Rapport du projet de programmation

Quentin MALLEGOL et Kwame MBOBDA-KUATE

10 mars 2024

1 Introduction

Ce rapport décrit les choix algorithmiques pris et les résultats obtenus dans le cadre du projet de programmation. Il présente d'abord les heuristiques et solveurs employés puis les analyse et les compare.

2 Définition

Dans toute la suite du rapport, n et m désignent deux entiers naturels non nuls. Une grille est un élément de l'ensemble G_{mn} des applications bijectives de $[\![0,\, mn-1]\!]$ dans lui-même. Soit $S^h_{m,n} = \{(i \quad i+1) \mid i \in [\![0,\, mn-1]\!],\ i \,\%\, n \neq n-1\}$ l'ensemble des échanges horizontaux possibles et $S^v_{m,n} = \{(i \quad i+n) \mid i \in [\![0,\, mn-1-n]\!]\}$ l'ensemble des échanges verticaux possibles puis $S_{m,n} = S^h_{m,n} \cup S^v_{m,n}$. Considérons les couples de nœuds adjacents $A_{m,n} = \{(\sigma, \sigma \circ s) \,|\, (\sigma, s) \in G_{mn} \times S_{m,n}\}$.
Le problème du *swap puzzle* consiste à, étant donnée deux grilles $\sigma, \tau \in G_{mn}$ à trouver un plus court chemin de τ à σ dans le graphe $(G_{mn}, A_{m,n})$, ou encore à trouver un n-uplet de permutations $(s_i)_{1 \leq i \leq n} \in S_{m,n}$ de taille minimale tel que $\sigma \circ s_1 \circ s_2 \circ ... \circ s_n = \tau$. Sous réserve d'existence d'un chemin, on pose alors $\delta(\sigma, \tau) = n$.

3 Heuristiques

3.1 Heuristiques naïves

Nous aborderons dans un premier temps les heuristiques « naïves », c'est-à-dire celles qui ne demandent pas de précalcul.

3.1.1 Distance de Manhattan

Definition 3.1. La distance de Manhattan est la fonction DM qui à un couple de grilles $(\sigma, \tau) \in G_{mn}^2$ associe

$$\mathrm{DM}(\sigma,\tau) = \sum_{k=0}^{mn-1} \left(\left| \left\lfloor \frac{\tau^{-1}(\sigma(k))}{n} \right\rfloor - \left\lfloor \frac{k}{n} \right\rfloor \right| + \left| (\tau^{-1}(\sigma(k)) \,\%\, n) - (k \,\%\, n) \right| \right)$$

.

Lemme 3.1. *La distance de Manhattan par rapport à une grille varie au plus de deux au cours d'un échange :*

$$\forall (\sigma,\tau) \in G_{mn}^2, \forall s \in S_{m,n}, |DM(\sigma,\tau) - DM(\sigma \circ s, \tau)| \leq 2$$

Démonstration. Soient $(\sigma, \tau) \in G_{mn}^2$ et $s \in S_{m,n}$. Il existe des entiers i et j avec $i < j$ tels que $s = (i \quad j)$.

$$\begin{aligned}
\mathrm{DM}(\sigma,\tau) - \mathrm{DM}(\sigma \circ s, \tau) = &\left| \left\lfloor \frac{\tau^{-1}(\sigma(i))}{n} \right\rfloor - \left\lfloor \frac{j}{n} \right\rfloor \right| + \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (j \,\%\, n) \right| \\
&+ \left| \left\lfloor \frac{\tau^{-1}(\sigma(j))}{n} \right\rfloor - \left\lfloor \frac{i}{n} \right\rfloor \right| + \left| (\tau^{-1}(\sigma(j)) \,\%\, n) - (i \,\%\, n) \right| \\
&- \left| \left\lfloor \frac{\tau^{-1}(\sigma(i))}{n} \right\rfloor - \left\lfloor \frac{i}{n} \right\rfloor \right| + \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) \right| \\
&- \left| \left\lfloor \frac{\tau^{-1}(\sigma(j))}{n} \right\rfloor - \left\lfloor \frac{j}{n} \right\rfloor \right| + \left| (\tau^{-1}(\sigma(j)) \,\%\, n) - (j \,\%\, n) \right|
\end{aligned}$$

Si $s \in S_{m,n}^h$, $j = i + 1$ et on a $\lfloor \frac{j}{n} \rfloor = \lfloor \frac{i}{n} \rfloor$ donc

$$\begin{aligned}
\mathrm{DM}(\sigma,\tau) - \mathrm{DM}(\sigma \circ s, \tau) = &\left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) - 1 \right| - \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) \right| \\
&+ \left| (\tau^{-1}(\sigma(i+1)) \,\%\, n) - (i \,\%\, n) \right| - \left| (\tau^{-1}(\sigma(i+1)) \,\%\, n) - (i \,\%\, n) - 1 \right|
\end{aligned}$$

