Visualisation de graphes de co-activité par matrices d'adjacence

Mohammad Ghoniem

Ecole des Mines de Nantes
4, rue Alfred Kastler
44300 Nantes
mghoniem@emn.fr

Jean-Daniel Fekete

INRIA Futurs, LRI
Bat. 490
91405 Orsay, Cedex
fekete@lri.fr

RESUME
Cet article décrit l'utilisation de matrices d'adjacence pour la visualisation de graphes de co-activité tels que les graphes d'interaction au sein de communautés. Nous décrivons leur utilisation pour la compréhension et l'analyse de programmes par contraintes. Nous montrons que l'usage de matrices d'adjacence pour visualiser le graphe variables-contraintes du problème étudié permet de voir la modélisation du problème et de comparer l'activité dans diverses parties du graphe au fil de la résolution.

MOTS CLES : graphes de co-activité, matrices d'adjacence, programmation par contraintes, diagramme nœuds-liens, visualisation d'information.

ABSTRACT
This paper describes the use of adjacency matrices for the visualization of co-activity graphs such as social networks. We describe their use for analysing and understanding constraint-oriented programs. We show that the use of adjacency matrices for the visualization of the variables vs. constraints graph makes it possible to visualize how the problem was modeled and to compare the activity in various regions of the graph while the problem is being solved.

KEYWORDS : graphs, adjacency matrices, constraint-oriented programming, node-link diagram, information visualization.

INTRODUCTION
Les graphes de co-activité ou d'interaction dans les communauté permettent de représenter les liens et les interactions qui peuvent exister au sein d'un groupe d'individus ou d'objets. Les relations sociales en sont un exemple et, dans le cadre de la programmation par contraintes, les graphes variables-contraintes en sont un autre.

Ces graphes ne sont pas planaires. La forte connectivité qui peut exister entre plusieurs ensembles de sommets rend donc la visualisation en nœuds-liens particulièrement inefficace, comme le montre la figure 1.

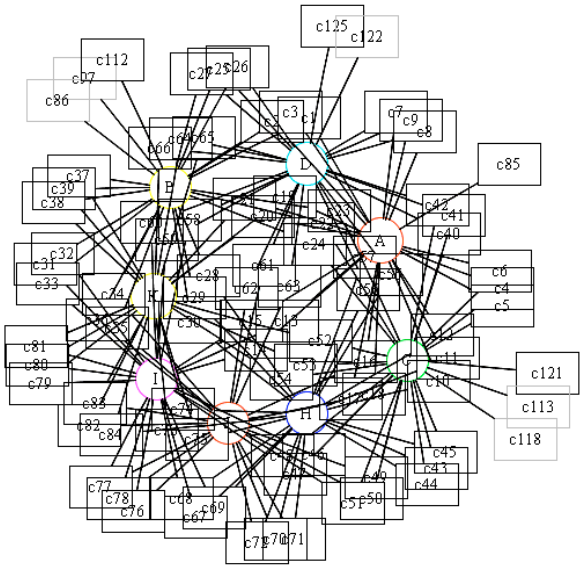

Figure 1: Le graphe contraintes-variables du problème des 8 reines sous forme de diagramme nœuds-liens.

Dans cet article, nous nous intéresserons en particulier à la visualisation de graphes de co-activité dans le cadre de la programmation par contraintes (PPC). Nous comparerons la visualisation classique de graphes sous forme de sommets et d'arêtes à la représentation des graphes sous forme de matrices d'adjacence. Après un bref descriptif de la PPC, nous illustrerons notre propos sur la résolution du problème des n-reines, problème classique décrit ci-après. La même technique de visualisation et d'interaction a été utilisée pour visualiser des relations au sein de communautés humaines.

La programmation par contraintes
La programmation par contraintes est un paradigme de programmation déclarative consistant, pour un problème donné, à décrire les contraintes qu'une solution doit satisfaire, si tant est qu'elle existe. La résolution à proprement parler est confiée à des solveurs spéciali

sés, une sorte de boîtes noires munies d'une panoplie de méthodes et d'algorithmes de résolution.

Un problème est modélisé par un ensemble de variables à valeurs discrètes dans un domaine, reliées entre elles par des contraintes. Il est donc naturel de représenter le problème par un graphe dont les sommets sont les variables et les contraintes du problème. Les arêtes du graphe reliant les contraintes aux variables qu'elles impliquent.

