux exigences fonctionnelles suivantes :

Le système optimise le flux des feuilles de travail en fonction des besoins de l'entreprise.

Le système permet aux utilisateurs de filtrer les chutes à une longueur minimale.

Le système gère de manière optimale les flux de la cascade, qui peuvent être réutilisés pour les flux suivants.

Le système gère (ajoute, modifie, supprime) le stock de papier et chutes.

Le système permet d'afficher la liste des chutes existantes et la nouvelle liste.

Le système permet d'actualiser le stock après chaque modification.

3.2.2 Les exigences non fonctionnelles

Les besoins non fonctionnels concernent les contraintes à prendre en considération pour mettre en place une solution adéquate.

Notre application doit nécessairement assurer ces besoins :

1. Souplesse des interfaces (Minimum de clics).
2. Portabilité.
3. Latence (délai de réponse raisonnable)
4. Intégrité, fiabilité et cohérence des données.

3.2.3 Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation résument les détails des acteurs du système et leurs interactions avec le système. Les diagrammes de cas d'utilisation identifient différents types d'utilisateurs d'un système et différents cas d'utilisation, et sont souvent accompagnés d'autres Type de graphique. Un concept clé de la modélisation des cas d'utilisation est qu'elle nous aide à concevoir des systèmes du point de vue de l'utilisateur final. Le diagramme de cas d'utilisation de notre projet est pésenté dans la figure3.1

Cas d'utilisation relatif à notre système :

- *Minimiser le nombre des feuilles à acheter* : Ce cas est inclus dans le cas "Optimiser le débit des feuilles" notre système essaie d'utiliser le nombre minimal des feuilles

- *ajouter chute :* Ce cas est inclus dans le cas "gérer le stock" le système ajoute les chutes restantes après l'opération d'optimisation

- *actualiser la liste des chutes :* Ce cas est inclus dans le cas "Gérer le stock" après chaque utilisation des chutes et chaque opération d'optimisation des feuilles le système mis à jour le stock

-*Gérer le stock des chutes :* Le système essaie de gérer le stock d'une manière optimale tel que la suppression des chutes utilisées et l'ajout des nouveaux chutes aux stock.

- *Optimiser le débit des feuiles :* notre système doit choisir la meilleure solution et pour cela vaut mieux utiliser tout d'abord les chutes existantes dans le stock, et puis utiliser les pièces a couper

- *Afficher le mode de découpage :* Le système découpe chaque feuille ou une chute à l'aide de notre Optimeizer qui retourne la solution la plus optimale qui a unminimum de chutes, un minimum d'utilisation des feuilles , notre système affiche leurs mode de découpage

Le diagramme dans la figure 3.1 représente les fonctionnalités ,des acteurs qui interagissent avec notre système :

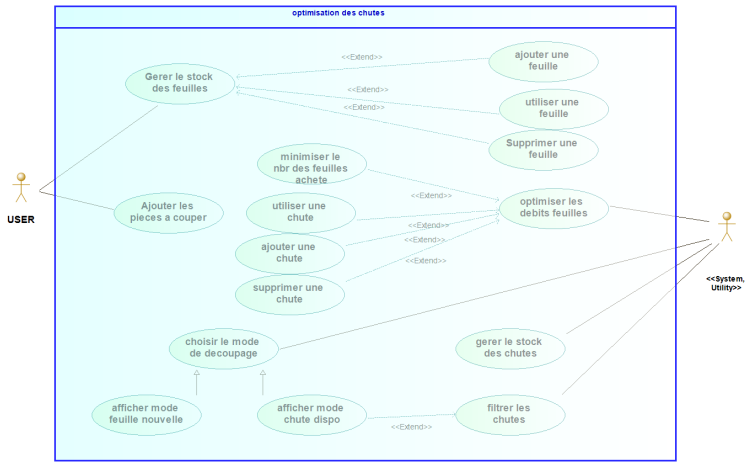


FIGURE 3.1 – Le diagramme de cas d'utilisation

3.2.4 Diagrammes de séquences

Les diagrammes de séquence UML sont des diagrammes d'interaction qui détaillent la façon dont les opérations sont effectuées. Ils définissent les interactions entre acteurs et systèmes Cadre de mise en œuvre des cas d'utilisation. Les diagrammes de séquence sont basés sur le temps et affichent visuellement la séquence d'interactions en utilisant l'axe vertical du graphique pour représenter les messages envoyés et quand ils ont été envoyés.

Dans cette section, nous allons détailler le diagrammes de séquences "Optimisation"

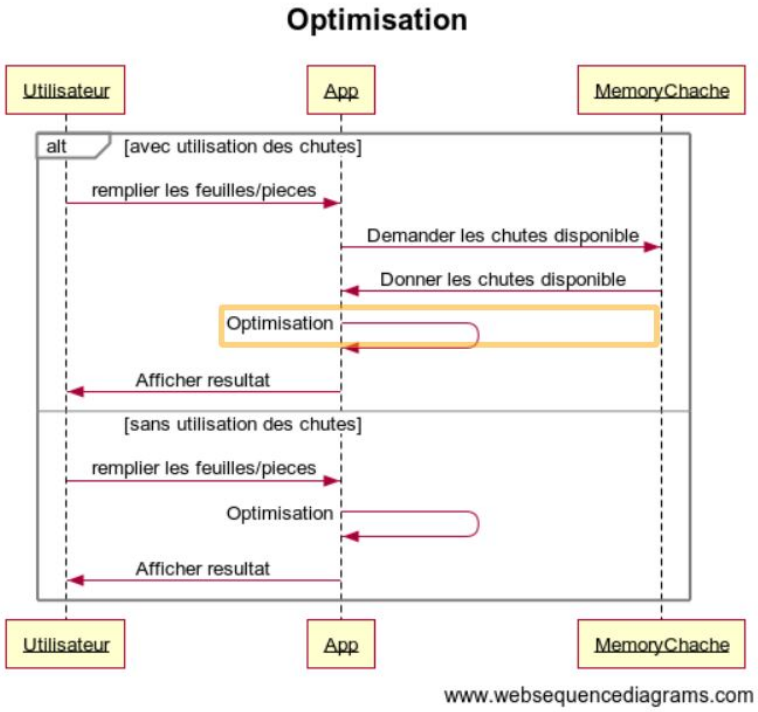


FIGURE 3.2 – Diagramme de séquence'optimisation'

Scénario nominal :

Avec utilisation des chutes :

1. L'utilisateur choisit l'option "utiliser les chutes".

2. L'utilisateur envoie la liste des piéces à découper .

3. Le système demande au MemoryCache la liste des chutes disponible.

4. Le système envoie la liste des chutes disponible.

5. L'algorithme découpe chaque piéce commandée de telle sorte que l'utilisation des chutes est maximale.

6. Le système affiche la liste des résultats obtenue.

***sans utlisation des chutes* :**

1. L'utilisateur ne veux pas utiliser les chutes.

2. L'utilisateur envoie la liste des piéces à découper .

3. L'algorithme découpe chaque piéce commandée de telle sorte que l'espace laissé libre soit minimal.

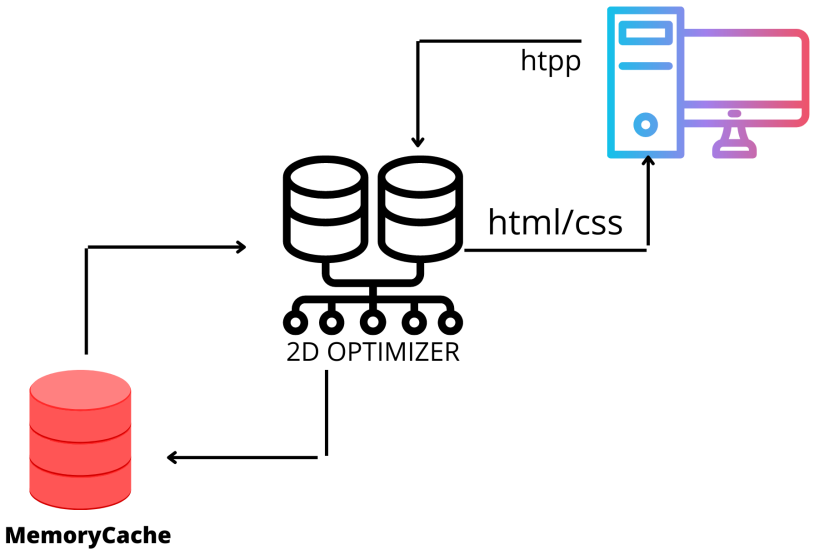
4. Le système affiche la liste des résultats obtenue.

