

Contraintes de flot et explications

Guillaume Rochart^{*,**}

^{*}Université de Nantes

LINA/FRE CNRS 2729

2, rue de la Houssinière – BP 92208

44322 Nantes Cedex 3, France

email : grochart@emn.fr

Narendra Jussien^{**}

^{**}École des Mines de Nantes

LINA/FRE CNRS 2729

4, rue Alfred Kastler – BP 20722

44307 Nantes Cedex 3, France

email : jussien@emn.fr

1 Introduction

Les explications pour la programmation par contraintes représentent une technique intéressante pour fournir de l'information de valeur pour l'utilisateur final : savoir qu'un problème n'a pas de solution n'est pas suffisant, il faut pouvoir indiquer à l'utilisateur *pourquoi*. [4] propose de telles explications dans le cas des contraintes classiques en programmation par contraintes : inégalités, égalités, combinaisons linéaires, . . .

Les contraintes globales, qui permettent de résoudre des problèmes en proposant des algorithmes de filtrage puissants et efficaces, sont mal adaptées à ce cadre. En effet, elles nécessitent d'être modifiées pour prendre en compte l'aspect *explication* au sein de leurs algorithmes de filtrage. [1; 6] proposent de telles explications pour certaines contraintes globales : `stretch`, `all_different` et `gcc` par exemple. Mais ces contraintes possèdent des structures de données relativement simples à maintenir. Nous présentons ici comment implémenter une contrainte *expliquée* de maintien de flot. Le challenge se situe à deux niveaux. Il faut d'abord déterminer comment générer de telles explications : quelle partie du réseau est responsable de l'état actuel? Ensuite, il est nécessaire de modifier la contrainte pour qu'elle supporte non seulement l'incrémentalité (retrait de valeur des domaines de variables) mais aussi la *décrémentalité* (ajout de valeurs dans un domaine) car une telle contrainte peut être utilisée dans un cadre dynamique [4]. Ceci n'est plus trivial dès lors que la contrainte s'appuie sur une structure de donnée complexe pour maintenir un support.

2 Concepts

2.1 Explications pour un CSP

Les explications servent ici à la fois à améliorer la résolution du problème grâce à des algorithmes comme `mac-dbt` [5] et à fournir de l'information à l'utilisateur. Ainsi il est nécessaire d'expliquer chaque décision ou inférence à l'aide des contraintes utilisateurs *et*

des contraintes de choix utilisées lors de la recherche. On peut alors définir une explication comme suit :

Définition 1 (Explication) Une explication d'une inférence \mathcal{X} est un sous-ensemble des contraintes de l'utilisateur (CSP original) $\mathcal{C}' \subset \mathcal{C}$ et un ensemble de contraintes de choix (instanciations par exemple) d_1, \dots, d_n tels que : $\mathcal{C}' \wedge d_1 \wedge \dots \wedge d_n \Rightarrow \mathcal{X}$.

Une explication e_1 est dite plus précise que e_2 si et seulement si $e_1 \subsetneq e_2$.

Bien entendu, plus une explication est précise plus elle est utile. C'est pourquoi il est primordial de chercher une explication aussi précise que possible à la place d'une explication canonique qui comprendrait la justification de tous les domaines de variables de la contrainte.

2.2 La théorie des flots

Les deux définitions centrales de la théorie des flots sont le *réseau* et le *flot*. On pourra se référer à [2] pour plus de détails ou les preuves de propriétés présentées ensuite.

Définition 2 (Réseau) Un réseau est un graphe bivalué acyclique $G = (N, A, l, u)$ tel que N est un ensemble de nœuds, $A \subset N \times N$ un ensemble d'arcs orientés, l et u sont deux fonctions de A vers \mathbb{N} (on notera $u_{i,j} = u((i, j))$), et il existe deux sommets particuliers s et t (appelés source et puits) tels que pour tout nœud n il existe un chemin de s vers n et un chemin de n vers t .

Définition 3 (Flot) Un flot dans un réseau G est une fonction $f : A \rightarrow \mathbb{N}$ (on note $f_{i,j} = f((i, j))$) telle que :

- les capacités sur les arcs sont satisfaites : $\forall (i, j) \in A, l_{i,j} \leq f_{i,j} \leq u_{i,j}$,
- à chaque nœud, le flot est conservé (loi de Kirchoff) : $\forall n \in N \setminus \{s, t\}$,

$$\sum_{j/(n,j) \in A} f_{n,j} = \sum_{i/(i,n) \in A} f_{i,n}$$

On peut alors définir le flot global F de G comme suit : $F = \sum_{j/(s,j) \in A} f_{s,j}$.

De plus, pour définir des explications, la notion de *coupe* est nécessaire :

Définition 4 (Coupe) Une coupe dans un réseau G est un sous-ensemble C de N tel que $s \in C$ et $t \notin C$.

Deux types d'arcs reliant C au reste de N peuvent être alors définis des arcs entrants (orientés de $N \setminus C$ vers C), et des arcs sortants (orientés de C vers $N \setminus C$).

La capacité de la coupe C est alors définie par : $c(C) = \sum_{(i,j) \in A / i \in C, j \notin C} u_{i,j} - \sum_{(i,j) \in A / i \notin C, j \in C} l_{i,j}$.

Deux propriétés principales relient le flot et la coupe. Ce sont ces propriétés qui nous permettront de générer des explications pour la contrainte de flot.

Propriété 1 Le flot global traversant un réseau G est inférieur ou égal à la capacité de toute coupe dans ce même réseau G .