D'après l'inégalité triangulaire,

$$\begin{aligned}
|\mathrm{DM}(\sigma,\tau) - \mathrm{DM}(\sigma \circ s, \tau)| &\leq \left| \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) - 1 \right| - \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) \right| \right| \\
&\quad + \left| \left| (\tau^{-1}(\sigma(i+1)) \,\%\, n) - (i \,\%\, n) \right| - \left| (\tau^{-1}(\sigma(i+1)) \,\%\, n) - (i \,\%\, n) - 1 \right| \right| \\
&\leq \left| (\tau^{-1}(\sigma(i)) \,\%\, n) - (i \,\%\, n) - 1 - (\tau^{-1}(\sigma(i)) \,\%\, n) + (i \,\%\, n) \right| \\
&\quad + \left| (\tau^{-1}(\sigma(i+1)) \,\%\, n) - (i \,\%\, n) - (\tau^{-1}(\sigma(i+1)) \,\%\, n) + (i \,\%\, n) - 1 \right| \\
&= 2.
\end{aligned}$$

Le cas où $s \in S_{m,n}^v$ se traite de manière analogue en remarquant qu'on a alors $j = i + n$ et donc $(i \,\%n) = (j \,\%n)$. □

Théorème 3.2. *Soit $\tau \in G_{m,n}$. Pour la recherche du plus court chemin jusqu'à τ, la fonction $\tau \in G_{m,n} \to \dfrac{DM(\sigma,\tau)}{2}$ est monotone (donc admissible) et $\tau \in G_{m,n} \to \left\lceil \dfrac{DM(\sigma,\tau)}{2} \right\rceil$ est admissible.*

Démonstration. La monotonie découle directement du lemme précédent et du fait que $\mathrm{DM}(\tau, \tau) = 0$. La longueur d'une solution étant toujours entière, $\tau \in G_{m,n} \to \left\lceil \frac{\mathrm{DM}(\sigma, \tau)}{2} \right\rceil$ est admissible. □

Conjecture 3.3. *Soit σ la permutation de $[\![0,\, mn-1]\!]$ définie par $\forall k \in [\![0,\, mn-1]\!], \sigma(k) = mn - 1 - k$. Alors $\forall \tau \in G_{mn}, DM(\tau, id) \leq DM(\sigma, id)$.*

Remarque. Le résultat a été vérifié numériquement pour $n \leq 3$ et $m \leq 3$ et est utilisé pour générer des grilles de difficulté contrôlée.

16	15	14	13
12	11	10	9
8	7	6	5
4	3	2	1

TABLE 1 – La permutation maximisant la distance de Manhattan en 4×4

3.2 Distance d'inversion

Definition 3.2. Soit $r \in G_{mn}$ la réflexion par rapport à la diagonale principale définie par $\forall (i,j) \in [\![0,\, m-1]\!] \times [\![0,\, n-1]\!]$, $r(ni+j) = mi+j$. Soit $\sigma \in G_{mn}$. On appelle inversion tout couple $(i,j) \in [\![0,\, mn-1]\!]^2$ tel que $i \leq j$ et $\sigma(i) \geq \sigma(j)$. On note $\mathrm{N}_h(\sigma)$ le nombre d'inversions de σ et $\mathrm{N}_v(\sigma)$ celui de $r \circ \sigma \circ r$ (c'est-à-dire la grille sur laquelle on a effectué une symétrie diagonale sur les indices et les nombres).

2	8	12	1
7	16	5	9
6	13	11	15
4	14	3	10

2	7	6	4
8	16	13	14
12	5	11	3
1	9	15	10

5	10	6	13
14	16	4	8
15	2	11	9
1	3	12	7

TABLE 2 – De gauche à droite : σ, $\sigma \circ r$ et $r \circ \sigma \circ r$

Lemme 3.4. *Effectuer un échange crée ou résout au plus $2n-1$ inversions horizontales et $2m-1$ inversions verticales :*

$$\forall \sigma \in G_{mn}, \forall s \in S^h_{m,n} | N_h(\sigma \circ s) - N_h(\sigma)| \leq 1$$

$$\forall \sigma \in G_{mn}, \forall s \in S^v_{m,n} | N_v(\sigma \circ s) - N_v(\sigma)| \leq 1$$

$$\forall \sigma \in G_{mn}, \forall s \in S^v_{m,n} | N_h(\sigma \circ s) - N_h(\sigma)| \leq 2n-1$$

$$\forall \sigma \in G_{mn}, \forall s \in S^h_{m,n} | N_v(\sigma \circ s) - N_v(\sigma)| \leq 2m-1$$

.

