Etude de cas Optimisation 2

Adam DJAROUD ENSIIE

Décembre 2025

1 Le code

Le code est divisé en trois parties : une phase de prétraitement, l'implémentation du pl puis la résolution.

2 Analyse Structurelle et Pré-traitement

Avant d'initier la résolution, une phase de pré-traitement est effectuée via la fonction `analyser_structure`. Lors de cette phase, l'identification des ponts est réalisée : les arêtes dont la suppression déconnecte le graphe sont identifiées. Ces arêtes, appartenant nécessairement à tout arbre couvrant, sont injectées comme contraintes forcées dès le départ. Ensuite, un partitionnement des sommets est effectué : les sommets sont segmentés en deux ensembles. Le premier ensemble, noté V_{01}, regroupe les sommets de degré initial $d_G(v) \leq 2$, qui ne peuvent jamais devenir des sommets de branchement. Le second ensemble, noté V_2, inclut les sommets de degré $d_G(v) > 2$, qui sont des candidats potentiels au branchement.

3 Le PL

À chaque itération, nous résolvons un programme linéaire en nombres entiers sur un graphe qui a été modifié par la boucle décrite ci-dessus et le ce même PL.

4 La Boucle Itérative

La contrainte de connexité globale est traitée de manière dynamique par une boucle `while` qui analyse la solution T fournie par le solveur.

4.1 Choix du Quota

Le paramètre *n_quota* détermine le nombre d'arêtes forcées pour relier les composantes isolées. Il est initialisé selon la racine carrée de la taille du graphe

:

$$n\_quota = \max\left(1, \left\lfloor \frac{\sqrt{|V|}}{2} \right\rfloor\right) \tag{1}$$

Il est choisie ainsi pour être proportionnel à la taille du graphe et rendre l'algorithme plus efficace.

4.2 Processus de Reconnexion

Si le graphe est déconnecté, nous identifions l'ensemble des arêtes *candidates* reliant deux composantes distinctes. Un échantillon de taille n_quota est prélevé aléatoirement et ajouté à l'ensemble des arêtes forcées pour l'itération suivante.

La sélection repose ensuite sur la condition `if` $c_u \neq c_v$. Si les deux sommets d'une arête résident dans des composantes distinctes, celle-ci est considérée comme une candidate idéale pour assurer la reconnexion du graphe lors de l'itération suivante. Enfin, ces arêtes candidates sont stockées sous forme de tuples triés afin de garantir l'unicité des liaisons et d'éliminer toute redondance structurelle dans la liste.

Les arêtes sélectionnées sont ensuite injectées comme contraintes strictes ($x_e = 1$) pour l'itération suivante. Étant donné que le programme linéaire impose un total fixe de $n - 1$ arêtes, l'intégration forcée de ces ponts oblige le solveur à désactiver d'autres liaisons internes moins performantes.