ENSTA Paris

Année 2020-2021

Projet RO203: Loopy & Bridge

Étudiants
Yoldoz TABEI
Qingyao WANG

Encadrant
Marc ETHEVE

Table des matières

Table des figures

1 Modélisation

1.1 Loopy

Input

— t : Int (n*n) -1 si nul, [0,3] qui indique le nombre de côtés de cette cellule dans la boucle.

Variables :

— x : Booléen de taille (n*n+1) qui représente les côtés verticaux, 1 s'il est dans la boucle, 0 sinon.

— y : Booléen de taille (n+1*n) qui représente les côtés horizontaux, 1 s'il est dans la boucle, 0 sinon.

— p : Booléen de taille (n+1*n+1) représente les nœuds , 1 si la boucle passe par ce nœud, 0 sinon.

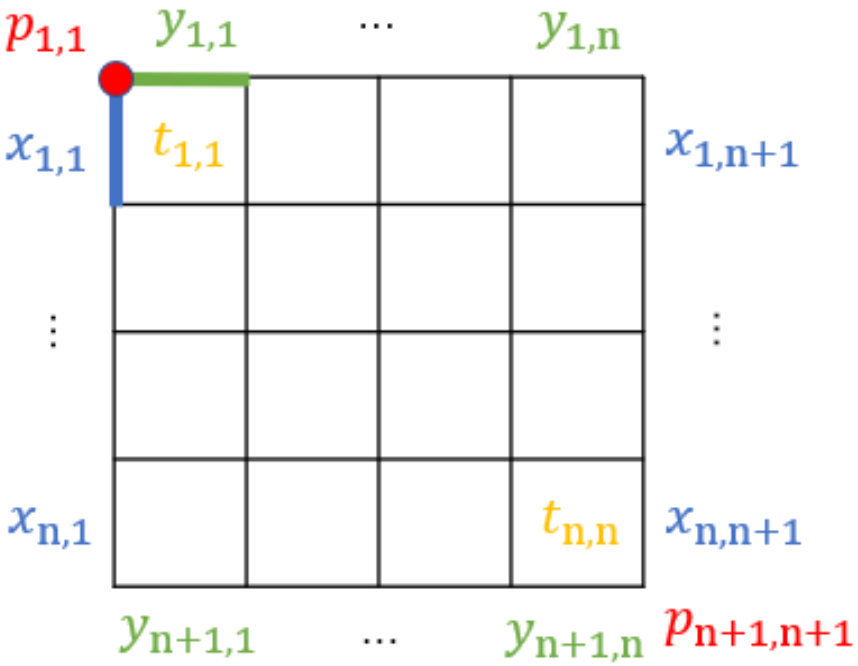

FIGURE 1 – Présentation des Variables

Contraintes :

— Contrainte sur les valeur des variables

$\forall i, j\ x_{i,j}, y_{i,j}, p_{i,j} \in \{0, 1\}$

— Contrainte sur la cellule : Le nombre des côtés par lesquels la boucle passe est égal au chiffre de cette cellule

$$\forall i, j \in [1, n]\ \mathrm{x}_{i,j} + x_{i,j+1} + y_{i,j} + y_{i+1,j} = t_{i,j}$$

— Contrainte sur le nœud : Soit la boucle passe par tous les noeuds soit non

$$\forall i, j \in [1, n+1]\ \mathrm{x}_{i-1,j} + x_{i,j} + y_{i,j-1} + y_{i,j} = 2{*}p_{i,j}$$

— Contrainte sur la boucle : Il existe une seule boucle, s'il y a sous-boucle, la somme des côtés est égale au nombre des nœuds dans cette sous-boucle. On doit l'éliminer.

$$\sum_{(i,j)\in S} \mathrm{x}_{i-1,j} + x_{i,j} + y_{i,j-1} + y_{i,j} \leq |S| - 1$$
$$\forall S \subset V, 2 \leq |S| \leq \sum_{(i,j)\in[1,n+1]} \mathrm{p}_{i,j}\ \text{-}1$$

Objective :

$$\mathrm{z} = \mathrm{Min}\Big(\sum_{i,j\in[1,n+1]} \mathrm{p}_{i,j}\Big)$$

1.2 Bridge

Le bridge est un jeu qui, à partir d'un nombre de lignes et de colonnes fourni par le joueur, fournit une grille contenant des noeuds numérotés. Le but de ce jeu est de créer des ponts entre ces noeuds de sorte à ce que la somme des ponts partant d'un noeus est égale au nombre indiqué sur ce noeuds.
Il doit également exister un chemin entre toute paire de noeuds et que les ponts ne s'intersectent pas. Le nombre maximal de ponts sortant d'un noeud dans une direction spécifique est égal à 2.