Démonstration. Par symétrie, nous ne traiterons que le cas horizontal. Soit $\sigma \in G_{mn}$ et $\forall s \in S_{m,n}$. Nous utilisons le résultat suivant : le nombre d'inversions de σ est l'entier minimal k tel qu'il existe une décomposition de σ en k transpositions de la forme $(j \quad j+1)$. Si $s \in S^h_{m,n}$ alors il existe un entier i tel que $s = (i \quad i+1)$ donc $N_h(\sigma \circ s) \leq N_h(\sigma)+1$. Sinon, il existe un entier i tel que $s = (i \quad i+n)$. Or, $(i \quad i+n) = (i \quad i+1)(i+1 \quad i+2)...(i+n-2 \quad i+n-1)(i+n-1 \quad i+n)(i+n-2 \quad i+n-1)...(i+1 \quad i)$ et comporte $2n-1$ termes donc $N_h(\sigma \circ s) \leq N_h(\sigma)+2n-1$. Le même raisonnement appliqué à $\sigma \circ s$ donne $N_h(\sigma) \leq N_h(\sigma \circ s) + 2n - 1$ si $s \in S^v_{m,n}$ et $N_h(\sigma) \leq N_h(\sigma \circ s) + 1$ si $s \in S^h_{m,n}$ (puisque $s^2 = \mathrm{id}$) ; d'où le résultat. □

Théorème 3.5. *Pour la recherche du plus court chemin jusqu'à id, la fonction* $\dfrac{N_h + N_v}{2\max(n,m)}$ *est monotone (donc admissible) et* $\left\lceil \dfrac{N_h + N_v}{2\max(n,m)} \right\rceil$ *est admissible.*

Démonstration. $\mathrm{N}_h(\mathrm{id}) = \mathrm{N}_v(\mathrm{id}) = 0$ et, pour $\sigma \in G_{mn}$

$$\begin{aligned}\forall s \in S^h_{m,n}, |\mathrm{N}_h(\sigma \circ s) - \mathrm{N}_h(\sigma) + \mathrm{N}_v(\sigma \circ s) - \mathrm{N}_v(\sigma)| &\leq |\mathrm{N}_h(\sigma \circ s) - \mathrm{N}_h(\sigma)| + |\mathrm{N}_v(\sigma \circ s) - \mathrm{N}_v(\sigma)| \\ &\leq 2n - 1 + 1 \leq 2\max(n,m)\end{aligned}$$

et aussi

$$\begin{aligned}\forall s \in S^v_{m,n}, |\mathrm{N}_h(\sigma \circ s) - \mathrm{N}_h(\sigma) + \mathrm{N}_v(\sigma \circ s) - \mathrm{N}_v(\sigma)| &\leq |\mathrm{N}_h(\sigma \circ s) - \mathrm{N}_h(\sigma)| + |\mathrm{N}_v(\sigma \circ s) - \mathrm{N}_v(\sigma)| \\ &\leq 1 + 2m - 1 \leq 2\max(n,m)\end{aligned}$$

donc $\dfrac{\mathrm{N}_h + \mathrm{N}_v}{2\max(n,m)}$ est monotone. La longueur d'une solution étant toujours entière, $\left\lceil \dfrac{\mathrm{N}_h + \mathrm{N}_v}{2\max(n,m)} \right\rceil$ est admissible. □

3.3 Heuristiques précalculées

Les heuristiques présentées ici sont tirées de travaux sur le taquin et fonctionnent en résolvant des problèmes simplifiés puis en stockant le nombre d'échanges nécessaire.

3.3.1 Walking Distance

Cette heuristique a été développée par Ken'ichiro Takahashi (takaken) [1]. Définissons les relations d'équivalence $\sim_h$ et $\sim_v$ sur G_{mn} par $\forall (\sigma, \tau) \in G^2_{m,n}$

$$\begin{aligned}\sigma \sim_h \tau &\iff \exists n \in \mathbb{N}, \exists (s_i)_{1 \leq i \leq n} \in (S^h_{n,m})^n, \sigma = \tau \circ s \\ \sigma \sim_v \tau &\iff \exists s \in S^v_{n,m}, \sigma = \tau \circ s\end{aligned}$$

Considérons les ensembles quotients $G_{m,n}/\sim_h$ et $G_{m,n}/\sim_v$ puis $A^h_{m,n} = \{([\sigma]_h, [\tau]_h), |\, (\sigma, \tau) \in A_{m,n}\}$ et $A^v_{m,n} = \{([\sigma]_v, [\tau]_v), |\, (\sigma, \tau) \in A_{m,n}\}$ où pour

tout $\sigma \in G_{mn}$, $[\sigma]_h$ désigne la classe de σ pour $\sim_h$ (et de même pour $\sim_v$). Enfin, pour toute permutation $\sigma \in G_{mn}$, soient $\mathrm{WD}_h(\sigma)$ la longueur du plus court chemin de $[\sigma]_h$ à id dans le graphe $(G_{m,n}/\sim_h, A^h_{m,n})$ et $\mathrm{WD}_v(\sigma)$ la longueur du plus court chemin de $[\sigma]_v$ à id dans le graphe $(G_{m,n}/\sim_v, A^v_{m,n})$.
Alors, l'heuristique *walking distance* est alors définie par $\forall \sigma \in G_{mn}, \mathrm{WD}(\sigma) = \mathrm{WD}_v(\sigma) + \mathrm{WD}_v(\sigma)$.
Informellement, pour $\sim_h$ deux grilles sont égales si et seulement si l'une peut s'obtenir à partir de l'autre en réalisant uniquement des échanges horizontaux, ou encore si et seulement si elles ont les mêmes chiffres dans chaque ligne. $\mathrm{WD}_h(\sigma)$ désigne alors le nombre minimal d'échanges à faire pour mettre chaque nombre dans sa ligne.

