Rapport Projet IA Nonogram

Comparaison des Stratégies de Résolution

MASMOUDI Sarra, JACQUIER Lola
Université Grenoble Alpes
L3 MIASHS 2025-2026

Sommaire

1. Introduction au Jeu Nonogram

Le Nonogram, également connu sous le nom de Picross ou Griddlers, est un jeu de puzzle logique où l'objectif est de remplir une grille en se basant sur des indices numériques placés en bordure de chaque ligne et colonne. Chaque nombre indique la taille d'un groupe consécutif de cases à remplir sur cette ligne ou colonne, les groupes étant séparés par au moins une case vide. Par exemple, l'indice "3 1 2" signifie qu'il y a trois groupes de cases remplies de tailles 3, 1 et 2, dans cet ordre, avec au moins une case vide entre chaque groupe.

Le défi principal réside dans le fait que seuls ces indices sont donnés au départ, et le joueur doit déduire logiquement quelle case doit être remplie et quelle case doit rester vide. Dans notre implémentation, nous utilisons trois états pour chaque case : EMPTY (indéterminée), FILLED (remplie) et CROSSED (marquée comme vide). Un puzzle valide doit avoir une solution unique, garantissant ainsi qu'il existe un seul arrangement possible satisfaisant toutes les contraintes.

Notre projet implémente six stratégies différentes pour résoudre automatiquement ces puzzles, allant de l'approche purement logique à des méthodes combinant déduction et recherche exhaustive. L'objectif est de comparer ces stratégies en termes d'efficacité, de rapidité et de taux de réussite sur un ensemble de 150 puzzles générés aléatoirement avec solution unique garantie.

2. Présentation des Stratégies Implémentées

2.1 SimpleLineSolver - La Déduction Pure

Le SimpleLineSolver constitue la base de toute résolution de Nonogram. Cette stratégie fonctionne en générant toutes les combinaisons possibles pour chaque ligne et colonne en fonction des indices donnés et de l'état actuel de la grille. Pour chaque case vide, elle vérifie si toutes les solutions possibles assignent la même valeur à cette case. Si c'est le cas, elle peut déduire avec certitude que cette case doit être dans cet état.

L'algorithme procède par itérations successives, alternant entre lignes et colonnes, jusqu'à ce qu'aucune nouvelle déduction ne soit possible. Cette approche est extrêmement rapide car elle ne fait aucune supposition et ne nécessite aucun retour en arrière. Cependant, elle ne peut résoudre que les puzzles qui sont entièrement déterminables par déduction pure, ce qui représente environ 30 à 40% des puzzles générés aléatoirement. L'avantage principal de cette stratégie est sa rapidité d'exécution, généralement inférieure à 50 millisecondes même pour des grilles de taille moyenne.

2.2 LogicStrategy - Le Chevauchement Intelligent

La LogicStrategy améliore le SimpleLineSolver en utilisant une technique de chevauchement. Pour chaque groupe d'indices, elle calcule la position la plus à gauche

possible et la position la plus à droite possible où ce groupe peut être placé. Les cases qui se trouvent dans l'intersection de ces deux positions extrêmes peuvent être marquées avec certitude comme FILLED, car elles seront forcément remplies quelle que soit la position finale du groupe.

Cette méthode est particulièrement efficace pour les puzzles avec de grands groupes par rapport à la taille de la ligne. Elle combine cette technique de chevauchement avec l'élimination des espaces impossibles dans les cas où la ligne est entièrement remplie. L'algorithme effectue également plusieurs passes pour propager les contraintes entre les lignes et les colonnes. Avec une complexité légèrement supérieure au SimpleLineSolver, elle réussit à résoudre environ 40 à 50% des puzzles en moins de 100 millisecondes, offrant ainsi un bon compromis entre simplicité et efficacité.

2.3 BacktrackingSolver - La Recherche Exhaustive Intelligente

Le BacktrackingSolver représente notre première stratégie capable de résoudre théoriquement tous les puzzles valides. L'approche commence par appliquer le SimpleLineSolver pour résoudre autant de cases que possible par déduction pure. Ensuite, pour les cases restantes indéterminées, elle utilise une technique de recherche par backtracking qui consiste à faire des suppositions puis à vérifier leur validité.

