Présentation de Mini-Projet

Solveur Picross

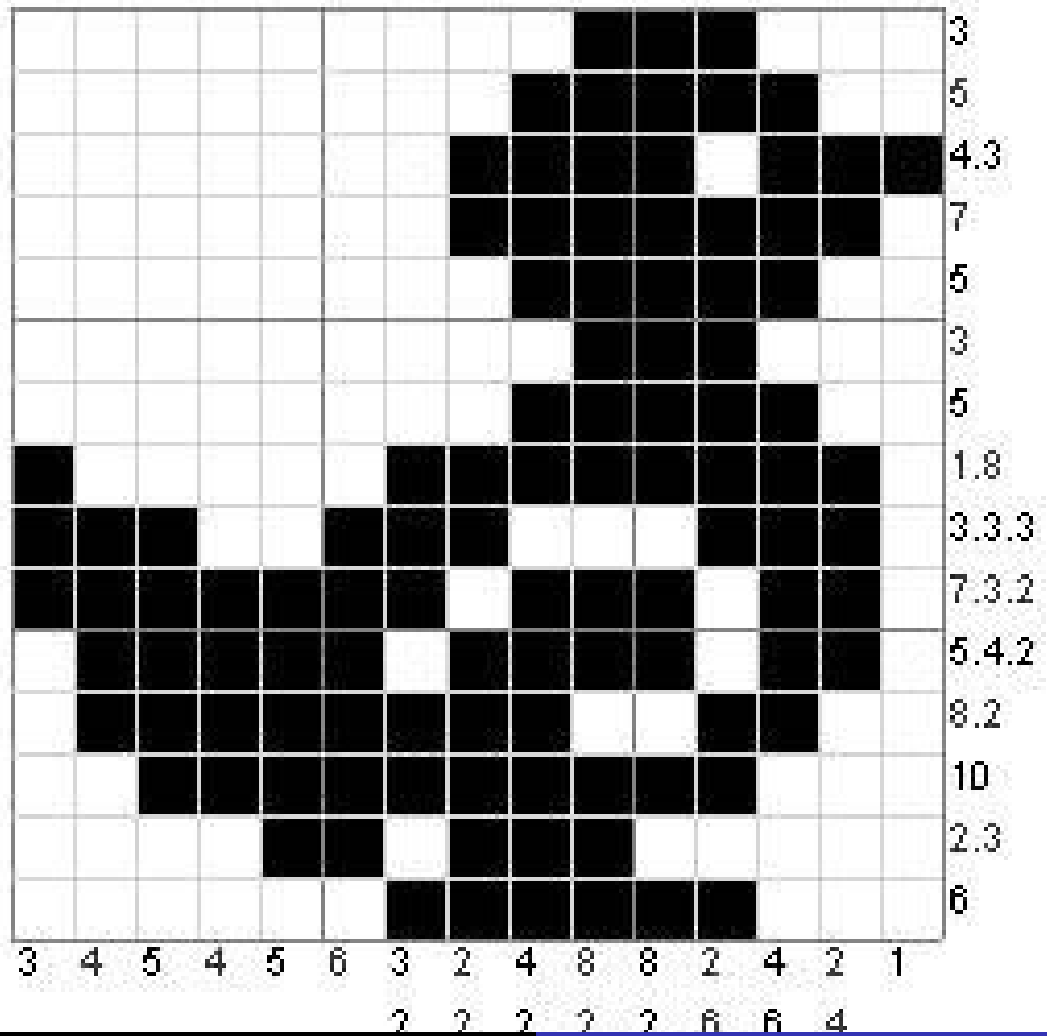
Pierre Larrenie et Simon Delecourt

20 novembre 2018

Sommaire

- Problématique
- Présentation des versions
- Démonstration
- Conclusion

Problématique



Présentation des versions

- Version 0 : Force Brute
- Version 1 : Backtracking
- Version 2 : Heuristique
- Version 3 : Génération d'un automate par contraintes

Version 0 : Force Brute

Par force brute l'idée est d'effectuer toutes les **permutations de 0 et de 1** sur la grille.
Pour une grille de taille N*M, 2^{N*M} **grilles à générer** et à trouver celles qui vérifient les contraintes.