2	8	12	1	12	1	12	8
7	16	5	9	7	9	5	16
6	13	11	15	6	13	11	15
4	14	3	10	10	14	4	3

TABLE 3 – Deux grilles égales pour $\sim_h$.

Théorème 3.6. *La walking distance est admissible et plus informée que la distance de Manhattan :*

$$\forall \sigma \in S_{mn}, \delta(\sigma, id) \geq WD(\sigma) \geq \frac{DM(\sigma, id)}{2}$$

.

Démonstration. Considérons une solution d'une grille σ et notons V le nombre de d'échanges horizontaux effectués et H le nombre d'échanges verticaux. Par définition, un échange vertical ne modifie pas WD_v et un échange horizontal ne modifie pas WD_h. En outre, un échange horizontal fait varier WD_v de 1, -1 ou 0 et un échange vertical fait varier WD_h de 1, -1 ou 0. Ainsi on a $H \geq \mathrm{WD}_h(\sigma)$ et $V \geq \mathrm{WD}_v(\sigma)$ donc $\delta(\sigma, id) \geq \mathrm{WD}(\sigma)$. Considérons

$$\mathrm{DM}_h : [\tau]_h \in G_{m,n}/\sim_h \to \frac{1}{2} \sum_{k=0}^{mn-1} |(\tau(k) \,\%\, n) - (k \,\%\, n)|$$

$$\mathrm{DM}_v : [\tau]_v \in G_{m,n}/\sim_v \to \frac{1}{2} \sum_{k=0}^{mn-1} \left| \left\lfloor \frac{\tau(k)}{n} \right\rfloor - \left\lfloor \frac{k}{n} \right\rfloor \right|$$

Alors DM_h et DM_v sont bien définies (elles ne dépendent pas du choix du représentant) et un raisonnement similaire à celui fait dans le lemme 3.1 montrerait que ce sont, respectivement, des heuristiques admissibles pour la recherche d'un plus court chemin de $[\tau]_h$ à $[id]_h$ dans $(G_{m,n}/\sim_h, A^h_{m,n})$ et de $[\sigma]_v$ à $[id]_v$ dans $(G_{m,n}/\sim_v, A^v_{m,n})$ pour tout $\tau \in G_{mn}$. Or, $\mathrm{WD}_v(\sigma)$ et $\mathrm{WD}_h(\sigma)$ représentent justement les distances des plus courts chemins dans ces deux graphes. On a donc $\mathrm{WD}_v(\sigma) \geq \mathrm{DM}_v(\sigma)$ et $\mathrm{WD}_h(\sigma) \geq \mathrm{DM}_h(\sigma)$ d'où $\mathrm{WD}(\sigma) \geq \frac{\mathrm{DM}(\sigma)}{2}$. □

3.3.2 Heuristique de base de données

Korf & Felner [2] [3] ont développé deux méthodes pour calculer des heuristiques. La première, dite paritionnée statiquement ou encore heuristique **A**dditive de **B**ase de **D**onnées à **M**otif (APDB en anglais), consiste à partitionner $[\![0, mn-1]\!]$ en ensembles $(S_i)_{i\in I}$. Pour chacun des S_i, on considère la relation d'équivalence $\sigma \iff {}_i\tau \forall k \in S_i, \tau^{-1}(k) = \sigma^{-1}(k)$ puis $A_i = \{([\sigma]_i, [\tau]_i \,|\, (\sigma, \tau) \in A_{m,n}\}$. Autrement dit, deux grilles sont égales pour $\sim_i$ si et seulement si les éléments de S_i y occupent la même position. La valeur de l'heurisitique est alors la demi-somme pour $i \in I$ des distances de $[\sigma]_i$ à $[\mathrm{id}]_i$ dans le graphe $(S_{mn}/\sim_i, A_i)$.

La seconde, dite partitionnée dynamiquement, demande d'abord de choisir un entier $k \in [\![0, mn-1]\!]$ puis de résoudre toutes les grilles partielles à k nombres. Dans une telle grille, il y a k vrais nombres et les autres sont remplacés par des symboles génériques dont la position finale importe peu. Étant donné une grille σ et un ensemble d'entiers I on considère l'ensemble des grilles partielles à k nombres (avec $k \in I$) extraites. On peut représenter cet ensemble par un hypergraphe pondéré où les sommets sont les couples $(i, \sigma(i))$, $i \in [\![0, mn-1]\!]$ et les hyperarêtes tous les k-uplets de sommets. Chaque hyperarête est pondérée par la moitié du nombre minimal d'échanges nécessaire pour résoudre la configuration associée aux sommets inclus. La valeur de l'heuristique est alors la somme des poids d'un couplage.

