PROJET IA : Sudoku

Charlotte Bichot - Estelle Mom - Antoine Rossi

Sommaire

1) Aspect théorique : Type d'IA, "à la Norvig"

2) Aspect technique : Code et résultats

3) Aspect collaboratif : Travail d'équipe

Aspect théorique : “à la Norvig”

Peter Norvig : scientifique américain, réputé pour ses travaux chez Google

- Il y a une dizaine d’années, Peter Norvig a écrit un essai sur la résolution de sudoku en Python afin de prouver à ses proches que le sudoku était chronophage.
- Ses buts étaient de créer une interface facile d’accès, qui couvrait les performances de plusieurs niveaux de difficulté et qui affichait le temps de résolution
- Depuis, son code a été traduit dans différents langages informatiques

Aspect Théorique

Intelligence Exploratoire : Arbre de jeux

2 stratégies : programmation par contraintes
(constraint propagation) et exploration en profondeur et backtracking

3 principes de la programmation par contraintes :

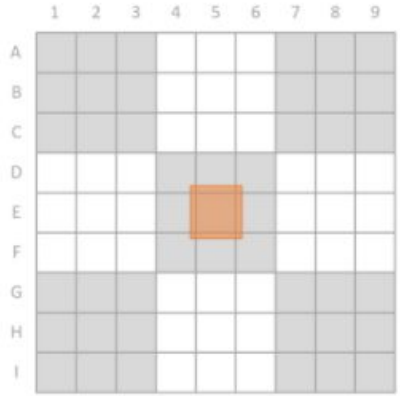
- Réduction de domaine / filtrage

- Chaque région est considérée comme une contrainte : valeurs différentes dans une région

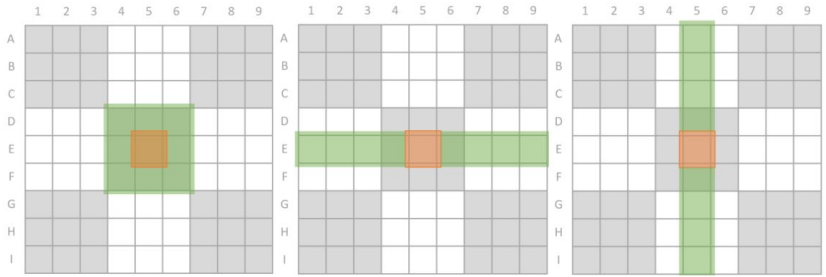
- Propagation : communication entre les régions pour trouver les valeurs disponibles

Aspect théorique

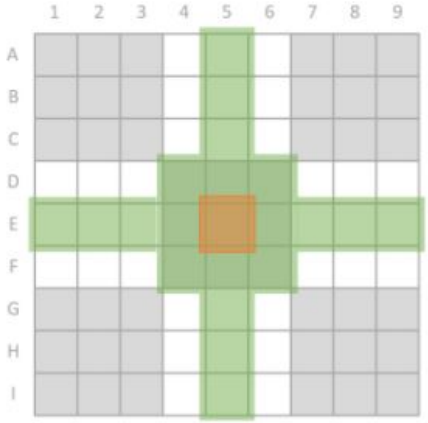
Carré



Unités



Voisins

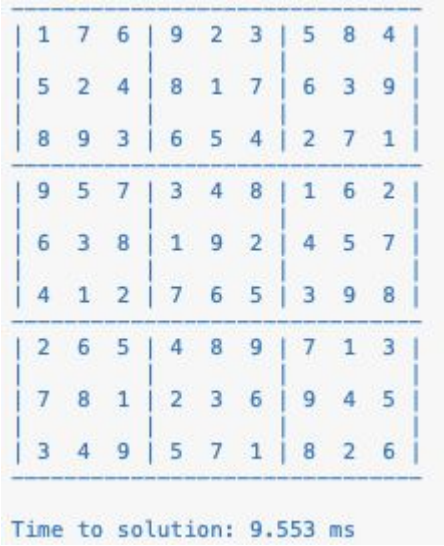


Aspect technique : Résolution “à la Norvig”

