Génération efficace de portraits de dominos

Hadrien Cambazard John Horan Eoin O'Mahony Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{h.cambazard|j.horan|e.omahony|b.osullivan}@4c.ucc.ie

Résumé

Un portrait de dominos est une approximation d'une image réalisée a partir des dominos d'un nombre donné de boites. Ce problème fut posé la première fois en 1981 et des portraits de dominos ont été obtenus depuis par des techniques de programmation linéaire, capables de produire des solutions optimales mais qui restent très lentes et incapables de passer à l'échelle. Nous proposons dans cet article une nouvelle approche qui supprime ces limitations et produit des portraits de haute qualité. Elle repose sur des techniques de recherche opérationnelle, de programmation par contraintes et de traitement d'image. Son efficacité résulte de la résolution d'un flot à coût minimum identifié comme le coeur du problème. Elle fournit des portraits qu'on peut difficilement distinguer visuellement de l'optimum dans des temps de résolution beaucoup plus faibles.

1 Introduction

En 1981, Kenneth Knowlton déposa un brevet aux Etats Unis intitulé "Representation of Designs" [4] dans lequel il proposait de produire des images monochromes en utilisant des dominos. Vingt ans plus tard, Robert Bosh expliquait à CP-AI-OR 2006 (Constraint Programming, Artificial Intelligence and Operations Research) comment les techniques d'optimisation peuvent être utilisées dans un contexte artistique de création d'images et de portraits en soulignant les qualités artistiques des images générées tout comme les défis scientifiques que représentent les problèmes sous-jacents. Bosh est connu en particulier pour ses portraits de dominos qui consistent à représenter une image par un nombre donné de boites de dominos. Des dominos à neuf points sont utilisés à cette fin offrant donc 55 dominos depuis le domino vierge 0-0 jusqu'au domino 9-9. Les dominos à neuf points offrent ainsi un bon spectre de niveaux de gris depuis le noir complet (domino vierge) jusqu'au blanc presque complet (domino 9-9).

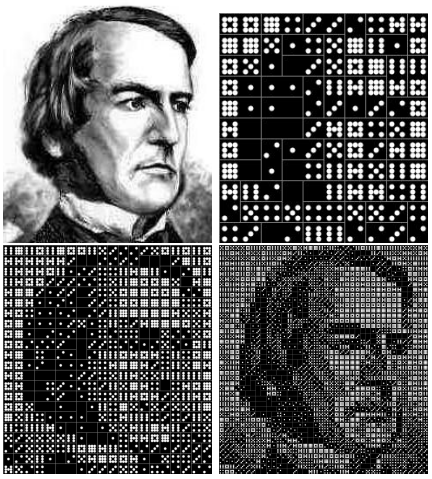


FIG. 1 – Un portrait de George Boole est présenté sur la gauche, puis une séquence de portraits de dominos générés à partir de cette image et utilisant 1, 4 et 16 boites de dominos.

Un ensemble de dominos se présente ainsi comme une palette de niveaux de gris qui est utilisée pour générer l'image. Cette palette est contrainte dans la mesure où une boite ne contient qu'un seul domino de chaque type et qu'on ne peut pas "briser" un domino en deux mais qu'on doit utiliser la pièce dans son intégralité.

Plusieurs exemples de portraits de dominos de George Boole sont présentés figure 1. Naturellement, plus le nombre de dominos dont on dispose est important, plus l'approximation de l'image est bonne. La figure 2 montre un portrait de Boole nettement plus large (fait à partir de 49 boites de dominos) sur lequel le lecteur peut distinguer chaque domino individuelle-

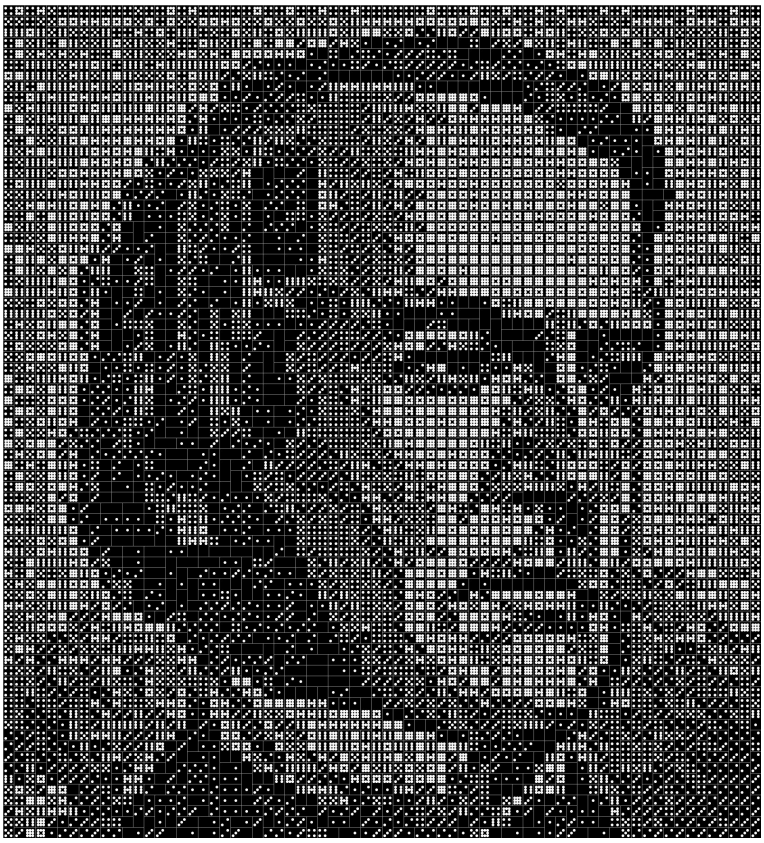


FIG. 2 – Le portrait de George Boole généré par notre approche en utilisant 49 boites de dominos a neuf points c'est à dire $49 \times 55 = 2695$ dominos.

ment.