4 Solveurs

4.1 Solveurs naïfs

4.1.1 Solveur glouton

Le premier solveur développé positionne dans l'ordre croissant chaque nombre à sa place finale en empruntant, à chaque étape, le plus court chemin composé d'échanges horizontaux puis verticaux. Cette dernière condition assure qu'au moment de ranger un nombre on ne déplace pas ceux qui sont déjà à leur place. Il y a $nm-1$ nombres et au cours de l'algorithme, chacun se déplacera d'au plus $n-1+m-1$ cases, ce qui donne une complexité temporelle en $O(nm(n+m))$. L'algorithme fait une copie de la grille donc sa complexité temporelle est en $O(nm)$. Évidemment, cet algorithme n'est pas exact.

4.1.2 Tri à bulle

Les cas où $m = 1$ se ramène au tri d'un tableau en effectuant des échanges entre cases adjacentes. La longueur d'une solution optimale d'une grille est alors donnée par son nombre d'inversion et le tri à bulle permet de résoudre le problème. En effet, un échange crée au supprime au plus une inversion, la grille finale n'en a aucune et le tri à bulle résout une inversion au moment de faire un échange. En moyenne, la complexité en temps de ce solveur est en $O((nm)^2)$ et en mémoire de $O(nm)$).

4.2 Parcours en largeur

Nous avons implémenté 4 algorithmes de parcours en largeur :

— Le parcours en largeur de la question de la question 7 qui demande de construire le graphe $(S_{nm}, A_{n,m})$. $\text{card}(S_{nm}) = (nm)!$ et on peut faire $nm - 1 - (m-1) = n(m-1)$ échanges horizontaux distincts et $nm-2-(n-1) = (n-1)m$ échanges verticaux distincts donc $2nm-m-n$ échanges au total. D'après le lemme des poignées de main, $\text{card}(A_{nm}) = \frac{1}{2}\sum_{\sigma\in S_{nm}} \deg(\sigma) = \frac{1}{2}\sum_{\sigma\in S_{nm}}(2nm-m-n) = (2nm-m-n)(mn)!$. L'algorithme a donc des complexités temporelles et spatiales $O(nm(nm)!)$.

— Le parcours en largeur de la question 8 qui construit au fur et à mesure le graphe $(S_{nm}, A_{n,m})$. Les complexités temporelles et spatiales dans le pire cas sont respectivement $O(nm(nm)!)$ et $O((nm)!)$ où encore $O(((2nm))^d)$ en temps et mémoire pour trouver un nœud à distance d de id.

— Le parcours en largeur bidirectionnel. Il consiste à effectuer un parcours en largeur depuis la grille qu'on cherche à résoudre et un autre depuis la grille finale. Comme le parcours en largeur considère tous les noeuds à distance d avant les noeuds à distance $d+1$, on a la garantie que le premier chemin formé par un noeud visité les deux parcours appartient à un plus court chemin entre la grille considérée et la grille finale. Il suffit alors de remonter de chaque côté pour en déduire un chemin. Les complexités dans le pire cas sont les mêmes que celles de la version unidirectionnelle mais, pour trouver un nœud à distance d, la complexité en temps et mémoire est de $O(\sqrt{((2nm))^d})$.

— Le parcours en largeur pseudo-bidirectionnel. Il consiste à simuler un parcours en largeur en bidirectionnel en ne faisant en réalité qu'un parcours unidirectionnel. Si l'on connaît un chemin de id à τ, il ne reste plus qu'à trouver un chemin de τ de σ ce qui revient à un chemin de $\sigma^{-1}\circ = \text{id}$. Toutefois, la réunion de deux plus courts chemins n'a aucune raison d'être un plus court chemin aussi donc cet algorithme n'est pas exact (contrairement à ceux du dessus). Il a la même complexité que l'algorithme précédent.

4.3 Solveur avec heuristique

Nous avons implémenté 3 algorithmes utilisant des heuristiques :

— L'algorithme A*. Dans le pire cas, ses complexités en mémoire et en temps sont respectivement $O(nm(nm)!\log((nm)!) = O((nm)^2(nm)!\log(nm))$ (d'après la formule de Stirling) et $O((nm)!)$ respectivement.

— Une version naïve bidirectionnelle d'A*. Cet algorithme exécute A* depuis la grille considérée et depuis la grille finale et s'arrête dès que les listes des nœuds visités (*closed list*) ont un nœud commun. On remonte depuis ce nœud de chaque côté pour obtenir un chemin. Cet algorithme n'est pas optimal. Une version plus évoluée, DIBBS [4], possède une condition d'arrêt différente et est optimal (si lancé avec deux heuristiques admissibles).

— Perimeter A* [5]. Cet algorithme fait un parcours en largeur de rayon d depuis la grille finale et exécute A* depuis la grille considérée avec l'heuristique définie par $h_d = d + \max_{\tau \in \mathcal{P}_d} h(\sigma, \tau)$ où $\mathcal{P}_d$ désigne les nœuds à distance d de id (d'où le nom de périmètre) et h est une heuristique.

5 Détails d'implémentation

— Les distances de Manhattan (abrégée en MD dans les graphiques) et d'inversion (ID) sont implémentées dans le fichier `utils.py` sous les noms respectifs `half_manhattan_distance` et `inversion_distance`. L'implémentation de la distance d'inversion utilise l'algorithme naïf quadratique bien qu'il existe une version quasi-linéaire.

