Singles - Range

Abdelkarim Eljandoubi & Mouin Ben Ammar

Mai 2021

1 Singles

Nous avons une grille de carrés blancs, qui contiennent tous des nombres. Notre tâche consiste à colorer certains carrés en noir (en supprimant le nombre) afin de satisfaire toutes les conditions suivantes:

- Aucun nombre n'apparaît plus d'une fois dans une ligne ou une colonne. (C1)
- Aucun carré noir n'est adjacent horizontalement ou verticalement à un autre carré noir. (C2)
- Les carrés blancs restants doivent tous former une région contiguë (reliée par des bords, pas seulement en se touchant aux coins). (C3)

1.1 Programme Linéaire

On considère une grille n_l ligne et n_c colonne et $m = \max(n_l, n_c)$. On note $(I_{i,j}) \in [|1, m|]^{n_l \times n_c}$ respectivement $(F_{i,j}) \in [|0, m|]^{n_l \times n_c}$ (0 représente une case masquée), la matrice contenant les valeurs initiales respectivement finales de grille. Pour modéliser le problème il faut choisir les variables. On voit du condition (C1) qu'on doit compter le nombre de répétition d'un nombre sur les lignes ou les colonnes. Alors similairement au Sodoku, on pose $x_{i,j,k} = \mathbf{1}_{F_{i,j}=k}$. En adaptant cette modélisation, les valeurs résultant $R_{i,j} = I_{i,j}$ ou 0 (ou exclusive) car soit on masque la case soit on la laisse. Donc $\forall i \in [|1, n_l|]$ $\forall j \in [|1, n_c|]$, $x_{i,j,I_{i,j}} + x_{i,j,0} = 1$. C'est l'équivalent du $\sum_{k=1}^{n} x_{i,j,k} = 1$ au Sodoku mais dans notre cas on sait les k candidats possibles. la condition (C1) rassemble au celle du sodoku est s'écrit en deux contraints $\forall k \in [|1, m|]$ $\forall i \in [|1, n_l|]$, $\sum_{j=1}^{n_c} x_{i,j,k} \leqslant 1$ (Aucun nombre n'apparaît "plu"s d'une fois dans une ligne) et $\forall k \in [|1, m|]$ $\forall j \in [|1, n_c|]$, $\sum_{i=1}^{n_l} x_{i,j,k} \leqslant 1$ (Aucun nombre n'apparaît "plus" d'une fois dans une colonne). On peut reformuler (C2) en une implication si la case (i, j) est masquée ($x_{i,j,0} = 0$) alors ses quatre voisines (haut, bas, gauche et droit) doit être blanche ($x_{i,j,0} = 1$). Formellement, $\forall i \in [|1, n_l|]$ $\forall j \in [|1, n_c|]$, $x_{i,j,0} + \frac{1}{4}(\sum_{|p|=1, 1 \leqslant i+p \leqslant n_l} x_{i+p,j,0} + \sum_{|q|=1, 1 \leqslant j+q \leqslant n_c} x_{i,j+q,0}) \leq 1$ (assure parfaitement la condition (C2)). La contrainte (C3) est la plus difficile. Avoir une seule zone connexe des carrés blancs ça veut dire que les cases masquées (avec les bords) ne doivent pas former de boucle. Ou encore le nombre de cases sur une boucle donné est strictement inférieur au longueur de boucle. Puisque les boucles ne peut pas se former par des cases adjacents sur les cas direction à cause de contraint précédente alors la seule possibilité qu'elles se forment par des cases diagonalement adjacents. On définit $\mathcal{C}(n_l, n_c)$ comme l'ensemble des boucles "élémentaires" (les boucles les plus petites au sens de l'inclusion) de grille de taille $n_l \times n_c$ composées soit par des cases diagonalement adjacents soit par des cases diagonalement adjacents qui s'étendent jusqu'aux bords.Si on ouvre les cycles (boucle) élémentaire par au moins une case blanche, on garantit que les tous les autre cycles sont ouvertes et on réduit les nombre de contraintes. D'où $\forall C \in \mathcal{C}(n_l, n_c)$, $\sum_{(i,j) \in C} (1 - x_{i,j,0}) \geq 1$. En tant que jour fainéant, on veut faire le moins effort possible. Donc on choisit de masquer le moins des cases autrement l'objectif est $\min \sum_{1 \leq i \leq n_l, 1 \leqslant j \leqslant n_c} x_{i,j,0}$. On résume le système en :