La résolution d'un problème en PPC comporte trois phases principales : la déclaration des variables et des contraintes initiales, la propagation des conséquences des contraintes initiales sur le domaine des variables, le *labelling* – tentatives successives d'affectation de valeur aux variables parmi les valeurs restantes dans leurs domaines respectifs. Au cours de la résolution, les domaines des variables sont modifiés, des contraintes sont ajoutées, retirées, activées ou suspendues dynamiquement. Des travaux récents [6] montrent que la mise au point et l'analyse de programmes avec contraintes nécessitent le suivi de l'évolution du graphe contraintes-variables et l'évolution des domaines des variables.

Le problème des n reines

A titre d'exemple, nous décrivons ici le problème des n reines, qui consiste à placer n reines sur un échiquier de dimensions n x n de manière à ce qu'aucune reine ne soit menacée par une autre. Une reine pouvant se déplacer en ligne, en colonne et en diagonale, il faut faire en sorte qu'aucune reine ne soit placée sur la même ligne, ni la même colonne, ni la même diagonale qu'une autre reine. Le problème est modélisé assez simplement : Les n reines sont représentées par les variables $v_1 \ldots v_n$ à valeurs dans l'intervalle [1, n] telles que pour tout i, j dans [1, n] avec $i < j$, $v_i \neq v_j$, $v_i + j-i \neq v_j$ et $v_i + i-j \neq v_j$ Pour $n > 3$, il existe au moins une solution à ce problème.

LA VISUALISATION EN NŒUDS-LIENS

Sur la figure 1, réalisée avec Graphviz [3], les huit variables du problème sont représentées par des ronds de couleur et les 125 contraintes sont réprésentées par des boîtes rectangulaires. Comme on peut le constater d'emblée, la représentation en nœuds-liens du graphe des contraintes-variables, pour un problème de taille réduite comme les huit reines, est totalement illisible. A fortiori, cette représentation n'est pas du tout adaptée pour des problèmes de taille industrielle, impliquant plusieurs centaines voire milliers de variables et contraintes.

LA VISUALISATION PAR MATRICE D'ADJACENCE

Les matrices d'adjacence sont utilisées depuis longtemps pour représenter des graphes en mathématiques et dans la théorie des graphes [4, 5]. Leur mise en œuvre dans le domaine de la visualisation d'information s'avère intéressante comme nous le verrons dans la suite de cet article.

La figure 2 montre la matrice d'adjacence du graphe des 8 reines identique à celui de la figure 1. Les 8 variables du problème sont disposées en ligne et les 125 contraintes qui leur sont appliquées sont disposées en colonne. Lorsqu'une contrainte porte sur une variable du problème, la cellule située à l'intersection de la ligne et de la colonne correspondantes est noircie.

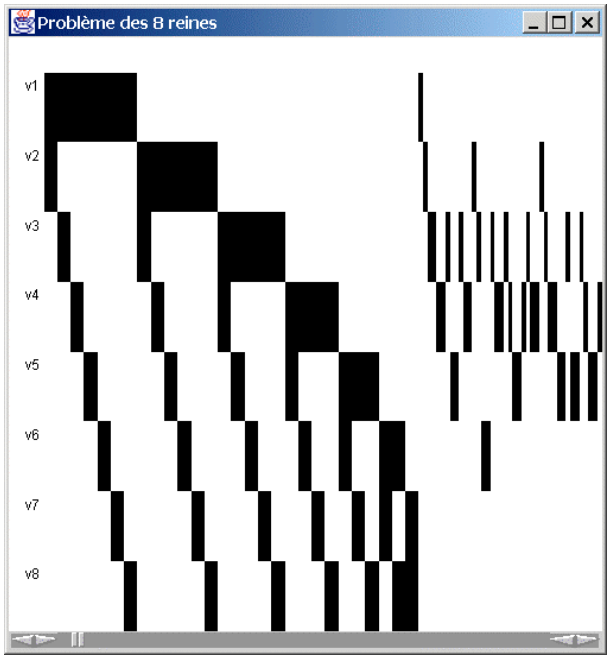

Figure 2 : Le graphe contraintes-variables du problème des 8 reines sous forme de matrice d'adjacence.