Bosh s'est intéressé à l'obtention de portraits optimaux (nous définirons ultérieurement la notion d'optimalité pour un portrait). Sa technique qui s'appuie sur la programmation linéaire passe difficilement à l'échelle et demande des temps de résolution très importants.

Nous proposons dans ce papier une nouvelle technique de génération de portraits de dominos qui fournit des solutions non optimales mais pratiquement indifférenciables visuellement de l'optimum dans des temps de résolution très courts. Il s'agit donc de construire une approximation d'une image en utilisant un nombre donné de boites de dominos neuf-neuf [3, 2]. Nous avons adopté une approche similaire à celle de Knowlton [4] (et plus tard Knuth [5]), dans laquelle l'image est partagée en rectangles vides auxquels les dominos sont affectés. Au lieu de traiter ce problème comme un problème d'affectation traditionnel que l'on peut résoudre à l'aide de la méthode hongroise, nous proposons une formulation sous la forme d'un flot à coût minimum. Cette étape d'affectation peut dès lors s'effectuer en temps constant, offrant une très grande robustesse à l'algorithme final par rapport à la taille de la donnée (ici le nombre de boites de dominos). En revanche, l'orientation des dominos doit être déterminée pour réduire le problème complet a un problème de flot et peut donc s'avérer très éloignée de l'organisation optimale des dominos. L'algorithme que nous proposons ici, s'appuie sur une recherche locale à large voisinage qui identifie les régions du portraits pour lesquelles un changement de l'organisation des dominos est susceptible d'améliorer le rendu complet de l'image.

L'article est organisé de la manière suivante. Section 2 présente le problème du portrait de dominos et explique en détail sa formulation. Nous résumons ensuite brièvement section 3 un modèle linéaire existant pour ce problème et capable de fournir des portraits optimaux, ainsi que les approches heuristiques qui ont été étudiées pour ce problème. La section 4 décrit notre approche en deux temps et sa contribution centrale basée sur le problème du flot à coût minimum. Section 5 présente une amélioration pratique de cette technique dans un contexte de recherche locale. Enfin, les résultats sont présentés section 6.

2 Le problème du portrait de dominos

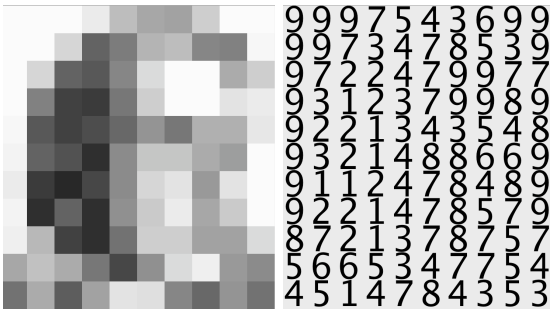
Un portrait de dominos peut être généré à partir de n'importe quelle image source. La première étape est de convertir cette image en noir et blanc en utilisant par exemple la commande UNIX *pgm*. Chaque pixel se voit donc attribué un niveau de gris compris entre 0 (noir) et 255 (blanc).

Nous considérons ici des dominos à neuf points. Il y a 55 dominos dans un jeu complet de dominos a neuf points : 10 dominos dont les deux faces ont des valeurs identiques, *i.e.* des valeurs comprises dans $\{(0,0),\ldots,(9,9)\}$ ainsi que 45 dominos non symétriques dont les valeurs sont comprises dans $\{(v_1,v_2)|v_1 \in \{0,\ldots,8\}, v_2 \in \{v_1+1,\ldots,9\}\}$. La surface couverte par un jeu unique de dominos est de 110 unités (cellules), puisque chaque domino couvre 2 unités. Pour un ensemble de s^2 dominos, l'image noir et blanc est divisée en $11s \times 10s$ cellules. La colonne et la ligne de chaque cellule sont dénotées c_i et r_i. $g_{i,j}$ dénote la valeur moyenne des niveaux de gris des pixels contenus dans une cellule, remise à l'échelle entre 0 et 9. $g_{i,j}$ correspond d'une certaine manière à la valeur parfaite du "demi" domino à placer sur cette cellule.

Chaque domino ayant deux valeurs identiques possède deux orientations possibles (étant symétrique), verticale et horizontale tandis qu'un domino non symétrique peut être placé selon quatre orientations puisqu'il peut être retourné selon chacun des axes. Pour $k = s^2$ boites de dominos, nous utiliserons une grille de taille $11s \times 10s$ qui doit être remplie par les $55 \times k$ dominos même si en pratique n'importe quelle grille avec $110 \times k$ cellules peut convenir. Les notations suivantes seront utilisées par la suite :

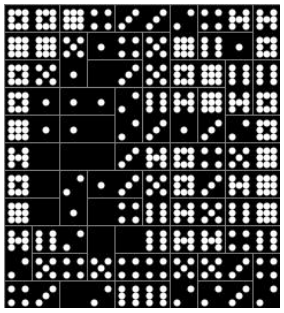
- k est le nombre d'ensembles (ou boites) de dominos et $N = 55 \times k$ est le nombre total de dominos.
- $d_i = (p_i^1, p_i^2)$ représente le domino numéro i, avec $p_i^q \in \{0,\ldots,9\}$
- g_{ij} est la valeur de la cellule (r_i, c_j) comprise entre 0 et 9. Elle représente le niveau de gris optimal qu'il faudrait idéalement avoir sur cette cellule. La matrice complète des valeurs g_{ij} est appelée par la suite la matrice des niveaux de gris.