$$\begin{cases} \min \sum_{1 \leq i \leq n_l, 1 \leqslant j \leqslant n_c} x_{i,j,0} \\ \forall i \in [|1, n_l|], \forall j \in [|1, n_c|], x_{i,j,I_{i,j}} + x_{i,j,0} = 1 \\ \forall k \in [|1, m|], \forall i \in [|1, n_l|], \sum_{j=1}^{n_c} x_{i,j,k} \leqslant 1 \\ \forall k \in [|1, m|], \forall j \in [|1, n_c|], \sum_{i=1}^{n_l} x_{i,j,k} \leqslant 1 \\ \forall i \in [|1, n_l|], \forall j \in [|1, n_c|], x_{i,j,0} + \frac{1}{4}(\sum_{|p|=1, 1 \leqslant i+p \leqslant n_l} x_{i+p,j,0} + \sum_{|q|=1, 1 \leqslant j+q \leqslant n_c} x_{i,j+q,0}) \leq 1 \\ \forall C \in \mathcal{C}(n_l, n_c), \sum_{(i,j) \in C} (1 - x_{i,j,0}) \geq 1 \end{cases} \tag{1}$$

La solution qui nous intéresse est la matrice $(x_{i,j,0})_{1 \leq i \leq n_l, 1 \leq j \leqslant n_c}$.

On peut implémenter simplement toutes les contraintes sauf la dernière qui fait l'objet de section suivante.

1.2 Générateur de cycles

Afin déterminer les éléments de $\mathcal{C}(n_l, n_c)$, on associe la grille à un graphe tel que la case (i, j) est associé à la sommet de numéro $(i-1).n_c + j$ et le domaine extérieur a le numéro $n_l.n_c + 1$. Chaque sommet $(i-1).n_c + j$ est liée par une arrête (non orientée) à la sommet $(i + p - 1).n_c + j + q$ avec $|p| = 1$ et $|q| = 1$ tel que $1 \leqslant i + p \leqslant n_l$ et $1 \leqslant j + q \leqslant n_c$. Si $i = 1$, $j = 1$, $i = n_l$ ou $j = n_c$ (ç-à-d la sommet est sur les bords), on relie par une arrête la sommet $(i - 1).n_c + j$ à la sommet $n_l.n_c + 1$ (l'extérieur).

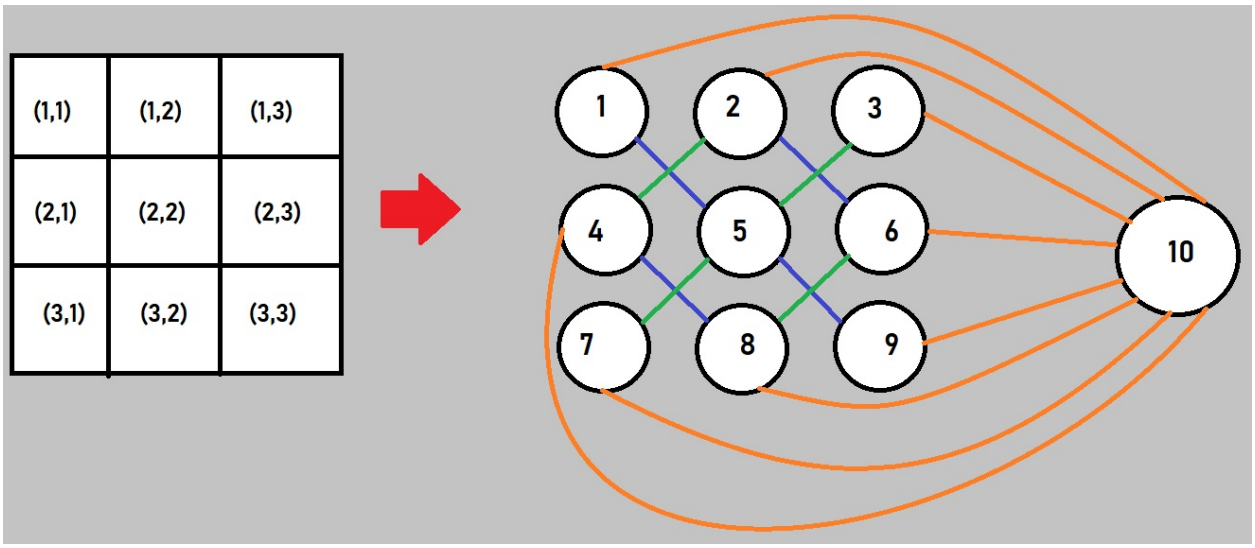

Figure 1: exemple pour une grille 3x3

Chercher les boucles élémentaires sur la grille est la même chose que chercher les cycles élémentaires sur la graphe associé avec une condition supplémentaire que ces cycles doivent passer par une sommet liée au $n_l.n_c$ au plus 2 fois.

