From Linear Relaxations to Global Constraint Propagation

Student: Claude-Guy Quimper
Supervisor: Alejandro López-Ortiz

School of Computer Science, University of Waterloo, Canada
{cquimper, alopez-o}@uwaterloo.ca

Abstract. We present a method for propagating linear constraints. Our technique exploits the fact that the interior point method converges on a central point of the polytope. A variable assigned to an extreme point is therefore assigned to this extreme point in all solutions. We show how linear relaxations and the interior point method can be combined to prune variable domains. We also describe a class of constraints where our propagator enforces global arc consistency.

1 Introduction

Recently, many algorithms have been designed to propagate global constraints [1, 4, 9, 16, 17]. Unfortunately, some global constraints, such the AT-MOST-1 [6] constraint and the EXTENDED-GCC [9] are NP-Hard to propagate. Often, these constraints can easily be written as integer linear programs. Using linear relaxation and other techniques developed by the operation research community, we show that it is possible to efficiently propagate such constraints.

We model constraints as integer programs that we relax into linear programs. We find a solution to the relaxation using the interior point method. We finally use a convergence property of the interior point method to prune the variable domains.

This paper summarizes some results jointly obtained with Emmanuel Hebrard and Toby Walsh. Due to space restrictions, some results and all proofs are omitted.

2 Related Work

Different approaches have been used to integrate linear programming to constraint programming systems. Linear relaxations are commonly used to determine bounds in optimization problems (see for instance [18]). They are also used to detect the unsatisfiability of a problem earlier in the backtracking search or to improve the branching heuristics [8]. The "Branch & Infer" framework [7] and the "Branch & Check" [19] framework combine constraint and linear programming both for modeling and solving.

Closer to our approach, Puget [15] uses linear relaxations to propagate the MULTI-CLIQUE constraint. The MULTI-CLIQUE constraint is an integer program where where B is a binary matrix, $\boldsymbol{e}$ is the vector with all components set to 1, and Z is a constrained integer variable on which we want to enforce bounds consistency.

$$\sum_{i=1}^{n} c_i x_i = Z,\ B\boldsymbol{x} = \boldsymbol{e},\ \boldsymbol{x} \in \{0,1\}^n$$

Puget computes the dual variables of the linear relaxation and maintains bounds consistency on linear equations based on these dual variables.

3 Theoretical Background

A Constraint Satisfaction Problem (CSP) is defined by a set of variables $X = \{x_1, \ldots, x_n\}$ and a set of constraints $C = \{C_1, \ldots, C_m\}$. The domain dom(x_i) of a variable x_i defines which values can be assigned to the variable x_i. A constraint C_i on variables $Var(C_i) \subseteq X$ restricts the number of valid assignments. A solution to the problem is an assignment that satisfies all constraints. A value v has a support in dom(x_i) with respect to a constraint C_i if there exists an assignment t with $x_i = v$ that satisfies C_i. Enforcing consistency on a constraint C consists in removing unsupported values from the domains with respect to constraint C. There exist different levels of consistency. By definition, Global Arc Consistency (GAC) holds if there are no unsupported values in the domains. Similarly, bound consistency (BC) holds if $\min(\text{dom}(x_i))$ and $\max(\text{dom}(x_i))$ have a support. A consistency A is as strong as a consistency B (written $A \succeq B$) if B holds whenever A holds. A consistency A is incomparable with a consistency B if there are cases when A holds but not B and vice-versa.

4 Linear programming

A binary integer program (IP) is a set of n binary variables subject to m linear equalities and inequalities. Every binary integer program can be written in the following standard form where A is a $m \times n$ coefficient matrix and $\boldsymbol{b}$ a vector of m dimensions.

$$\left.\begin{array}{rcl} A\boldsymbol{x} & \leq & \boldsymbol{b} \\ \boldsymbol{x} & \in & \{0,1\}^n \end{array}\right\} \textbf{IP}$$

Finding a solution to a binary integer program is NP-Hard, this is why we study the following linear program (LP) which is a relaxation of **IP**.

$$\left.\begin{array}{l} A\boldsymbol{x} \leq \boldsymbol{b} \\ 0 \leq x_i \leq 1 \end{array}\right\} \textbf{P}$$

These equations describe a polytope of n dimensions, i.e. a convex shape delimited by hyper-plans. Each face of a polytope is itself a polytope of lower dimension.

Many algorithms have been developed to solve such equations. All of them use a common strategy which finds a solution by creating a polytope $\mathbf{P}'$ of higher dimension in which $\mathbf{P}$ is a face of $\mathbf{P}'$. By using some properties based on the construction of $\mathbf{P}'$,

the algorithm trivially finds a point inside the polytope. Then, by an iterative process, the point moves until it reaches the face **P** and becomes a solution.

The simplex algorithm [11] iterates through the vertices of the polytope. Karmarkar's algorithm [20], also known as the interior point method, starts from a central point in $\mathbf{P}'$ and moves the point through the polytope until it reaches the face **P**.

5 Interior Point Method and Constraint Propagation

We now show how some properties of the interior point method can be used to create constraint propagators.