Le coût de positionnement d'un demi-domino p_l^q sur une cellule (r_i, c_j) est égal a $(p_l^q - g_{ij})^2$. On peut noter que ce coût est quadratique et croît plus vite que l'erreur de sorte que les grandes erreurs sont largement pénalisées. Le problème est de positionner les dominos sur la grille de sorte que le coût total (la somme des coûts de chaque cellule) soit minimum et que chaque domino soit utilisé une et une seule fois. Une representation graphique du procédé de calcul de la matrice des niveaux de gris qui permet de définir les coûts de placement est donnée figure 3.



(a) Les niveaux de gris sont calculés par cellule.

(b) Les niveaux de gris sont remis à l'échelle entre $0 \ldots 9$.



(c) Un exemple de placement des dominos.

FIG. 3 – Illustration du processus permettant de spécifier le problème à partir d'une image et d'un nombre donné de dominos.

3 Un modèle de programmation linéaire

Robert Bosh propose dans [3] un modèle linéaire en nombres entiers pour ce problème. Son modèle s'appuie sur des variables booléennes spécifiant l'orientation d'un domino ainsi que la cellule sur laquelle est positionnée sa face (demi domino) de référence. Des contraintes stipulent ensuite que chaque domino doit être utilisé une fois et que chaque cellule de la grille doit être couverte par un domino. La taille des modèles obtenus de cette manière est assez importante, même pour un programme linéaire. Un modèle pour $k = 49$ comporte plus d'un million de variables et cinq mille contraintes. Bosh rapporte néanmoins que ces problèmes sont relativement faciles à résoudre mais exigent presque deux heures de calcul pour $k = 49$ par exemple.

Nous avons utilisé ce modèle comme référence dans nos expérimentations, avec une amélioration simple, non décrite par Bosh dans ses papiers, mais utilisée par Knowlton, consistant à conserver uniquement l'orientation optimale d'un domino. Un domino peut en effet être placé selon deux orientations sur une paire donnée de cellules mais l'une des orientations domine souvent l'autre en terme de coût et seule cette dernière

peut être considérée sans perdre la solution optimale. Il s'agit d'une forme de symétrie brisée au niveau de chaque domino.

Le passage a l'échelle de ce modèle est néanmoins très limité et nous présenterons une approche non optimale mais beaucoup plus efficace dans la section suivante avant de montrer comment l'appliquer dans un schéma de recherche locale à large voisinage.

4 Une générations en deux étapes

Dans son brevet original, Knowlton utilise un processus en deux étapes pour la génération de portraits. La première étape consiste à générer un positionnement initial sur la grille de dominos vierges ou rectangles de deux unités de longueur, i.e. à diviser la grille en paires de cellules adjacentes, sur lesquelles vont être placés les dominos dans la phase suivante. Il choisit comme heuristique à cette étape de maximiser la différence moyenne des deux coûts de chaque rectangle (*i.e.* les deux niveaux de gris $g_{i,j}$ présents dans les deux cellules d'un même rectangle). La deuxième étape consiste à affecter les dominos sur ces rectangles de manière à minimiser le coût provenant de la différence entre le nombre de points d'une face d'un domino et la valeur optimale de niveaux de gris attendue sur la cellule correspondante. Donald Knuth reconnaît par la suite dans cette étape un problème d'affectation [5], mais comme les deux étapes sont indépendantes, il n'y a aucune garantie d'obtenir un résultat proche de l'optimal.

Nous nous appuyions ici sur une modification de la méthode de Knowlton dans laquelle le pattern initial de rectangles est généré aléatoirement. Les dominos sont ensuite placés dans ce pattern en résolvant le problème d'affectation. Cette approche repose sur le fait que le problème devient polynomial si le *pattern* initial est connu puisque l'affectation elle même est polynomiale. On peut donc envisager une méthode de recherche au niveau des patterns seulement pour identifier le portrait optimal. En pratique nous montrerons que n'importe quel pattern aléatoire fournit une bonne borne supérieure sur le coût du portrait. Nous présentons à présent les deux étapes en détails.

4.1 Génération du pattern de rectangles

La génération d'un pattern aléatoire de rectangles sur la grille est effectuée en utilisant l'algorithme 1. Le *pattern* désigne une couverture de la grille par des rectangles de deux unités de longueur. La génération de ce pattern peut se voir comme un problème de *packing* et un exemple d'un tel pattern est donné figure 4.

Algorithme 1 procède en remplissant la grille du bas

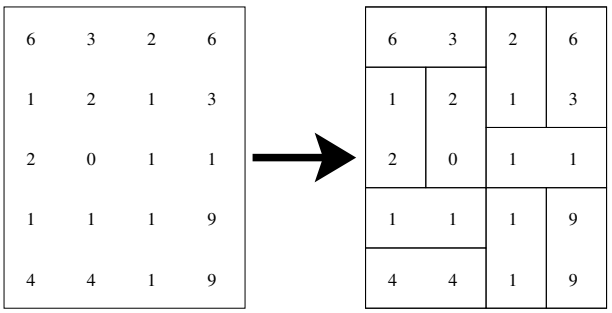


FIG. 4 – Un exemple de pattern sur la droite qui couvre la matrice des niveaux de gris sur la gauche.

vers le haut, ligne par ligne depuis la cellule gauche jusqu'à la cellule droite de chaque ligne (lignes 2 et 3 de l'algorithme). Á chaque étape, l'algorithme affecte aléatoirement un rectangle verticalement ou horizontalement (ligne 4) avant de passer par une étape de propagation. Dès qu'une cellule est entourée par trois cellules déjà couvertes par un rectangle, l'orientation du rectangle couvrant cette cellule est connue et peut être propagée (ligne 5–6). Cette propagation s'effectue jusqu'à l'obtention d'un point fixe ou l'obtention d'une. Une contradiction est levée dès qu'un nombre impair de cellules constituant un espace connexe apparaît sur la grille. En effet chaque domino doit couvrir deux cellules et tout espace connexe doit comporter un nombre pair de cellules. A l'obtention d'une contradiction, une petite région du pattern est effacée en éliminant les rectangles positionnés sur un certain nombre de lignes.