3.2.5 Architecture du système

Nous présentons l'architecture générale de notre système dans la figure . Les composants principaux sont :

-Le 2D-OPTIMIZER : contient les modules attachés au système.

-MemoryCache : où les chutes sont stocké.



3.2.6 Algorithme de découpage

Pour résoudre notre problème, nous nous sommes appuyés sur un algorithme qui privilégie la qualité de la solution. En effet, pour la modélisation nous avons utilisé un algorithme moderne qui est un algorithme génitique a l'aide d'une bibliothéque développé en typescript basé sur AG et comme méthode, nous avons commencé par réduire notre problème de découpage au problème de 2D Bin packing , et utiliser le cut-optimizer-2d par la suite pour résoudre ce problème d'optimisation.

3.2.6.1 Contrainte du problème

En résumé, le problème de la découpe bidimensionnelle d'un matériau dans lequel L'objectif de ce travail est celui qui répond aux exigences suivantes :

Il n'y a qu'une seule pièce originale, à partir de laquelle les autres pièces sont obtenues ;une fois que le problème à résoudre a été déterminé avec une certaine précision, il faut que :

1. La pièce originale est de forme rectangulaire . Les pièces à obtenir sont de forme rectangulaire .
2. Les pièces à obtenir ont une taille plus petite que la pièce original.
3. Une quantité de pièces à obtenir n'est pas désignée : si deux pièces de mêmes caractéristiques sont souhaitées, elle est indiquée avec deux pièces différentes avec les mêmes paramètres .
4. Le matériau perdu est celui qui reste entre deux ou plusieurs pièces taillées.
5. Les pièces ne peuvent pas être tournées.

3.2.6.2 Formalisation du problème Bin Packing 2D BPP2D

Nous avons déjà mentionné que le problème de découpe 2D est similaire à un type particulier du problème du bin packing(BIN PACKING 2D)

Le problème de bin packing en 2 dimensions s'agit de minimiser le nombre de grands rectangles pour ranger une liste d'objets rectangulaires. Les objets doivent être rangés de telle manière que les côtés des rectangles soient parallèles à ceux des bacs.On note (I, w, h, W, H) une instance de BPP-2D, I=1,2,. . .,n est la liste des objets à ranger, et une fonction w (Respectivement h) qui associe à chaque objet i une valeur w_i (respectivement h_i) qui correspond à sa longueur (respectivement sa largeur),W et H représentent la longueur et la largeur du bin.

3.2.6.3 Énoncé du problème

Les objectifs de ce problème d'optimisation sont au nombre de deux :

1. Minimiser la surface occupée par le plus petit rectangle possible qui englobe toutes les pièces selon le tracé réalisé. Le but de cet objectif est de minimiser la zone qui reste entre les pièces une fois disposées, car elle est considérée comme un matériau inutile car de petite taille, tandis que la zone résultante sur les côtés de ce rectangle est considérée comme un matériau utile„
2. Maximiser la proportion entre la hauteur et la largeur du rectangle minimum possible qui englobe toutes les pièces selon l'agencement réalisé Dans certains cas, la pièce d'origine a une certaine largeur et la seule chose qui est recherchée est de minimiser la longueur à dépenser lors de l'obtention d'un ensemble de pièces. Pour généraliser le problème à cet égard, il a été décidé de délimiter la pièce d'origine avec une largeur et une longueur prédéterminées et d'essayer d'obtenir une disposition la plus carrée possible, pour éviter des répartitions trop verticales ou trop horizontales

En résumé, le problème de la découpe bidimensionnelle d'un matériau dans lequel L'objectif de ce travail est celui qui répond aux exigences suivantes :

— Il n'y a qu'une seule pièce originale, à partir de laquelle les autres pièces sont obtenues ;
— La pièce originale est de forme rectangulaire ;
— Les pièces à obtenir sont de forme rectangulaire ;
— Les pièces à obtenir ont une taille plus petite que la pièce original ;
— Une quantité de pièces à obtenir n'est pas désignée : si deux pièces de mêmes caractéristiques sont souhaitées, elle est indiquée avec deux pièces différentes avec les mêmes paramètres ;
— Le matériau perdu est celui qui reste entre deux ou plusieurs pièces taillées ;
— Les pièces ne peuvent pas être tournées
— La répartition finale des pièces doit permettre de l'obtenir en n'utilisant que des coupes guillotines

— Le positionnement des pièces se fait par rapport au coin supérieur gauche de la pièce d'origine, de sorte que la pointe du coin supérieur gauche de la première pièce disposée coïncide avec la pointe du coin supérieur gauche de la pièce d'origine.

3.2.6.4 Formalisation

Soit P le morceau de matériel d'origine ;

$p_1, p_2, \ldots, p_n$,les pièces à obtenir

r ,la désignation d'un espace défini au plan, sousla forme d'un rectangle.

$w_i \in$,la largeur du morceau i ou du rectangl r.

$l_i \in$,la longueur du morceau i ou du rectangle$r$.

x_i ,la position sur l'axe des abscisses du morceau i ou du rectangle r a par rapport au point initial.

y_i, la position sur l'axe des ordonnées de la pièce i ou du rectangle r par rapport au point initial.

$SPC(p_1, p_2, \ldots, p_k)$,la fonction qui obtient le plus petit rectangle possible r contenant $p_1, p_2, \ldots, p_k$.

L'objectif numéro un du problème est de minimiser les déchets, ceci étant le matériau qui se trouve dans le rectangle minimum qui couvre toutes les pièces mais ne fait partie d'aucune pièce. Cet objectif peut être exprimé comme suit :

$$\underset{(x_1,x_2..x_n)(y_1,y_2..y_n)}{\operatorname{argmax}} = \frac{\sum_{i=1}^{n} w_i l_i}{W_s pc(p_1, p_2..p_n) L_s pc(p_1, p_2..p_n)}$$

1

Et l'objectif numéro deux du problème est de maximiser la proportion régulière du rectangle r de plus petite taille possible qui contient tous les morceaux que nous voulons les obtenir, lui donnant une forme aussi carrée que possible, donc il est défini comme :

$$\underset{(x_1,x_2..x_n)(y_1,y_2..y_n)}{\operatorname{argmax}} g(p_1, p_2..p_n) = \frac{min(W spc(p1, p2..pn) Lspc((p1, p2..pn))}{maxW spc(p1, p2..pn) Lspc((p1, p2..pn)}$$

2 Les deux objectifs sont soumis à la fois à la contrainte numéro un (R1), qui force le plus petit rectangle possible r contenant toutes les pièces souhaitées à être contenu dans les limites de la pièce d'origine P. Cette restriction est notée :

$W_s pc(p_1, p_2..p_n) \leq W_p \wedge L_s pc((p_1, p_2..p_n) \leq L_p$ *3*

Et à la contrainte numéro deux (R2), qui garantit qu'il n'y a pas de chevauchement de pièces dans l'arrangement :

$p_i, p_j x_i < x_j + w_j \wedge x_i + w_j > x_j \wedge y_i > y_j - l_i \wedge y_i - l_i < y_j$ *4*

Une fois le problème formalisé et bien spécifié, il est possible commencer le processus de conception et de développement de l'outil qui le résoudra

3.2.7 Conception de l'algorithme génétique

3.2.7.1 Introduction aux algorithmes génétiques

Un algorithme génétique est une procédure de calcul heuristique dédiée à la recherche stochastique (plutôt qu'aléatoire), normalement utilisée pour résoudre des problèmes d'optimisation ou d'apprentissage automatique. Cette méthode imite la nature, de sorte que l'évolution des êtres vivants s'applique aux solutions d'un problème, représentées sous forme de chromosomes, afin qu'ils évoluent et deviennent de mieux en mieux (plus en forme).[18]

3.2.7.1.1 Comment fonctionne un algorithme génétique

Un algorithme génétique typique a une boucle principale, dans laquelle les actions caractéristiques les plus importantes sont effectuées. L'énumération suivante représente dans l'ordre comment ils agissent :

1. Une population initiale de solutions obtenues aléatoirement est générée.

2. Les solutions sont évaluées avec la fonction de fitness ou d'évaluation.

3. Il vérifie si une solution optimale (ou suffisamment bonne) a été obtenue ou si une autre condition d'arrêt est remplie. Si oui, ça se termine. Si ce n'est pas le cas, passez à l'étape quatre .