Plusieurs renseignements peuvent être obtenus à la lecture de cette représentation.

La modélisation et la résolution du problème

On peut distinguer deux types de contraintes :

1. des contraintes mettant en jeu exactement deux variables, apparaissant sur la moitié gauche de la figure. Il s'agit des contraintes « structurelles » du problème comme, par exemple, les différences deux à deux entre les 8 reines.

2. des contraintes impliquant une variable à la fois, matérialisées par les points affichés sur la moitié droite de la figure. Il s'agit d'essais d'affectation de valeurs à certaines des 8 reines par le moteur de résolution. Dans la terminologie de la programmation par contraintes, ces contraintes sont qualifiées de contraintes de « labelling ». On voit notamment qu'au cours de la résolution la pose de la première reine n'a jamais été remise en question. Cela se traduit visuellement par la présence d'une seule cellule isolée sur la ligne v1. La deuxième reine a connu trois essais alors que les reines 3, 4, 5 et 6 ont été posées puis retirées à plusieurs reprises.

Par ailleurs, nous constatons un profil en « racine carrée » de la moitié gauche de la matrice que l'on perçoit de manière plus évidente sur des instances de plus grande taille du problème des n reines. (cf. la figure 3) et qui exprime une structuration forte du problème.

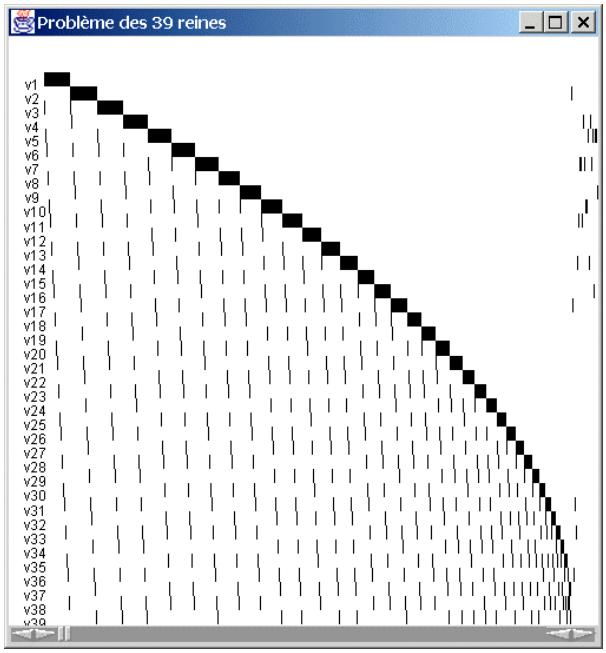

Figure 3 : Le problème des 39 reines sous forme de matrice d'adjacence mettant en jeu 2 334 contraintes.

Comparaison de divers modèles et algorithmes

Pour un problème donné, il est possible de comparer l'efficacité de diverses modélisations et algorithmes de résolution.

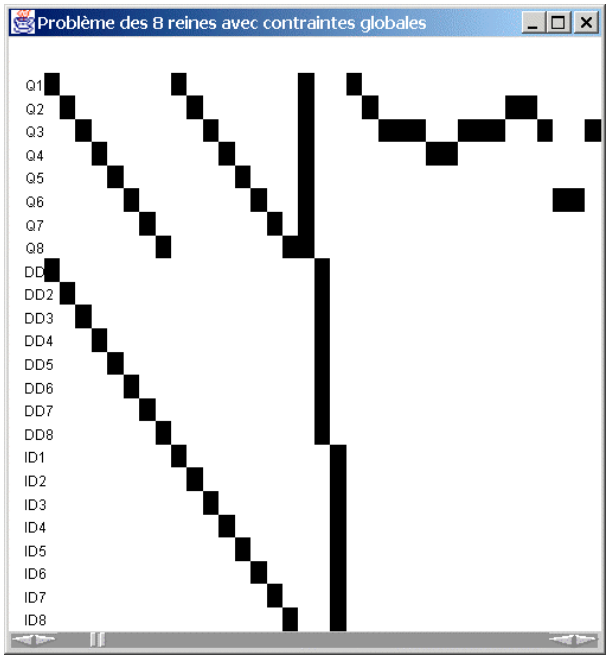

Figure 4 : Le graphe contraintes-variables du problème des 8 reines déclaré avec des contraintes globales.