Cette approche non-déterministe est très efficace pour la génération d'un pattern aléatoire en pratique. En particulier, l'étape de redémarrage partiel à chaque contradiction est beaucoup plus efficace qu'un algorithme de *backtrack* complet quand le nombre de dominos devient important.

4.2 Résolution du problème d'affectation comme un flot à coût minimum

Une fois que le pattern est connu, le placement optimal des dominos est un problème polynomial, c'est un problème d'affectation optimale.

La figure 5 présente un exemple de ce problème d'affectation. On peut noter que le coût $c(d_i, \langle a, b\rangle)$ d'affectation d'un domino $d_i = (p_i^1, p_i^2)$ dans un rectangle donné avec deux niveaux de gris $\langle a, b\rangle$ est défini comme le meilleur coût entre les deux orientations possibles du domino :

$$c(d_i, \langle a, b\rangle) = min((p_i^1 - a)^2 + (p_i^2 - b)^2, (p_i^1 - b)^2 + (p_i^2 - a)^2). \quad (1)$$

Algorithm 1 Génération aléatoire de patterns

while il existe une cellule vide sur la grille **do**
  $i \leftarrow$ la première ligne contenant une cellule vide
  $j \leftarrow$ la première colonne telle que (i, j) est vide
  Place un rectangle aléatoirement sur la position $(i, j), (i + 1, j)$ ou $(i, j), (i, j + 1)$
  while il existe (i, j), une cellule vide avec trois voisines orthogonales couverte par un rectangle **et** toute région connexe de cellules vides est de taille paire **do**
    Place un rectangle pour couvrir (i, j) et la cellule adjacente à (i, j)
  end while
  if Il y a une région connexe de la grille avec un nombre impair de cellules sur la grille **then**
    Efface une partie de la couverture de rectangles
  end if
end while

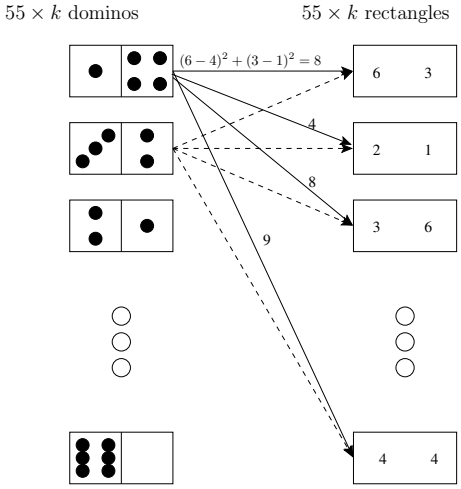


FIG. 5 – Un exemple de problème d'affectation

La résolution de l'affectation peut se faire de manière très efficace en utilisant la méthode hongroise en $O(n^3)$. Cependant, n représente ici le nombre individuel de dominos qui peut devenir rapidement très important. Un bon portrait peut exiger une centaine de boites de dominos ce qui implique 5500 dominos. La méthode hongroise est clairement incapable de passer à l'échelle dans notre contexte.

Nous proposons une nouvelle formulation de cette étape comme un problème de flot à coût minimum. On peut observer sur le graphe bipartite de la figure 5 que chaque domino sur la partie gauche du graphe est présent en k exemplaires et que de nombreux rectangles sur la partie droite ont des coûts identiques. En fait, puisque le nombre de points varie entre 0 et 9 dans chaque cellule, il n'y a que 55 paires de coûts possibles pour deux cellules adjacentes. Nous pouvons tirer parti de ces symétries dans une autre formulation du problème qui nécessite les notations suivantes :

– Nous dénoterons par *région*, un ensemble de rectangles qui possèdent des paires de coûts identiques dans le pattern. La région j correspond aux rectangles de coût $\langle j_1, j_2 \rangle$ et le nombre de tels rectangles est dénoté $capa_j$. Par ailleurs, le nombre total de régions est dénoté par $nbArea$ et $nbArea \leq 55$.
– x_{ij} est le nombre de dominos de *type* i affecté a la région j.
– $c(d_i, j)$ est le coût d'affectation du domino d_i dans la région j. $c(d_i, j)$ est le même coût que précédemment de sorte que $c(d_i, j) = c(d_i, \langle j_1, j_2 \rangle)$ comme défini par l'équation 1.

Dans le pattern donné à la figure 4, il y a six régions ($nbArea = 6$) définies par les paires de coûts suivantes : $\{\langle 6, 3\rangle, \langle 2, 1\rangle, \langle 2, 0\rangle, \langle 4, 4\rangle, \langle 1, 1\rangle, \langle 9, 9\rangle\}$.