4. n individus sont sélectionnés (selon la méthode de sélection).

5. Les individus sélectionnés sont croisés et de nouveaux individus sont obtenus. – Les nouveaux individus subissent ou non une mutation au hasard.

6. La population est remplacée par les nouveaux individus obtenus et on revient à la étape numéro deux.

de cette façon, bien que cela dépende de la méthode de sélection, il est souhaitable que les individus les plus aptes soient choisis avec une plus grande probabilité, de sorte qu'en combinant les solutions dans la phase de croisement, on obtienne des solutions probablement également aptes, voire plus aptes. [19]

Un Algorithme génitique peut être résumé comme suit :

1) Générer aléatoirement et Evaluer une population initiale P(0) de N individus t <— 0.

2) Sélectionner N individus dans P(t) pour former une population Ps(t).

3) Combiner les individus de Ps(t) et appliquer les opérateurs génétiques de croisement et de mutation pour obtenir une population P(t+1).

4) Evaluer les individus de P(t+1), t <— t +1.

5) Aller en 2) ou arrêter selon un critère d'arrêt (qualité, temps, nombre de génération, etc.. . ..

pour mieux comprendre, le pipline suivant explique bien le parcours :

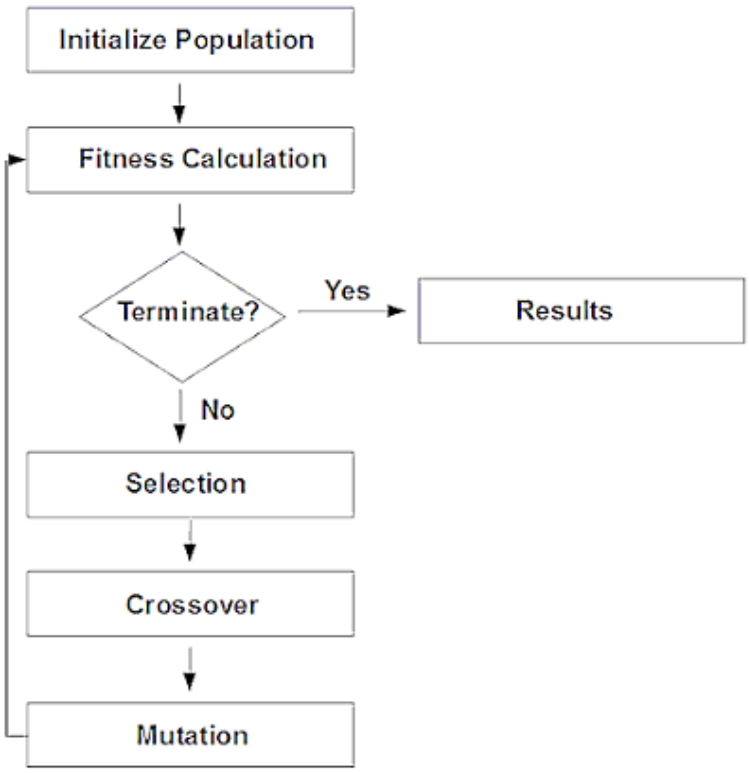


FIGURE 3.3 – AG pipline

3.2.7.2 Conception d'algorithmes génétiques

Lors de la résolution d'un problème en appliquant un algorithme génétique, plusieurs problèmes doivent être pris en compte. Il existe différentes techniques dans chaque aspect des opérateurs génétiques (croisement, mutation et sélection), ce qui rend ces algorithmes normalement flexibles et utilisables dans un grand nombre de problèmes, mais la représentation de la solution du problème à résoudre en tant qu'individu (chromosome) n'est généralement pas trivial, et c'est un aspect remarquablement important car il peut gravement affecter le succès ou non de l'algorithme.[20]

3.2.7.2.1 Paramètres démographiques de base

La population d'individus compte n individus, qui sont modifiés à chaque génération pour avancer dans la résolution du problème. La valeur de n est déterminée de manière empirique et, à moins qu'elle ne soit extrêmement faible, des résultats similaires ont tendance à être obtenus à chaque exécution de l'algorithme. Pour des valeurs très élevées de n, l'algorithme s'avère plus lent et n'apporte aucune amélioration apparente par rapport à la solution obtenue. C'est pourquoi il est nécessaire d'établir une valeur de$n$ qui ne soit pas trop faible pour qu'elle compromette la population à avoir peu de diversité, et qu'une convergence prématurée ou précoce se produise ; ni qu'il est trop élevé et nécessite plus de ressources en vain.

Généralement, une taille de population acceptable se situe entre 30 et 100 individus, bien que cela dépende du problème auquel elle s'applique. Dans ce cas, des tests ont été effectués avec différentes tailles

de population dans la plage mentionnée. [21]

3.2.7.2.2 Codage des solutions

Lors de la codification des solutions pour appliquer l'algorithme génétique, il est nécessaire d'être clair sur les structures de données qui doivent être utilisées pour gérer les solutions du problème et comment ces solutions vont être représentées, c'est-à-dire qu'il est nécessaire de concevoir une fonction qui transforme le génotype d'un individu en le phénotype correspondant, afin de visualiser et de comprendre la solution que représente un individu donné.

Pour ce faire, il faut d'abord déterminer comment représenter les individus, c'est-à-dire les solutions du problème. Pour que les gènes du chromosome représentent un arrangement d'éléments dans la pièce d'origine sans ambiguïté, une notation postfixée (également connue sous le nom de notation polonaise inversée) a été adoptée pour construire les chromosomes.[21] [22]

Dans ces chromosomes, deux opérateurs sont utilisés : l'opérateur H et l'opérateur V.

L'opérateur H prend deux rectangles (c'est-à-dire, morceau, élément) comme arguments et produit le plus petit rectangle capable de couvrir les deux pris, disposés horizontalement côte à côte. autre. L'opérateur V effectue la même action, mais la différence est que les rectangles pris comme opérande sont disposés verticalement. Par exemple, le chromosome « 1 2 H » indique que le rectangle numéro deux est disposé horizontalement, à droite du rectangle 1 comme suit :

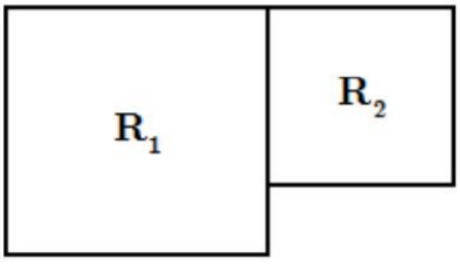


FIGURE 3.4 – Phénotype du chromosome « 1 2 H »

De même, le chromosome "1 2 V" indique que le rectangle numéro deux est disposés verticalement, sous le rectangle 1 comme suit :

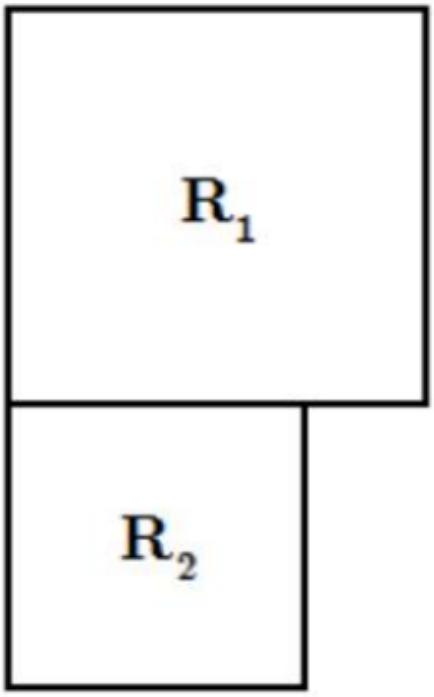


FIGURE 3.5 – Phénotype du chromosome « 1 2 H »

