
Résolution du puzzle tridimensionnel des pentaminos

Bruno Gilleta

*Laboratoire d'Informatique de Marseille
Parc Scientifique et Technologique de Luminy
163, avenue de Luminy Case 901
13288 Marseille cedex 09
gilleta@lim.univ-mrs.fr*

RÉSUMÉ. Les pentaminos sont des pièces connexes formées de 5 cubes unitaires assemblés à plat. Il en existe 12 différents et leurs formes ressemblent aux 12 lettres F, I, L, P, N, T, U, V, W, X, Y, Z. Nous nous intéressons à placer de toutes les façons possibles ces 12 pentaminos dans un parallélépipède rectangle d'un volume de 60 cubes unités. Nous présentons un ensemble naturel de contraintes sur les nombres entiers dont les solutions donnent les différentes façons de placer les pentaminos et présentons un système pour résoudre nos contraintes par itérations de réductions locales et par coupures d'intervalles en intervalles distincts. Des benchmarks montrent qu'un algorithme énumératif classique reste de loin la méthode la plus rapide mais le fait que nos arbres de recherche sont de taille inférieure montrent que nous sommes dans la bonne voie.

ABSTRACT. Pentaminoes are pieces made of 5 connected unit cubes lead on a plane surface. Their 12 different shapes look like the 12 letters F, I, L, P, N, T, U, V, W, X, Y, Z. We are interested in all the different ways of putting these 12 pentaminoes in a box having a volume of 60 unit cubes. We express the different pentaminoes configurations as the solutions of a natural set of constraints on unknown integers and we present an algorithm for solving our constraints by iterations of local narrowing and by splitting intervals into disjoint intervals. Benchmarks show that a classical enumerative algorithm is still the most efficient way to solve our problem. However, our smaller search spaces show that we are working on the right track.

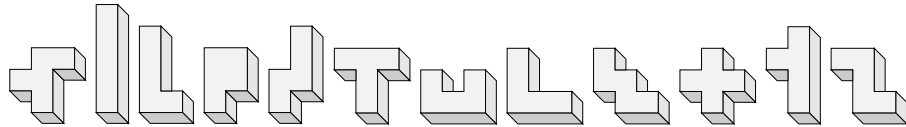
MOTS-CLÉS: Placement d'objets dans l'espace, Puzzle des pentaminos tridimensionnel, contraintes entières, Réduction d'intervalles

KEYWORDS: Fitting of pieces in space, Three-dimensional pentamino puzzle, Integer constraints, Interval narrowing

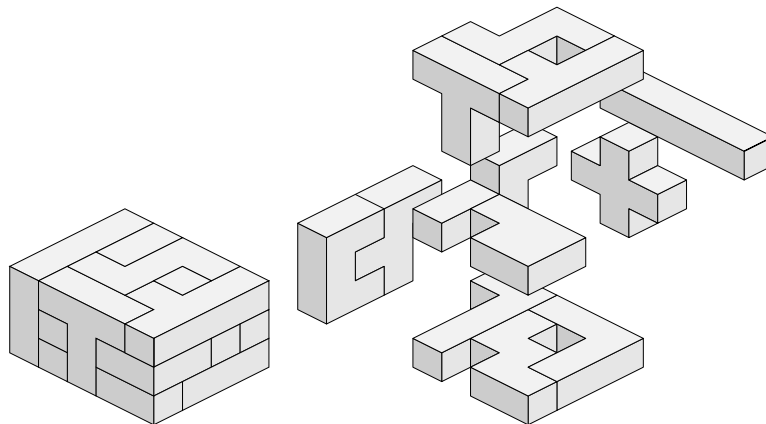
Introduction

Un *polyomino* (ou *n*-omino) désigne une figure simple composée de n carrés 1×1 connectés le long de leurs cotés. Le terme polyomino a été créé par Solomon W. Golomb en 1953 dans une présentation qu'il en fit au Harvard Mathematics Club, l'article [GOL 54] présente différentes façons de paver un échiquier avec des polyominos de deux (les classiques dominos), trois, quatre et cinq carrés : les *pentaminos*. Mais le premier problème connu portant sur les pentaminos fut publié en 1907 dans les Canterbury Puzzles par Henry Ernest Dudeney.

Dans [GAR 58] M. Gardner donne une épaisseur aux pentaminos, ce sont maintenant des pièces connexes formés de 5 cubes unitaires assemblés à plat. Il en existe 12 différents et leurs formes ressemblent aux 12 lettres F, I, L, P, N, T, U, V, W, X, Y, Z :



Nous nous intéressons à placer de toutes les façons possibles ces 12 pentaminos dans un parallélépipède rectangle d'un volume de 60 cubes unités. Par exemple, voici un placement des pentaminos dans une boîte de dimensions $3 \times 4 \times 5$:



Ce document traite d'un programme de contraintes sur les nombres entiers et de sa résolution par une méthode de réduction d'intervalles classique qui consiste à isoler l'ensemble des solutions dans plusieurs produits cartésiens d'intervalles de plus en plus réduits. Ce programme résout le puzzle des pentaminos en trois dimensions et par la même occasion met en évidence de bonnes contraintes sur les nombres entiers pour résoudre les problèmes de placement d'objets dans l'espace.