Par exemple, la figure 2 illustre des contraintes de différence entre les variables prises deux à deux. Cette approche met en jeu 125 contraintes. Or, l'introduction de contraintes globales du type « all-different » (typiquement : tous les v_i sont différents deux à deux) permet une modélisation beaucoup plus efficace, modulo l'introduction de 16 variables temporaires, comme cela est illustré sur la figure 4. La matrice d'adjacence obtenue comporte 35 contraintes au lieu des 125 précédentes. De surcroît, la résolution du problème se fait plus efficacement puisque l'on fait beaucoup moins d'affectations de valeur sur la figure 4 que sur la figure 2.

La visualisation de graphes pondérés

Lors de la mise au point de programmes avec contraintes, il est utile de repérer les régions les plus actives du graphe. Concrètement, il s'agit des contraintes qui sont fréquemment activées et appliquées par le solveur. On repère ces régions grâce à la pondération des arêtes du graphe proportionnellement au nombre d'activations des contraintes.

Sur la visualisation classique en nœuds-liens, le poids d'une arête peut être exprimé par l'épaisseur du trait qui la matérialise. Cette technique reste exploitable dans le cas de graphes de petite taille. Dans le cas de la visualisation par matrice d'adjacence, nous pouvons exprimer le poids relatif des arêtes en niveaux de gris. Plus une arête du graphe est active, plus la cellule correspondante dans la matrice s'approche du noir. Inversement, une faible activité donne une couleur proche du blanc. (cf. figure 5 ci-dessous)

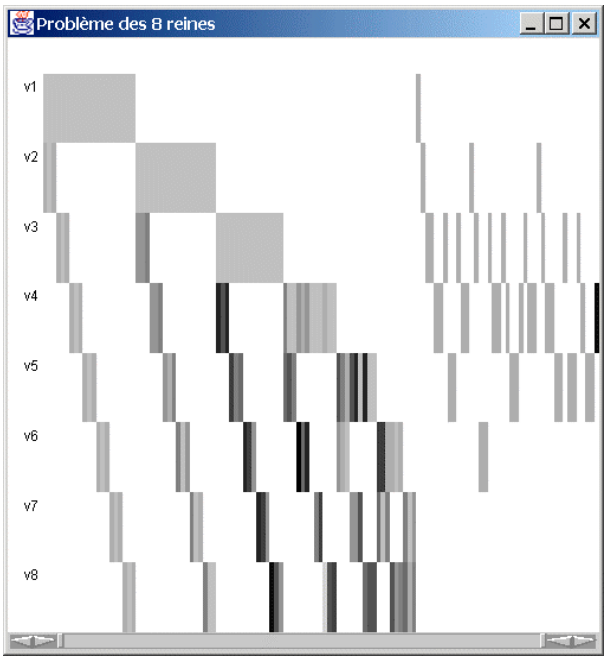

Figure 5 : Le graphe contraintes-variables du problème des 8 reines pondéré selon l'activité des arêtes sur l'ensemble de la résolution.

De cette manière, il devient possible de révéler des zones critiques du problème ou encore des régions où le

solveur tourne en rond inutilement. D'où des optimisations éventuelles de l'heuristique de résolution.

Interaction

L'activité au sein du graphe variant au fil de la résolution, il est utile d'avoir une vue chronologique du déroulement de la résolution et de l'évolution de l'activité au sein du graphe. En gardant un historique d'événements associés à chacune des arêtes du graphe, nous pouvons, grâce à une barre de défilement d'intervalle [1, 2], préciser l'amplitude et la position de la fenêtre de temps qui nous intéresse. Sur la figure 5, nous avons choisi de voir l'activité du graphe sur l'ensemble de la résolution. Mais il est également possible de choisir un intervalle de temps plus réduit et de le faire glisser d'une extrémité à l'autre de l'historique.