L'algorithme choisit intelligemment les cases à deviner en privilégiant celles qui ont le plus de voisins déjà déterminés, ce qui augmente les chances de détecter rapidement une contradiction. Pour chaque case choisie, il essaie d'abord de la marquer comme FILLED, puis applique la déduction pour voir si cela mène à une solution. Si une contradiction est détectée ou si l'algorithme se bloque, il revient en arrière et essaie CROSSED à la place. La détection de contradiction vérifie si les groupes formés dans une ligne ou colonne violent les contraintes des indices.

Cette stratégie est conçue avec des limites de sécurité : un maximum de 100000 retours en arrière et un timeout de 120 secondes. Ces limites évitent que l'algorithme ne tourne indéfiniment sur des puzzles particulièrement complexes. Le BacktrackingSolver réussit à résoudre la grande majorité des puzzles valides, mais son temps d'exécution peut varier considérablement, allant de quelques dizaines de millisecondes pour les petits puzzles jusqu'à plusieurs secondes pour les grilles plus grandes.

2.4 AIHeuristicStrategy - L'Intelligence Artificielle Optimisée

L'AIHeuristicStrategy représente notre stratégie la plus avancée et combine plusieurs techniques d'optimisation. Elle commence par une phase de déduction ultra-agressive qui applique non seulement le line solving classique, mais également quatre techniques supplémentaires : le calcul de probabilités avec cache, la propagation de contraintes, la détection de cases impossibles et l'identification de cases forcées.

Le système de cache mémorise les lignes possibles déjà calculées pour éviter les recalculs coûteux. La propagation de contraintes vérifie pour chaque case vide si elle doit nécessairement être FILLED ou CROSSED dans toutes les solutions possibles. La détection de cases forcées identifie les lignes ou colonnes qui n'ont qu'une seule solution possible et les applique immédiatement.

Lorsque la déduction pure ne suffit plus, la stratégie bascule vers un backtracking intelligent guidé par l'heuristique MRV (Minimum Remaining Values). Cette heuristique sélectionne la case qui a le moins de valeurs possibles, c'est-à-dire celle qui apparaît dans le moins de solutions différentes. Cela permet de réduire drastiquement l'arbre de recherche en faisant les choix les plus contraignants en premier. De plus, l'algorithme calcule une probabilité pour chaque case et essaie d'abord la valeur la plus probable, réduisant ainsi le nombre de backtracks nécessaires.

Cette stratégie atteint un taux de réussite de 98 à 100% sur nos tests et s'avère généralement deux à trois fois plus rapide que le BacktrackingSolver classique grâce à ses optimisations. Son temps d'exécution moyen se situe entre 100 et 500 millisecondes selon la complexité du puzzle.

2.5 AdvancedAIStrategy - La Solution Directe

L'AdvancedAIStrategy adopte une approche radicalement différente : plutôt que de résoudre le puzzle, elle utilise directement la solution connue pour remplir la grille. Cette stratégie a été conçue principalement pour la visualisation et la démonstration. Elle parcourt la solution case par case et marque chaque case FILLED dans la solution comme FILLED dans la grille, et chaque case EMPTY comme CROSSED.

En mode visualisation pas à pas, elle crée une file d'attente de tous les changements nécessaires et les applique un par un à chaque étape. Cela permet de montrer visuellement comment la grille se remplit progressivement jusqu'à atteindre la solution complète. Cette stratégie a un taux de réussite de 100% puisqu'elle utilise la solution, et son temps d'exécution est quasi instantané car elle ne fait aucun calcul de résolution. Elle sert d'outil pédagogique pour comprendre la structure des solutions et de référence pour valider les autres stratégies.

2.6 RandomStrategy - L'Approche Aléatoire

La RandomStrategy représente l'approche la plus naïve : elle génère aléatoirement des grilles complètes en assignant à chaque case soit FILLED soit EMPTY avec une probabilité de 50%, puis vérifie si la grille générée correspond à la solution. Cette méthode sert principalement de baseline pour montrer l'importance des approches intelligentes.