1. Notations et définitions

Intervalles et blocs

On désigne par \mathbf{Z} l'ensemble des entiers. Si a et b sont des entiers, on appelle *intervalle* l'ensemble éventuellement vide des entiers x tels que $a \leq x$ et $x \leq b$ et on le note $[a, b]$. On note \underline{I} (resp. \bar{I}) le plus petit (resp. le plus grand) élément de l'intervalle I quand il existe. Soit z un réel, on note $\lceil z \rceil$ (resp. $\lfloor z \rfloor$) le plus petit (resp. le plus grand) entier supérieur (resp. inférieur) ou égal à z . On appelle *bloc* tout produit cartésien fini $I_1 \times \cdots \times I_n$ d'intervalles I_i .

Soit maintenant une contrainte de la forme

$$x_1 \in I_1 \wedge \cdots \wedge x_n \in I_n \wedge (x_1, \dots, x_n) \in r,$$

où r est un sous-ensemble de \mathbf{Z}^n , où les x_i sont des variables et les I_i des intervalles. Nous nous intéressons à calculer une contrainte équivalente de la même forme mais dont les nouveaux intervalles I'_i sont le plus petit possible au sens de l'inclusion. Plus précisément nous cherchons à calculer au sens de l'inclusion le plus petit produit cartésien $\text{apx}(r)$ qui contient r . Pour ceci, nous utiliserons le fait que $X'_1 \times \cdots \times X'_n$ est le plus petit produit cartésien d'intervalles (au sens de l'inclusion) tel que :

$$X_1 \times \cdots \times X_n \cap r = X'_1 \times \cdots \times X'_n \cap r$$

Si r est un sous-ensemble de \mathbf{Z}^n , la i -ème *projection* de r , notée $\text{proj}_i(r)$, est l'ensemble des entiers e tels qu'il existe $(x_1, \dots, x_n) \in r$ avec $x_i = e$. L'*approximation* d'un ensemble de r , notée $\text{apx}(r)$ est le plus petit produit cartésien d'intervalles d'entiers contenant r .

De l'ensemble agréables des propriétés [BEN 93] de l'application $r \mapsto \text{apx}(r)$ nous en retiendrons deux qui nous seront utiles dans différents calcul de pavés de la forme $\text{apx}(r \cap I_1 \times \cdots \times I_n)$:

$$\text{apx}(r_1 \cup \cdots \cup r_n) = \text{apx}(\text{apx}(r_1) \cup \cdots \cup \text{apx}(r_n)) \quad [1]$$

et

$$\text{apx}(r) = \text{apx}(\text{proj}_1(r)) \times \cdots \times \text{apx}(\text{proj}_m(r)), \quad [2]$$

où r et les r_i sont des sous-ensembles de \mathbf{Z}^n et $\text{proj}_i(r)$ est la i ème *projection* de r , c'est-à-dire l'ensemble des éléments b de \mathbf{Z} tels qu'il existe un n -uplet (a_1, \dots, a_n) de r avec $b = a_i$.

Translations et rotations

Par *matrice de rotation* M nous entendons une matrice dont les éléments sont -1 , 0 ou 1 , qui est de la forme

$$M = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix} \quad [3]$$

avec au plus un élément non nul par ligne et par colonne, et dont le déterminant vaut 1.

Appelons *point* tout élément de \mathbf{Z}^3 et *configuration* tout ensemble fini de points. Soient

$$\begin{aligned} f &= \{(a_1, b_1, c_1), \dots, (a_n, b_n, c_n)\}, \\ f' &= \{(a'_1, b'_1, c'_1), \dots, (a'_n, b'_n, c'_n)\} \end{aligned}$$

deux configurations de n points. On dit que f' est une *translatée* de f si il existe des éléments c, d, e de \mathbf{Z} tel que, pour chaque i pris dans $[1, n]$, on ait

$$\begin{aligned} a'_i &= a_i + c, \\ b'_i &= b_i + d, \\ c'_i &= c_i + e. \end{aligned}$$

On dit que f' est une *pivotée* de f s'il existe une matrice de rotation M de la forme (3) telle que, pour chaque i pris dans $[1, n]$, on ait

$$\begin{aligned} a'_i &= \theta_{11}a_i + \theta_{12}b_i + \theta_{13}c_i, \\ b'_i &= \theta_{21}a_i + \theta_{22}b_i + \theta_{23}c_i, \\ c'_i &= \theta_{31}a_i + \theta_{32}b_i + \theta_{33}c_i. \end{aligned}$$

On note *translatés* (f) l'ensemble des translatées de la configuration f et *pivotés* (f) l'ensemble des pivotées de f . On peut alors définir la classe de la configuration f comme l'ensemble *classe* (f) des configurations qui sont égales à f à une translation et une rotation près. On a donc

$$\text{classe}(f) = \bigcup_{g \in \text{pivotés}(f)} \text{translatés}(g)$$

Pentaminos

Un *pentamino* est une configuration $\{(a_1, b_1, c_1), \dots, (a_5, b_5, c_5)\}$ de 5 points qui satisfait aux conditions de planarité et de connexité suivantes :

- les a_i sont tous égaux ou les b_i sont tous égaux ou les c_i sont tous égaux,
- il existe une suite i_1, \dots, i_k d'entiers pris dans 1..5 qui fait intervenir chaque élément de 1..5 et qui est telle que $(a_{i-1} - a_i)^2 + (b_{i-1} - b_i)^2 + (c_{i-1} - c_i)^2 = 1$, pour chaque i pris dans 2.. k .

