PRCON: Résolution de Picross

Amine Mike El Maalouf
Oscar Le Dauphin
Alexis Petignat
Max Nagaishi

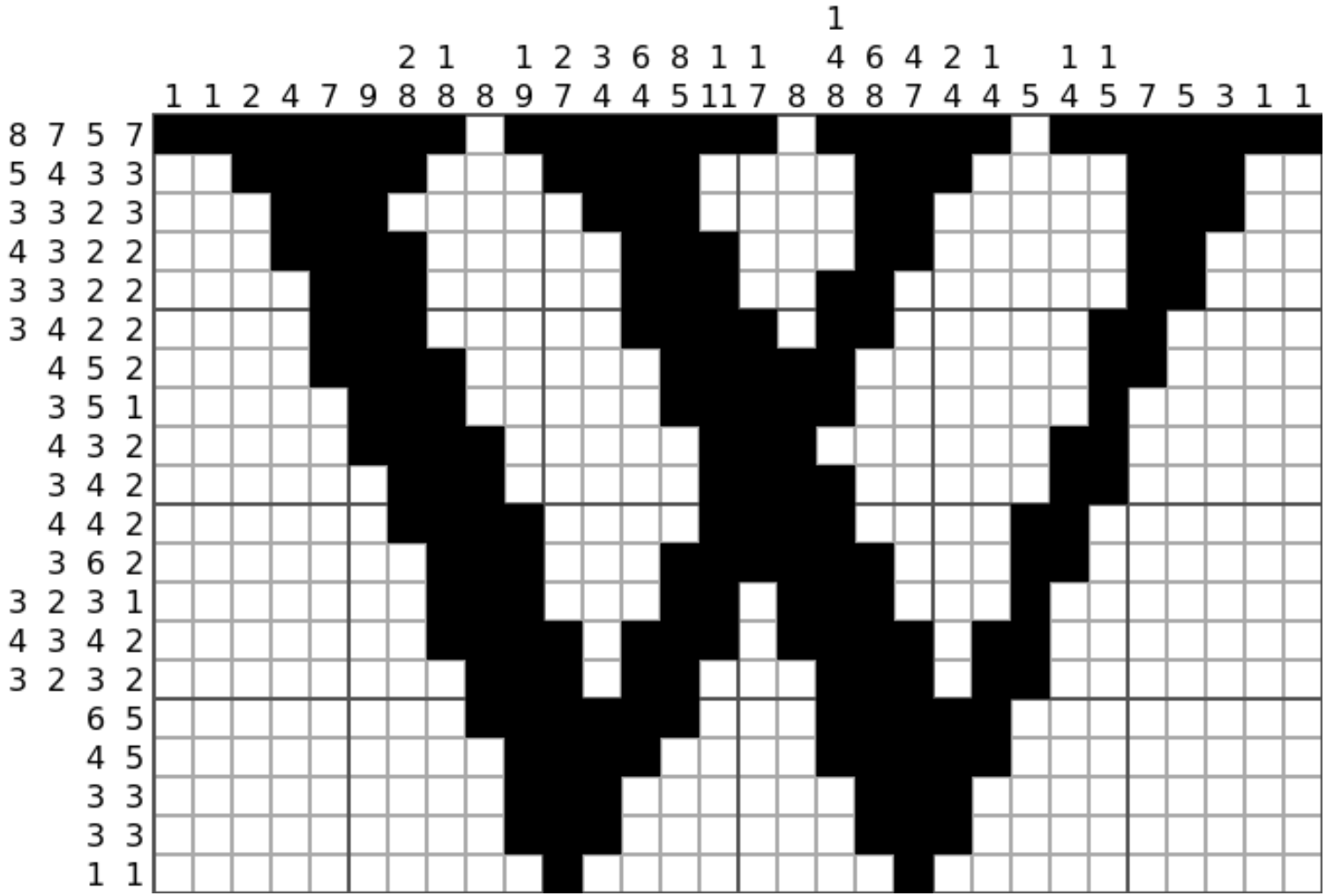
Avril 2025

Introduction

Sommaire

Qu'est-ce qu'un Picross

Qu'est-ce qu'un Picross



Représentation et Génération de Picross

Représentation et Génération de Picross

- Comment représenter un Picross non résolu ?
- Comment représenter un Picross pendant la résolution ?
- Comment représenter un Picross résolu ?
- Comment générer un Picross ?

Comment représenter un Picross non résolu ?

On veut représenter les contraintes.

- Le nombre de cases noires successives sur chaque ligne et colonne
- Le nombre de lignes et de colonnes

Comment représenter un Picross non résolu ?

```
┌──(venv)─(alex㉿localhost)-[~/afs/Master/PPC/PrCon]
└─$ cat smiley.pc
4 | 2 2 | 1 1 | 1 1 2 1 | 1 1 1 | 1 1 1 | 1 1 2 1 | 1 1 1 | 2 2 | 4
4 | 2 2 | 1 1 | 1 1 | 1 1 2 1 | 1 1 | 1 1 1 1 | 1 4 1 | 2 2 | 4
```

Comment représenter un Picross en cours de résolution ?