— La *walking distance* (WD) est implémentée dans le fichier `wd.py`. La résolution des grilles lors du précalcul se fait avec un parcours en largeur.

— L'heuristique de base de données statique (APDB) est implémentée dans `apdb.py`. Deux implémentations sont proposées : une utilisant un dictionnaire et l'autre un tableau numpy. La symétrie peut être employée pour déduire d'une base de données une autre et ainsi gagner du temps et de la place. Une heuristique de base de données statique est décrite par la dimension des grilles qu'elle considère et le partitionnement de $[\![0, nm-1]\!]$ choisi.
L'heuristique de base de données dynamique (GADB) est implémentée dans `gadb.py`. Au vu de la taille exponentielle des base de données en jeu, un tableau numpy est utilisé ici. Contrairement à l'implémentation de l'heurisitque statique où, pour stocker $\dfrac{(nm)!}{(nm-k!)}$ éléments on crée un tableau k dimensionnel de côté nm (soit $(nm)^k$ emplacements au total), on alloue ici exactement la taille demandée. Pour ce faire, on stocke une bijection de l'ensemble des k-arrangements de $[\![0, nm-1]\!]$ dans $\left[\!\left[1, \frac{(nm)!}{(nm-k)!}\right]\!\right]$ et une autre des parties à k éléments de $[\![0, nm-1]\!]$ dans $\left[\!\left[1, \binom{nm}{k}\right]\!\right]$. Puisque l'ensemble des grilles à k nombres s'identifie au produit cartésien entre les k-arrangements de $[\![0, nm-1]\!]$ et les parties à k éléments de $[\![0, nm-1]\!]$ (le premier indique les positions et le second les nombres), on peut simplement créer un tableau bidimensionnel $\frac{(nm)!}{(nm-k)!} \times \binom{nm}{k}$. L'espace occupé par les bijections f et g sous forme de dictionnaire est largement contrebalancé par le gain de place. Le calcul de l'heuristique dans le cas statique se fait avec un algorithme glouton : les hyperarêtes sont considérées triées et sélectionnées si elles ne partagent pas de sommet avec celles déjà prises. Le tri est fait par poids décroissant ou par ratio poids / nombre de sommets décroissant : c'est la version fractionnaire (frac). Une heuristique de base de données dynamique est décrite par la dimension des grilles qu'elle considère et un entier donnant le maximum de chiffres dans les grilles partielles. (3, 3, 2) correspond à toutes les grilles partielles 3×3 à 1 ou 2 chiffres par exemple.

— Une implémentation d'A* utilise un tas binaire pour la recherche du minimum et une autre une *bucket list* (BL, implémentée dans `solver_utils.py`). Entre deux nœuds qui ont la même distance estimée f, on privilégie celui avec la plus grande distance g. Cette manière de départager des nœuds est noté TB (pour *tie-breaking*). Par défaut, un solveur A* l'implémente. Une autre implémentation spéciale d'A* calcule la distance de Manhattan de manière incrémentale. En effet, la variation de la distance de Manhattan au cours d'un échange ne dépend que des nombres échangées et de leurs positions donc on peut précalculer rapidement cette quantité. Au moment d'un échange il suffit alors de la mettre à jour.

6 Résultats

6.1 Résultats sur les heuristiques

Heuristique	Temps de calcul (en s)	Taille dans la RAM (en MB)	Taille sur disque (en MB)
GADB 4×4, 3	3,27	1,9	0,4
GADB 4×4, 4	241	79,4	16,7
APDB 3×3 (1, 2, 3, 4, 5, 6, 7, 8, 9)	16,6	387,4	1,2
APDB 4×4 (1, 2, 3, 5, 6)	24	1,0	0,2
APDB 4×4 (1, 2, 3, 4, 5, 6, 7)	3652	268,4	26,3
APDB 4×4 (9, 10, 13, 14, 15, 16)	313	16,8	2,3

TABLE 4 – Ressources en temps et en mémoire nécessaires au calcul des heuristiques de base de données

Heuristique	Temps de calcul moyen (en µs)	Valeur moyenne
DM	4,2	7,9955
DI	10,7	6,3409
WD	6	8,0177
APDB 3×3 (1 2 3 4 5 6 7 8 9)	6,2	9,0231
GADB 3×3 2	70	7,9034
GADB 3×3 2 frac	148,4	7,9955
GADB 3×3 4	425	7,219
GADB 3×3 4 frac	853	7,9955

TABLE 5 – Temps de calcul et valeurs moyennes des heuristiques sur des grilles 3×3