On se donne les 12 pentaminos de bases π_i, \dots, π_{12} suivants :

$$\begin{aligned}
 \pi_1 &= \{(1, 2, 0), (2, 2, 0), (3, 2, 0), (3, 1, 0), (2, 3, 0)\}, \\
 \pi_2 &= \{(1, 1, 0), (2, 1, 0), (3, 1, 0), (4, 1, 0), (5, 1, 0)\}, \\
 \pi_3 &= \{(1, 1, 0), (1, 1, 0), (2, 2, 0), (3, 3, 0), (4, 3, 0)\}, \\
 \pi_4 &= \{(2, 1, 0), (1, 1, 0), (1, 2, 0), (2, 3, 0), (3, 3, 0)\}, \\
 \pi_5 &= \{(1, 1, 0), (2, 1, 0), (2, 2, 0), (3, 3, 0), (4, 3, 0)\}, \\
 \pi_6 &= \{(1, 1, 0), (1, 1, 0), (1, 2, 0), (2, 3, 0), (3, 3, 0)\}, \\
 \pi_7 &= \{(2, 1, 0), (1, 1, 0), (1, 2, 0), (2, 3, 0), (2, 3, 0)\}, \\
 \pi_8 &= \{(1, 1, 0), (1, 1, 0), (1, 2, 0), (2, 3, 0), (3, 3, 0)\}, \\
 \pi_8 &= \{(1, 1, 0), (1, 1, 0), (2, 2, 0), (2, 3, 0), (3, 3, 0)\}, \\
 \pi_9 &= \{(1, 2, 0), (2, 1, 0), (3, 2, 0), (2, 3, 0), (2, 3, 0)\}, \\
 \pi_{10} &= \{(1, 1, 0), (2, 1, 0), (3, 2, 0), (4, 3, 0), (2, 3, 0)\}, \\
 \pi_{12} &= \{(1, 1, 0), (2, 1, 0), (2, 2, 0), (2, 3, 0), (3, 3, 0)\}.
 \end{aligned} \tag{4}$$

L'ensemble de sous-ensembles $\{classe(\pi_1), \dots, classe(\pi_{12})\}$ est alors une partition de l'ensemble des pentaminos en 12 classes. En accord avec les formes physiques qui leurs sont associées les ensembles $classe(\pi_1), \dots, classe(\pi_{12})$ sont souvent notées F, I, L, P, N, T, U, V, W, X, Y, Z.

2. Énoncé du problème

Étant donnés trois entiers positifs a, b et c tels que $abc = 60$, nous voulons partitionner l'ensemble $[0, a-1] \times [0, b-1] \times [0, c-1]$ de *points* en 12 pentaminos f_1, \dots, f_n appartenant respectivement aux ensembles $classe(\pi_1), \dots, classe(\pi_{12})$. En posant, pour chaque $i \in 1..12$,

$$f_i = \{(x_{5i-4}, y_{5i-4}, z_{5i-4}), \dots, (x_{5i-0}, y_{5i-0}, z_{5i-0})\}$$

avec $((x_{5i-4}, y_{5i-4}, z_{5i-4}), \dots, (x_{5i-0}, y_{5i-0}, z_{5i-0}))$ lexicographiquement croissant, le calcul d'une partition revient au calcul d'une solution en $x_1, y_1, z_1, \dots, x_{60}, y_{60}, z_{60}$ de la contrainte

$$\left(\begin{array}{l}
 x_1 \in [0, a-1] \wedge \dots \wedge x_{60} \in [0, a-1] \\
 \wedge y_1 \in [0, b-1] \wedge \dots \wedge y_{60} \in [0, b-1] \\
 \wedge z_1 \in [0, c-1] \wedge \dots \wedge z_{60} \in [0, c-1] \\
 \wedge (x_1, y_1, z_1, \dots, x_{60}, y_{60}, z_{60}) \in \mathbf{PointsDistincts} \\
 \wedge (x_1, y_1, z_1, \dots, x_5, y_5, z_5) \in \mathbf{Classe}(\pi_1) \\
 \vdots \\
 \wedge (x_{56}, y_{56}, z_{56}, \dots, x_{60}, y_{60}, z_{60}) \in \mathbf{Classe}(\pi_{12})
 \end{array} \right)$$

où

- $\mathbf{PointsDistincts}$ est l'ensemble des $3n$ -uplets $(a_1, b_1, c_1, \dots, a_n, b_n, c_n)$ d'entiers qui sont tels que, pour tout i, j pris dans $1..n$, le point (a_i, b_i, c_i) soit distinct du point (a_j, b_j, c_j) , si i est distinct de j ,

- *Classe* (f), avec f une configuration de n points, est l'ensemble des $3n$ -uplets $(a_1, b_1, c_1, \dots, a_n, b_n, c_n)$ d'entiers qui sont tels que

$$\{(a_1, b_1, c_1), \dots, (a_n, b_n, c_n)\} \in \textit{classe}(f) \text{ et} \\ ((a_1, b_1, c_1), \dots, (a_n, b_n, c_n)) \text{ est lexicographiquement croissant,}$$

- les π_i sont les pentaminos de base définis en (4).