- Deux états pour une case: Noir ou Blanc
- Matrice de booléens

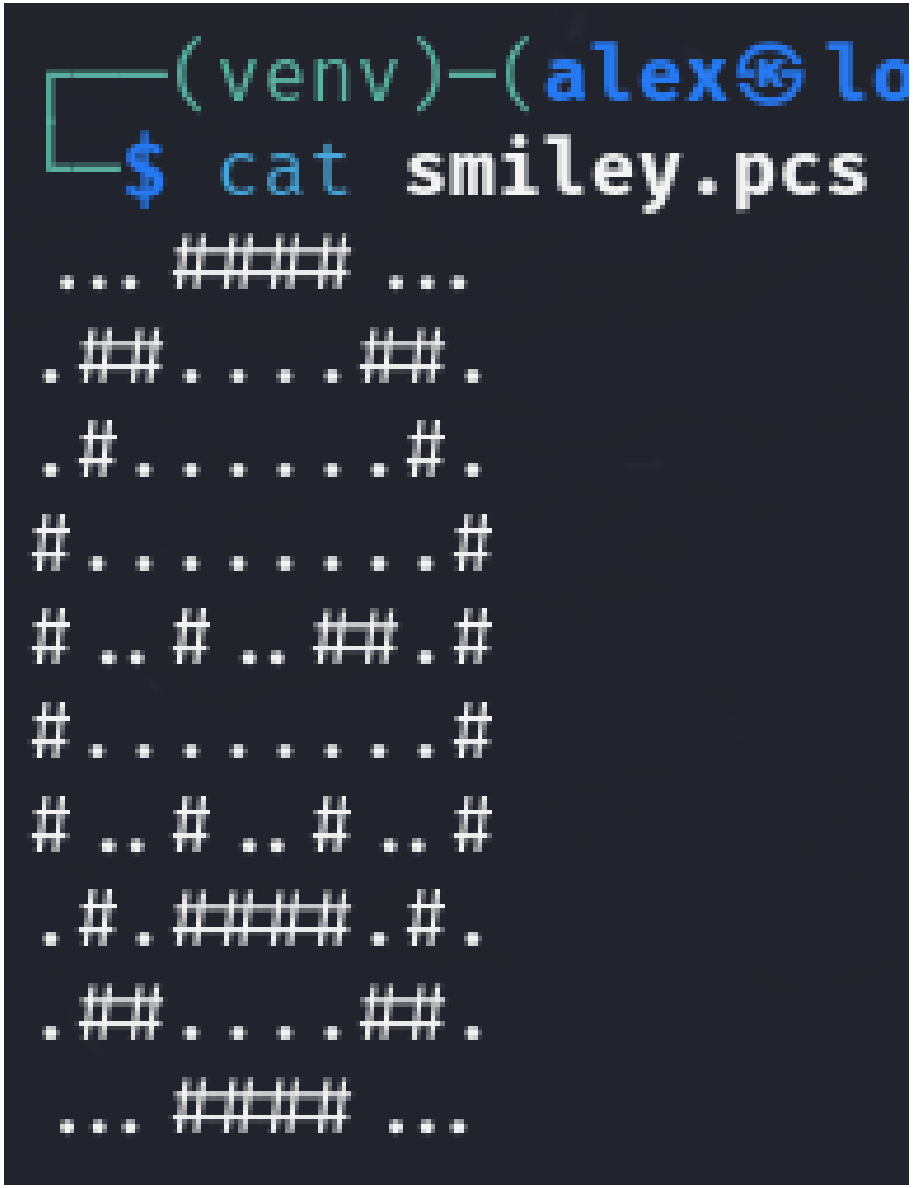
Comment représenter un Picross en cours de résolution ?

- 3^{e} état: Non traité
- 4^{e} état: Noir ou blanc
- $\rightarrow$ Matrice d'entiers (Enums)

Comment représenter un Picross résolu ?

- Deux états pour une case: Noir ou Blanc
- Doit être adapté pour les tests et l'affichage

Comment représenter un Picross résolu ?



Comment générer un Picross ?

- Transformer une image en format pc puis pcs
- Utile pour le testing
- Fait en Python puis traduit en Rust

Première approche (naïve)

Implémentation en Rust

- Sécurité mémoire sans garbage collector – zéro fuite ou segmentation fault.
- Performances proches du C/C++ – sans sacrifier la sécurité.
- Concurrence simple et sûre – évite les data races par design.
- Écosystème moderne – cargo, crates.io, et documentation de qualité.
- Adopté par des géants – Mozilla, Dropbox, Cloudflare, Microsoft...