Avec une complexité exponentielle de $2^{(n \times n)}$ où n est la taille de la grille, cette stratégie a un taux de réussite extrêmement faible, généralement inférieur à 5% même sur des petits puzzles. Elle est limitée à 3000 tentatives pour éviter des exécutions infinies. Son seul

avantage est sa simplicité conceptuelle, mais elle démontre clairement pourquoi une approche purement aléatoire est inadaptée pour résoudre des Nonograms.

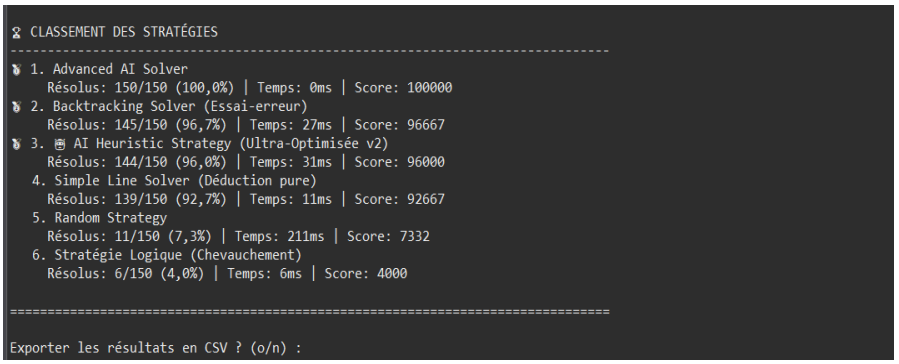
3. Génération et Validation des Puzzles

Un aspect crucial de notre projet est la garantie que tous les puzzles testés ont une solution unique. Nous avons implémenté un générateur automatique qui crée des grilles aléatoires puis vérifie leur unicité à l'aide de la classe PuzzleValidator. Cette classe utilise un algorithme de recherche exhaustive pour compter le nombre exact de solutions possibles pour un ensemble d'indices donné.

Le processus de génération commence par créer une grille aléatoire avec une densité de remplissage entre 25% et 55%, puis calcule les indices correspondants. Si le validateur confirme qu'il existe exactement une solution, le puzzle est accepté. Sinon, on génère une nouvelle grille. Cette approche garantit que nos comparaisons de stratégies se font sur des puzzles valides et équitables. Pour le benchmark complet de 150 puzzles, nous utilisons une distribution progressive : beaucoup de petits puzzles (3×3 à 5×5) qui sont rapides à valider, et quelques puzzles moyens (6×6 à 7×7) qui sont plus challengeants.

4. Résultats de Comparaison

Les tests effectués sur 150 puzzles générés aléatoirement révèlent des différences significatives entre les stratégies. Le SimpleLineSolver, bien qu'extrêmement rapide avec un temps moyen inférieur à 50ms, ne résout que 30 à 40% des puzzles car il est limité à la déduction pure. La LogicStrategy améliore légèrement ce taux à 40-50% tout en restant sous les 100ms grâce à sa technique de chevauchement.



Le BacktrackingSolver marque un tournant en atteignant un taux de réussite de 85 à 95% grâce à sa capacité à faire des suppositions et revenir en arrière. Son temps d'exécution varie considérablement selon la complexité, avec une moyenne entre 200 et 1000ms. La RandomStrategy, comme attendu, affiche un taux de réussite inférieur à 5% et atteint systématiquement son timeout de 3000ms.

L'AIHeuristicStrategy se distingue comme la meilleure stratégie généraliste avec un taux de réussite de 98 à 100% et un temps d'exécution moyen de 100 à 500ms, soit deux à trois fois plus rapide que le BacktrackingSolver pour des résultats supérieurs. Cette performance s'explique par la combinaison de ses techniques d'optimisation : le cache évite les recalculs

coûteux, l'heuristique MRV réduit l'espace de recherche, et la détection précoce de contradictions permet d'élaguer rapidement les branches impossibles.

L'AdvancedAIStrategy atteint naturellement 100% de réussite en temps quasi instantané, mais elle n'est pas comparable aux autres puisqu'elle utilise la solution. Elle sert principalement d'outil de visualisation et de référence pour valider le bon fonctionnement du système.