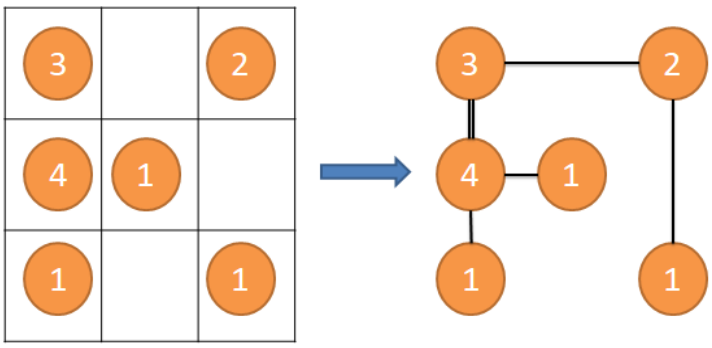

FIGURE 2 – Exemple de grille initiale et résolue

Soit le graph G=(V,E) ; V l'ensemble des noeuds et E l'ensemble des ponts.
Soit $\delta(i)$ l'ensemble des voisins du noeud i.

les variables de notre modèle :

- $x_{i,j}$ le nombre de ponts entre les noeuds i et j
- $y_{i,j}$ si les noeuds i et j sont connectés, et 0 sinon
- d_i le nombre de ponts partant de i

Les contraintes :

$$\sum_{i\in\delta(k),i<k} \mathrm{x}_{i,k}+\sum_{j\in\delta(k),k<j} \mathrm{x}_{k,j} = \mathrm{d}_k \ , \forall k \in V(1)$$

$$x_{i,j} = x_{j,i} \text{ et } y_{i,j} = y_{j,i} \ , \forall i,j \in V(2)$$

$$y_{i,j} + y_{k,l} \leq 1, \forall (i,j) \bigcap (k,l) \neq \varnothing \ (3)$$

$$y_{i,j} \leq \mathrm{x}_{i,j} \leq 2 \ \mathrm{y}_{i,j} \ , \forall (i,j) \in E \ (4)$$

$$\sum_{i\in S, j\in V/S} \mathrm{y}_{i,j} \geq 1, S \subset V, 1 \leq |S| \leq n-1 (5)$$

$$\sum_{i,j\in[1:n]} \mathrm{y}_{i,j} \geq 2*(n-1), S \subset V, 1 \leq |S| \leq n-1 (6)$$

$$x_{i,j} \subset \{0,1,2\} (7)$$

$$y_{i,j} \subset \{0,1\} (8)$$

L'objective :

$$\mathrm{z}= \mathrm{Min}(\sum_{i,j\in[i,n]} \mathrm{y}_{i,j})$$

Explication des contraintes :

Contrainte 1 : Le nombre de ponts sortant du noeud est égal au nombre du noeud.

Contrainte 2 : La symétrie des variables pour améliorer les performances.

Contrainte 3 : Le nombre de ponts sortant d'un noeuds dans une direction ne doit pas dépasser 2.

Contrainte 4 : Il n'y a pas d'intersection entre les ponts.

Contrainte 5 : Assure que la solution soit connectée.

Contrainte 6 : Assure que la solution soit connexe.

Contrainte 7 : Le nombre de ponts maximal entre 2 noeud est égal à 2.

2 Génération d'instances

2.1 Loopy

On utilise la méthode de coloration pour générer l'instance

— On colore des cellules connectées, et son contour fait la boucle.

— On note le nombre de côté pour chaque cellule.

— On tire par hasard les valeurs des cellules pour construire l'instance.

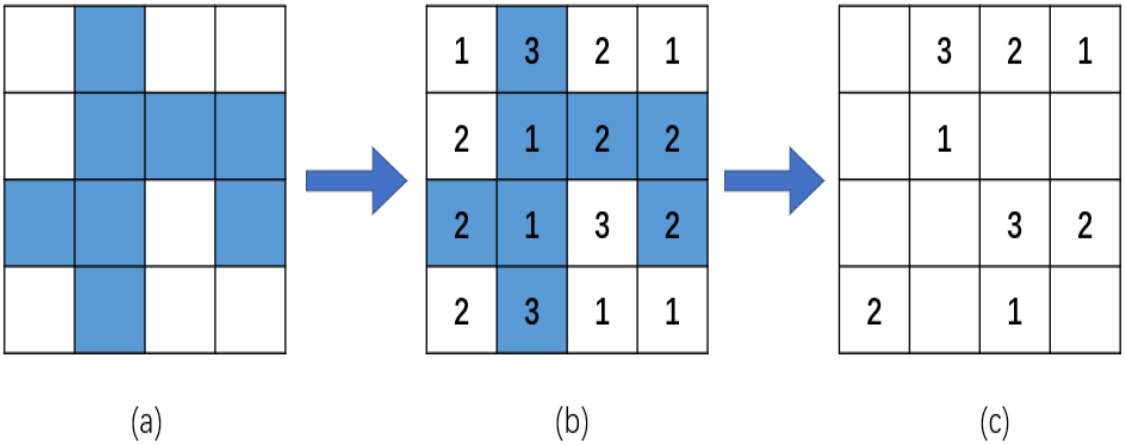

FIGURE 3 – Génération d'instance

```
julia> t=generateInstance(8, 0.6)
8×8 Array{Int64,2}:
 -1   0   2  2  -1  -1   2   0
  3  -1  -1  0   0  -1   3  -1
 -1   1  -1  1   0  -1  -1  -1
  2  -1  -1  2  -1  -1  -1   3
 -1   1   0  2   1   1   3  -1
  0  -1   1  3  -1   0   3   1
  0   0   0  2  -1  -1   3   0
  0  -1   0  1   2   1  -1  -1
```

FIGURE 4 – la sortie de le fonction generateInstance

2.2 Bridge

On construit une fonction qui génère une instance et qui a comme entrée n le nombre de ligne, m le nombre de colonnes et la densité.
Tout d'abord, on choisit un point de départ aléatoirement et on le met dans une liste list_noeud.