Le problème d'affectation optimale peut être réécrit comme suit :

$$\begin{array}{ll} Minimise & \sum_{i,j} c(d_i, j) \times x_{ij} \\ & \text{subject to} \\ & \sum_j x_{ij} = k, \forall i \leq 55 \\ & \sum_i x_{ij} \leq capa_j, \forall j \leq nbArea \end{array} \quad (2)$$

La première contrainte de ce programme linéaire impose que k dominos de chaque type soit affectés. La seconde fait en sorte que pas plus de $capa_j$ dominos ne soient affectés à la région j. En pratique, il y a exactement $capa_j$ dominos pour remplir la région j puisque nous avons $\sum_j capa_j = 55 \times k$.

Ce problème est davantage compréhensible et efficacement résolu comme un problème de flot à coût minimum sur le graphe présenté à la figure 6, ou les variables x peuvent être interprétées comme la quantité de flot circulant du domino i a la région j.

Il y a deux observations importantes à faire sur cette formulation. En premier lieu, il est uniquement nécessaire de connaître la région dans laquelle on décide de placer un domino et pas où précisément dans cette région. En second lieu, nous avons uniquement besoin de savoir combien de dominos de chaque type sont affectés dans chaque région et pas où chaque domino

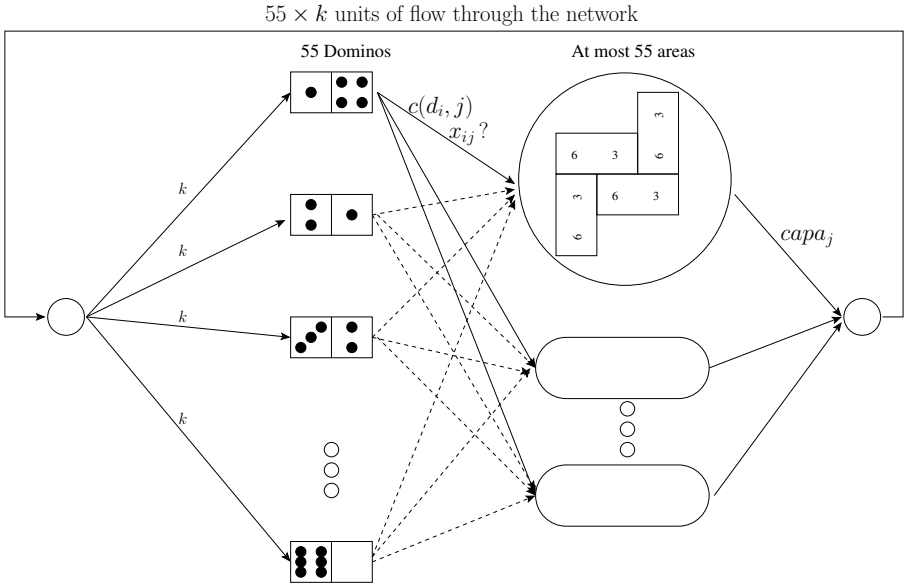


FIG. 6 – Le problème d'affectation des dominos au pattern vu comme un flot de coût minimum.

individuellement est affecté. La formulation sous la forme d'un flot à coût minimum prend ces symétries en compte et permet une résolution beaucoup plus efficace du problème d'affectation. On peut également remarquer que la taille du graphe (nombre d'arcs et de noeuds) supportant le flot est indépendante de k ; Seules les capacités et la quantité de flot augmentent avec k (ici k est la taille de la donnée) ce qui rend l'approche extrêmement robuste à l'augmentation de k.

Une fois réduit à un flot, le problème peut être résolu de nombreuses façons. Il est facile par exemple d'utiliser la formulation linéaire (voir modèle 2) qui possède la propriété d'intégralité et n'exige donc que la résolution de sa relaxation continue. D'un autre côté, de nombreux algorithmes existent pour le flot à coût minimum *e.g.* l'algorithme SSP (Successive Shortest Path (SSP) [1]) envoie la quantité maximale de flot le long du chemin le plus court entre la source et le puit a chaque itération. La complexité d'un tel algorithme est en $O(n \times max_{j \in \{1 \ldots nbArea\}}(capa_j))$ ou n est le nombre de noeuds.

Un algorithme alternatif, le *Enhanced Capacity Scaling* [1] possède une complexité en $O((m \log n)(m + n \log n))$, ou n est le nombre de noeuds et m le nombre d'arcs du graphe de flot. Cela montre notamment que la formulation sous la forme d'un flot à coût minimum permet de résoudre cette étape en temps constant puisque sa complexité est indépendante de k, *i.e* du nombre de boites de dominos utilisées pour le portrait.

5 Améliorer la recherche du pattern par recherche locale

L'utilisation d'un flot à coût minimum permet de résoudre l'étape d'affectation en temps constant. Il y a une dernière difficulté sur laquelle il faut se pencher pour pouvoir générer des portraits de qualité, c'est le choix du pattern constituant les données d'entrée du flot.

Ce pattern joue un rôle important là où les niveaux de gris ne sont pas homogènes ; le pattern dans des zones de coût uniforme n'a pratiquement aucune influence sur le coût final. Du point de vue du flot, cela signifie qu'un changement du pattern qui n'affecterait pas la taille des régions du graphe de flot, *i.e.* les valeurs $capa_j$, n'a aucun effet sur l'affectation optimale. Par conséquent, nous allons faire en sorte de perturber le pattern par recherche locale de manière a faire varier les valeurs des $capa_j$ pour tenter d'améliorer le flot.