Suivant cette manière de construire les solutions, la longueur (taille horizontale) L_h et la largeur (taille horizontale) W_h des parties intégrées (Désormais, les rectangles ou morceaux qui sont obtenus à partir de deux morceaux ou morceaux intégrés effectuant une opération en H ou en V, seront appelés « morceau intégré ») avec un opérateur H peut être déterminée comme suit :

$L_h = L_1 + L_2, W_h = max(W_1, W_2)$

où L1 et L2 sont les longueurs des pièces un et deux, et max est une fonction qui prend deux paramètres en entrée et renvoie celui qui est le plus grand. Dans le cas où les pièces sont intégrées avec un opérateur V, la longueur et la largeur de la pièce intégrée s'obtiennent en effectuant les mêmes opérations mais en sens inverse :

$Lh = max(L1, L2), Wh = W1 + W2$

Vous trouverez ci-dessous un exemple plus complet de ce que ressemble un cas réel proposé. Soit le chromosome d'un individu « 1 5 H 7 6 V 2 H 4 HV 9 8 V 3 V H », cela représente la solution suivante pour des tailles données des pièces demandées :

FIGURE 3.6 – Exemple d'un individu

Pour obtenir la série de coupes nécessaires à l'obtention d'une pièce, il suffit de tracer un chemin depuis la racine de l'arbre jusqu'au nœud feuille où se trouve le numéro de la pièce souhaitée et d'effectuer les coupes indiquées par ledit chemin

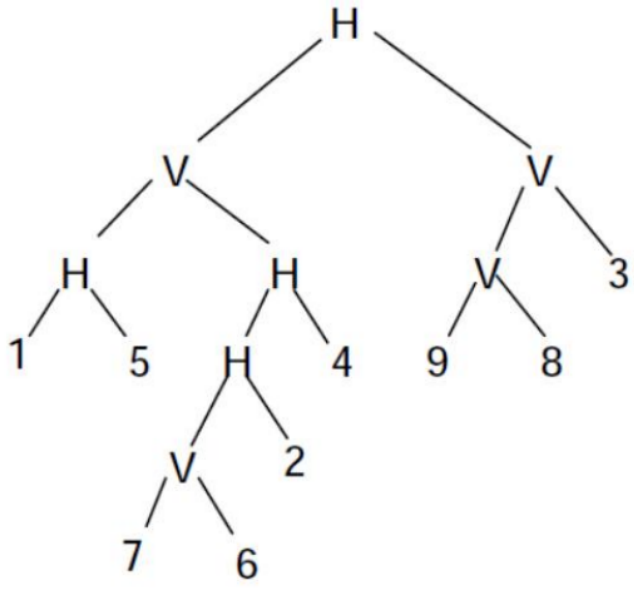


FIGURE 3.7 – Exemple individuel

La formulation suivie pour générer les chromosomes, étant sous le format de notation postfixée, doit suivre une série de restrictions pour que le chromosome soit effectivement valable. De plus, le fait de suivre une notation non ambiguë implique que tous les chromosomes ont la même longueur pour un nombre donné

de morceaux n , ce qui est un aspect souhaitable dans un algorithme génétique pour faciliter la lecture et le traitement des individus (chromosomes). Soit N_p le nombre de numéros de pièce et N_o le nombre d'opérateurs que possède un chromosome, tout d'abord la contrainte suivante doit être remplie, sinon le chromosome ne donne pas lieu à une solution valide : $N_o = N_p - 1$

Deuxièmement, notons que le fait d'adopter la notation postfixée rend les expressions des chromosomes exemptes d'ambiguïté, donc l'utilisation de parenthèses n'est pas nécessaire. En tenant compte de cela et de la restriction donnée par l'équation en haut, la longueur du chromosome N_g pour un nombre de pièces donné peut être obtenue : $N_g = N_p + N_o = 2N_p - 1$

Et à partir de ces deux équations, nous pouvons obtenir à la fois N_p et N_o comme N_g seulement si la valeur de l'un d'entre eux est connue.

Une autre chose à considérer est une restriction sur les positions relatives des numéros de pièce et des opérateurs. En prenant n'importe quel opérateur, soit n_p ; le nombre de références qui existent sur le côté gauche de cet opérateur et n_o ; le nombre d'opérateurs qui existent sur le côté gauche de cet opérateur, en tenant compte également de ce même opérateur, la relation suivante doit être vérifiée : $1 \leq n_o \leq n_p - 1$

Étant donné que les relations décrites ne sont pas pertinentes pour le type d'opérateurs, tous les arrangements possibles peuvent être obtenus en sélectionnant les opérateurs pertinents (H ou V) et en modifiant à la fois leurs positions et les positions des numéros de pièce sur le chromosome.

3.2.7.2.3 Fonction de fitness

Compte tenu des objectifs décrits dans la section 3.3.6.4 la fonction de fitness est aussi simple que de regrouper dans la même fonction la valeur de fitness qu'une solution donnée a pour chacun des objectifs. Ce regroupement d'objectifs est calculé avec un poids différent pour chacun, de sorte qu'un objectif est plus important que l'autre . La différence de poids peut impliquer un grand changement lorsqu'il s'agit d'obtenir des résultats dans l'algorithme, puisque la fonction de fitness est l'élément qui simule les facteurs environnementaux qui conditionnent les individus à survivre ou non. Ainsi, soit $S = (p_1, p_2, \ldots, p_n)$ une solution réalisable, la fonction de fitness est donnée par :

$fitness(S) = w_f \quad f(s) + W_g \quad g(s)$

Où : w_f est le poids associé à l'objectif un (1). $f(S)$ est la fonction qui calcule la valeur de fitness de l'individu en tenant compte la satisfaction du but un (1) (fonction *2* partie objectifs). w_g est le poids associé à l'objectif deux (2). $g(S)$ est la fonction qui calcule la valeur de fitness de l'individu en tenant compte e la satisfaction de l'objectif deux (2) (fonction *3* partie objetif).

3.2.7.2.4 Méthode de sélection

La méthode de sélection utilisée est la sélection à la roulette. Cette méthode de sélection établit une probabilité de sélection équivalente à la proportion de fitness d'un individu par rapport à la fitness de la population totale. La formule suivante détermine la probabilité qu'un individu donné soit choisi :

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i}$$

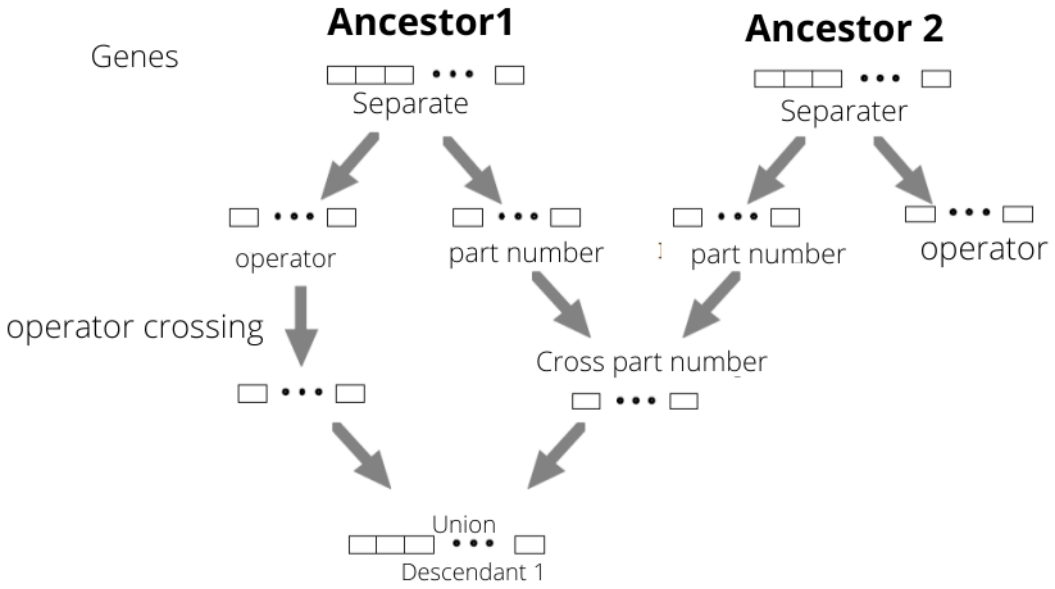
où :

-p_i est la probabilité que l'individu soit sélectionné .