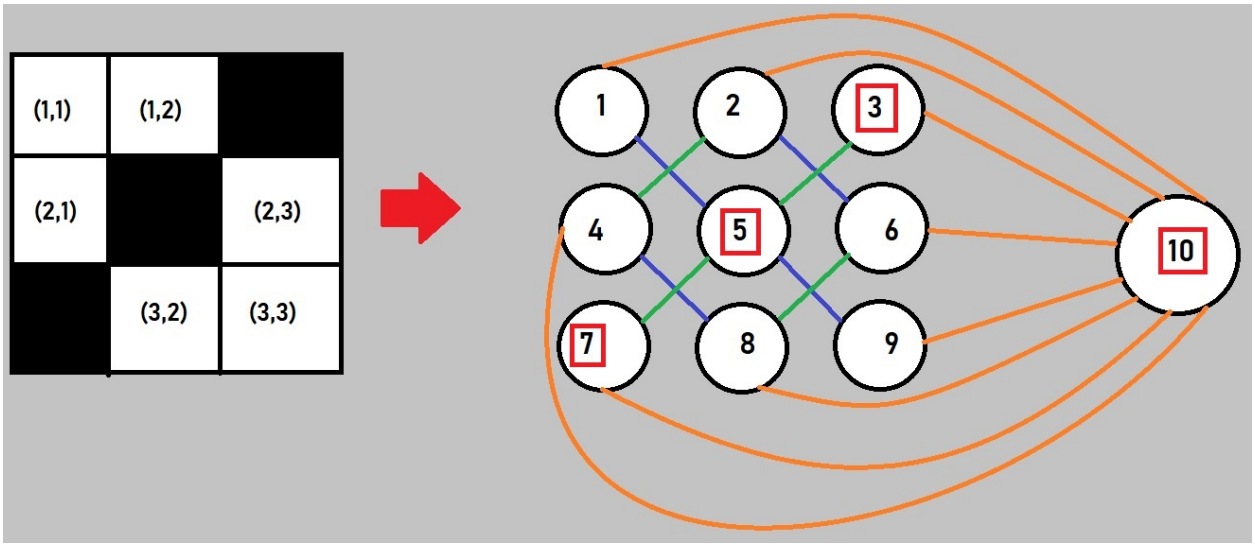

Figure 2: exemple de cycle sur une grille 3x3

Pour déterminer $\mathcal{C}(n_l, n_c)$, on a implémenter l'algorithme de parcours en profondeur puis l'adapter. Si au cours de parcours, on a retombé sur la sommet de départ on inscrit le cycle d'une ficher texte pour les réutiliser directement et ne pas refaire les mêmes calculs à chaque fois. Sinon on continue. Si la sommet actuel ne figure pas dans dans la liste des sommet de notre chaîne susceptible d'être une cycle et on recommence en considérant la nouvelle chaîne.

1.3 Heuristique

L'algorithme heuristique de résolution de l'instance à pour principe de tout d'abord mémoriser l'ensemble des indices des nombres répéter E et qui doivent être éliminer(on traite chaque nombre à part), puis de

choisir une permutation aléatoire de taille Cardinal(E)-1 , et de l'éliminer de la matrice , une fois que toute les colonnes et les lignes sont traiter de cette façon, on s'assure que la solution obtenue vérifie les contrainte imposer par le problème qui sont définis dans des fonctions à parts, si c'est pas le cas , on réitère l'algorithme si non l'instance obtenue est bien une solution. on mets une contrainte temporelle de 3 minute pour trouver la solution , si cette contrainte temporelle est dépasser , probablement il n'est pas capable de résoudre cette instance.