L'algorithme que nous avons utilisé peut se voir comme une recherche locale à large voisinage [8] dans l'espace des patterns qui permettent de couvrir le portrait. Elle fonctionne de la manière suivante :

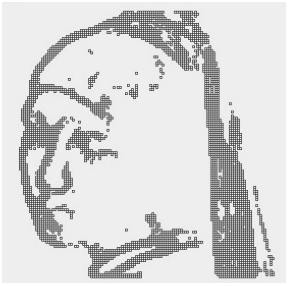
1. Les régions de la grille où les valeurs des niveaux de gris sont hétérogènes et donc où le pattern est susceptible d'être amélioré sont identifiées. Nous noterons par X l'ensemble des cellules (i, j) correspondant à ces régions.
2. Un élément $x \in X$ est sélectionné et retiré de X. Si X est vide, alors nous sélectionnons une cellule

aléatoirement.

3. M dominos situés autour de x sont retirés du pattern courant. x peut se voir comme le centre d'une nouvelle région vide dans le pattern.
4. Tous les patterns possibles qui peuvent remplir cette zone vide sont énumérés. Pour chacun de ces patterns, les valeurs de $capa_j$ sont mises à jour et le nouveau flot à coût minimum correspondant au coût de ce nouveau pattern est calculé. Notons que cette étape est une ré-optimisation globale étant donné que les dominos qui étaient affectés préalablement dans cette région peuvent maintenant être placés à un endroit complètement différent.
5. On revient à l'étape 2 (ci-dessus) tant que l'amélioration moyenne du coût sur les 20 dernières itérations reste au dessus d'un certain seuil (très petit en pratique).

Les cellules de l'ensemble X sont pondérées de manière à ce que les cellules adjacentes à celles déjà sélectionnées aient moins de chance d'être sélectionnées que celles qui sont plus indépendantes. Ce procédé vise simplement à maximiser l'impact des premières itérations et à assurer une convergence globale plus rapide.

(a) "La jeune fille à la perle" de Vermeer.



(b) La région X détectée par FAST pour $k = 225$.

FIG. 7 – La sélection de cellules intéressantes pour orienter le travail de recherche locale.

Le premier point est réalisé en utilisant un algorithme issu du traitement d'image et destiné à la détection de contour, *corner détection,* ou la détection d'éléments intéressants d'une image pour extraire certaines caractéristiques et tenter de déduire le contenu d'une image. Nous utilisons l'algorithme FAST (Features from Accelerated Segment Test) [6, 7]. Cette approche semble très adéquate au traitement de portraits dans la mesure où elle souligne les caractéristiques importantes du visage (yeux, bouche, cheveux etc...) qui sont critiques pour le rendu final du portrait. La figure 7 montre le résultat de FAST sur la "Jeune fille à la perle".

TAB. 1 – Comparaison de la méthode hongroise et du flot à coût minimum pour résoudre le problème d'affectation des dominos au pattern.

#Boites de Dominos	Temps (en secondes)	
	Flots Min	Hongrois
9	0.23	0.47
25	0.15	6.87
49	0.15	50.17
121	0.17	734.69
2,500	0.31	-
10,000	0.63	-

Le voisinage exploré est défini par tous les patterns possibles qui peuvent s'insérer dans l'emplacement vide donc dans une petite zone de $2 \times M$ cellules de la grille ($M = 15$ est utilisé en pratique dans le paramètres de l'algorithme pour les expérimentations).

L'énumération est réalisée en utilisant la propagation décrite dans l'algorithme 1 et dans un contexte de recherche arborescente complète. Enfin le problème de calcul du flot à coût minimum par rapport à de petits changements des $capa_j$ est un problème d'analyse de sensibilité sur le flot à coût minimum et peut se faire de manière incrémentale [1]. Le flot optimal est maintenu pendant la recherche locale sur les $capa_j$ reflétant les changements effectués au niveau du pattern. Ceci est rendu possible par l'efficacité du modèle de flot et son comportement incrémental.

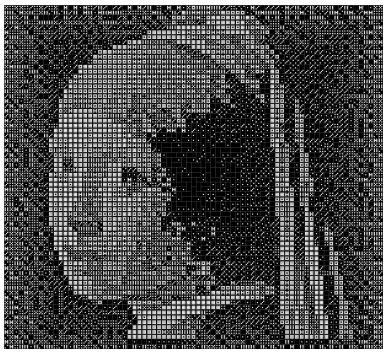
6 Expérimentations

Robert Bosh à proposé un modèle de programmation linéaire en nombres entiers (ILP) pour résoudre ce problème, que nous avons discuté précédemment et que nous utiliserons comme référence. Nous avons choisis "La jeune fille à la perle" (Figure 7) de Vermeer's pour servir de servir de base à nos expérimentations. Tous les temps rapportés sont les temps nécessaires à générer et résoudre le problème ; ils n'incluent pas le temps nécessaire à la conversion de la solution en image proprement dite.

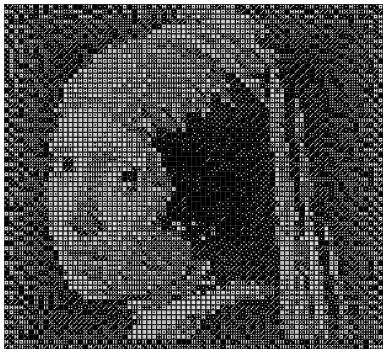
Dans un premier temps, nous comparons les performances du flot à coût minimum et de la méthode Hongroise pour mettre en évidence la robustesse de l'algorithme de flot et le gain obtenu via ce modèle (Table 1). L'algorithme de flot utilisé est le SSP, mentionné auparavant, dont l'efficacité est suffisante pour nos besoins et qui reste facile à implémenter. Clairement, la méthode Hongroise ne passe pas à l'échelle, tandis que le flot se comporte pratiquement en temps constant. La variation minime du temps de résolution (par rapport à l'augmentation de la taille) pour le flot provient du temps nécessaire à la génération du problème.