3. Expression du problème avec des contraintes plus élémentaires

Ne disposant pas de moyen efficace de calcul de la contrainte « être n vecteurs distincts » et remarquant que les triplets $(x, y, z) \in [a-1] \times [b-1] \times [c-1]$, nous utilisons l'application $(x, y, z) \mapsto x + ay + abc$ pour ré-exprimer cette contrainte au moyen de la contrainte « être n entiers distincts ». La contrainte

$$(x_1, y_1, z_1, \dots, x_{60}, y_{60}, z_{60}) \in \textit{PointsDistincts}$$

deviens alors

$$\left(\begin{array}{l} \exists m_1 \dots \exists m_{60} \\ m_1 \in [0, abc-1] \wedge \dots \wedge m_{60} \in [0, abc-1] \\ \wedge (m_1, x_1, y_1, z_1) \in \textit{SommePondérée}(1, a, ab) \\ \vdots \\ \wedge (m_{60}, x_{60}, y_{60}, z_{60}) \in \textit{SommePondérée}(1, a, ab) \\ \wedge (m_1, \dots, m_{60}) \in \textit{EntiersDistincts} \end{array} \right)$$

où *EntiersDistincts* est l'ensemble des n -uplets d'entiers distincts deux à deux et *SommePondérée*(a_1, \dots, a_n) l'ensemble des n -uplets d'entiers (x, y_1, \dots, y_n) tels que $x = a_1 y_1 + \dots + a_n y_n$.

Calculer efficacement une contrainte de « somme pondérée » étant quelque chose d'aussi difficile que de résoudre un problème de sac à dos (qui en est un sous-problème); nous décomposons cette contrainte en contraintes élémentaires de « somme » et de « produit par une constante » en introduisant à nouveau des variables quantifiées qui n'apparaissent pas dans les solutions :

$$(m, x, y, z) \in \textit{SommePondérée}(1, a, ab)$$

\Updownarrow

$$\left(\begin{array}{l} \exists s \exists t \\ s \in [0, a(b-1)] \\ \wedge t \in [0, ab(c-1)] \\ \wedge (s, y) \in \textit{Produit}(a) \\ \wedge (t, z) \in \textit{Produit}(ab) \\ \wedge (m, x, s, t) \in \textit{Sum} \end{array} \right)$$

où *Produit*(a) est l'ensemble des couples (x, y) tels que $x = ay$ et *Sum* est l'ensemble des n -uplets (x, y_1, \dots, y_n) tels que $x = y_1 + \dots + y_n$.

Après application de ces transformations, notre ensemble de contraintes se trouve sous la forme :

$$\left(\begin{array}{l} x_1 \in [0, a-1] \wedge \dots \wedge x_{60} \in [0, a-1] \\ \wedge y_1 \in [0, b-1] \wedge \dots \wedge y_{60} \in [0, b-1] \\ \wedge z_1 \in [0, c-1] \wedge \dots \wedge z_{60} \in [0, c-1] \\ \wedge (x_1, y_1, z_1, \dots, x_5, y_5, z_5) \in \mathbf{Classe}(\pi_1) \\ \wedge \vdots \\ \wedge (x_{56}, y_{56}, z_{56}, \dots, x_{60}, y_{60}, z_{60}) \in \mathbf{Classe}(\pi_{12}) \\ \wedge \exists m_1 \dots \exists m_{60} \\ \quad \left(\begin{array}{l} m_1 \in [0, abc-1] \wedge \dots \wedge m_{60} \in [0, abc-1] \\ \wedge (m_1, \dots, m_{60}) \in \mathbf{EntiersDistincts} \\ \wedge \exists s_1 \exists t_1 \\ \quad \left(\begin{array}{l} s_1 \in [0, a(b-1)] \\ \wedge t_1 \in [0, ab(c-1)] \\ \wedge (s_1, y_1) \in \mathbf{Produit}(a) \\ \wedge (t_1, z_1) \in \mathbf{Produit}(ab) \\ \wedge (m_1, x_1, s_1, t_1) \in \mathbf{Sum} \end{array} \right) \\ \wedge \vdots \\ \wedge \exists s_{60} \exists t_{60} \\ \quad \left(\begin{array}{l} s_{60} \in [0, a(b-1)] \\ \wedge t_{60} \in [0, ab(c-1)] \\ \wedge (s_{60}, y_{60}) \in \mathbf{Produit}(a) \\ \wedge (t_{60}, z_{60}) \in \mathbf{Produit}(ab) \\ \wedge (m_{60}, x_{60}, s_{60}, t_{60}) \in \mathbf{Sum} \end{array} \right) \end{array} \right) \end{array} \right)$$

4. Réduction des contraintes élémentaires

Multiplication entière par une constante

Étant donné un entier a , on rappelle que $\mathbf{Produit}(a)$ est l'ensemble des couples d'entiers (x, y) qui sont tels que

$$(x, y) \in \mathbf{Produit}(a) \iff x = ay$$

Proposition : Si X, Y, X', Y' sont des intervalles tels que $X' \times Y' = \text{apx}(X \times Y \cap \mathbf{Produit}(a))$, avec X et Y non vides, alors

$$\left(\begin{array}{l} X' \\ Y' \end{array} \right) = \begin{cases} \left(\begin{array}{l} \emptyset \\ \emptyset \end{array} \right), & \text{si } \overline{X} < \min(a\underline{Y}, a\overline{Y}) \text{ ou } \max(a\underline{Y}, a\overline{Y}) < \underline{X} \\ \left(\begin{array}{l} \{0\} \\ Y \end{array} \right), & \text{si } \overline{X} \geq \min(a\underline{Y}, a\overline{Y}) \text{ et } \max(a\underline{Y}, a\overline{Y}) \geq \underline{X} \text{ et } a = 0 \\ \left(\begin{array}{l} [\max(a \lceil \underline{X}/a \rceil, \min(a\underline{Y}, a\overline{Y})), \min(a \lfloor \overline{X}/a \rfloor, \max(a\overline{Y}, a\underline{Y})] \\ [\max(\min(\lceil \underline{X}/a \rceil, \lceil \overline{X}/a \rceil), \underline{Y}), \min(\max(\lfloor \overline{X}/a \rfloor, \lfloor \underline{X}/a \rfloor), \overline{Y})] \end{array} \right), & \text{sinon} \end{cases}$$