Let Ω_P be the set of solutions of the linear program **P**. Variables can be partitioned into two sets: the variables B that can be assigned to positive values and variables N whose value must always be assigned to 0.

$$B = \{x_i \mid \exists \boldsymbol{x} \in \Omega_P, x_i > 0\} \tag{1}$$

$$N = \{x_i \mid \forall \boldsymbol{x} \in \Omega_P, x_i = 0\} \tag{2}$$

The bipartition $\{B, N\}$ is called the optimal partition [12]. Guler and Ye [13] proved a convergence property of the interior point method that allows to easily compute the optimal partition (B, N). The solution returned by the interior point method is central to the polytope. Therefore, if a component of $\boldsymbol{x}$ is assigned to the boundary of the polytope, it is simply because it has no choice to be on this boundary. In other words, if the interior point method returns a solution where $x_i = 0$, variable x_i is assigned to zero in all solutions.

This property can be used to design a constraint propagator. For instance, the PERMUTATION constraint can be encoded as follows on the following domains.

$$\text{dom}(a) = \{1, 2, 3\}, \text{dom}(b) = \{1, 2\}, \text{dom}(c) = \{1, 2\}$$

We create a binary variable for each value in the domains. For instance, we create the binary variable a_1 for value $1 \in \text{dom}(a)$ where $a_1 = 1$ iff a is assigned to 1.

$$\begin{array}{rrrrrrrrr}
a_1 & +a_2 & +a_3 & & & & & & = 1 \\
 & & & b_1 & +b_2 & & & & = 1 \\
 & & & & & & c_1 & +c_2 & = 1 \\
a_1 & & & b_1 & & +c_1 & & & = 1 \\
 & a_2 & & & +b_2 & & & +c_2 & = 1 \\
 & & a_3 & & & & & & = 1 \\
\multicolumn{9}{c}{0 \leq a_1, a_2, a_3, b_1, b_2, c_1, c_2 \leq 1}
\end{array}$$

The first three equations ensure that each variable is assigned to a single value while the three last rows ensure that each value is assigned to only one variable. Using the interior point method to solve this problem, we retrieve the following solution:

$$[a_1, a_2, a_3, b_1, b_2, c_1, c_2] = [0, 0, 1, 0.5, 0.5, 0.5, 0.5]$$

Using the convergence property of the interior point method, we conclude that a_1 and a_2 belong to set N and therefore values 1 and 2 should be removed from the domain of a.

There is a class of constraints where the interior point method can be used to enforce GAC. Those constraints are encoded with a linear program $A\boldsymbol{x} \leq \boldsymbol{b}$ with a binary variable x_i^v for each value v in dom(X_i), a *totally unimodular* coefficient matrix A and an integer right-hand side vector $\boldsymbol{b}$.

Unimodular matrix ([14] p. 316) "An integer matrix B is called *unimodular* if its determinant $\det(B) = \pm 1$."

Totally unimodular matrix ([14] p. 316) "An integer matrix A is called *totally unimodular* if every square, nonsingular sub-matrix of A is unimodular."

The polytope defined by a totally unimodular matrix and an integer vector $\boldsymbol{b}$ has all its vertices at integer coordinates. This occurs, for instance, when A is the incidence matrix of a network flow.

Lemma 1. *Consider the linear program $A\boldsymbol{x} \leq \boldsymbol{b}, 0 \leq x_i \leq 1$ where A is totally unimodular and $\boldsymbol{b}$ has integer components. If there are no integer solutions with $x_i = 1$ then the interior point method returns a solution with $x_i = 0$.*

Lemma 1 can be applied in other contexts. Consider the linear program $\mathbf{P_2}$ where A is totally unimodular and $\boldsymbol{b_1}$ has integer components. No assumptions are made about B, C, and $\boldsymbol{b_2}$.

$$\left.\begin{array}{c} \begin{bmatrix} A & 0 \\ B & C \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{b_1} \\ \boldsymbol{b_2} \end{bmatrix} \\ 0 \leq x_i, y_j \leq 1 \end{array}\right\} \mathbf{P_2}$$

Corollary 1. *Consider the linear program $\mathbf{P_2}$. If there are no integer solutions with $x_i = 1$ in the linear program $Ax \leq \boldsymbol{b_1}, 0 \leq x_i \leq 1$, then the interior point method returns a solution to $\mathbf{P_2}$ with $x_i = 0$.*

6 Example: The Cardinality Matrix Constraint

To illustrate our method of building constraint propagators, we use the cardinality matrix constraint introduced in Régin [17]. This constraint applies to a $m \times n$ table of integer variables X_{ij} for $1 \leq i \leq m$ and $1 \leq j \leq n$. There is a global cardinality constraint [16] (GCC) on each row and each column. The GCC restricts the number of occurrences of a value in a set of variables. For instance, GCC$(X_1, \ldots, X_n, D, l, u)$ insures that value $v \in D$ is assigned to a least $l[v]$ and at most $u[v]$ variables among X_i for $1 \leq i \leq n$. The cardinality matrix constraint is NP-Hard to propagate since it is a generalization of the quasigroup completion problem.