TAB. 2 – Comparaison de la qualité et la vitesse des approches ILP et génération aléatoire d'un pattern.

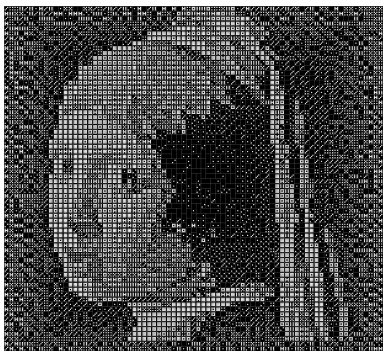
	ILP		**Pattern + flot (100 runs)**			**%Ecart**	
k	Coût	Temps (s)	Coût Moy	Coût Min	Temps (s)	Moy	Min
1	1,192	1.04	1,260	1,222	0.02	5.96	2.52
4	4,844	13.8	5,228	5,139	0.04	7.99	6.09
9	11,255	65.9	12,183	12,013	0.07	8.26	6.73
25	33,673	325.62	36,265	35,998	0.12	7.71	6.90
49	69,585	7,030.29	74,075	73,639	0.13	6.45	5.83
121	171,961	9,797.55	181,768	180,991	0.16	5.72	5.25
225	376,176	44,895.86	386,870	386,326	0.17	2.84	2.69



(a) Optimal (ILP)



(b) Pattern aléatoire + flot



(c) Recherche à large voisinage (LNS)

FIG. 8 – Comparaison de l'approche ILP, la génération d'un pattern aléatoire et l'approche complète combinant le flot et la recherche locale.

Dans un deuxième temps, nous indiquons la qualité moyenne d'un pattern aléatoire (sur 100 exécutions) pour différents nombres de dominos afin de montrer qu'un pattern aléatoire fournit déjà une borne supérieure de qualité sur le portrait par rapport aux valeurs optimales données par ILP(Table 2). Le modèle ILP est résolu avec CPLEX en utilisant le modèle de Bosch discuté a la section 3. La Table 2 présente le coût de la solution optimale ainsi que le coût moyen et le meilleur coût des 100 patterns aléatoires générés. Il est intéressant de remarquer que la qualité des portrait obtenus pour un pattern aléatoire croît avec l'augmentation du nombre de dominos. Des portraits à moins de 2.69% de l'optimum sont trouvés en utilisant 225 boites de dominos. Une différence importante entre les méthodes réside aussi dans le temps de résolution qui se chiffre en fraction de seconde pour le pattern et peut monter jusqu'à plusieurs heures pour ILP avec un grand nombre de dominos.

Finalement, la Table 3 montre les résultats du modèle ILP et l'approche basée sur le flot mise en oeuvre dans le cadre d'une recherche à large voisinage (LNS). Cette dernière est capable de produire des portraits d'excellente qualité à quelques pour-cents de l'optimum dans des temps de résolutions beaucoup plus faibles. On peut difficilement distinguer les images produites de l'optimum pour "La jeune fille à la perle". Sur la figure 8, trois portraits sont présentés utilisant 49 boites de dominos et correspondant à la valeur optimale obtenue par ILP, un pattern aléatoire et la recherche à large voisinage (LNS). Pour les tailles intéressantes (entre 9 et 225 boites de dominos), la recherche locale est nettement plus efficace que le modèle linéaire sans perte de qualité visible sur l'image (la différence est toujours en dessous de 2.45%).

Nous n'avons pas pu appliquer ILP pour des portraits de plus de 225 boites de dominos à cause de problèmes de mémoire. En revanche, des portraits constitués de $10,000$ boites de dominos ($55,000$ dominos) sont tout à fait abordables par notre technique. En fait, l'optimisation du pattern devient de moins en moins importante plus le nombre de dominos augmente. La figure 9 présente un portrait d'Alan Tu-

TAB. 3 – Comparaison de la qualité et la vitesse de l'approche ILP et l'approche de flot dans un cadre de recherche locale.

	ILP		LNS		
k	Coût Opt	Tps(s)	Coût	Tps(s)	%écart
1	1,192	1.04	1,207	8.32	1.26
4	4,844	13.80	4,903	14.00	1.22
9	11,255	65.90	11,512	14.54	2.28
25	33,673	325.62	34,498	15.72	2.45
49	69,585	7,030.29	70,977	17.66	2.00
121	171,961	9,797.55	175,669	27.34	2.16
225	376,176	44,895.86	380,408	32.71	1.13

ring généré à partir de 361 boites de dominos, et rapporte les temps de résolution nécessaires pour le flot à coût minimum et la phase de recherche locale dans sa légende. Nous avons aussi généré un portrait avec $10,000$ dominos dans des temps encore plus courts étant donné que la phase de recherche locale devient de plus en plus courte.

7 Conclusion

Nous avons proposé une nouvelle technique de résolution pour le problème de portrait de dominos qui s'appuie sur une formulation originale d'une partie du problème comme un flot à coût minimum combinée a une recherche locale à large voisinage. Un gain de plusieurs ordres de grandeur est observé sur les temps de résolutions pour obtenir des portraits a quelques pourcent de la valeur optimale. Cette approche ne fournit pas des valeurs optimales mais produit des portraits de haute qualité en quelques secondes. Elle est par ailleurs extrêmement robuste à l'augmentation de la taille du problème.