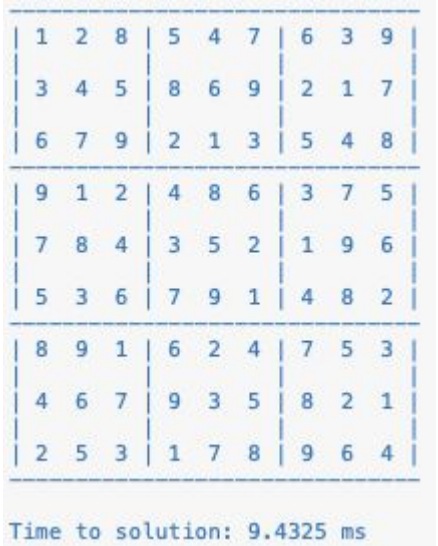
Une classe NorvigSolver.cs : Résolution de Sudoku pour 3 niveaux de difficulté

```
namespace Sudoku.Norvig
{
    0 références
    public class NorvigSolver : ISudokuSolver
    {
        private readonly IEnumerable<int> cellIndices = Enumerable.Range(0, 9);
        11 références
        public void Solve(GrilleSudoku s)
        {
            string game = string.Concat(s.Cellules.Select(c => (c == 0) ? "." : c.ToString()));
            Dictionary<string, string> solution = LinqSudokuSolver.Search(LinqSudokuSolver.Parse_grid(game));

            for (int r = 0; r < cellIndices.Count(); r++)
            {
                for (int c=0; c<cellIndices.Count(); c++)
                {
                    var cellules = Int32.Parse(solution[LinqSudokuSolver.Rows[r].ToString()  + LinqSudokuSolver.Cols[c].ToString() ]);

                    s.SetCell(r, c, cellules);
                }


            }
```

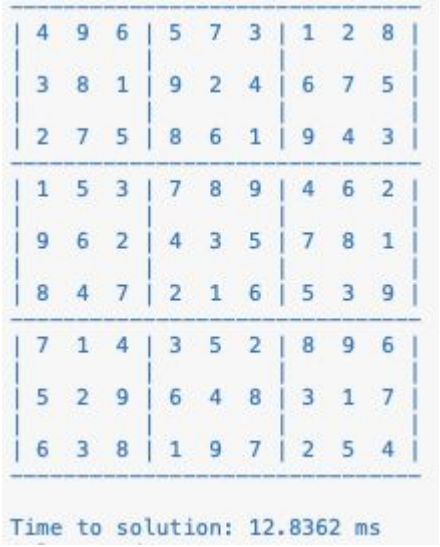
Temps d’exécution



Mode Easy

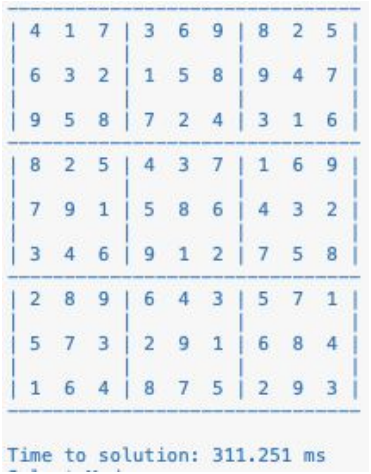


Mode Medium

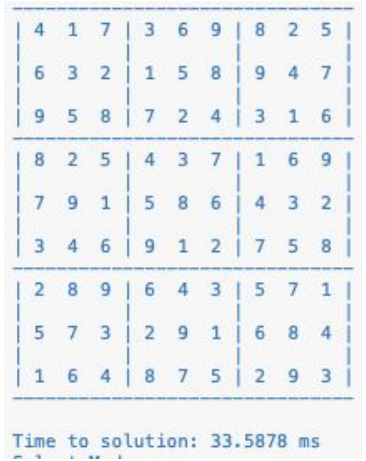


Mode Hard

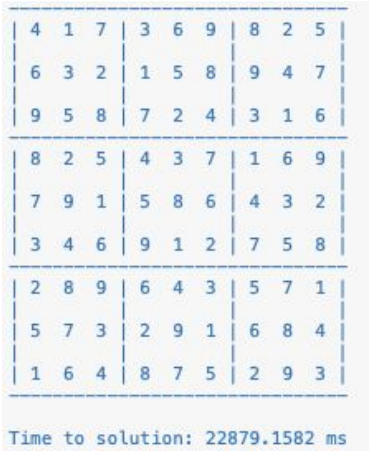
Comparaison des différentes solutions pour le numéro 1 (hard)