Ensuite, on répète les étapes suivantes jusqu'à ce que le n_noeud(la taille de list_noeud) soit égal à n*m*densité :

- On tire un point dans la liste
- On choisit un voisin telque aucune des cellules qui les séparent soit marquée
- Pour le pont les séparant, on lui associe la valeur ceil.(Int, 2*rand())
- On ajoute le nouveau noeud dans list_noeud

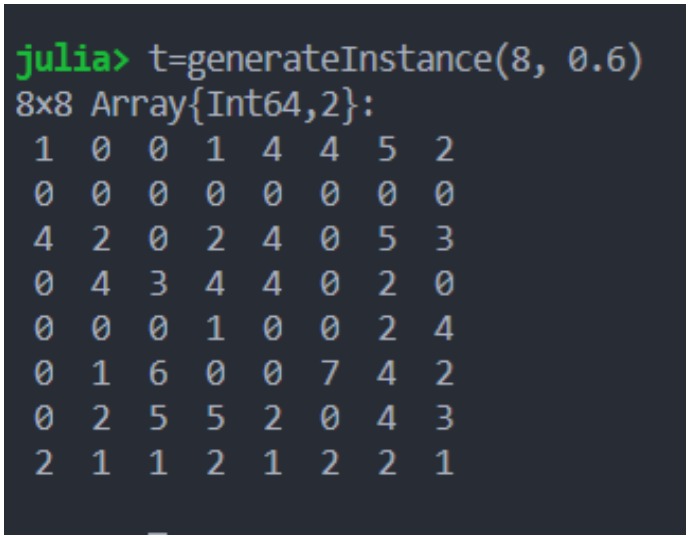

FIGURE 5 – la sortie de le fonction generateInstance

3 Solution cplex

3.1 Loopy

On utilise le Cplex pour résoudre ce problème.
Le 4eme contrainte a une complexité de 2^n, qui est tellement grande. Alors on utilise les callbacks pour qu'à chaque fois où on trouve une sous-boucle, on utilise la 4eme contrainte pour éliminer seulement la sous-boucle S trouvée.

$$\sum_{i \in S} x + \sum_{y \in S} y \leq |S| - 1$$

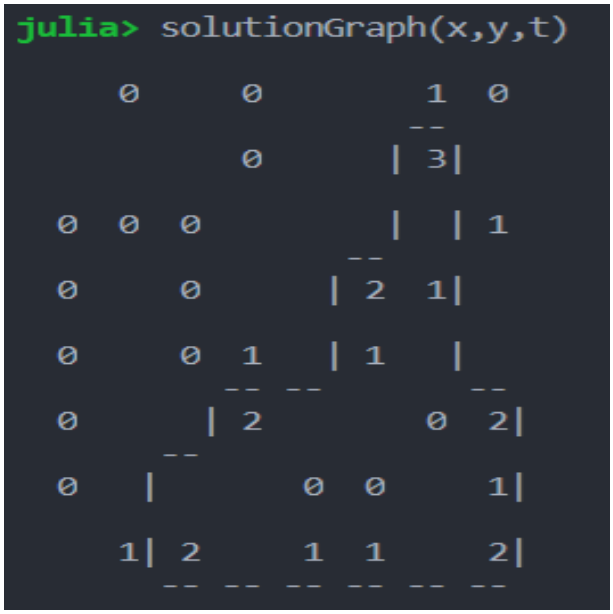

FIGURE 6 – la sortie de le fonction solutionGraph

3.2 Bridge

La 5ème contrainte du modèle est très complexe. Sa complexité est très grande. C'est pour cette raison qu'on utilise le callback. En effet, cette fonction permet de trouver 2 sous-ensembles de noeuds et assure qu'il y a au moins un pont entre eux et donc que le graphe soit connexe.

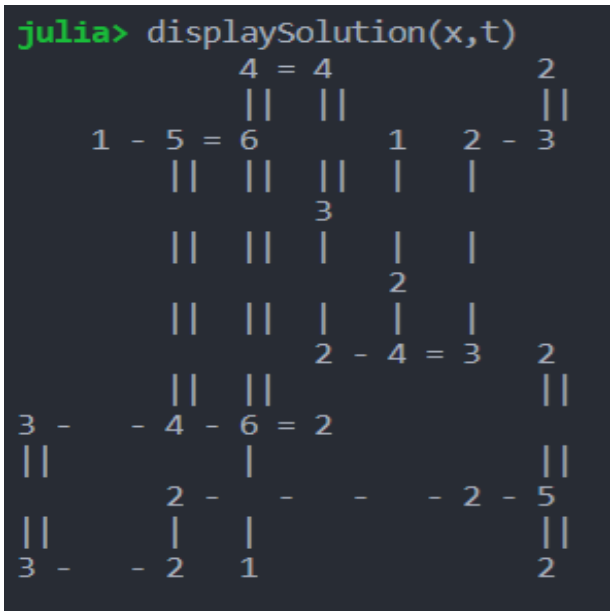

FIGURE 7 – la sortie de le fonction solutionGraph

Le tableau ci-dessous souligne le fait que le cplex est très rapide (Temps $<$ 1seconde) pour un nombre d'instances qui n'est pas grand.