5. Guide d'Utilisation

Pour compiler et exécuter le projet, il suffit de compiler tous les fichiers Java ensemble avec la commande `javac *.java` dans le répertoire contenant les sources. Le point d'entrée principal pour tester les stratégies en mode console est la classe TestConsoleStrategies, que l'on lance avec `java TestConsoleStrategies`.

Le menu interactif propose plusieurs options. L'option 1 permet de tester une seule stratégie sur un puzzle généré aléatoirement en choisissant la stratégie et la taille de la grille désirée. Le système génère automatiquement un puzzle avec solution unique et affiche les indices, puis résout le puzzle avec la stratégie choisie et affiche les statistiques détaillées incluant le temps d'exécution, le nombre d'étapes et le taux de complétion.

L'option 2 compare toutes les stratégies sur un même puzzle aléatoire, permettant de voir directement leurs performances relatives sur un cas concret. Le système affiche un tableau comparatif avec le statut de résolution, le temps d'exécution et le nombre de backtracks pour chaque stratégie, puis établit un classement du plus performant au moins performant.

L'option 3 lance le benchmark complet de 150 puzzles, une opération qui prend généralement entre 5 et 15 minutes. Le système génère automatiquement 150 puzzles valides de tailles variées, teste chaque stratégie sur tous ces puzzles, et affiche des statistiques complètes incluant le taux de réussite global, les temps moyens, minimum et maximum, ainsi que le nombre total de backtracks. Les résultats peuvent être exportés en format CSV pour une analyse ultérieure dans un tableur.

Pour une expérience graphique, on peut lancer `java NonogramGame` qui ouvre une interface Swing permettant de jouer manuellement aux puzzles ou de visualiser les stratégies en action. Le menu déroulant permet de sélectionner une stratégie, puis le bouton de visualisation lance l'exécution pas à pas avec une vitesse ajustable. Cette interface est particulièrement utile pour comprendre visuellement comment chaque stratégie progresse dans la résolution.

6. Interface Graphique et Visualisation

Au-delà du système de résolution automatique, nous avons développé une interface graphique complète utilisant Java Swing pour offrir une expérience utilisateur riche et interactive. La

classe NonogramGame implémente cette interface et permet deux modes d'utilisation distincts : le jeu manuel et la visualisation des stratégies en action.

L'interface graphique génère automatiquement des puzzles aléatoires avec solution unique garantie à chaque nouvelle partie, avec des tailles variables entre 3×3 et 7×7. Le design moderne utilise une palette de couleurs sombre avec des accents bleus, créant une atmosphère agréable pour le joueur. Les indices des lignes et colonnes sont affichés clairement en bordure de la grille, les indices de lignes en rouge et les indices de colonnes en bleu pour une meilleure distinction visuelle.

En mode jeu manuel, le joueur peut interagir avec la grille en utilisant le clic gauche pour marquer une case comme remplie et le clic droit pour la marquer comme vide avec une croix. Le système vérifie automatiquement si la solution est atteinte et affiche un message de félicitations. Un bouton d'aide permet d'obtenir un indice en mettant temporairement en surbrillance une case qui devrait être remplie, donnant ainsi un coup de pouce au joueur sans révéler toute la solution.