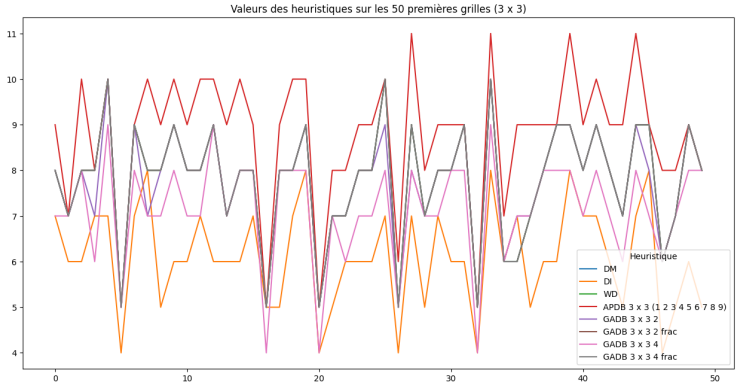

FIGURE 1 – Valeurs des heuristiques sur des grilles 3 × 3

Heuristique	Temps de calcul moyen (en µs)	Valeur moyenne
DM	6,4	20,0093
DI	22,3	15,4012
WD	9	20,0165
APDB 4 × 4 (1, 2, 3, 5, 6) + (4, 7, 8, 11, 12) + (9, 10, 13, 14, 15, 16)	15,6	17,501
APDB 4 × 4 (1, 2, 3, 4, 5, 6, 7) + (8, 9, 10, 13, 14, 15, 16) + (11, 12)	20,4	16,7098
GADB 4 × 4 2	210	19,8503
GADB 4 × 4 2 frac	437	20,0093
GADB 4 × 4 4	5922	18,6303
GADB 4 × 4 4 frac	10309	20,0093

TABLE 6 – Temps de calcul et valeurs moyennes des heuristiques sur des grilles 4 × 4

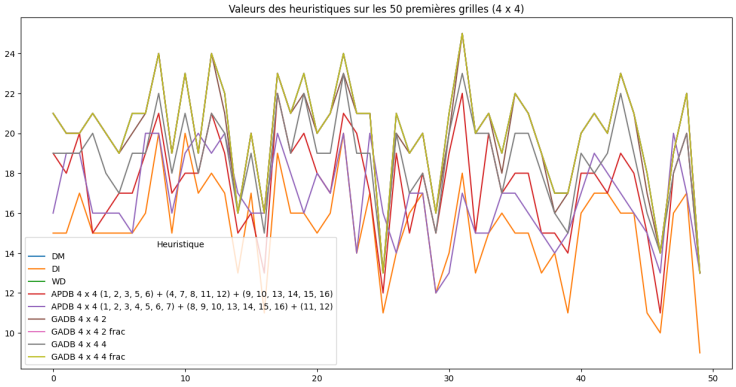

FIGURE 2 – Valeurs des heuristiques sur des grilles 4 × 4

6.2 Résultat sur les solveurs

Solveur	Temps de calcul (en ms)	Nombre de nœuds visités	Longueur de la solution
BFS	152,3	230412	8,88
BFS Bidirectionnel	41	16900	8,88
BFS Pseudo Bidirectionnel	29,8	5398	8,92
A* avec DM	4,1	836	8,88
A* avec DM incrémentale	2	836	8,88
A* avec 3×DM	5	113	9,2
PA^*_1	29,2	628	8,93
A* Bidirectionnel avec DM	3,2	611	8,93
DIBBS avec DM	12,7	1316	9,28

TABLE 7 – Performances moyennes des solveurs sur des grilles 3 × 3

Solveur	Temps de calcul (en s)	Nombre de nœuds visités	Longueur de la solution
A* avec DM sans départagement	446,9	60459	14,42
A* avec DM et BL sans départagement	266,3	41100	14,42
A* avec DM	256	35808	14,42
A* avec DM et BL	253,8	35814	14,42
A* avec DM incrémentale	165,7	35808	14,42
A* avec 3×DM	1,6	309	15,66
A* avec WD	290	32086	14,42
PA^*_1	290	34457	14,42
A* Bidirectionnel avec DM	71,2	9984	14,42
DIBBS avec DM	978	62289	15,46

TABLE 8 – Performances moyennes des solveurs sur des grilles 4 × 3

Solveur	Temps de calcul (en s)	Nombre de nœuds visités	Longueur de la solution
A* avec DM sans départagement	446,9	60459	14,42
A* avec DM et BL sans départagement	266,3	41100	14,42
A* avec DM	256	35808	14,42
A* avec DM et BL	253,8	35814	14,42
A* avec DM incrémentale	165,7	35808	14,42
A* avec 3×DM	1,6	309	15,66
A* avec WD	290	32086	14,42
PA^*_1	290	34457	14,42
A* Bidirectionnel avec DM	71,2	9984	14,42
DIBBS avec DM	978	62289	15,46