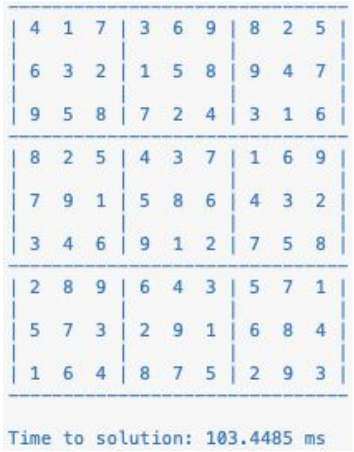
DancingLink
311.251 ms



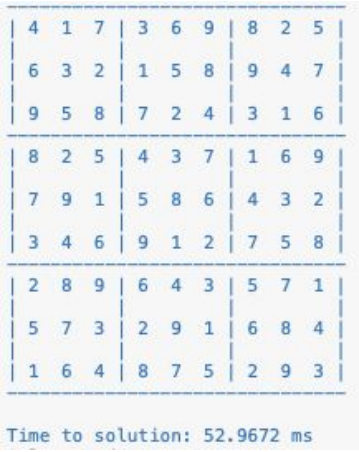
CSP
33.5878 ms



Recursive
22879.1582 ms



Techniques Humaines
103.4485 ms



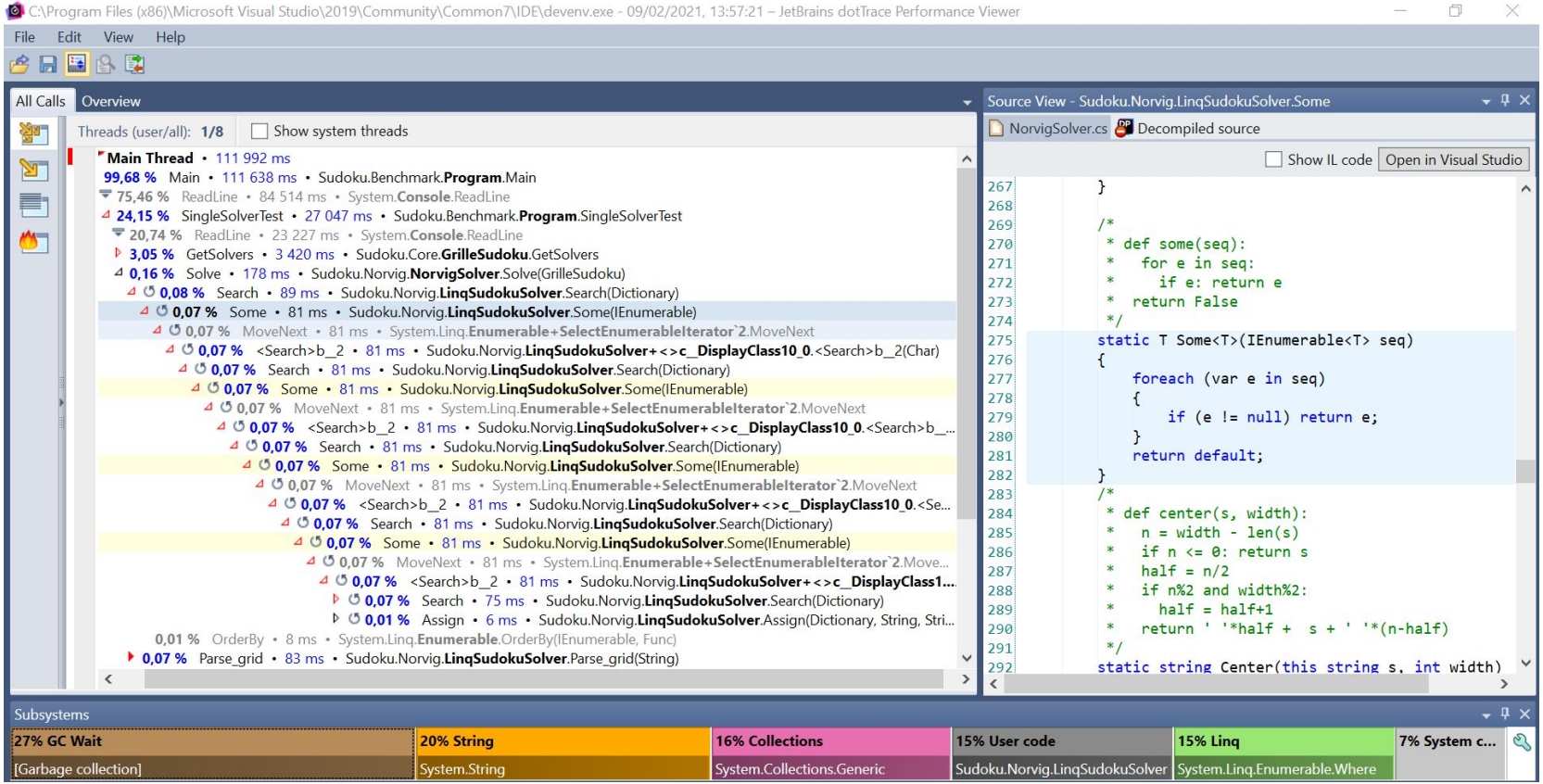
Norvig
52.9672 ms

Optimisation du code

dotTrace part of dotUltimate

Profileur de performances .NET

Avant “transformation” des foreach en for ↠ Solve 178ms

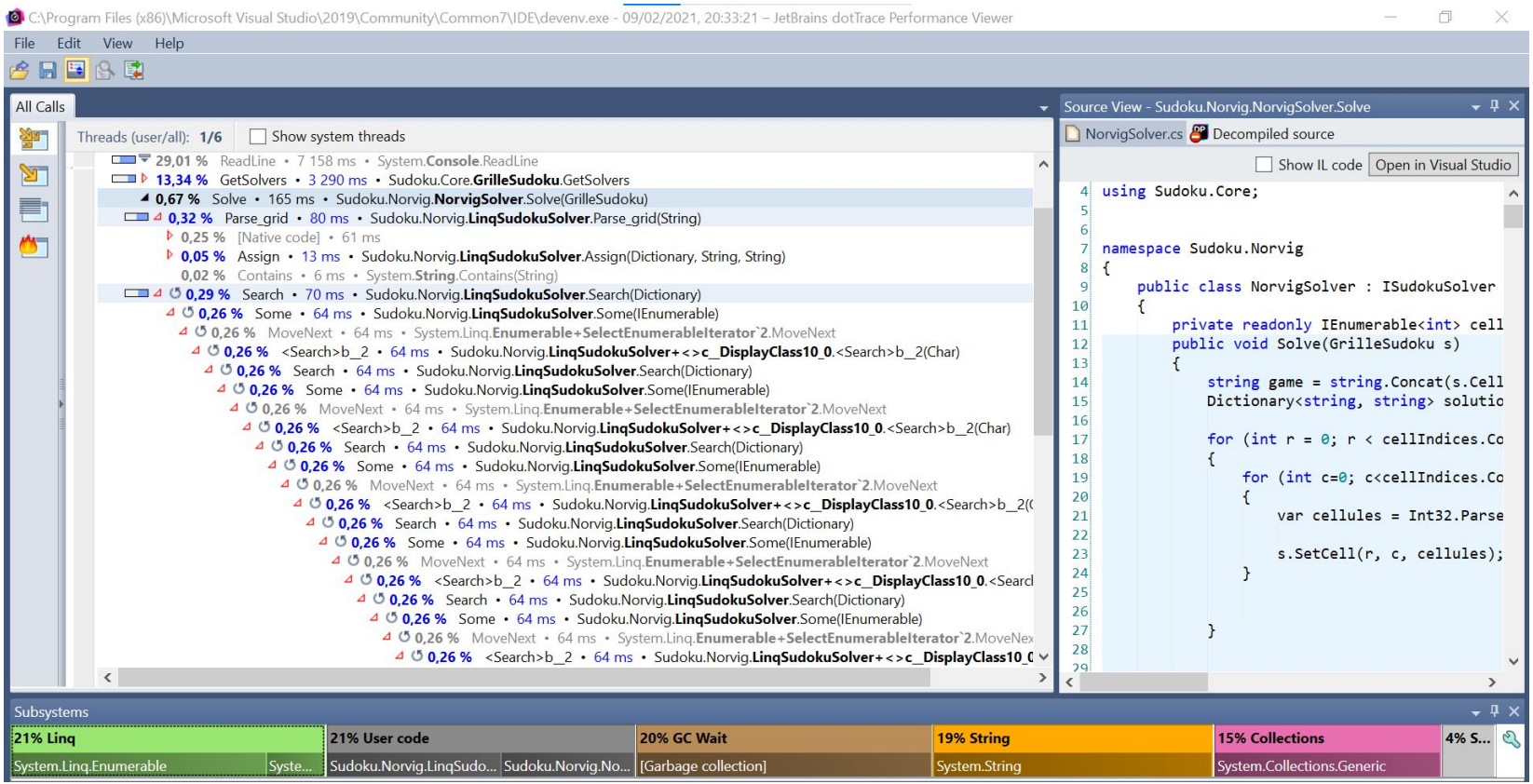


Optimisation du code