	cplex	
Instance	**Temps (s)**	**Optimal ?**
instance_t12_d0.3_1.txt	0.01	
instance_t12_d0.3_10.txt	0.01	
instance_t12_d0.3_2.txt	0.17	×
instance_t12_d0.3_3.txt	0.01	
instance_t12_d0.3_4.txt	0.01	
instance_t12_d0.3_5.txt	0.01	
instance_t12_d0.3_6.txt	0.01	
instance_t12_d0.3_7.txt	0.01	
instance_t12_d0.3_8.txt	0.01	
instance_t12_d0.3_9.txt	0.08	×
instance_t12_d0.5_1.txt	0.02	
instance_t12_d0.5_10.txt	0.03	
instance_t12_d0.5_2.txt	0.12	×
instance_t12_d0.5_3.txt	0.01	
instance_t12_d0.5_4.txt	0.02	
instance_t12_d0.5_5.txt	0.47	×
instance_t12_d0.5_6.txt	0.01	
instance_t12_d0.5_7.txt	0.01	
instance_t12_d0.5_8.txt	0.18	×
instance_t12_d0.5_9.txt	0.01	
instance_t12_d0.6_1.txt	0.01	
instance_t12_d0.6_10.txt	0.02	
instance_t12_d0.6_2.txt	0.27	×
instance_t12_d0.6_3.txt	0.02	
instance_t12_d0.6_4.txt	0.01	
instance_t12_d0.6_5.txt	0.02	
instance_t12_d0.6_6.txt	0.02	
instance_t12_d0.6_7.txt	0.02	
instance_t12_d0.6_8.txt	0.02	

	cplex	
Instance	**Temps (s)**	**Optimal ?**
instance_t12_d0.6_9.txt	0.22	×
instance_t4_d0.3_1.txt	0.01	×
instance_t4_d0.3_10.txt	0.0	×
instance_t4_d0.3_2.txt	0.01	×
instance_t4_d0.3_3.txt	0.01	×
instance_t4_d0.3_4.txt	0.0	×
instance_t4_d0.3_5.txt	0.0	
instance_t4_d0.3_6.txt	0.0	×
instance_t4_d0.3_7.txt	0.0	×
instance_t4_d0.3_8.txt	0.0	×
instance_t4_d0.3_9.txt	0.0	×
instance_t4_d0.5_1.txt	0.0	×
instance_t4_d0.5_10.txt	0.0	×
instance_t4_d0.5_2.txt	0.0	×
instance_t4_d0.5_3.txt	0.0	×
instance_t4_d0.5_4.txt	0.01	×
instance_t4_d0.5_5.txt	0.0	×
instance_t4_d0.5_6.txt	0.01	×
instance_t4_d0.5_7.txt	0.01	×
instance_t4_d0.5_8.txt	0.01	×
instance_t4_d0.5_9.txt	0.0	×
instance_t4_d0.6_1.txt	0.02	×
instance_t4_d0.6_10.txt	0.0	
instance_t4_d0.6_2.txt	0.0	×
instance_t4_d0.6_3.txt	0.0	×
instance_t4_d0.6_4.txt	0.0	×
instance_t4_d0.6_5.txt	0.01	×
instance_t4_d0.6_6.txt	0.0	×
instance_t4_d0.6_7.txt	0.0	
instance_t4_d0.6_8.txt	0.0	×

	cplex	
Instance	**Temps (s)**	**Optimal ?**
instance_t4_d0.6_9.txt	0.01	×
instance_t7_d0.3_1.txt	0.0	
instance_t7_d0.3_10.txt	0.01	
instance_t7_d0.3_2.txt	0.08	×
instance_t7_d0.3_3.txt	0.01	
instance_t7_d0.3_4.txt	0.01	
instance_t7_d0.3_5.txt	0.02	×
instance_t7_d0.3_6.txt	0.0	
instance_t7_d0.3_7.txt	0.01	×
instance_t7_d0.3_8.txt	0.0	×
instance_t7_d0.3_9.txt	0.01	×
instance_t7_d0.5_1.txt	0.0	
instance_t7_d0.5_10.txt	0.03	×
instance_t7_d0.5_2.txt	0.04	×
instance_t7_d0.5_3.txt	0.02	×
instance_t7_d0.5_4.txt	0.01	
instance_t7_d0.5_5.txt	0.08	×
instance_t7_d0.5_6.txt	0.02	×
instance_t7_d0.5_7.txt	0.02	×
instance_t7_d0.5_8.txt	0.01	×
instance_t7_d0.5_9.txt	0.02	×
instance_t7_d0.6_1.txt	0.04	×
instance_t7_d0.6_10.txt	0.05	×
instance_t7_d0.6_2.txt	0.04	×
instance_t7_d0.6_3.txt	0.01	
instance_t7_d0.6_4.txt	0.03	×
instance_t7_d0.6_5.txt	0.0	
instance_t7_d0.6_6.txt	0.03	×
instance_t7_d0.6_7.txt	0.04	×
instance_t7_d0.6_8.txt	0.01	

	cplex	
Instance	**Temps (s)**	**Optimal ?**
instance_t7_d0.6_9.txt	0.0	

4 Solution heuristique

4.1 Loopy

On utilise l'algorithme de recherche approfondie pour résoudre ce problème.