Somme multiple d'entiers

On rappelle que *Sum* est l'ensemble des $(n+1)$ -uplets d'entiers (x, y_1, \dots, y_n) qui sont tels que

$$(x, y_1, \dots, y_n) \in \text{Sum} \Leftrightarrow x = y_1 + \dots + y_n$$

Proposition : Si $X, Y_1, \dots, Y_n, X', Y'_1, \dots, Y'_n$ sont des intervalles tels que $X' \times Y'_1 \times \dots \times Y'_n = \text{apx}(X \times Y_1 \times \dots \times Y_n \cap \text{Sum})$ avec X, Y_1, \dots, Y_n non vides alors

$$\begin{pmatrix} X' \\ Y'_i \end{pmatrix} = \begin{cases} \begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}, & \text{si } \overline{X} < \underline{Y}_1 + \dots + \underline{Y}_n \text{ ou } \overline{Y}_1 + \dots + \overline{Y}_n < \underline{X} \\ \begin{pmatrix} [\max(\underline{X}, \underline{Y}_1 + \dots + \underline{Y}_n), \min(\overline{X}, \overline{Y}_1 + \dots + \overline{Y}_n)] \\ [\max(\underline{Y}_i, \underline{X} - \sum_{j=1, j \neq i}^n \underline{X}_j), \min(\overline{Y}_i, \overline{X} - \sum_{j=1, j \neq i}^n \overline{Y}_j)] \end{pmatrix}, & \text{sinon.} \end{cases}$$

Contrainte d'être des entiers distincts

On définit par *EntiersDistincts* l'ensemble des n -uplets d'entiers distincts deux à deux :

$$\begin{aligned} (x_1, \dots, x_n) &\in \text{EntiersDistincts} \\ &\Downarrow \\ x_1 &\neq x_2 \wedge x_1 \neq x_3 \wedge \dots \wedge x_{n-2} \neq x_n \wedge x_{n-1} \neq x_n \end{aligned}$$

Soit n intervalles d'entiers non vides X_1, \dots, X_n , pour trouver les intervalles d'entiers X'_1, \dots, X'_n qui sont tels que $X'_1 \times \dots \times X'_n$ soit le plus petit produit cartésien d'intervalles tel que :

$$X_1 \times \dots \times X_n \cap \text{EntiersDistincts} = X'_1 \times \dots \times X'_n \cap \text{EntiersDistincts}$$

nous utilisons l'algorithme en $\mathcal{O}(n^2)$ présenté dans [LEC 96] dans le cas général.

Pour notre problème de placement des pentaminos où le nombre de variables contraintes à prendre des valeurs distinctes est égal au nombre de valeurs qu'elles peuvent prendre, nous aurions pu utiliser un algorithme réduisant en $\mathcal{O}(n \log(n))$ opérations la contrainte de « tri » (décrites dans [BLE 97]), en imposant aux x_i d'être un tri des n premiers entiers :

$$\begin{aligned} &\begin{pmatrix} x_1 \in [1, n] \wedge \dots \wedge x_n \in [1, n] \\ \wedge (x_1, \dots, x_n) \in \text{EntiersDistincts} \end{pmatrix} \\ &\Downarrow \\ &\begin{pmatrix} x_1 \in [1, n] \wedge \dots \wedge x_n \in [1, n] \\ \wedge (x_1, \dots, x_n) \in \text{Tri}(1, \dots, n) \end{pmatrix} \end{aligned}$$

5. Réduction de la contrainte de placement

Contrainte de translation

Pour réduire la contrainte *Classe* (f), nous introduisons tout d'abord la contrainte *Translaté*(f) et donnons la façon de la réduire. Si f est une configuration de n points, alors *Translaté*(f) est l'ensemble des $3n$ -uplets d'entiers tels que

$$\begin{aligned} (x_1, y_1, z_1, \dots, x_n, y_n, z_n) &\in \text{Translaté}(f) \\ &\Downarrow \\ \{(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)\} &\in \text{translatés}(f) \text{ et} \\ ((x_1, y_1, z_1), \dots, (x_n, y_n, z_n)) &\text{ est lexicographiquement croissant} \end{aligned}$$

Proposition : Soit $X_1, Y_1, Z_1, \dots, X_n, Y_n, Z_n, X'_1, Y'_1, Z'_1, \dots, X'_n, Y'_n, Z'_n$ des intervalles tels que $X'_1 \times Y'_1 \times Z'_1 \times \dots \times X'_n \times Y'_n \times Z'_n = \text{apx}(\text{Translaté}(X_1 \times Y_1 \times Z_1 \times \dots \times X_n \times Y_n \times Z_n))$. Si $((a_1, b_1, c_1), \dots, (a_n, b_n, c_n))$ est un n -uplet de points de f croissants dans l'ordre lexicographique alors, pour tout $i \in \{1, \dots, n\}$