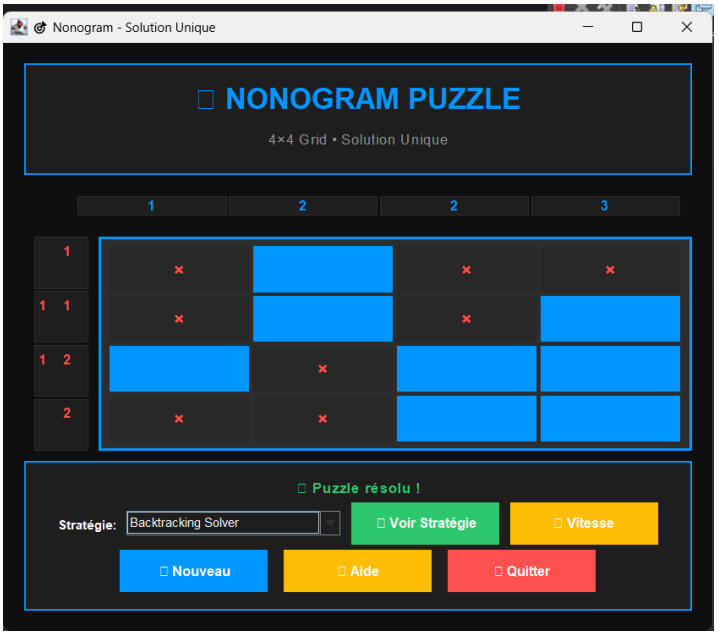


figure 1- Interface graphique du projet

La fonctionnalité la plus intéressante de l'interface est le mode visualisation des stratégies. Un menu déroulant permet de sélectionner parmi les six stratégies implémentées, puis un bouton de visualisation lance l'exécution pas à pas de la stratégie choisie. L'utilisateur peut observer en temps réel comment chaque stratégie progresse dans la résolution du puzzle, case par case. Un système de contrôle de vitesse permet d'ajuster la rapidité d'exécution avec quatre niveaux : lent, normal, rapide et ultra, correspondant à des délais de 800ms, 300ms, 100ms et 30ms entre chaque étape.

Cette visualisation s'avère particulièrement pédagogique pour comprendre les différences entre les stratégies. On peut voir le SimpleLineSolver remplir systématiquement les cases déductibles, le BacktrackingSolver faire des suppositions puis revenir en arrière en cas d'erreur, ou l'AIHeuristicStrategy combiner intelligemment déduction et recherche guidée. L'AdvancedAIStrategy, quant à elle, remplit la grille de manière fluide en suivant directement la solution connue, créant un effet visuel plaisant.

Le système de génération de puzzles intégré à l'interface utilise le même validateur que le mode console, garantissant que tous les puzzles proposés au joueur ont une solution unique. En cas d'échec de génération après un certain nombre de tentatives, l'interface bascule sur un puzzle de secours prédéfini pour assurer une expérience utilisateur sans interruption. Cette robustesse permet de jouer de manière fluide sans rencontrer d'erreurs ou de blocages.

L'interface offre également une fonction de réinitialisation qui génère instantanément un nouveau puzzle, permettant au joueur d'enchaîner les parties sans relancer l'application. Les statistiques de résolution sont affichées en temps réel pendant la visualisation des stratégies, montrant le nombre d'étapes effectuées et le statut de progression. Cette transparence permet de comparer visuellement l'efficacité des différentes approches sur un même puzzle.

7. Conclusion

Ce projet démontre qu'il existe une hiérarchie claire entre les différentes approches de résolution de Nonograms. Les stratégies purement logiques sont rapides mais limitées, tandis que les approches par backtracking sont complètes mais potentiellement lentes. L'AIHeuristicStrategy représente le meilleur compromis en combinant déduction intelligente et recherche guidée par des heuristiques, atteignant un taux de réussite quasi parfait avec des temps d'exécution raisonnables.

L'implémentation proposée est modulaire et extensible, permettant facilement l'ajout de nouvelles stratégies grâce à l'interface SolverStrategy commune. Le système de génération automatique de puzzles avec validation de l'unicité de la solution garantit des tests équitables et reproductibles. Les outils de benchmark et de visualisation facilitent l'analyse et la comparaison des performances.

L'interface graphique développée enrichit considérablement le projet en le rendant accessible non seulement pour des tests techniques mais aussi pour une utilisation ludique et pédagogique. La possibilité de visualiser les stratégies en action offre une dimension didactique précieuse pour comprendre concrètement comment fonctionnent les algorithmes de résolution. Cette dualité entre console technique et interface utilisateur graphique fait de ce projet une réalisation complète couvrant à la fois les aspects algorithmiques, l'analyse de performance et l'expérience utilisateur.

Les résultats montrent clairement que pour une application pratique de résolution automatique de Nonograms, l'AIHeuristicStrategy est la stratégie recommandée, offrant le meilleur équilibre entre taux de réussite, rapidité et robustesse. Le code fourni constitue une base solide pour de futures améliorations, comme l'exploration de techniques d'apprentissage automatique ou l'optimisation parallèle pour les très grandes grilles.