Des idées mises en oeuvre pourraient s'avérer utiles dans un contexte de problèmes de packing avec un coût de positionnement absolu. Le problème de packing est ici très facile puisqu'il n'est constitué que de rectangles de même taille, mais l'approche globale pourrait se révéler intéressante dans des applications plus complexes où les objets sont de formes différentes.

Cette application présente et manipule des algorithmes connus de Recherche Opérationnelle (Hongrois, Flot à coût minimum et analyse de sensibilité du flot), des techniques de recherche (recherche locale à large voisinage ; recherche arborescente avec propagation de contraintes) ainsi qu'un algorithme issue du traitement d'image (FAST). Elle semble donc pour cette raison particulièrement adaptée pour enseigner la Recherche Opérationnelle. Elle a été utilisée avec beaucoup de succès à la *Discovery Exhibition 2007* de Cork[1] destinée à la découverte de la science pour des adolescents âgés de 10 à 16 ans.

Acknowledgements

Ce travail fut financé par Science Foundation Ireland (Numéro de Grant 05/IN/I886). Nous remercions Robert Bosh de nous avoir donné l'inspiration pour attaquer ce problème ainsi qu'un retour intéressant.

Références

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows - Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs NJ, 1993.

[2] E. Berlekamp and T. Rogers. The mathemagician and pied puzzler : A collection in tribute to martin gardner. *AK Peters*, 1999.

[3] R. Bosch. Constructing domino portraits. *Tribute to a Mathemagician*, pages 251–256, 2004.

[4] Kenneth C. Knowlton. Representation of designs. *U.S. Patent # 4,398,890*, 1983.

[5] Donald E. Knuth. *The Stanford GraphBase : A Platform for Combinatorial Computing*. Addison-Wesley, 1993.

[6] Edward Rosten and Tom Drummond. Fusing points and lines for high performance tracking. In *ICCV*, pages 1508–1515, 2005.

[7] Edward Rosten, Gerhard Reitmayr, and Tom Drummond. Real-time video annotations for augmented reality. In *ISVC*, pages 294–302, 2005.

[8] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP*, pages 417–431, 1998.

[1] http ://www.corkcity.ie/discovery/

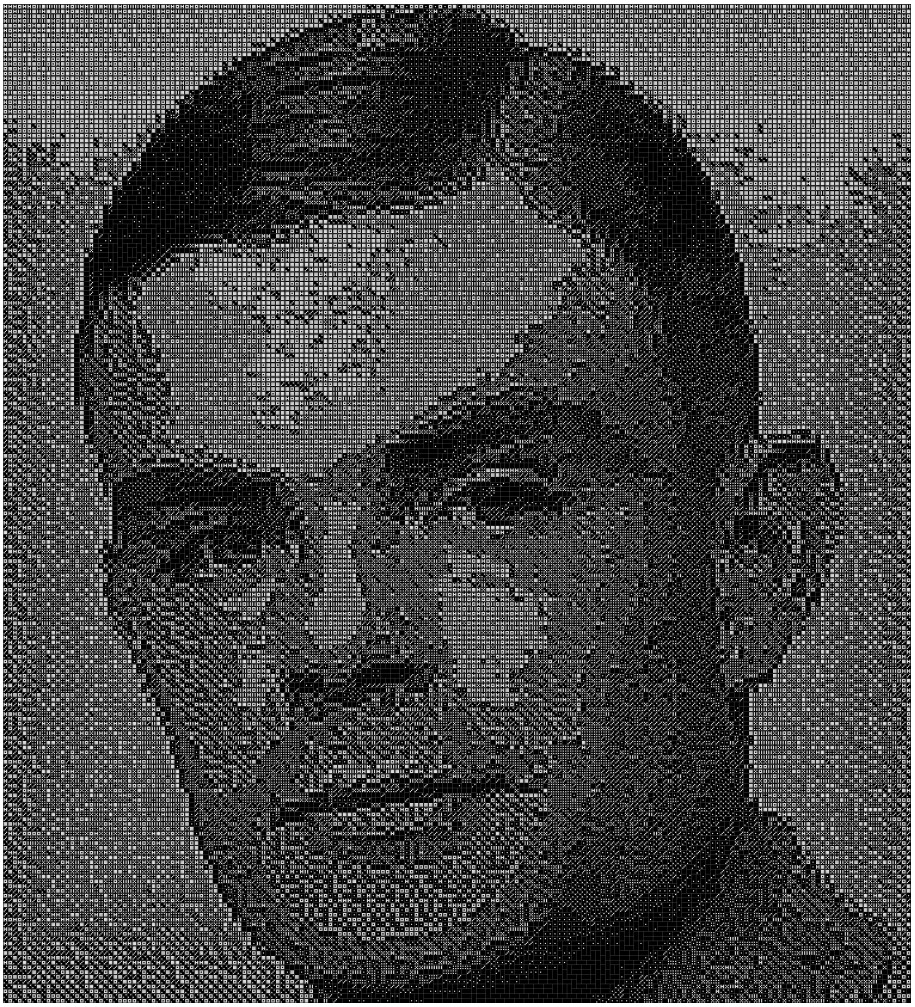


FIG. 9 – Un portrait de dominos d'Alan Turing généré par notre approche en utilisant 361 boites de dominos, i.e. 19,855 dominos. La phase de flot à coût minimum pour ce portrait demande 0.194 secondes, et la recherche locale 33.965 secondes. Nous avons aussi générés un portrait beaucoup plus large de 10,000 boites de dominos (55,000 dominos), pour lequel le flot prends 0.539 secondes et la recherche locale 26.285 secondes. Ce portrait n'est pas inclus dans le papier puisqu'il ressemble pratiquement à une image en niveaux de gris.