Table des matières

1 Étude bibliographique

2 A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems[1]

Objectif de l'article : Présenter différents algorithmes, constructifs ou non, ainsi que plusieurs heuristiques en vue de résoudre le TDC (expliqué après).

Problème considéré : *Two-dimensional cutting problem* = **une** grande feuille de papier et un ensemble de pièces **rectangulaires** à découper. Chaque pièce a une **valeur** et il s'agit de **maximiser** la valeur totale découpée. À la différence avec notre problème, il n'y a **ni défauts, ni contrainte de tailles, ni contraintes d'ordre**... Le problème est donc différent. De plus, les différentes pièces ont encore une fois des offres et des demandes différentes. Toutefois, on dispose des **contraintes guillotines** (notons que les pièces ne peuvent pas roter ici) (la rotation semble être une liberté très importante).

Les TDC se classent en quatre catégories :

1. *The unconstrained unweighted version* (UU_TDC) : la valeur d'une pièce est égale à sa surface. On cherche à maximiser la surface utilisée (ce que nous on veut, d'une certaine manière).
2. *The unconstrained weighted version* (UW_TDC) : Chaque pièce a une certaine valeur, indépendante de sa surface (a priori). Pas ce qu'on veut.
3. *The constrained unweighted version* (CU_TDC) : On contraint sur les quantités à produire de chaque pièce. Il s'agit encore plus de notre problème puis-ce qu'on impose ici de couper 1 fois chaque pièce.
4. *The constrained weighted version*

— Des algorithmes exacts existent pour des petites et moyennes instances. Pour les autres, il y a des heuristiques : l'une d'elle se base sur une recherche en profondeur d'abord suivie d'une *hill climbing strategy* (algorithme DH)

A constructive heuristic algorithm : L'idée est de considérer, pour chaque sous-rectangle à découper toutes les pièces possibles.

— 1. Pour une pièce placée dans le coin inférieur gauche, une guillotine va faire deux nouvelles pièces. On calcule pour chacune d'entre elle une borne supérieure de ce qu'on pourrait y mettre en résolvant un problème du sac-à-dos. L'idée étant de faire en sorte de pouvoir en mettre un maximum.

— Une autre idée pour obtenir une autre borne sup (non comprise)

3 Remarques prélimininaires sur le sujet

3.1 Définitions

— Un **jumbo** est une grande fenêtre. Ils sont ordonnés dans un **bin**.
— Une **fenêtre/vitre/item** est une petite fenêtre.
— Les **fenêtres** sont ordonnées au sein de stacks (ordre d'extraction strict) qui eux ne sont pas ordonnés.
— Peut-on commencer par un Tooceuht ?

Références

[1] R. Alvarez-Valdés, A. Parajón, et al. A tabu search algorithm for large-scale guillotine (un) constrained two-dimensional cutting problems. *Computers & Operations Research*, 29(7) :925–947, 2002.