Implémentation en Rust

- Utilise le backtracking
- Prend beaucoup de temps pour détecter une solution incorrecte
- Nombre de cas à tester exponentiel:
  - 2x2: 16 possibilités
  - 4x4: 65536 possibilités
- Quelques techniques trouvées par l'observation du processus de backtracking permettent un gros gain de performance

Implémentation en Rust

Taille de la grille	Nombre de solutions existantes
2x2	16
4x4	65536
7x7	562949953421312
12x12	2230074519853062314153571827264836150598041 6

Table: Nombre de solutions possibles par rapport à la taille de la grille

Taille de la grille	Temps estimé
12x12	40ms
15x15	1.1s
16x16	10s
20x20	27h (estimation)

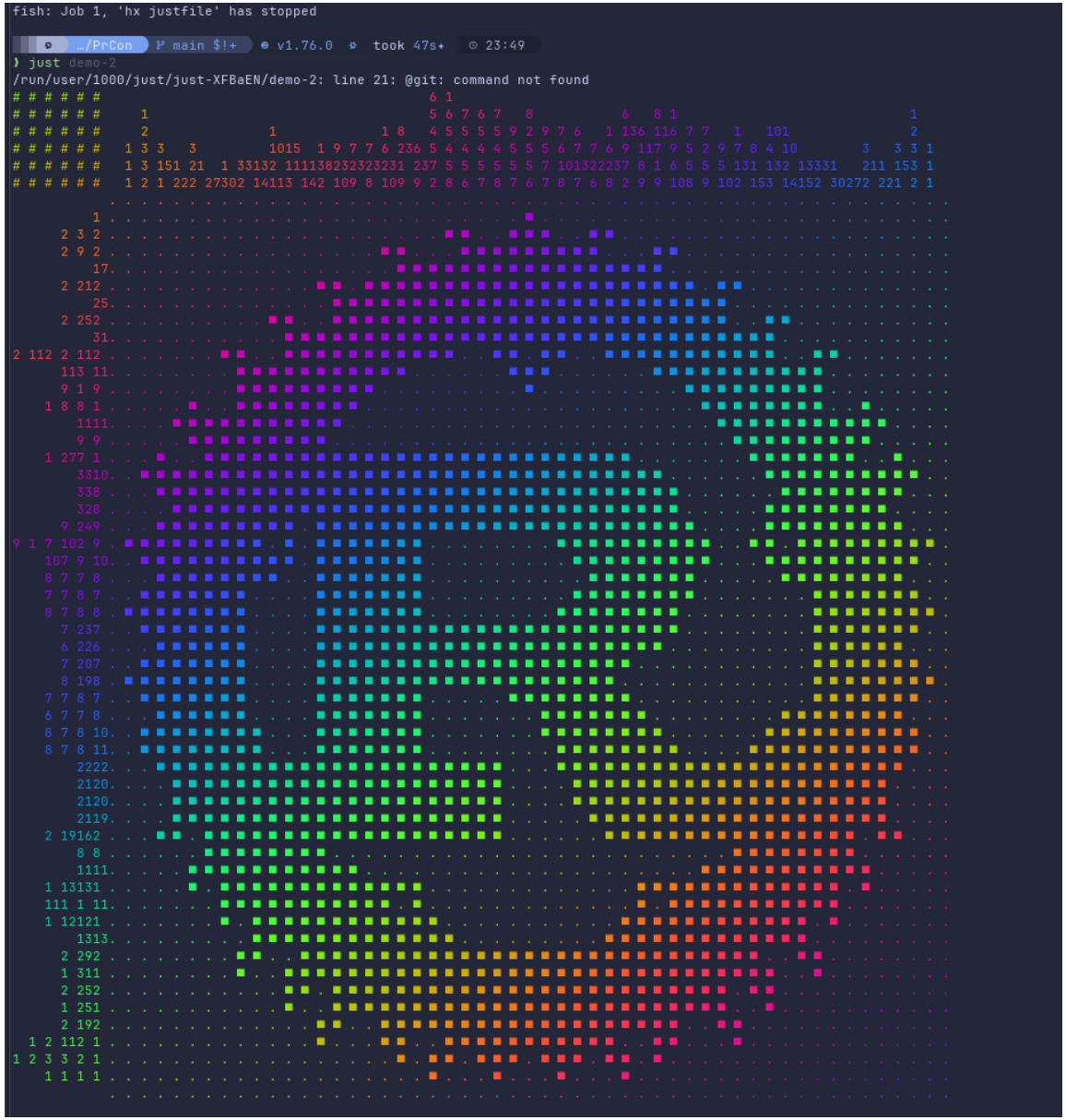
Table: Estimation du temps requis en fonction de la taille de la grille

Comparaison aux solutions existantes

Comparaison aux solutions existantes

- Librairie existante en Rust (Nonogrid)
- Utilise une combinaison d'algorithmes de résolution
- Beaucoup plus rapide que notre solution...

Comparaison aux solutions existantes



Programmation par contraintes

Programmation par contraintes

- Algorithme en Python
- Utilise la librairie Ortools de Google
- Adapté par nos soins au cas d'usage

Conclusion