TABLE 9 – Performances moyennes des solveurs sur des grilles 4 × 4

7 Analyse des résultats et conclusion

En dehors du cas trivial de APDB 3 × 3 (1 2 3 4 5 6 7 8 9) où toutes les grilles possibles ont été précalculées, la distance de Manhattan surpasse toutes les heuristiques de base de données, tant en temps de calcul qu'en valeur. On remarque qu'exécuter GADB en mode fractionnaire revient à calculer la distance de Manhtattan. La *walking distance* est effectivement plus informée que la distance de Manhattan mais la différence est minime. Comment expliquer que des heuristiques qui ont réduit d'un facteur de 100 le temps de résolution d'un taquin par rapport à la distance de Manhattan sont ici surpassées par

cette dernière ? Celà vient très certainement du fait que, dans le *swap puzzle*, il y a une plus grande liberté d'échange et il n'y a pas de conflit entre les cases. La distance de Manhattan fournit donc une minoration correcte de la distance réelle.
On remarque qu'exécuter A* avec une heurisitque non admissible (ici le triple de la distance de Manhattan) fournit un algorithme rapide qui ne s'écarte pas beacoup de l'optimal. Les performances de la bucket list sont un peu décevantes. Le fait que A* avec tas binaire soit plus lent qu'avec une *bucket list* mais que la situation s'inverse si on départage les nœuds suivant leur profondeur vient probablement du surcoût lié à la manipulation de tableaux bidimensionnels. La lenteur de PA*$_d$ n'est pas surprenante : le calcul de h_d est lourd et n'apporte pas un gain significatif par rapport à h. Toutefois, il convient de garder à l'esprit que nos implémentations de ces solveurs expérimentaux (DIBBS et PA*) sont probablement sous-optimales voire incorrectes : les chercheurs sont assez vagues sur l'implémentation dans leurs articles et nous n'avons trouvé aucune implémention en ligne.

En somme, le *swap puzzle* est un problème relativement intéressant du fait de sa haute symétrie, son interprétation en termes de permutation et sa similarité avec d'autres problèmes existants. Toutefois, la résolution du puzzle est immédiate avec l'algorithme glouton proposé au tout début donc, afin de maintenir artificiellement un intérêt, on s'intérsse aux solutions minimales. Citons quelques pistes qui semblent prometteuses mais qui n'ont pas été exploitées faute de temps :

- Employer des heuristiques plus fines dans le calcul approché d'un couplage pondéré maximal dans un hypergraphe ;
- Exploiter la symétrie. Par exemple, les distance de Manhattan et d'inversion sont invariantes par passage à l'inverse. Par ailleurs, lors du précalcul dans GADB, de la distance d d'une grille partielle σ on déduit les distances de ses grilles conjuguées $s \circ \sigma \circ s^{-}1$ (qui valent toutes d) où s est une rotation ou une réflexion. Cela permettrait de définir une représentation invariante par rotation et par symétrie pour accélérer le calcul et diminuer l'espace occupé. On pourrait aussi exploiter les translations ;
- L'algorithme glouton fournit quasi-instantanément un chemin. Peut-on l'optimiser pour en déduire un chemin minimal ?
- Optimiser A*. C'est une tâche assez vaste mais on peut déjà continuer le travail fait sur l'implémentation de la file de priorité en essayant de nouvelles structures comme le tas de Fibonacci ou l'arbre Van EDMe Boas ;
- Envisager le problème sous un tout nouvel angle. Abandonnons l'idée de graphe et intéressons nous plutôt aux langages. Puisque $A_{m,n}$ est un sous-groupe générateur de S_{nm}, on peut définir ce dernier par une présentation, c'est-à-dire une partie génératrice et des relations ou règles de réécriture. Les relations désignent des égalités non triviales faisant intervenir les éléments de la partie génératrice comme par exemple s^2 $\forall s \in A_{m,n}, s^2 = \mathrm{id}$. L'obtention d'une solution optimale est équivalente

au calcul d'une forme irréductible donc à la résolution du problème du mot. Du fait de la croissance rapide du cardinal de S_{nm} et de l'indécidabilité du problème du mot dans le cas général, cette approche pourrait certainement ne pas aboutir.

Références

[1] Ken'ichiro TAKAHASHI. Il s'agt d'une traduction. URL : https://web.archive.org/web/20141224035932/http://juropollo.xe0.ru:80/stp_wd_translation_en.htm..

[2] A. FELNER, R. E. KORF et S. HANAN. "Additive Pattern Database Heuristics". In : *Journal of Artificial Intelligence Research* 22 (nov. 2004), p. 279-318. ISSN : 1076-9757. DOI : 10.1613/jair.1480. URL : http://dx.doi.org/10.1613/jair.1480.

[3] Richard E. KORF et Ariel FELNER. "Disjoint pattern database heuristics". In : *Artificial Intelligence* 134.1 (2002), p. 9-22. ISSN : 0004-3702. DOI : https://doi.org/10.1016/S0004-3702(01)00092-3. URL : https://www.sciencedirect.com/science/article/pii/S0004370201000923.

[4] John F. DILLENBURG et Peter C. NELSON. "Perimeter search". In : *Artificial Intelligence* 65.1 (1994), p. 165-178. ISSN : 0004-3702. DOI : https://doi.org/10.1016/0004-3702(94)90040-X. URL : https://www.sciencedirect.com/science/article/pii/000437029490040X.

[5] E.C. SEWELL et S.H. JACOBSON. "Dynamically improved bounds bidirectional search". In : *Artificial Intelligence* 291 (2021), p. 103405. ISSN : 0004-3702. DOI : https://doi.org/10.1016/j.artint.2020.103405. URL : https://www.sciencedirect.com/science/article/pii/S0004370220301545.