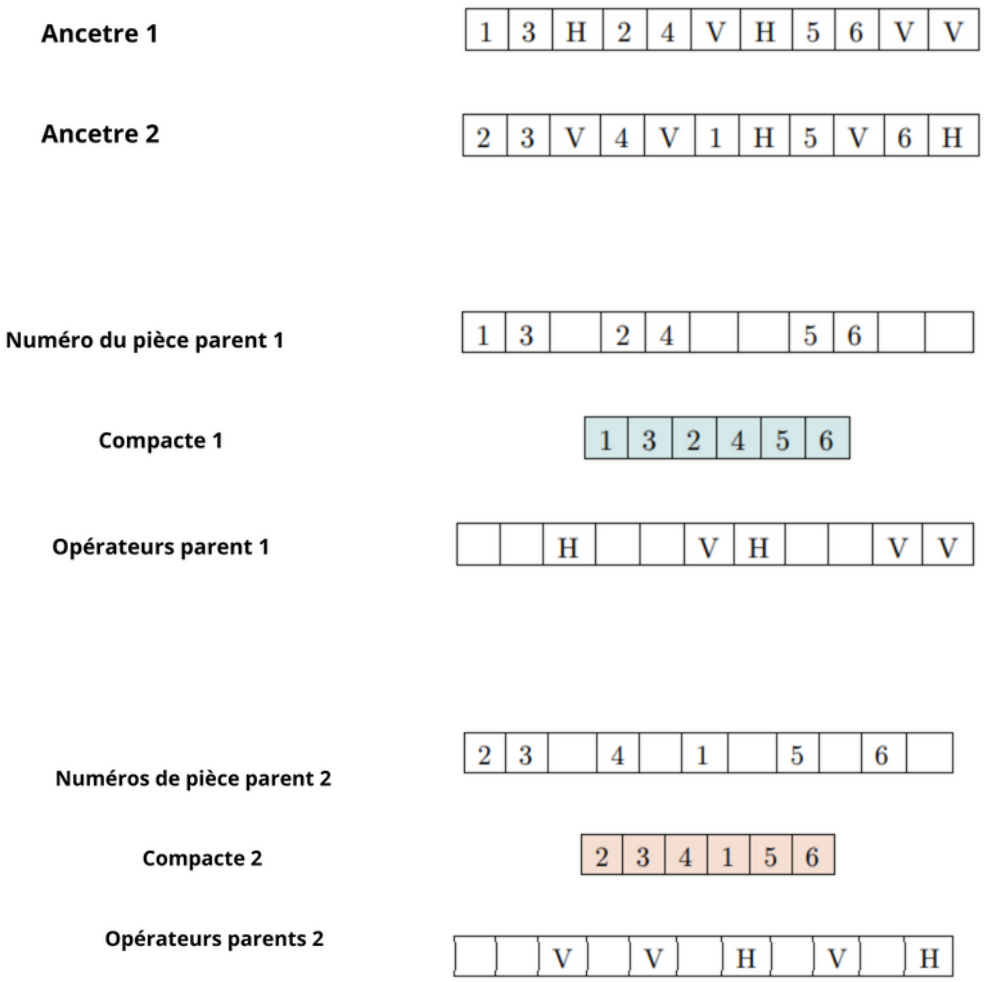
-f_i est la valeur de fitness de l'individu i .

3.2.7.2.5 méthode de croisement

La méthode de croissement implique deux parents sélectionnés à l'aide de la technique de la roulette pour former deux descendants. Le nombre de parents sélectionnés par croisement et la méthode de sélection sont des valeurs fixes de l'algorithme et ne peuvent pas être modifiés. Étant des chromosomes composés de gènes de deux types différents (opérateurs et numéros de pièces), le croisement doit être particulier. Étant donné que les gènes indiquant les numéros de pièce et les gènes indiquant les opérateurs ne partagent pas les mêmes caractéristiques et contraintes sur le chromosome, ils sont séparés des individus avant de se croiser, de sorte qu'à l'aide d'un masque de bits, tous les opérateurs d'une part et toutes les parties les nombres d'autre part sont sélectionnés pour les combiner avec ceux d'un autre individu puis les rejoindre pour former la progéniture

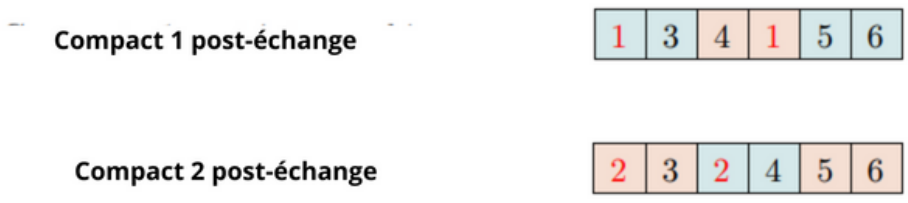


En ce qui concerne le croisement, tout d'abord, étant donné la nature ordinale des gènes chromosomiques qui contiennent des numéros de pièce, aucune méthode de croisement ne peut être utilisée, comme le croisement par échange direct, car les chromosomes résultants ne seraient pas valides, car ils auraient une partie répétitive. numéros et autres numéros manquants. C'est pourquoi des méthodes doivent être utilisées pour recombiner des chromosomes ordonnés. Deuxièmement, pour les opérateurs, puisqu'ils n'ont à suivre aucun ordre et peuvent être répétés ou non, il n'est pas nécessaire d'effectuer un croisement spécial, ils sont donc échangés directement et sont placés dans les mêmes positions que l'un des opérateurs.chromosomes des parents pour s'assurer que le chromosome est valide .L'exemple suivant montre un croisement complet étant donné les chromosomes de deux individus :



Le croissement s'applique aux « Compact 1 » et « Compact 2 » :

1. Échange direct des gènes compris dans le segment (aléatoire) [3, 4] (tous deux inclus) des listes de numéros de pièces compactes est obtenu :



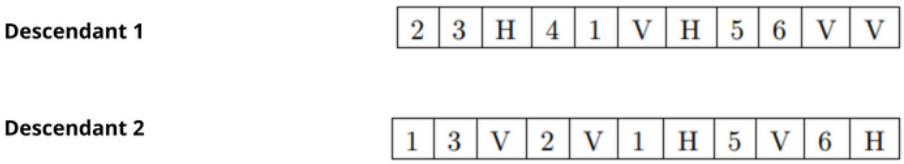
2. La relation des nombres inclus dans le segment est établie :

$4 \leftrightarrow 2 \leftrightarrow 1$

3. Les valeurs engagées qui sont en dehors du segment sont échangées contre une valide de la relation :

4. Les opérateurs du segment délimité par les numéros de partie choisis au hasard à l'étape 1 sont échangés. Dans ce cas, comme le chromosome du parent n'a pas d'opérateurs entre le gène indiquant le numéro de partie 2 et le numéro de partie de la partie 4, il n'y a pas de croisement d'opérateurs .