$$\begin{pmatrix} X'_i \\ Y'_i \\ Z'_i \end{pmatrix} = \begin{cases} \begin{pmatrix} \emptyset \\ \emptyset \\ \emptyset \end{pmatrix}, & \text{si } \begin{aligned} &\max_{j=1..n}(\underline{X}_j - a_j) > \min_{j=1..n}(\overline{X}_j - a_j) \\ &\text{ou } \max_{j=1..n}(\underline{Y}_j - b_j) > \min_{j=1..n}(\overline{Y}_j - b_j) \\ &\text{ou } \max_{j=1..n}(\underline{Z}_j - c_j) > \min_{j=1..n}(\overline{Z}_j - c_j) \end{aligned} \\ \begin{pmatrix} [a_i + \max_{j=1..n}(\underline{X}_j - a_j), a_i + \min_{j=1..n}(\overline{X}_j - a_j)] \\ [b_i + \max_{j=1..n}(\underline{Y}_j - b_j), b_i + \min_{j=1..n}(\overline{Y}_j - b_j)] \\ [c_i + \max_{j=1..n}(\underline{Z}_j - c_j), c_i + \min_{j=1..n}(\overline{Z}_j - c_j)] \end{pmatrix}, & \text{autrement} \end{cases}$$

Contrainte de placement

Nous pouvons maintenant traiter la contrainte *Classe* (f). Soit f une configuration de n points, on rappelle que *Classe* (f) est l'ensemble des $3n$ -uplets d'entiers tels que

$$\begin{aligned} (x_1, y_1, z_1, \dots, x_n, y_n, z_n) &\in \text{Classe}(f) \\ &\Downarrow \\ \{(x_1, y_1, z_1), \dots, (x_n, y_n, z_n)\} &\in \text{classe}(f) \text{ et} \\ ((x_1, y_1, z_1), \dots, (x_n, y_n, z_n)) &\text{ est lexicographiquement croissant} \end{aligned}$$

Proposition : Soit $X_1, Y_1, Z_1, \dots, X_n, Y_n, Z_n, X'_1, Y'_1, Z'_1, \dots, X'_n, Y'_n, Z'_n$ des intervalles tels que $X_1, Y_1, Z_1, \dots, X_n, Y_n, Z_n$, sont non vides et $X'_1 \times Y'_1 \times Z'_1 \times \dots \times X'_n \times Y'_n \times Z'_n = \text{Classe}(f) \cap X_1 \times Y_1 \times Z_1 \times \dots \times X_n \times Y_n \times Z_n$, soit $\{g_1, \dots, g_{24}\} = \text{pivotés}(f)$, soit $r = \text{Classe}(f) \cap X_1 \times Y_1 \times Z_1 \times \dots \times X_n \times Y_n \times Z_n$, soit $r_j = \text{Translaté}(g_j) \cap X_1 \times Y_1 \times Z_1 \times \dots \times X_n \times Y_n \times Z_n$, soit $\bar{X}_{1j} \times \bar{Y}_{1j} \times \bar{Z}_{1j} \times \dots \times \bar{X}_{nj} \times \bar{Y}_{nj} \times \bar{Z}_{nj} = \text{apx}(r_j)$, soit J l'ensemble des $j \in \{1, \dots, 24\}$ tels que $\text{apx}(r_j) \neq \emptyset$.

Alors pour tout $i \in 1..n$

$$\begin{pmatrix} X'_i \\ Y'_i \\ Z'_i \end{pmatrix} = \begin{cases} \begin{pmatrix} \emptyset \\ \emptyset \\ \emptyset \end{pmatrix}, & \text{si } J = \emptyset \\ \begin{pmatrix} [\min_{j \in J} X_{ij}, \max_{j \in J} \overline{X_{ij}}] \\ [\min_{j \in J} \overline{Y_{ij}}, \max_{j \in J} Y_{ij}] \\ [\min_{j \in J} \overline{Z_{ij}}, \max_{j \in J} \overline{Z_{ij}}] \end{pmatrix}, & \text{sinon} \end{cases}$$

Ce résultat découle de la définition de $\text{Translaté}(f)$, de l'égalité (1) et de l'égalité (4).

6. L'algorithme général de résolution

Une formule f est dite *normalisée* lorsque elle est écrite sous la forme suivante :

$$\begin{pmatrix} \exists x_m \exists x_{m+1} \dots \exists x_n \\ x_1 \in I_1 \wedge \dots \wedge x_n \in I_n \\ \wedge p_1 \wedge \dots \wedge p_k \\ \wedge q_1 \wedge \dots \wedge q_l \\ \wedge \text{vrai} \end{pmatrix}$$

où les x_i sont des variables ; les I_i sont des intervalles d'entiers ; les p_i sont des contraintes de la forme $(y_1, \dots, y_m) \in r$, r étant un ensemble de m -uplets d'entiers et les y_i des variables distinctes prises dans l'ensemble des x_i ; et les q_i sont des formules normalisées sans occurrence des x_i sous la portée d'un quantificateur.

Soit f une formule normalisée. Nous appliquons à la formule $f \vee \text{faux}$ une suite de transformations qui à chaque étape produisent une formule $f_1 \vee \dots \vee f_n \vee \text{faux}$ équivalente ou les f_i sont des formules normalisées. Après un nombre fini de transformations, nous obtenons une formule équivalente à f de la forme :

$$\begin{cases} \text{faux, si } f \text{ est équivalente à faux} \\ \text{vrai} \vee \dots \vee \text{vrai} \vee \text{faux, si } f \text{ est équivalente à vrai} \\ \left(\begin{pmatrix} x_1 \in \{a_1\} \\ \wedge \vdots \\ \wedge x_n \in \{a_n\} \\ \wedge \text{vrai} \end{pmatrix} \vee \dots \vee \begin{pmatrix} x_1 \in \{z_1\} \\ \wedge \vdots \\ \wedge x_n \in \{z_n\} \\ \wedge \text{vrai} \end{pmatrix} \right) \vee \text{faux, dans les autres cas.} \end{cases}$$

ou $\{(a_1, \dots, a_n), \dots, (z_1, \dots, z_n)\}$ est l'ensemble des n -uplets de valeurs pour lesquelles la formule obtenue en substituant dans f ces valeurs aux x_i est équivalente à *vrai*.