— On cherche un nœud avec une seule branche.

— On choisit une parmi les branches possibles de ce nœud, et on va regarder cellule par cellule (figure3 :Fill By Cell) et puis point par point (figure4 :Fill By Point)si on peut déduire des valeurs des côtés.

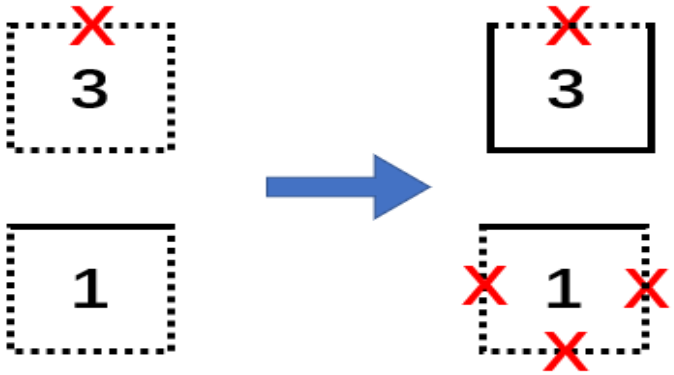

FIGURE 8 – Fill By Cell

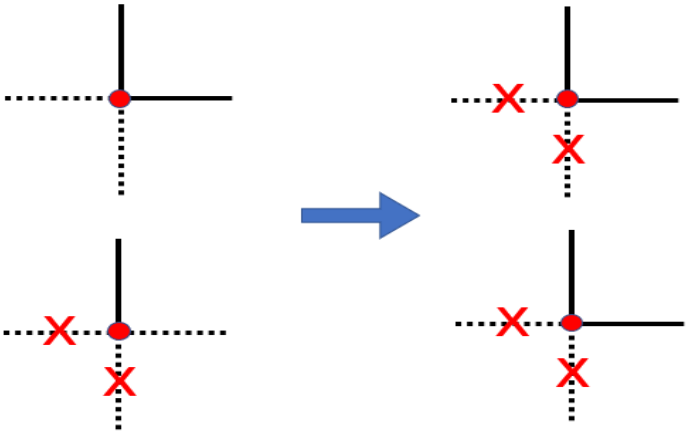

FIGURE 9 – Fill By Point

— S'il y a un nœud ou une cellule qui viole la contrainte, on note cette branche impossible. Et on fait des essais sur les autres points. Sinon, on le rajoute dans la boucle.

— Si toutes les contraintes sont vérifiées, on a réussi. Sinon on recommence ce processu.

Notice :
Au début, il n'y a peut-être pas de branche sur la grille, alors les essais commencent par les 4 côtés d'une cellule dont la valeur est la plus grande.

4.2 Bridge

Tout d'abord, on fait une copie de la grille, Cgrille sur laquelle on effectuera nos opérations.
On augmentant les contraintes de certains noeuds spéciaux (*) et on modifie Cgrille (à chaque fois q'un pont sort d'un noeud sa valeur diminue de 1).
Ensuite, après qu'elle ne soit plus améliorable, on prend un noeud au hasard, on étudie toutes les possibilités (tous les arbres possibles) en s'arrêtant lorsque Cgrille devienne un tableau de zeros et que tous les moeuds sont connectés(le nombre de ponts soit supérieur ou égal au nombre de noeuds).

(*) :
-une case de valeur 8 a forcément 2 ponts de chaque côté
-une case de valeur 7 a forcément 1 pont de chaque côté
-une case de valeur 6 sur un des 4 côtés de la grille a forcément 2 ponts de chaque côté
-les cases de valeurs 4, 5 sur un des côtés de la grille ont forcément 1 pont de chaque côté
-une case de valeur 3 sur un des 4 coins de la grille a forcément 1 pont de chaque côté
-une case de valeur 4 sur un des 4 coins de la grille a forcément 2 ponts de chaque côté

-une case de valeur 1 qui n'a qu'un voisin doit forcément se connecter à lui par 1 pont
-une case de valeur 2 qui n'a qu'un voisin doit forcément se connecter à lui par 2 ponts
-une case de valeur 3 qui n'a que 2 voisins doit forcément se connecter à eux par 1 pont chacun
-une case de valeur 4 qui n'a que 2 voisins doit forcément se connecter à eux par 2 ponts chacun
-une case de valeur 5 qui n'a que 3 voisins doit forcément se connecter à eux par 1 pont chacun
-une case de valeur 6 qui n'a que 3 voisins doit forcément se connecter à eux par 2 ponts chacun

-2 cases adjacentes de mêmes valeurs 1 ou 2 ne peuvent pas être connectés (sous-ensemble isolé du reste des noeuds)

5 Résultat

Ci-dessous le résultat du Loopy.