5. Les listes d'opérateurs sont combinées avec les listes compactes des parents pour obtenir les deux descendants :



3.2.7.2.6 Méthode de mutation

La mutation adoptée consiste en l'échange des éléments de deux gènes choisis au hasard, qu'il s'agisse de gènes contenant un opérateur ou un numéro de pièce. La probabilité de mutation s'applique à tous les individus à l'exception des individus élites une fois constituée la nouvelle génération constituée intégralement des descendants issus des croisements comme indiqué au point 3.3.7.2.6

si p_1 et p_2 sont deux éléments quelconques du chromosome et que p_1 est situé à gauche de p_2, un échange mutuel est possible si

p_1 et p_2 sont des références,

p_1 est un opérateur, quel que soit le type p_2

p_1 est un numéro de pièce, p_2 est un opérateur et après avoir été échangés ils sont rencontre Cependant, comme il est souhaitable de savoir si le résultat est valide avant d' échanger le contenu des gènes, il peut être généralisé et exprimé comme suit :

p_1 est un numéro de pièce, p_2 est un opérateur, et tous les opérateurs entre p_1 et p_2 doivent satisfaire à ce qui suit :

$n_o \leq \mathrm{n}_p$ - 3

3.2.7.2.7 Méthode de remplacement

La méthode de remplacement est que toute la population (sauf les n individus élites) est remplacé par

la progéniture générée, de sorte qu'entre les générations seuls les individus d'élite survivent. Par exemple, si la taille de la population est de 50 et le nombre d'élites désignées est de 2, pour passer de la génération k à la génération $k+1$.., il faut faire 24 croisements de 48 parents (évidemment les parents peuvent être répétés dans différents croisements). donnent naissance à 48 nouveaux individus, qui sont ceux qui forment la population de la génération $k+1$. [20]

3.3 Téchnologies et Outils utilisées

3.3.1 Technologie et Langages

Cette section est destinée à la présentation des différentes technologies que nous avons choisi pour réaliser notre application :

3.3.1.1 HTML

HyperText Markup Language (HTML) est le code utilisé pour structurer une page web et son contenu. Par exemple, le contenu de votre page pourra être structuré en un ensemble de paragraphes, une liste à puces ou avec des images et des tableaux de données. Comme le suggère le titre, cet article vous fournit les bases de compréhension du HTML et de ses fonctions , mais qu'est-ce que HTML, réellement ? HTML n'est pas un langage de programmation. C'est un langage de balises qui définit la structure de votre contenu. HTML se compose d'une série d'éléments, utilisés pour entourer, ou envelopper, les diverses parties du contenu pour les faire apparaître ou agir d'une certaine façon. Les balises entourantes peuvent être rendues par un mot ou une image lien hypertexte vers quelque chose d'autre, un texte en italique, une police plus grande ou plus petite, et ainsi de suite.[23]

3.3.1.2 CSS

CSS (Cascading Style Sheets) permet de créer des pages web à l'apparence soignée. Cet article vous propose de lever le voile en expliquant ce qu'est CSS ; un exemple simple en présentera la syntaxe, puis quelques termes clés du langage seront introduits.CSS peut être utilisé pour une mise en forme élémentaire des documents — par exemple, changer la couleur et la taille des titres et des liens. Il peut être utilisé pour concevoir une maquette — par exemple, transformer un texte affiché sur une colonne en une composition avec un cadre principal et une barre latérale pour les informations reliées. Avec CSS, on peut aussi produire des animations

3.3.1.3 TypeScript

Typescript est un langage de programmation open source fait par Microsoft. Pour être plus précis, c'est un sure-nsemble de Javascript. C'est-à-dire que tout programme Javascript existant est déjà un programme Typescript valide. Autrement dit, un développeur Javascript, n'as aucune barrière d'entrée. Typescript est un langage multi paradigmes. on peux faire du fonctionnel, comme de l'orienté objet sans problème et comme il permet de faire du Javascript, donc de travailler dans le navigateur, en gardant environnement fortement typé et orienté objet.[24]

En adaptons le typescript ,nous bénéficions :

-Meilleure intégration avec les éditeurs de texte

-Les types aident à documenter le code

-Il est plus rapide de refactoriser

-Attrape les bugs avant la compilation

-Peut compiler vers une version ES cible

-Accédez en avant-première aux futures fonctionnalités de JS

3.3.1.4 Angular

Angulaire est un open-source framework frontal conçu pour les applications modernes et des applications Web puissantes. Il continue d'affirmer sa domination en tant que nom le plus établi dans le monde du front-end Frameworks JavaScript. Ce framework est compétent pour développer des applications monopages puissantes et vieillissantes.[25]

a principale raison de sa popularité est qu'il est facile et simple à utiliser, il simplifie également le processus de développement et la structure de Code JavaScript,parmis les raisons on trouve :

1) Cohérence et réutilisabilité du code :La cohérence du code est la base d'un environnement de développement puissant pour réussir, car un codage incohérent augmente les risques de lancements retardés ou de coûts élevés.

2) Facile à apprendre, à utiliser et à tester : AngularJS peut être facilement appris par des personnes connaissant JavaScript, HTML et CSS.

3) Fonctionnalités orientées SPA : AngularJS et les SPA vont tout simplement ensemble, AngularJS prend en charge le développement d'applications à page unique. L'objectif principal du développement d'applications d'une seule page est transition de site Web plus rapide.

4) Liaison de données bidirectionnelle : AngularJS est un cadre très demandé dans de nombreux secteurs. Et la principale raison derrière cela est sa fonctionnalité de liaison de données bidirectionnelle.Cela signifie que lorsque le framework est déclenché par un événement dans le navigateur Web, il peut mettre à jour le module et les actions de l'utilisateur sur la page Web et modifier les modèles nécessaires et essentiels.

3.3.2 Les outils utilisés

Nous présentons ci-dessous les différents outils utilisés pour réaliser notre application :

3.3.2.1 Outils de développement

C'est les outils qui offrent différentes fonctionnalités afin de faciliter le processus de développement logiciel.

3.3.2.1.1 Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la factorisation du code et un Git intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.[26]

3.3.2.2 NPM

NPM est l'abréviation de Node Package Manager, qui est un outil (programme) gérant les bibliothèques de programmation JavaScript pour Node.js. Cet outil est réellement nécessaire pour le monde open source.[27]

3.3.2.3 Outils de Modélisation

3.3.2.3.1 Modelio Open source

Modelio est un outil de modélisation UML disponible sur les plates-formes Windows, Linux et Mac. Il intègre également la modélisation BPMN, et le support de la modélisation des exigences, du dictionnaire, des règles métier et des objectifs. Modelio propose une gamme d'outils étendant ses fonctionnalités permettant, la mise en œuvre de l'approche MDA.[28]

3.4 Présentation de l'application

Pour résoudre le problème et mettre en pratique les différentes idées qui ont été écrites plus tôt dans ce document, une application a été développée qui donne à l'utilisateur la possibilité de sélectionner la taille et le nombre de pièces qu'il souhaite obtenir à partir de l'original ou bien a travers les chutes

La première étape de l'exécution de l'application c'est l'affichage de l'unique interface de notre application la figure3.3 montre ceci

FIGURE 3.8 – Interface de l'application

l'application donne la possibilité de saisir les feuilles souhaité a utliser en mentionns la largeur et la longeur

figure3.4 montre ceci :