$\Rightarrow$ **Irréalisable**

Version 1 : Backtracking

Idée :

- Générer les grilles dont les lignes qui vérifient les contraintes de lignes
- Tester si ces grilles vérifient les contraintes de bloc

Version 2 : Heuristique

Trier les contraintes selon leur score θ, dans l'ordre décroissant, et générer en alternant ligne et colonne. Soit $C = (c_1, \ldots, c_n)$ une contrainte d'une ligne ou d'une colonne T de longueur N.

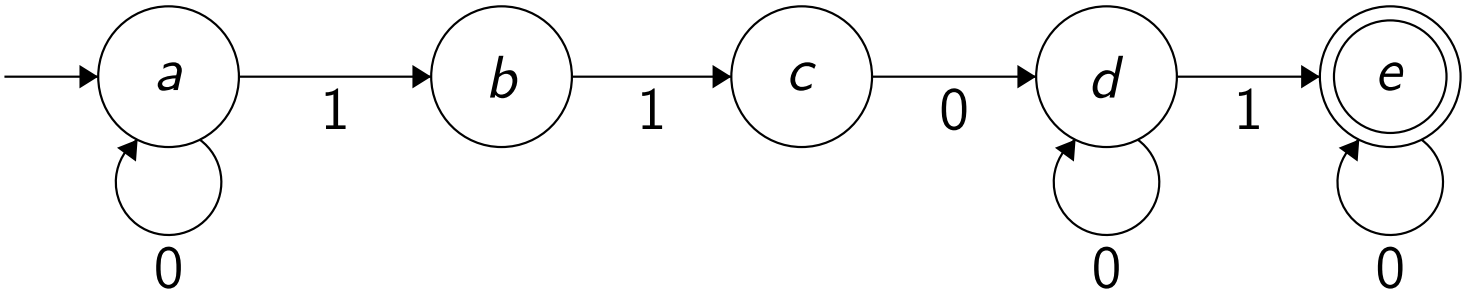
$$\theta(C) = \begin{cases} N & \text{si } \lg(C) = 0 \\ \left(\sum_{i=1}^{n} c_i\right) + N - 2 & \text{si } \lg(C) \neq 0 \end{cases}$$

Version 3 : Génération d'un automate par contraintes

Objectif : Construire un automate qui reconnait toutes le lignes admissibles d'après les contraintes

Version 3 : Génération d'un automate par contraintes

Illustrons par un exemple la manière dont un automate peut être construit à partir de l'ensemble des contraintes : [2 , 1].



Version 3 : Génération d'un automate par contraintes

Langage régulier reconnaissant les lignes admissibles :

$$|X| = 0^* \left\{ \bigodot_{i=0}^{n-1} 1^{c_i} 0^+ \right\} c_n 0^* \qquad (1)$$

Version 3 : Génération d'un automate par contraintes

Aperçu du code pour la génération de l'automate

Démonstration

Comparaison des résultats

	Version 1		Version 2		Version 3	
	Inférences	Temps	Inférences	Temps	Inférences	Temps
Picross basique (7*7)	5 191 515	0.285	7 056	0.002	53 133	0.009
Picross 1 (5*10)	2 342 100 782	131.962	115 309	0.009	117 365	0.015
Picross 5839 (15*15)	–	–	44 832	0.004	311 912	0.034
Picross 16638 (20*40)	–	–	–	–	3 268 442	0.334
Picross 5318 (35*45)	–	–	–	–	7 136 110	0.656
Picross 2992 (77*77)	–	–	–	–	30 756 915	2.877
Picross 30713 (99*99)	–	–	–	–	169 763 351	15.729

Conclusion

- Programmation logique et par contraintes
- Problème de la complexité des algorithmes
- Intérêt d'une bonne modélisation du problème

Bibliographie


Approche par contraintes :
https://rosettacode.org/wiki/Nonogram_solver?fbclid=IwAR2Q8HEIcQaU29eriJjBHsgc4VJ2o1lgHPed-Uj4PydpYcWeRvtagjProlog


Examen polytechnique :
http://www.enseignement.polytechnique.fr/informatique/INF580/exams/examen_14.pdf

Questions ?


MERCI POUR VOTRE ATTENTION!
NE POSEZ PAS DES QUESTIONS DIFFICILES S'IL VOUS PLAÎT !