Les opérateurs \wedge et \vee sont associatifs et commutatifs. Pour des raisons de commodité, nous ommettons d'écrire certaines parenthèses dans nos formules.

Règles de transformation

Les six transformations suivantes sont appliquées sur une formule $f_1 \vee \dots \vee f_n \vee \text{faux}$ ou les f_i sont des formules normalisées ; quand une de ses sous-formules est sous la forme du membre gauche d'une de ces règles, on la remplace par le membre droit qui lui est équivalent. La formule $f'_1 \vee \dots \vee f'_m \vee \text{faux}$ ainsi obtenue est équivalente à $f_1 \vee \dots \vee f_n \vee \text{faux}$ et les f'_i sont des formules normalisées :

1. $p \wedge \exists x_1 \dots \exists x_n (q \wedge \text{vrai}) \Rightarrow \exists x_1 \dots \exists x_n (p \wedge q)$
2. $(x_1, \dots, x_n) \in r \wedge x_1 \in \{a_1\} \wedge \dots \wedge x_n \in \{a_n\} \Rightarrow x_1 \in \{a_1\} \wedge \dots \wedge x_n \in \{a_n\}$
3. $\text{faux} \vee \exists x_1 \dots \exists x_n (x \in \emptyset \wedge p) \Rightarrow \text{faux}$
4. $\exists x_1 \dots \exists x_i \dots \exists x_n (x \in \{a\} \wedge p) \Rightarrow \exists x_1 \dots \exists x_{i-1} \exists x_{i+1} \dots \exists x_n p$
5. $(x_1, \dots, x_n) \in r \wedge x_1 \in I_1 \wedge \dots \wedge x_n \in I_n \Rightarrow (x_1, \dots, x_n) \in r \wedge x_1 \in I'_1 \wedge \dots \wedge x_n \in I'_n$
6. $\text{faux} \vee \exists x_1 \dots \exists x_n (x \in I \wedge p) \Rightarrow \text{faux} \vee \exists x_1 \dots \exists x_n (x \in J \wedge p) \vee \exists x_1 \dots \exists x_n (x \in K \wedge p)$

où p et q sont des conjonction de formules, r est un sous-ensemble de \mathbf{Z}^n , les entiers a_1, \dots, a_n sont tels que $(a_1, \dots, a_n) \in r$, les intervalles I_1, \dots, I_n sont non vides et I'_1, \dots, I'_n sont tels que $I'_1 \times \dots \times I'_n = \text{apx}(I_1 \times \dots \times I_n \cap r)$ et il qu'il existe un i tel que $I'_i \neq I_i$, les intervalles I, J, K sont non vides et tels que $I = J \cup K$ et $J \cap K = \emptyset$ et p est sans occurrence de x_i dans la quatrième règle.

Correction

On associe à toute formule de la forme $f_1 \vee \dots \vee f_n \vee \text{faux}$ ou les f_i sont des formules normalisées le couple d'entiers positifs (n_1, n_2) ou :

- n_1 est la somme sur i des produit des carrés de la taille des intervalles apparaissant dans f_i ,
- n_2 est le nombre d'occurrences du \wedge dans la formule $f_1 \vee \dots \vee f_n \vee \text{faux}$

Considérons le couple d'entiers positifs (n'_1, n'_2) associé à la formule $f'_1 \vee \dots \vee f'_m \vee \text{faux}$ obtenue par application de la (i) -ème transformation

- $n'_1 < n_1$ pour $i \in \{5, 6\}$,
- $n'_1 \leq n_1$ et $n'_2 < n_2$ pour $i \in \{1, 2, 3, 4\}$.

Toute suite de transformations que l'on applique sur une formule $f \vee \text{faux}$ où f est normalisée produit donc une suite de couples (n_1, n_2) strictement décroissante lexicographiquement ; n_1 et n_2 étant positifs, cette suite converge et le nombre de transformation que l'on peut appliquer itérativement sur $f \vee \text{faux}$ est donc fini.

Après avoir épuisé toutes les transformations possibles, notre formule sera sous la forme $f_1 \vee \dots \vee f_n \vee \text{faux}$, n pouvant être nul, où les f_i sont des formules normalisées sans contrainte de la forme $x \in I$ avec I de taille supérieure à 1 (on pourrait appliquer la règle 6) ou nulle (on pourrait appliquer la règle 3), sans contrainte de la forme $(x_1, \dots, x_n) \in r$ (la valeur de chaque variable étant fixée, on pourrait appliquer soit la règle 2, soit la règle 5), sans quantificateur en tête (la valeur de chaque variable quantifiée étant fixée, on pourrait appliquer la règle 4) ni dans le corps de la formule (on pourrait appliquer la règle 1).

Les f_i sont donc toutes de la forme $(x_1 \in \{a_1\} \wedge \dots \wedge x_m \in \{a_m\} \wedge \text{vrai})$ et ont toutes le même nombre de variables (qui peut être nul) qui est le nombre de variables libre de la formule initiale.