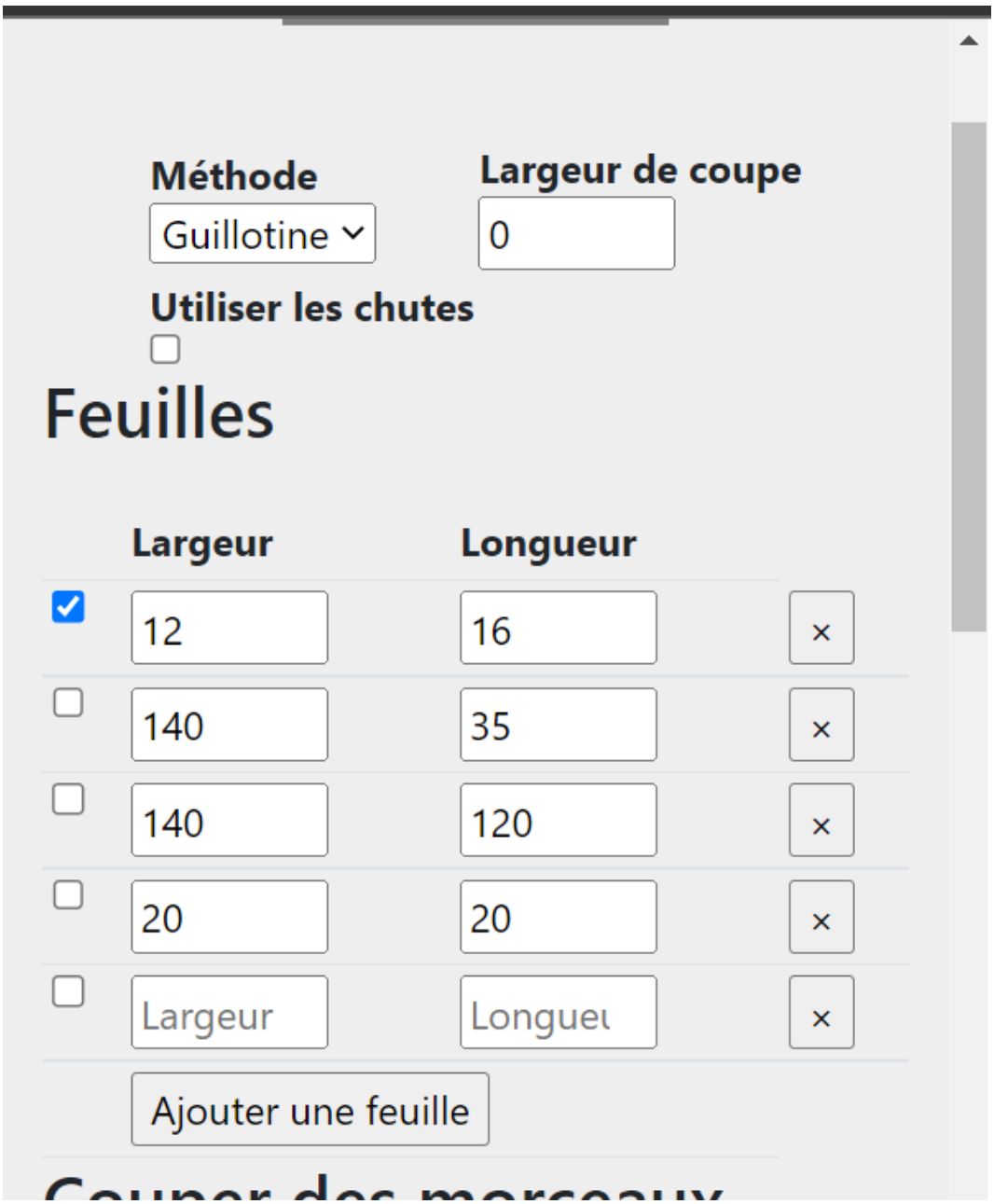


FIGURE 3.9 – saisir Feuilles

On peut aussi ajouter plusieurs pièces a couper avec le nombre ,largeur et longeur souhaité figure3.5 montre ceci :

FIGURE 3.10 – saisir pièces

Une fois les feuilles et les pièces l'application donne la possibilitée de selectionner ou non une feuille ou bien une pièce déjà saisi La figure 3.6 illustre ça :


Optimiser
Méthode
Guillotine
Largeur de coupe
0
Utiliser les chutes
Feuilles
Largeur
Longueur
12
16
140
35
140
120
20
20
Largeur
Longueu
Ajouter une feuille
Couper des morceaux
Quantité
Largeur
Longueur
2
24
13
2
22
22
1
32
2
3
12
45
1
111
12

Aprés avoir sélectionné le nombre des piéces a couper et le nombre des feuilles à utiliser l'étape suivante est affichage des piéces coupé "Optimiser" on faisons appel à "optimizeMethod" et "optimizerService" comme il est présenté dans la figure 3.7

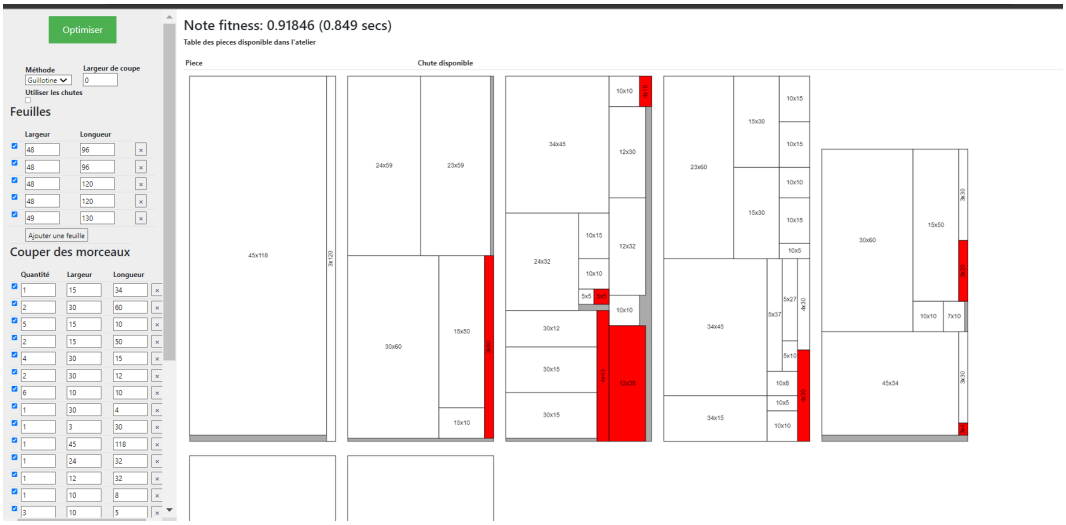


FIGURE 3.12 – Affichage des pièces découpé

Conçernant les chutes présenté aprés notre découpe et à l'aide de "chutes-table.component" on peux les stocker et l'utiliser dans la prochaine découpe ,en revanche on peut les supprimées , les figures 3.8 et 3.9 montre ceci :

FIGURE 3.13 – affichage de la liste des chutes

FIGURE 3.14 – Utlisation des chutes

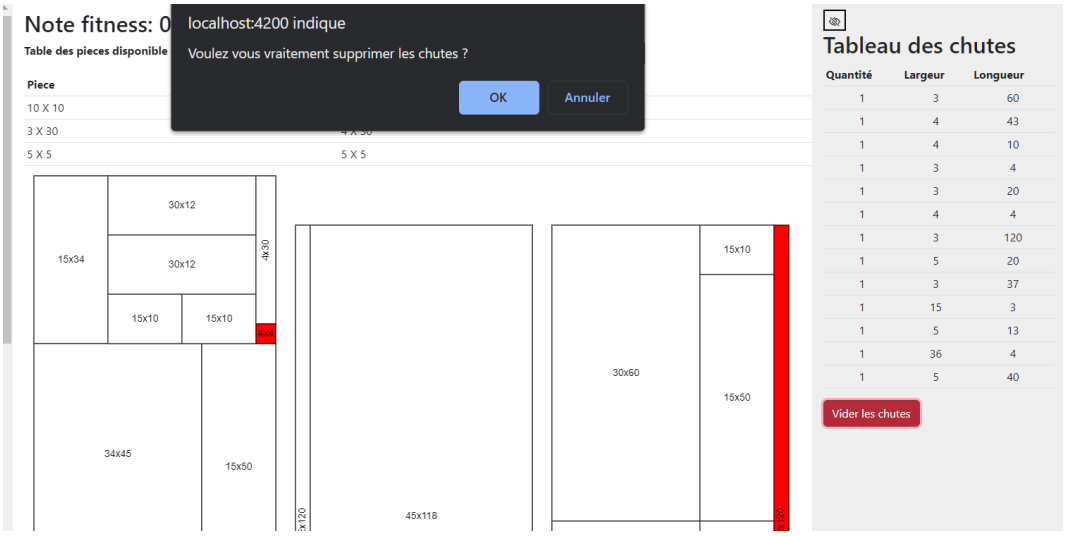


FIGURE 3.15 – supprission des chutes

3.5 Conclusion

Dans ce chapitre, nous avons présenté la méthodologie que nous avons adoptée pour mener à bien notre projet, ainsi que l'analyse des besoins. Puis nous avons présenté l'architecture du système, notre problématique ainsi que les outils de développement. Enfin, nous avons exposé les différentes interfaces et fonctions de notre application.

Conclusion générale & perspectives

Ce travail a été réalisé dans le but de répondre à notre problématique ; Comment découper un nombre donné d'éléments de manière à utiliser une quantité minimale des feuilles, et une quantité maximale de chute ? Ainsi, Nous avons fait une étude sur les problèmes combinatoires et les solutions existantes et potentiellement intéressantes pour notre problème . Cette étude nous a permis d'adopter une solution à base d'un algorithme génétique dont nous avons fait la conception et l'implémentation.