dotTrace part of dotUltimate

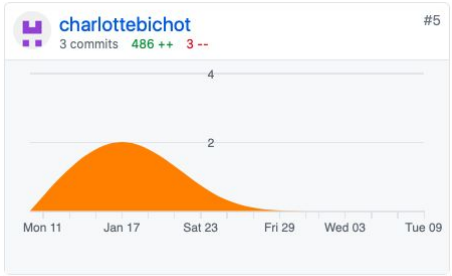
Profileur de performances .NET

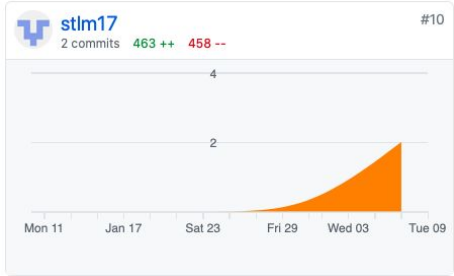
Après “transformation” des foreach en for ↠ Solve 165ms

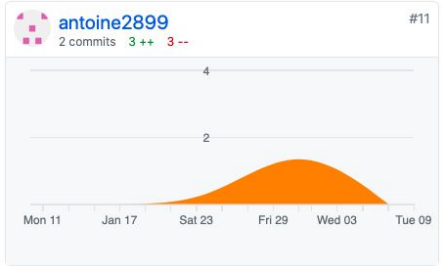


Aspect collaboratif

- Travail d'équipe







- Réflexion en commun

- Amélioration de l'optimisation du temps d'exécution

Sources :

- http://narendra.jussien.free.fr/sudoku/ia-sudoku.html
- https://norvig.com/sudoku.html
- https://alexisalulema.com/2018/04/19/c-sudoku-solver/
- https://naokishibuya.medium.com/peter-norvigs-sudoku-solver-25779bb349ce
- https://www.hebergementwebs.com/nouvelles/creer-une-application-de-reaction-pour-resoudre-chaque-puzzle-de-sudoku
- https://www.c-sharpcorner.com/UploadFile/dacca2/5-tips-to-improve-performance-of-C-Sharp-code/?fbclid=IwAR2k4rpLuGP5u6ArPbe2ozahElop9SWLX4c1OKGl3OwrCjKZQxT0QXneGMM
- https://www.jetbrains.com/fr-fr/profiler/