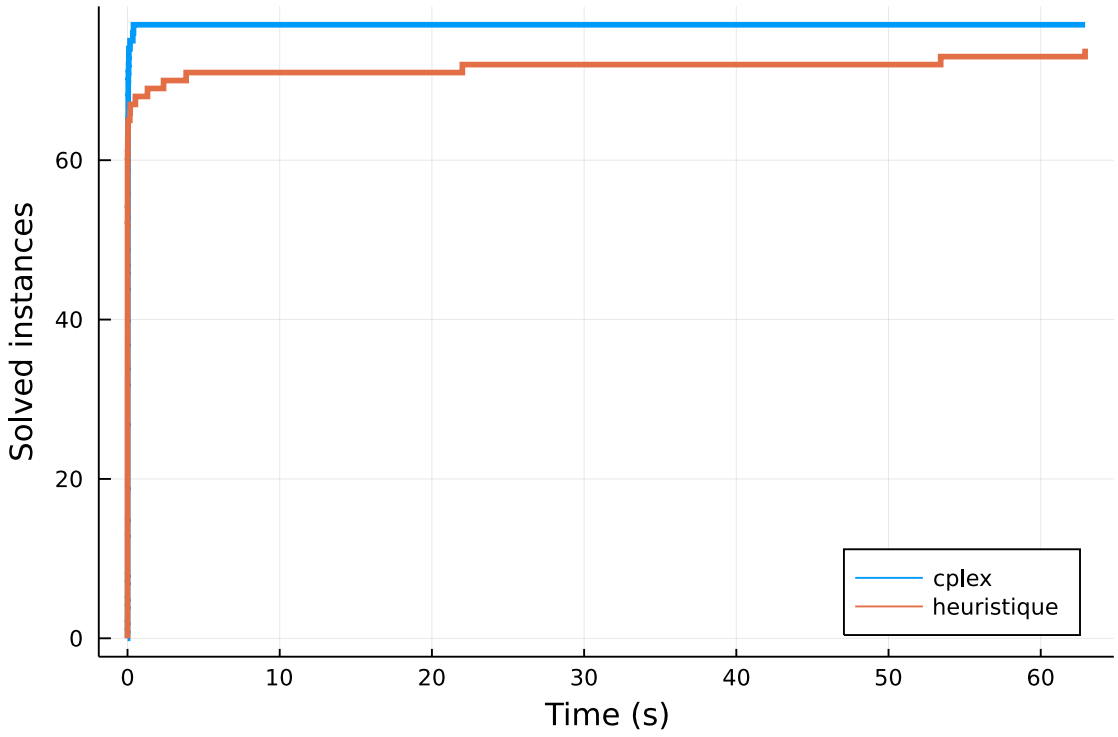

FIGURE 10 – Performance des 2 algorithmes

Nous pouvons tirer 3 remarques grâce à cette figure :

- Le cplex est mieux que l'algorithme heuristique puisqu'il résoud plus d'instances.
- Lorsque la taille est très grande, les 2 algorithmes ne peuvent pas résoudre le problème. Ceci est claire à travers les pentes de chacune des deux courbes (100 pour le cplex et 80 pour le heuristique).
- Lorsque la densité augmente, il devient plus facile de résoudre le problème car il y a plus de contraintes.

Le tableau ci-dessous montre que lorsque la taille est très petite, l'algorithme heuristique devient mieux que le cplex.