1.4 Générateur d'instance

La génération d'instance a pour but de remplier une matrice de dimension N_1 , N_2 choisie par l'utilisateur, par des nombres aléatoire choisis uniformément entre 1 et MaxN_1 , N_2

1.5 Performance

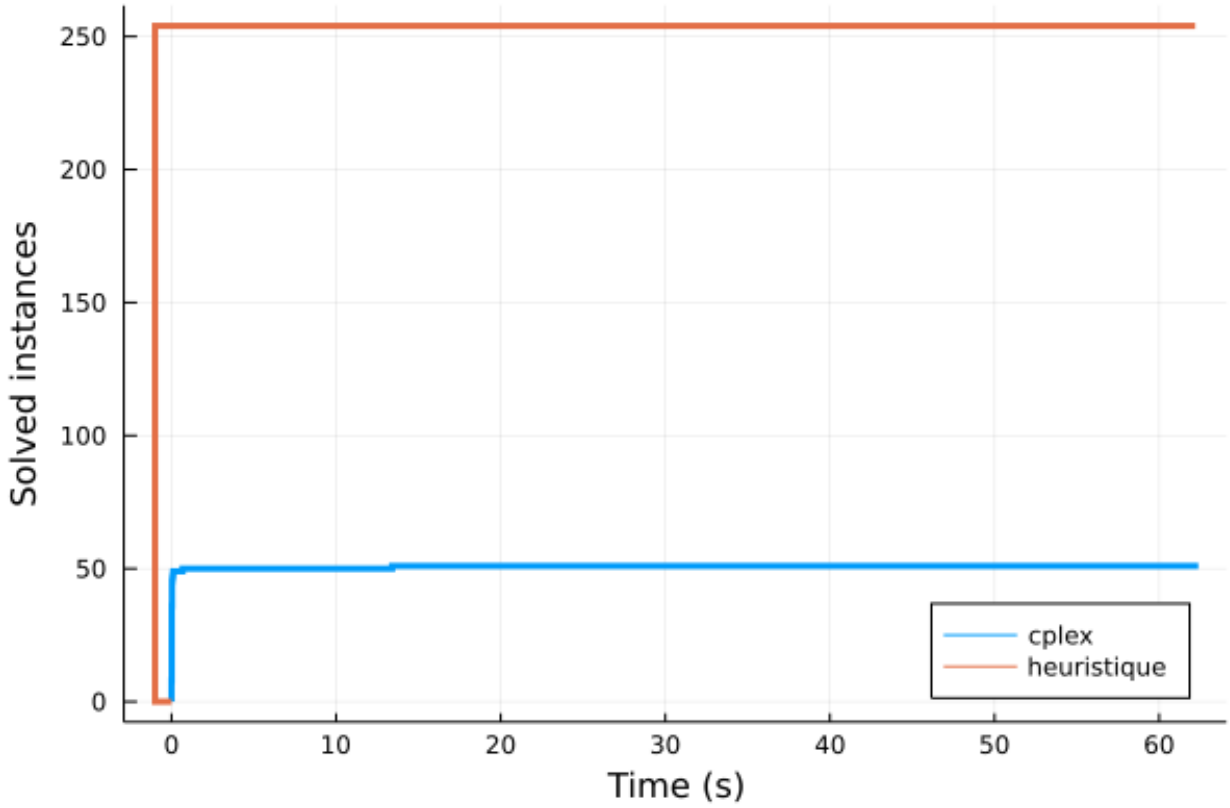

Figure 3: diagramme cplex-heuristique

On a met une contrainte de ne pas dépasser de 2s. C'est pour ça on observe que le heuristique fissent plus vite alors que en réalité elle n'assure pas l'optimalité (elle pas vraiment toujours une solution).

2 Range

Nous avons une grille de carrés tel que certains carrés contiennent des nombres. Notre travail consiste à colorer certains carrés en noir, de sorte que plusieurs critères soient satisfaits:

- aucun carré avec un nombre n'est coloré en noir.(C1)
- aucun carré avec un nombre n'est coloré en noir. il n'y a pas deux carrés noirs adjacents (horizontalement ou verticalement). (C2)
- pour deux carrés blancs quelconques, il y a un chemin entre eux en utilisant uniquement des carrés blancs. (C3)
- pour chaque carré avec un nombre, ce nombre indique le nombre total de carrés blancs accessibles à partir de ce carré en ligne droite dans n'importe quelle direction horizontale ou verticale jusqu'à toucher un mur ou un carré noir; le carré avec le nombre est inclus dans le total (une fois). (C4)