Bien qu'il existe déjà des algorithmes exacts pour résoudre des problèmes issus de la découpe de matière, ceux-ci ne sont pas viables pour des problèmes complexes, surtout si la taille des problèmes à résoudre est relativement importante. Cependant, l'application d'algorithmes génétiques dans ce cas est aussi faisable que pour tout autre problème d'optimisation NP-Complet , et ce n'est pas seulement faisable, mais dans certains cas même souhaitable. Les différents algorithmes exacts existants pour la découpe de matériau sont des algorithmes spécifiques, conçus pour un problème spécifique, généralement de petits problèmes, tandis qu'un algorithme génétique est facilement adaptable et est utilisé pour toute variante du problème d'optimisation tant qu'il peut être commodément transformé.

Bien que les résultats obtenus ne soient pas toujours optimaux, ils sont considérablement bons pour atteindre les objectifs dans un temps de calcul admissible, ce qui est l'objectif de l'application de techniques métaheuristiques telles que celle utilisée pour résoudre le problème dans ce travail. Cependant, comme il s'agit d'un algorithme à tout moment, une meilleure solution peut généralement être attendue si l'algorithme dispose de plus de temps.

Nous avons acquis une expérience technique avec la maîtrise d'un ensemble de framework et de langages. Les bénéfices de cette expérience se sont également étalés sur le plan social, en effet vu les résultats obtenus, on peut voir qu'il s'agit d'un processus non déterministe, ce qui signifie que le fait que la solution optimale soit trouvée ou que la population évolue vers des individus relativement bien adaptés est un processus aléatoire , qui lui seul est guidé par la fonction objectif, de sorte qu'à chaque exécution de l'algorithme, une solution différente est obtenue, et vu q'un travail n'est jamais achevé totalement, il y a toujours une possibilité d'amélioration. C'est pourquoi, en perspective, nous pouvons dire que le travail réalisé dans ce projet peut être amélioré par : -Une extension possible de l'algorithme qui améliore les solutions obtenues en prenant l'aspect mentionné en sa faveur est de convertir l'algorithme génétique en un algorithme génétique parallèle, qui est capable de partir d'un certain nombre de populations distinctes (îles) et de les rejoindre lorsque ils convergent, pour assurer une plus grande divergence de population lorsque les populations sont bloquées à un optimum local et élargir l'espace exploré en croisant des populations qui ont convergé

- Developper une platforme avec un système d'identification sur web où l'utlisateur peut gérer le stock des feuilles et des chutes à travers une base de donnée réelle.

Webographie

[1] B. Korte, J. Vygen, J. Fonlupt, and A. Skoda, "Optimisation combinatoire," 2010.

[2] M. Bellalouna, "Problèmes d'optimisation combinatoires probabilistes," 1993.

[3] A. Khanafer, "Algorithmes pour des problèmes de bin packing mono-et multi-objectif," 2010.

[4] J. Dipama, "Optimisation multi-objectif des systèmes énergétiques," 2010.

[5] M. A. Boschetti and A. Mingozzi, "The two-dimensional finite bin packing problem. part ii : New lower and upper bounds," 2003.

[6] H. I. Christensen, A. Khan, S. Pokutta, and P. Tetali, "Approximation and online algorithms for multidimensional bin packing : A survey," 2017.

[7] J.-M. Boussard, J.-J. Daudin, *et al.*, "La programmation linéaire dans les modèles de production," 1988.

[8] I. Devarenne, "Etudes en recherche locale adaptative pour l'optimisation combinatoire," 2007.

[9] R. W. Haessler and P. E. Sweeney, "Cutting stock problems and solution procedures," 1991.

[10] A. Vignier, J.-C. Billaut, and C. Proust, "Les problèmes d'ordonnancement de type «flow-shop» hybride : état de l'art," 1999.

[11] E. Mennas and L. Oussada, "Problème de découpe, application et résolution," 2019.

[12] Z. Degraeve and L. Schrage, "Optimal integer solutions to real-life cutting-stock problems," 1996.

[13] J. Antonio, "Les problèmes de placement : étude et résolution de quelques problèmes réels," 1997.

[14] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," 2007.

[15] K. Lai and J. W. Chan, "Developing a simulated annealing algorithm for the cutting stock problem," 1997.

[16] S. Aboumsabah and A. Baba-Ali, "La résolution de problème de bin packing rectangulaire avec contraintes en deux étapes génétiques hybrides," 2004.

[17] X. Song, "Problèmes de découpe : études et applications," 2004.

[18] J. Bennell, L.-S. Lee, and C. Potts, "A genetic algorithm for two-dimensional bin packing problem to minimise the maximum lateness," 2005.

[19] E. Hopper and B. Turton, "A genetic algorithm for a 2d industrial packing problem," 1999.

[20] J. A. Bennell, L. S. Lee, and C. N. Potts, "A genetic algorithm for two-dimensional bin packing with due dates," 2013.

[21] I. Sánchez Rodríguez, "Aplicación de algoritmos genéticos para la optimización del corte de material," 2016.

[22] "Genitic algorithm for the 2d packing problem." https://sci-hub.se/https://www.sciencedirect.com/science/article/abs/pii/S0925527313002041/.

[23] "html : définition simple et détaillée." https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1203255-html-hypertext-markup-langage-definition-traduction/.

[24] "Les avantages et les inconvénients de l'adoption de typescript." https://fr.quish.tv/benefits-drawbacks-adopting-typescript.

[25] "Angular : définition." https://cynoteck.com/fr/blog-post/reasons-to-use-angular-for-yourapp/#:~:text=Le%20principal%20avantage%20de%20AngularJS,applications%20d'une%20seule%20page/.

[26] "Visual studio code — wikipédia." https://fr.wikipedia.org/wiki/Visual_Studio_Code. (Accessed on 25/06/2022).

[27] "Qu'est-ce que npm ?." https://devstory.net/11925/qu-est-ce-que-npm. (Accessed on 23/06/2022).

[28] https://www.modelio.org. (Accessed on 20/06/2022).

Résumé

Le problème de la découpe de matière consiste, étant donné une pièce de matière de taille standard, à obtenir les différentes pièces souhaitées par coupes successives, à minimiser la matière inutile rejetée, appelée déchet, ou à maximiser le nombre de pièces pouvant être obtenues à la fois. temps de la pièce d'origine. C'est un problème d'optimisation réaliste. l'optimisation combinatoire, parfois multicritères, qui appartient à l'ensemble des problèmes dits "couper et emballer", et dont la solution optimisée nécessite souvent l'application de techniques métaheuristiques. Ce projet couvre la résolution du problème par le développement, la mise en œuvre d'un algorithme génétique de nature à tout moment, capable d'obtenir et améliorer successivement la solution atteinte jusqu'à ce qu'elle soit réalisab

Mots-clés : Optimisation, algorithmes, génétique, découpage, matériau, découpe,bin packing 2D.

Abstract

The cutting stock problem consists in, given a standard-sized sheet of material, obtain all of the desired pieces making successive cuts, minimizing wasted material, known as scrap, waste or trim loss ; or maximizing the number of pieces that can be obtained from the original sheet. It is a realistic combinatorial optimization problem, sometimes multi-objective optimization that belongs to the set of problems known as "cutting and packing problems" and which optimized solution often requires the use of metaheuristic techniques. This project covers the resolution of the problem by developing, implementing a genetic anytime algorithm able to obtain and improve successively the solution until it is viable.

Keywords : optimization, genetic, algorithm, cutting, stock,2D bin packing .

ملخص

تتمثل مشكلة مخزون القطع، نظرًا لورقة من المواد بالحجم القياسي، في الحصول على جميع القطع المطلوبة لإجراء تخفيضات متتالية، وتقليل المواد المهدرة، والمعروفة باسم الخردة، والنفايات. أو فقدان القطع ؛ أو زيادة عدد القطع التي يمكن الحصول عليها من الورقة الأصلية. إنها مشكلة واقعية لتحسين التوافقية، وأحياناًا تحسين متعدد الأهداف. تنتمي إلى مجموعة من المشاكل المعروفة باسم ≫مشاكل القطع والتعبئة≪ والتي يتطلب حلها الأمثل في كثير من الأحيان استخدام تقنيات ميتاهويستية. يغطي هذا المشروع حل المشكلة من خلال تطوير وتنفيذ خوارزمية وراثية في أي وقت قادرة على الحصول على الحل وتحسينه على التوالي حتى يصبح قابلاً للتطبيق.

لكلمات الرئيسية:تحسين ، خوارزمية وراثية،، قطع، تعبئة