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t4_d0.4_1.txt	0.39	×	0.0	×
instance_t4_d0.4_10.txt	0.01	×	0.0	×
instance_t4_d0.4_2.txt	0.02	×	0.0	×
instance_t4_d0.4_3.txt	0.02	×	0.0	×
instance_t4_d0.4_4.txt	0.01	×	0.0	×
instance_t4_d0.4_5.txt	0.01	×	0.0	×
instance_t4_d0.4_6.txt	0.02	×	0.0	×
instance_t4_d0.4_7.txt	0.01	×	0.0	×
instance_t4_d0.4_8.txt	0.02	×	0.0	×
instance_t4_d0.4_9.txt	0.02	×	0.0	×
instance_t4_d0.6_1.txt	0.01	×	0.0	×
instance_t4_d0.6_10.txt	0.01	×	0.0	×
instance_t4_d0.6_2.txt	0.02	×	0.0	×
instance_t4_d0.6_3.txt	0.02	×	0.0	×
instance_t4_d0.6_4.txt	0.01	×	0.0	×
instance_t4_d0.6_5.txt	0.02	×	0.0	×
instance_t4_d0.6_6.txt	0.01	×	0.0	×
instance_t4_d0.6_7.txt	0.02	×	0.0	×
instance_t4_d0.6_8.txt	0.02	×	0.0	×
instance_t4_d0.6_9.txt	0.02	×	0.0	×
instance_t4_d0.7_1.txt	0.0	×	0.0	×
instance_t4_d0.7_10.txt	0.01	×	0.0	×
instance_t4_d0.7_2.txt	0.0	×	0.0	×
instance_t4_d0.7_3.txt	0.01	×	0.0	×
instance_t4_d0.7_4.txt	0.02	×	0.0	×
instance_t4_d0.7_5.txt	0.02	×	0.0	×
instance_t4_d0.7_6.txt	0.0	×	0.0	×
instance_t4_d0.7_7.txt	0.01	×	0.0	×
instance_t4_d0.7_8.txt	0.03	×	0.0	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t4_d0.7_9.txt	0.02	×	0.0	×
instance_t6_d0.4_1.txt	0.02	×	1.31	×
instance_t6_d0.4_10.txt	0.03	×	0.0	×
instance_t6_d0.4_2.txt	0.07	×	0.0	×
instance_t6_d0.4_3.txt	0.04	×	0.52	×
instance_t6_d0.4_4.txt	0.04	×	0.18	×
instance_t6_d0.4_5.txt	0.02	×	0.0	×
instance_t6_d0.4_6.txt	0.06	×	0.03	×
instance_t6_d0.4_7.txt	0.03	×	0.0	×
instance_t6_d0.4_8.txt	0.02	×	0.0	×
instance_t6_d0.4_9.txt	0.04	×	0.02	×
instance_t6_d0.6_1.txt	0.09	×	0.0	×
instance_t6_d0.6_10.txt	0.03	×	0.15	×
instance_t6_d0.6_2.txt	0.01	×	0.0	×
instance_t6_d0.6_3.txt	0.03	×	0.0	×
instance_t6_d0.6_4.txt	0.05	×	0.0	×
instance_t6_d0.6_5.txt	0.03	×	0.0	×
instance_t6_d0.6_6.txt	0.03	×	0.0	×
instance_t6_d0.6_7.txt	0.02	×	0.0	×
instance_t6_d0.6_8.txt	0.03	×	0.0	×
instance_t6_d0.6_9.txt	0.02	×	0.0	×
instance_t6_d0.7_1.txt	0.16	×	0.0	×
instance_t6_d0.7_10.txt	0.02	×	0.0	×
instance_t6_d0.7_2.txt	0.01	×	0.0	×
instance_t6_d0.7_3.txt	0.02	×	0.0	×
instance_t6_d0.7_4.txt	0.03	×	0.0	×
instance_t6_d0.7_5.txt	0.02	×	0.0	×
instance_t6_d0.7_6.txt	0.01	×	0.0	×
instance_t6_d0.7_7.txt	0.02	×	0.0	×
instance_t6_d0.7_8.txt	0.01	×	0.0	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t6_d0.7_9.txt	0.05	×	0.02	×
instance_t9_d0.4_1.txt	559.77		100.0	
instance_t9_d0.4_10.txt	0.02	×	100.0	
instance_t9_d0.4_2.txt	0.04	×	3.85	×
instance_t9_d0.4_3.txt	0.04	×	53.43	×
instance_t9_d0.4_4.txt	0.02	×	100.0	
instance_t9_d0.4_5.txt	0.04	×	0.0	×
instance_t9_d0.4_6.txt	0.03	×	0.0	×
instance_t9_d0.4_7.txt	0.0		100.0	
instance_t9_d0.4_8.txt	0.0		100.0	
instance_t9_d0.4_9.txt	0.08	×	0.0	×
instance_t9_d0.6_1.txt	0.04	×	100.0	
instance_t9_d0.6_10.txt	0.01	×	0.0	×
instance_t9_d0.6_2.txt	0.02	×	100.0	
instance_t9_d0.6_3.txt	165.13		62.92	×
instance_t9_d0.6_4.txt	0.0		100.0	
instance_t9_d0.6_5.txt	117.46		100.0	
instance_t9_d0.6_6.txt	0.01		100.0	
instance_t9_d0.6_7.txt	115.14		100.0	
instance_t9_d0.6_8.txt	0.03	×	0.04	×
instance_t9_d0.6_9.txt	185.12		100.0	
instance_t9_d0.7_1.txt	0.02	×	22.0	×
instance_t9_d0.7_10.txt	0.35	×	0.0	×
instance_t9_d0.7_2.txt	0.02	×	0.0	×
instance_t9_d0.7_3.txt	0.01		100.0	
instance_t9_d0.7_4.txt	0.03	×	0.0	×
instance_t9_d0.7_5.txt	0.02	×	0.0	×
instance_t9_d0.7_6.txt	0.0		100.0	
instance_t9_d0.7_7.txt	0.0		100.0	
instance_t9_d0.7_8.txt	0.01	×	2.37	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t9_d0.7_9.txt	155.76		100.0	