De plus, nous surlignons la ligne et la colonne parcourues par le pointeur afin de faciliter le repérage et la sélection dans la matrice. (cf. figure 6)

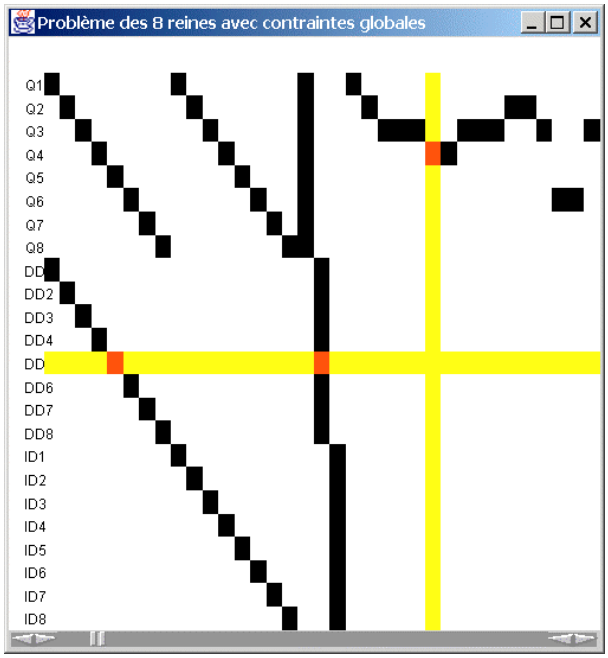

Figure 6 : Interaction avec la matrice d'adjacence.

Lorsque la matrice comporte les mêmes objets en ligne et en colonne, nous surlignons également les symétriques de la ligne et de la colonne survolées pour faciliter la détection de symétries dans le graphe.

OBJECTIFS FUTURS

En termes d'interaction, nous envisageons d'ajouter à cette visualisation un fisheye [7, 8] mono- ou multi-focus pour faciliter la visualisation et l'interaction avec de grands graphes. Nous disposons de problèmes de taille moyenne comportant environ 20 000 contraintes et près de 2 000 variables.

Non seulement ce genre de problèmes nécessitera des techniques d'interaction adaptées, mais aussi le recours à l'agrégation, notamment sur les lignes et les colonnes de la matrice d'adjacence. L'utilisateur pourra également permuter manuellement des lignes et des colonnes pour mieux mettre en évidence des clusters dans le graphe. Nous pourrons également expérimenter avec des techniques de clusterisation automatique.

CONCLUSION

Nous avons montré que l'utilisation de matrices d'adjacence pour représenter des graphes variables-contraintes dans le cadre de la PPC permet, pour un problème donné, de comparer différentes modélisations, l'efficacité de divers algorithmes de résolution et l'activité au sein du graphe. L'utilisation de matrices d'adjacence pour visualiser les graphes de co-activité a une portée générale. Ses applications dépassent le seul domaine de la programmation par contraintes. Nous tenterons dans la suite de nos travaux de valider cette approche dans le cadre plus général des relations sociales.

REMERCIEMENTS

Ce travail s'insrit dans le cadre du projet RNTL OADymPPaC (http://oadymppac.inria.fr/). Nous remercions tous nos partenaires qui ont rendu ce travail possible par la contribution de jeux de données et par des échanges d'idées enrichissants.

BIBLIOGRAPHIE

1. Ahlberg, C., Williamson, C. and Shneiderman, B. *Dynamic Queries for information exploration: an implementation and evaluation*. In CHI'92, pp. 619–626, 1992.
2. Ahlberg, C. and Truve, S.. (1995) *Exploring terra incognita in the design space of query devices*. Proc. EHCI '95.
3. AT&T Labs. *Graphviz - open source graph drawing software*. http://www.research.att.com/sw/tools/graphviz/
4. Buckley, F. and Haray, F. Distance In Graphs. Addison-Wesley, p. 117, 1990.
5. Di Battista, Eades, Tamassia and Tollis, Graph Drawing Algorithms For the Visualization of Graphs, Prentice-Hall, p. 5, 1999.
6. The Discipl Project, http://discipl.inria.fr/
7. Furnas, G.W., *The FISHEYE View: A New Look at Structured Files*. Bell Laboratories Tech. Report, Murray Hill, New Jersey, 1981.
8. Leung, Y. K. and Apperley, M. D. *A review and taxonomy of distortion-oriented presentation techniques*. ACM Trans. on Computer-Human Interaction, 1(2) pp.126–160, June 1994