2.1 Programme Linéaire

On considère une grille n_l ligne et n_c colonne. On note $(T_{i,j}) \in [|1,m|]^{n_l \times n_c}$ tel que $T_{i,j}$ vaut 1 si la case (i,j) est blanche sans numéro sinon la numéro. Pour la modélisation, on pose $x_{i,j} = 1$ si la case (i,j) est masquée, 0 sinon. (C1) s'écrit simplement:$\forall i \in [|1,n_l|]\ \forall j \in [|1,n_c|]$ si $T_{i,j} > 1$, $x_{i,j} = 0$. Pour (C2) On a une contrainte identique pour le 1er jeu donc: $\forall i \in [|1,n_l|]\ \forall j \in [|1,n_c|]$, $x_{i,j} + \frac{1}{4}(\sum_{|p|=1,1\leqslant i+p\leqslant n_l} x_{i+p,j} + \sum_{|q|=1,1\leqslant j+q\leqslant n_c} x_{i,j+q}) \leq 1$. De même pour (C3): $\forall C \in \mathcal{C}(n_l,n_c)$, $\sum_{(i,j)\in C}(1-x_{i,j}) \geq 1$. Dans le but d'écrire (C4) en équation, on va introduire des variables servants à modélisé la notion de 2 cases se voient. Soit $z_{l,j,k} \in [|0,1|]^{n_l \times n_c \times n_c}$ tel que $z_{l,j,k} = 1$ si (l,j) et (l,k) se voient (2 cases sur la même lien l), 0 sinon. Soit $y_{i,k,c} \in [|0,1|]^{n_l \times n_l \times n_c}$ tel que $y_{i,k,c} = 1$ si (i,c) et (k,c) se voient (2 cases sur la même colonne), 0 sinon. Sur chaque ligne l, (l,j) et (l,k) se voient ssi il n'y a pas des cases masquées entre eux. Formellement: $\forall l \in [|1,n_l|]\ \forall j \in [|1,n_c|]\ \forall j \in [|1,n_c|]\ k \leqslant k$, $z_{l,j,k} = 1 \Leftrightarrow \sum_{i=j}^{k} x_{l,i} = 0$. La traduction de l'équivalent en des inégalités donne : $\forall l \in [|1,n_l|]\ \forall j \in [|1,n_c|]\ \forall k \in [|1,n_c|]\ j \leqslant k$, $1 \leqslant z_{l,j,k} + \sum_{i=j}^{k} x_{l,i}$ et $\forall l \in [|1,n_l|]\ \forall j \in [|1,n_c|]\ \forall k \in [|1,n_c|]\ j \leqslant k$, $z_{l,j,k} + \frac{1}{1+k-j}\sum_{i=j}^{k} x_{l,i} \leqslant 1$. De plus, (l,j) voit (l,k) ssi (l,k) voit (l,j) ou encore $\forall l \in [|1,n_l|]\ \forall j \in [|1,n_c|]\ \forall k \in [|1,n_c|]\ k < j$, $z_{l,j,k} = z_{l,k,j}$. De façon similaire sur $y_{i,k,c}$. D'où le système en masquant le moindre des cases:

$$\begin{cases} \min \sum_{1\leq i\leq n_l, 1\leq j\leqslant n_c} x_{i,j} \\ \forall i \in [|1,n_l|], \forall j \in [|1,n_c|], T_{i,j} > 1, x_{i,j} = 1 \\ \forall i \in [|1,n_l|], \forall j \in [|1,n_c|], x_{i,j,0} + \frac{1}{4}(\sum_{|p|=1,1\leqslant i+p\leqslant n_l} x_{i+p,j,0} + \sum_{|q|=1,1\leqslant j+q\leqslant n_c} x_{i,j+q,0}) \leq 1 \\ \forall C \in \mathcal{C}(n_l,n_c), \sum_{(i,j)\in C}(1-x_{i,j,0}) \geq 1 \\ \forall l \in [|1,n_l|], \forall j \in [|1,n_c|], \forall k \in [|1,n_c|], j \leqslant k, 1 \leqslant z_{l,j,k} + \sum_{i=j}^{k} x_{l,i} \\ \forall l \in [|1,n_l|], \forall j \in [|1,n_c|], \forall k \in [|1,n_c|], j \leqslant k, z_{l,j,k} + \frac{1}{1+k-j}\sum_{i=j}^{k} x_{l,i} \leqslant 1 \\ \forall l \in [|1,n_l|], \forall j \in [|1,n_c|], \forall k \in [|1,n_c|], k < j, z_{l,j,k} = z_{l,k,j} \\ \forall c \in [|1,n_c|], \forall i \in [|1,n_l|], \forall k \in [|1,n_l|], i \leqslant k, 1 \leqslant y_{i,k,c} + \sum_{j=i}^{k} x_{j,c} \\ \forall c \in [|1,n_c|], \forall i \in [|1,n_l|], \forall k \in [|1,n_l|], i \leqslant k, y_{i,k,c} + \frac{1}{1+k-i}\sum_{j=i}^{k} x_{j,c} \leqslant 1 \\ \forall c \in [|1,n_c|], \forall i \in [|1,n_l|], \forall k \in [|1,n_l|], i \leqslant k, k < i, y_{i,k,c} = y_{k,i,c} \\ \forall l \in [|1,n_l|], \forall c \in [|1,n_c|], T_{i,j} > 1, \sum_{i=1}^{n_l} y_{i,l,c} + \sum_{j=1}^{n_c} z_{l,c,j} - 1 = T_{l,c} \end{cases} \tag{2}$$