Stratégie de résolution

Ce système de transformations termine toujours si la formule entrée est bien sous forme normalisée. Toutefois, pour des raisons d'efficacité, il est préférable de donner un ordre de priorité à chacune des transformations. L'ordre choisi est celui de l'énoncé de ces règles.

Pour notre problème de placement des pentaminos, les règles 5 et 6 ont également été ordonnées de la façon suivante

- la règle 5 de réduction des domaines a été appliquée en priorité sur les contraintes les plus simples (*i.e.* celles dont l'algorithme de calcul est de plus faible complexité par rapport au nombre de variables),
- pour la règle 6, nous avons coupé en priorité les intervalles des m_i dont le min de l'intervalle était le plus petit, essayant ainsi de placer en priorité les pentaminos en comblant les vides du pavé de plus faible abscisse, puis de plus faible ordonnée, puis de plus faible hauteur. Une fois que tous les intervalles des m_i sont réduits à un entier, toutes les autres valeurs du problème sont déterminées.

7. Résultats expérimentaux

Notre programme de résolution de contraintes a été mis en compétition avec un algorithme énumératif classique (décrit en appendice). Pour chaque combinaison de a , b et c telle que $a \times b \times c = 60$, la table suivante donne le nombre total de solutions trouvées, le temps pris par l'algorithme classique et le nombre de nœuds de son arbre de recherche, le temps pris par notre algorithme de résolution par contraintes et le nombre de coupures nécessaires au parcours de la totalité de l'arbre de recherche.

<i>dimensions du volume</i>	<i>nombre de solutions</i>	<i>algorithme énumératif</i>		<i>notre algorithme</i>	
		<i>temps</i>	<i>nœuds</i>	<i>temps</i>	<i>coupures</i>
2 × 30	0	0 s	100	0 s	0
3 × 20	2	0 s	38 792	16 s	43 080
4 × 15	368	7 s	708 508	3 m 56 s	771 425
5 × 12	1 010	33 s	3 285 507	17 m 03 s	3 532 267
1 × 6 × 10	2 339	1 m 36 s	9 918 168	46 m 27 s	9 752 037
2 × 2 × 15	0	0 s	3 474	0 s	0
2 × 3 × 10	2 × 12	23 s	1 860 942	9 m 42 s	1 559 135
2 × 5 × 6	2 × 264	36 m 5 s	114 424 641	25 h 9 m	102 516 710
3 × 4 × 5	2 × 3940	10 h 42 m	2 039 115 519	216 h 30 m	1 672 290 710

Ces deux algorithmes ont été écrits en C, compilés avec gcc v7.2 et exécutés sur un Pentium III cadencé à 350 MHz avec 512 Ko de cache, sous le système d'exploitation Linux.

Pour éviter les solutions symétriques, nous avons forcé un des pentaminos ne présentant pas de symétries internes (comme par exemple le F) à ne pas se placer dans des orientations qui peuvent être obtenues par des symétries du volume où l'on place les pentaminos. Ceci nous permet d'éviter toutes les solutions symétriques pour les problèmes bidimensionnels, on obtient par contre deux fois trop de solutions dans l'espace. Ces solutions symétriques peuvent également s'éliminer, mais le gain de temps est alors insignifiant.

8. Conclusion

Le rapport ρ des temps d'exécution entre notre algorithme et l'algorithme énumératif se situe entre 20 et 40 et on note que les plus faibles valeurs de ρ sont obtenues lorsqu'on travaille en trois dimensions ; il est de 20 dans le cas $3 \times 4 \times 5$, qui est le plus combinatoire de tous et dans ce cas, l'espace de recherche semble aussi plus petit.

Sans changer les grandes lignes de notre algorithme, nous pensons qu'il est possible de réduire ρ d'un facteur 10. En remarquant que l'algorithme passe 70% de son temps d'exécution dans la réduction de la contrainte *EntiersDistincts*, 20% de son temps dans la contrainte *Classe (f)* et 10% dans les tâches restantes, nous pourrions

- au lieu d'un algorithme en $\mathcal{O}(n^2)$ pour réduire *EntiersDistincts*, utiliser un algorithme en $\mathcal{O}(n \log n)$,
- implémenter des algorithmes incrémentaux pour les contraintes *EntiersDistincts* et *Classe (f)*,
- par une implémentation plus soignée, nous pouvons diviser le temps des tâches restantes par 5.

Mais pour réellement atténuer la complexité de notre algorithme, nous devons réduire la taille de l'espace de recherche, notamment par un meilleur traitement de la contrainte *PointsDistincts*.

9. Bibliographie

- [BEN 93] BENHAMOU F., OLDER W. J., Applying Interval Arithmetic to Real, Integer and Boolean Constraints, *Logic Programming : The ALP Newsletter*, vol. 6, 2, 1993, p. 13–14, (Extended Abstract).
- [BLE 97] BLEUZEN-GUERNALEC N., COLMERAUER A., Narrowing a n -block of sortings in $\mathcal{O}(n \log n)$, SMOLKA G., Ed., *Lecture Notes in Computer Science, Principle and Practice of Constraint Programming, CP'97*, Austria, 1997, Springer.
- [GAR 58] GARDNER M., Solid Polyominoes, *Scientific American*, , 1958.
- [GOL 54] GOLOMB S. W., Checker boards and polyominoes, *Amer. Math. Monthly*, vol. 61, 1954, p. 675–682.
- [LEC 96] LECONTE M., A bounds-based reduction scheme for constraints of difference, *Constraint-96, Second International Workshop on Constraint Based Reasoning*, Key West, Florida, 1996.