Propriété 2 (Flot max – coupe min) *Si un flot compatible existe dans un réseau G , alors le flot global maximal est égal à la capacité de la coupe de capacité minimale dans ce même réseau G .*

Ces deux propriétés permettent de relier une caractéristique globale sur le réseau (le flot global) à des propriétés locales (un sous-ensemble d’arcs), ce qui permet de connaître précisément les arcs permettant de justifier par exemple le flot global maximal dans un réseau.

3 Une contrainte de maintien de flot expliquée

3.1 Expliquer le filtrage

La contrainte de flot vérifie qu’il existe un flot vérifiant à la fois la conservation du flot et les capacités des arcs. De plus, elle propage le flot global minimal et maximal pouvant parcourir le réseau [3].

Expliquer la conservation du flot en chaque nœud ne présente pas de challenge. En effet, il ne s’agit que d’une combinaison linéaire de variables et les approches classiques [4] suffisent. Nous ne détaillerons donc pas la génération d’explication dans ce cas.

Expliquer le flot global maximal pouvant parcourir le réseau G , revient à déterminer les contraintes impliquant les valeurs courantes des bornes des arcs qui justifient la valeur maximale du flot global (et rendent donc incohérentes les valeurs supérieures).

Comme nous l’avons vu, le flot global maximal dans un réseau G est égal à la capacité de la coupe de capacité minimale dans ce même réseau (Propriété 2). Donc la justification de cette valeur maximale dépend exclusivement de cette coupe. On peut alors très facilement déduire de la propriété 1, que la capacité de cette coupe permet de justifier que tout flot global dans ce réseau est inférieur à cette capacité.

Ainsi, étant donné un réseau G , expliquer le flot global maximal traversant G , revient à justifier les capacités minimales l et maximales u rentrant en compte dans le calcul de la capacité d’une coupe de capacité minimale dans le flot G .

Soit C une telle coupe. On en déduit les explications suivantes $\forall z > c(C)$:

$$\text{expl}(x_F \neq z) = \bigcup_{(i,j) \in A / i \in C, j \notin C} \text{expl}(x_{(i,j)} \leq u_{i,j}) \cup \bigcup_{(i,j) \in A / i \notin C, j \in C} \text{expl}(x_{(i,j)} \geq l_{i,j})$$

où x_F représente la variable associée au flot global F , $x_{(i,j)}$ la variable associée à l’arc (i, j) et où $\text{expl}(x \leq \alpha) = \bigcup_{z > \alpha} \text{expl}(x \neq z)$.

Des résultats très similaires sont trouvés dans le cas du flot global minimal ou pour expliquer l’absence de flot compatible. En effet ces problèmes peuvent se ramener au problème de flot maximal [3].

3.2 Implémentation dans un cadre dynamique

De telles contraintes peuvent être utilisées dans un cadre dynamique. Ceci signifie notamment que des contraintes passées (que ce soit des contraintes du CSP original ou des contraintes de choix) peuvent être retirées dans un ordre quelconque. Ainsi, l’enregistrement d’états de la contrainte dans une pile (comme c’est le cas avec un retour-arrière

classique), n'est plus possible. Il est donc nécessaire de maintenir la consistance de la contrainte et de ses structures de données de manière incrémentale (et décrémente).

Pour ce qui est de l'incrémentalité, les techniques habituelles (maintien d'une structure et notamment d'un support) restent valides ici, puisque seul le retrait de contraintes modifie le comportement habituel. Pour la décrémente, un support est sauvegardé à chaque fois qu'un flot compatible est trouvé, et la contrainte stocke tous les événements reçus jusqu'au prochain support atteint. Ainsi, il est possible de revenir à un état stable précédent (si un support est compatible avec un ensemble de contraintes, il le sera tout autant avec moins de contraintes) et la liste des événements reçus permet de modifier le support pour retrouver un état consistant avec les contraintes introduites depuis.

4 Résultats expérimentaux

Dans de nombreux cas, une contrainte de flot expliquée naïvement ne présente pas d'intérêt car elle n'apporte pas d'information supplémentaire à l'utilisateur et ralentit notablement le système à cause du maintien de ces explications ; par contre une contrainte expliquée précisément se comporte de manière comparable à une version classique (sans explication) [3] et améliore même les temps de résolution pour certains problèmes, tout en maintenant une information utile à l'utilisateur.

5 Conclusion

Dans cet article, nous avons présenté comment générer des explications pour une contrainte globale de flot. La problématique sur cette génération est double : il est nécessaire d'étudier précisément le fonctionnement de la contrainte pour connaître les inférences à justifier et surtout savoir où trouver l'information nécessaire pour la création de l'explication ; de plus, de telles contraintes sont prévues pour être utilisées, entre autre, dans un cadre dynamique, et il est donc indispensable de pourvoir maintenir une structure de donnée ou un support de manière incrémentale mais aussi décrémente.

Références

- [1] Magnus Ågren. Tracing and explaining the execution of clp(fd) programs in sicstus prolog. Master's thesis, Uppsala University, Sweden, July 2002.
- [2] Claude Berge. *Graphes*. Gauthier-Villars, troisième édition, 1983.
- [3] Étienne Gaudin, Narendra Jussien, and Guillaume Rochart. Explained global constraints at work. Research Report 04/3/INFO, École des Mines de Nantes, France, 2004.
- [4] Narendra Jussien. The versatility of using explanations within constraint programming. Research Report 03/4/INFO, École des Mines de Nantes, France, 2003.
- [5] Narendra Jussien, Romuald Debruyne, and Patrice Boizumault. Maintaining arc-consistency within dynamic backtracking. In *CP'00*, pages 249–261, 2000.
- [6] Guillaume Rochart, Narendra Jussien, and François Laburthe. Challenging explanations for global constraints. In *CP03 Workshop on User-Interaction in Constraint Satisfaction (UICS'03)*, 2003.