La dernière contrainte assure (C4).

2.2 Générateur d'instance

La génération d'instance a pour but de sélectionner quelques case aléatoirement à partir d'une matrice de dimension N_1 , N_2 choisie par l'utilisateur, puis de remplir ces case par des nombres aléatoire choisis uniformément entre 1 et MaxN_1 , N_2 , la densité des nombres est définie par le paramètre "Density" passer à la fonction generateInstance.

2.3 Performance

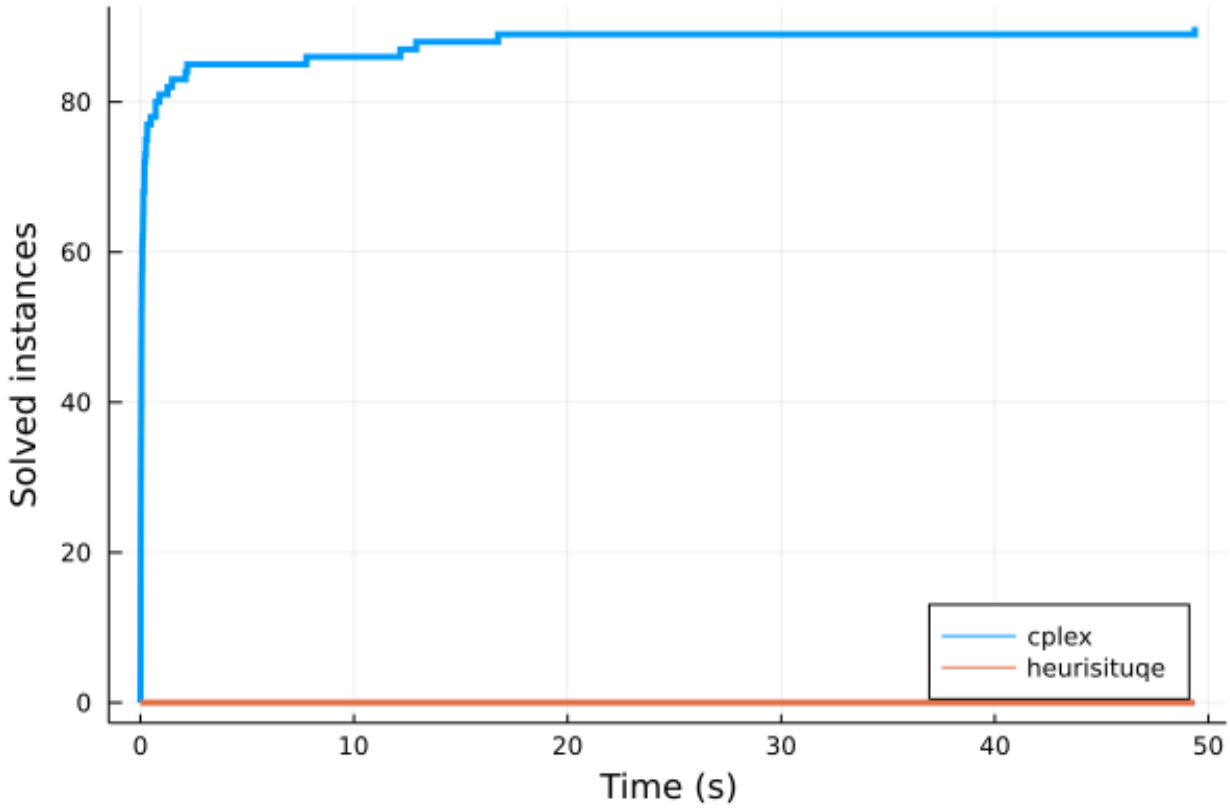

Figure 4: diagramme 2 cplex-heuristique

Heuristique vaut 0 car le dossier est vide.