To propagate this constraint using our interior point method, we create the following linear relaxation. Let $x_{ij}^v = 1$ if $X_{ij} = v$ and $x_{ij}^v = 0$ otherwise. Let $lbR[i, v]$ and $ubR[i, v]$ be a lower and upper bound on the occurrence of value v on row i and let

$lbC[j,v]$ and $ubC[j,v]$ be a lower and upper bound on the occurrence of value v on column j. We have:

$$\forall\, i,j \quad \sum_{v \in \text{dom}(X_{ij})} x_{ij}^{v} = 1 \tag{3}$$

$$\forall\, i,v \;\; lbR[i,v] \leq \sum_{j=1}^{n} x_{ij}^{v} \leq ubR[i,v] \tag{4}$$

$$\forall\, j,v \;\; lbC[j,v] \leq \sum_{i=1}^{m} x_{ij}^{v} \leq ubC[j,v] \tag{5}$$

$$\forall\, i,j,v \;\; 0 \leq x_{ij}^{v} \leq 1 \tag{6}$$

As explained in Section 5, we can use the interior point method to find a solution x and remove value v from $\text{dom}(X_{ij})$ for all $x_{ij}^{v} = 0$.

To propagate this constraint, Régin uses three constraints: a GCC on each row, a GCC on each column and finally a 0/1-CARDINALITY-MATRIX for each value. This last constraint restricts the sum of each row and column of a 0/1 matrix to lie between two constant bounds. As shown by Régin, even though this constraint is redundant, enforcing GAC on 0/1-CARDINALITY-MATRIX significantly improves the pruning. We show that the interior point method can be used to enforce a stronger consistency than GAC on GCCs and GAG on the 0/1-CARDINALITY-MATRIX constraints.

Lemma 2. *Our interior point method enforces a strictly stronger ($\succ$) consistency than GAC on the* GCC*s on rows and columns and GAC on the* 0/1-CARDINALITY-MATRIX *on each value.*

7 Conclusion and Future Works

We have shown how the interior point method can be used as a core algorithm to design constraint propagators. We would like to explore in the future how other tools associated to operation research can be used to design constraint propagators. Sensitivity analysis and the analysis of dual variables may help in pruning more the variable domains.

References

1. N. Beldiceanu. Pruning for the *minimum* constraint family and for the *Number of Distinct Values* constraint family. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, Paphos, Cyprus, 2001.
2. N. Beldiceanu, M. Carlsson, and S. Thiel. Cost-fi ltering algorithms for the two sides of the sum of weights of distinct values constraint. Research Report T2002-14, Swedish Institute of Computer Science, 2002.
3. C. Bessière, A. Chmeiss, and L. Saïs. Neighborhood-basedvariable ordering heuristics for the constraint satisfaction problem. In *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming*, pages 565–569, Paphos, Cyprus, 2001. Short paper.

4. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Filtering algorithms for the nvalue constraint. In *Proceedings of the 7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Prague, Czech Republic, 2005.
5. C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The tractability of global constraints. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, Toronto, Canada, 2004.
6. C. Bessière, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *Proceedings of the 21th National Conference on Artificial Intelligence*, San jose, 2004.
7. A. Bockmayr and T. Kasper. A unifying framework for integer and fi nite domain constraint programming. Technical Report MPI-I-97-2-008, Saarbruecken, 1997.
8. C. Gomes and D. Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and Extensions*, 2002.
9. C-G. Quimper, A. L´opez-Ortiz, P. van Beek, and A. Golynski. Improved algorithms for the global cardinality constraint. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, Toronto, Canada, 2004.
10. C. Bessiere R. Debruyne. Some practicable fi ltering techniques for the constraint satisfaction problem. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.
11. G. B. Dantzig Linear Programming and Extensions Princeton University Press, 1963
12. A. J. Goldman and A. W. Tucker Theory of linear programming, in Linear Equalities and Related Systems, H. W. Kuhn and A. W. Tucker, eds., Princeton University Press, 1956.
13. O. Guler and Y. Ye *Convergence behavior of interior-point algorithms* Mathematical Programming 60, pages 215–228, 1993.
14. C. H. Papadimitriou and K. Steiglitz *Combinatorial optimization: algorithms and complexity* Prentice-Hall 1982,
15. J.-F. Puget Improved Bound Computation in Presence of Several Clique Constraints In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, Toronto, 2004.
16. J.-C. R´egin. Generalized arc consistency for global cardinality constraint. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, 1996.
17. J.-C. R´egin and C. P. Gomes The Cardinality Matrix Constraint In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming*, 2004.
18. R. Rodosek, M. G. Wallace, and M. T. Hajian. A new approach to integrating mixed integer programming and constraint logic programming. *Annals of Operations Research, to appear*, 86:63–87, 1999.
19. Erlendur S. Thorsteinsson. Branch-and-Check: A hybrid framework integrating mixed integer programming and constraint logic programming. In Toby Walsh, editor, *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming (CP-01)*, volume 2239 of *Lecture Notes in Computer Science (LNCS)*, pages 16–30. Springer-Verlag, November 2001.
20. N. Karmarkar A New Polynomial-Time Algorithm for Linear Programming *Combinatorica 4* 373–395, 1984