instance_t4_d0.4_1.txt	0.39	×	0.0	×
instance_t4_d0.4_10.txt	0.01	×	0.0	×
instance_t4_d0.4_2.txt	0.02	×	0.0	×
instance_t4_d0.4_3.txt	0.02	×	0.0	×
instance_t4_d0.4_4.txt	0.01	×	0.0	×
instance_t4_d0.4_5.txt	0.01	×	0.0	×
instance_t4_d0.4_6.txt	0.02	×	0.0	×
instance_t4_d0.4_7.txt	0.01	×	0.0	×
instance_t4_d0.4_8.txt	0.02	×	0.0	×
instance_t4_d0.4_9.txt	0.02	×	0.0	×
instance_t4_d0.6_1.txt	0.01	×	0.0	×
instance_t4_d0.6_10.txt	0.01	×	0.0	×
instance_t4_d0.6_2.txt	0.02	×	0.0	×
instance_t4_d0.6_3.txt	0.02	×	0.0	×
instance_t4_d0.6_4.txt	0.01	×	0.0	×
instance_t4_d0.6_5.txt	0.02	×	0.0	×
instance_t4_d0.6_6.txt	0.01	×	0.0	×
instance_t4_d0.6_7.txt	0.02	×	0.0	×
instance_t4_d0.6_8.txt	0.02	×	0.0	×
instance_t4_d0.6_9.txt	0.02	×	0.0	×
instance_t4_d0.7_1.txt	0.0	×	0.0	×
instance_t4_d0.7_10.txt	0.01	×	0.0	×
instance_t4_d0.7_2.txt	0.0	×	0.0	×
instance_t4_d0.7_3.txt	0.01	×	0.0	×
instance_t4_d0.7_4.txt	0.02	×	0.0	×
instance_t4_d0.7_5.txt	0.02	×	0.0	×
instance_t4_d0.7_6.txt	0.0	×	0.0	×
instance_t4_d0.7_7.txt	0.01	×	0.0	×
instance_t4_d0.7_8.txt	0.03	×	0.0	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t4_d0.7_9.txt	0.02	×	0.0	×
instance_t6_d0.4_1.txt	0.02	×	1.31	×
instance_t6_d0.4_10.txt	0.03	×	0.0	×
instance_t6_d0.4_2.txt	0.07	×	0.0	×
instance_t6_d0.4_3.txt	0.04	×	0.52	×
instance_t6_d0.4_4.txt	0.04	×	0.18	×
instance_t6_d0.4_5.txt	0.02	×	0.0	×
instance_t6_d0.4_6.txt	0.06	×	0.03	×
instance_t6_d0.4_7.txt	0.03	×	0.0	×
instance_t6_d0.4_8.txt	0.02	×	0.0	×
instance_t6_d0.4_9.txt	0.04	×	0.02	×
instance_t6_d0.6_1.txt	0.09	×	0.0	×
instance_t6_d0.6_10.txt	0.03	×	0.15	×
instance_t6_d0.6_2.txt	0.01	×	0.0	×
instance_t6_d0.6_3.txt	0.03	×	0.0	×
instance_t6_d0.6_4.txt	0.05	×	0.0	×
instance_t6_d0.6_5.txt	0.03	×	0.0	×
instance_t6_d0.6_6.txt	0.03	×	0.0	×
instance_t6_d0.6_7.txt	0.02	×	0.0	×
instance_t6_d0.6_8.txt	0.03	×	0.0	×
instance_t6_d0.6_9.txt	0.02	×	0.0	×
instance_t6_d0.7_1.txt	0.16	×	0.0	×
instance_t6_d0.7_10.txt	0.02	×	0.0	×
instance_t6_d0.7_2.txt	0.01	×	0.0	×
instance_t6_d0.7_3.txt	0.02	×	0.0	×
instance_t6_d0.7_4.txt	0.03	×	0.0	×
instance_t6_d0.7_5.txt	0.02	×	0.0	×
instance_t6_d0.7_6.txt	0.01	×	0.0	×
instance_t6_d0.7_7.txt	0.02	×	0.0	×
instance_t6_d0.7_8.txt	0.01	×	0.0	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t6_d0.7_9.txt	0.05	×	0.02	×
instance_t9_d0.4_1.txt	559.77		100.0	
instance_t9_d0.4_10.txt	0.02	×	100.0	
instance_t9_d0.4_2.txt	0.04	×	3.85	×
instance_t9_d0.4_3.txt	0.04	×	53.43	×
instance_t9_d0.4_4.txt	0.02	×	100.0	
instance_t9_d0.4_5.txt	0.04	×	0.0	×
instance_t9_d0.4_6.txt	0.03	×	0.0	×
instance_t9_d0.4_7.txt	0.0		100.0	
instance_t9_d0.4_8.txt	0.0		100.0	
instance_t9_d0.4_9.txt	0.08	×	0.0	×
instance_t9_d0.6_1.txt	0.04	×	100.0	
instance_t9_d0.6_10.txt	0.01	×	0.0	×
instance_t9_d0.6_2.txt	0.02	×	100.0	
instance_t9_d0.6_3.txt	165.13		62.92	×
instance_t9_d0.6_4.txt	0.0		100.0	
instance_t9_d0.6_5.txt	117.46		100.0	
instance_t9_d0.6_6.txt	0.01		100.0	
instance_t9_d0.6_7.txt	115.14		100.0	
instance_t9_d0.6_8.txt	0.03	×	0.04	×
instance_t9_d0.6_9.txt	185.12		100.0	
instance_t9_d0.7_1.txt	0.02	×	22.0	×
instance_t9_d0.7_10.txt	0.35	×	0.0	×
instance_t9_d0.7_2.txt	0.02	×	0.0	×
instance_t9_d0.7_3.txt	0.01		100.0	
instance_t9_d0.7_4.txt	0.03	×	0.0	×
instance_t9_d0.7_5.txt	0.02	×	0.0	×
instance_t9_d0.7_6.txt	0.0		100.0	
instance_t9_d0.7_7.txt	0.0		100.0	
instance_t9_d0.7_8.txt	0.01	×	2.37	×

	cplex		heuristique	
Instance	**Temps (s)**	**Optimal ?**	**Temps (s)**	**Optimal ?**
instance_t9_d0.7_9.txt	155.76		100.0	