Global Constraints for Discrete Lattices

Alessandro Dal Palù, Univ. di Parma, IT
Agostino Dovier, Univ. di UD, IT
Enrico Pontelli, NMSU, USA

WCB 2006
September 25th, 2006
Nantes, France

Outline

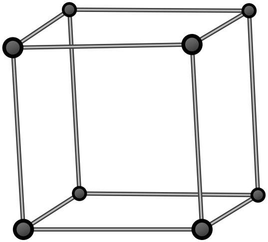
Lattices

- A *lattice* P is a non-empty, partially ordered set, such that $x \vee y$ and $x \wedge y$ exist for all $x, y \in P$

Lattices

- ~~A *lattice* P is a non-empty, partially ordered set, such that $x \vee y$ and $x \wedge y$ exist for all $x, y \in P$~~
- A *lattice* is a graph
  that can be viewed in 2D or in 3D
  and it has strong symmetries and
  repeated patterns

Lattice Models
A cubic lattice (6 connected)

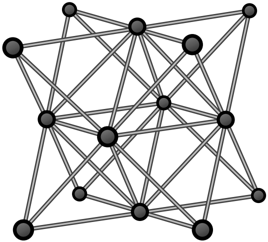


A *cubic lattice* (P, E) is defined:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N}\}$
- $E = \{(A, B) \mid A, B \in P,$
  $(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2 = 1\}$

Lattice Models

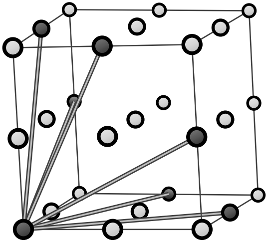
A FCC lattice (12 connected)



An FCC lattice (P, E) is defined:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N} \wedge x + y + z \text{ is even}\}$
- $E = \{(A, B) \mid A, B \in P,$
  $(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2 = 2\}$

Lattice Models
A chess knight lattice (24 connected)



A chess knight lattice (P, E) is defined:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N}\}$
- $E = \{(A, B) \mid A, B \in P,$
  $(B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2 = 5\}$

Lattice models
Encoding using finite domains

- Lattice models are used to represent simplified models for biological structures (e.g., proteins)
- The domain of a point P is a set of lattice points of the form $\langle x, y, z \rangle$
- x, y, z have values on (a finite portion of) $\mathbb{N}$
- However, poor propagation is obtained with this encoding: $P_1 \neq P_2$ is equivalent to

$$x_1 \neq x_2 \vee y_1 \neq y_2 \vee z_1 \neq z_2$$

  that introduces a disjunction
- If M is a "large enough" number, we could write:

$$M^2 x_1 + M y_1 + z_1 \neq M^2 x_2 + M y_2 + z_2$$

- In this case, e.g., AC is polynomial, but the domains size might be huge.

Lattice models
The point as central entity

- COLA (COnstraint Solving on LAttices [DDP05]) is a constraint solver for discrete lattices where the main object is the point.
- The domain of a point is a *box* identified by two points
- COLA implements a 3D bounds consistency
- Protein folding prediction using COLA improves a 3D finite domain encoding in SICStus Prolog
- COLA does not have yet global constraints
- In this paper we discuss about *which* global constraints must be considered and the theoretical complexity of their satisfiability and filtering problems
- The results are independent of COLA

[DDP05] A. Dal Palù, A. Dovier, and E. Pontelli. A New Constraint Solver for 3D Lattices and Its Application to the Protein Folding Problem. LPAR2005, LNCS 3835, pp. 48–63, 2005.

Global Constraints

- Given n variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, a *global constraint* C on the variables $X_1, \ldots, X_n$ is a subset $C \subseteq D_1 \times \cdots \times D_n$
- We are mainly interested in verifying two properties:
  - *consistency (CON): $C \neq \emptyset$*
  - *generalized arc consistency (GAC):*

$$\begin{array}{l} \forall i \in \{1, \ldots, n\} \, \forall a_i \in D_i \\ \quad \exists a_1 \in D_1 \cdots \exists a_{i-1} \in D_{i-1} \exists a_{i+1} \in D_{i+1} \cdots \exists a_n \in D_n \\ \qquad (a_1, \ldots, a_n) \in C \end{array}$$

- *filtering* is the process of removing values from the domains of variables in order to obtain an equivalent constraint which is *GAC*.
- We are interested in the complexities of these problem, and (later) in approximation algorithms for them whenever they are NP hard

alldifferent

- Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

$$\texttt{alldifferent}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i, j.\ (1 \leq i < j \leq n \wedge a_i = a_j)\}$$

- CON and GAC properties, as well as performing GAC filtering for `alldifferent` constraint can be done in polynomial time (w.r.t. sum of domains size)—Regin
- The `alldifferent` constraint has a significant role in the modeling of protein folding to express the fact that a point in the lattice cannot be used to accommodate two distinct amino acids.

contiguous

- Let $X_1, \ldots, X_n$ be variables with domains $D_1, \ldots, D_n$:

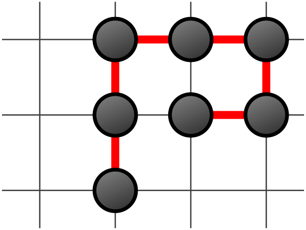
$$\begin{array}{l}\texttt{contiguous}(X_1, \ldots, X_n) = (D_1 \times \cdots \times D_n) \setminus \\ \quad \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \\ \qquad \exists\, i.\ (1 \leq i < n \ \wedge\ (a_i, a_{i+1}) \notin E)\}\end{array}$$

  where E is the set of lattice edges.
- It is equivalent to the conjunction of the $n-1$ binary constraints $C_{i,i+1}$, with $i \in \{1, \ldots, n-1\}$, such that

$$\begin{array}{rcl}C_{i,i+1} & = & (D_i \times D_{i+1}) \setminus \\ & & \{(a_i, a_{i+1}) \in D_i \times D_{i+1} \ : \ (a_i, a_{i+1}) \notin E\}\end{array}$$

- The graph induced by these constraint is acyclic. Thus, under these conditions, AC implies GAC
- GAC, CON, and filtering can be done in polynomial time
- This constraints is used in protein folding to state that amino acids contiguous in a sequence are contiguous in the lattice

Self Avoiding Walks
Basics



- A self avoiding walk (SAW) on a lattice $\langle P, E\rangle$ is simply an acyclic path.
- Even removing symmetries, there are an exponential number of self avoiding walks
- If the lattice is k-connected, and the walk length is n, there are $O((k-1)^{n-1})$ $(\Theta((k-2)^{n-1}))$ SAWs.

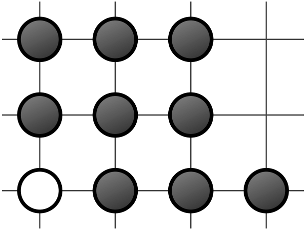
Self Avoiding Walk
Constraint Definition

- Given n variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, the global constraint `saw` is the following:

$$\texttt{saw}(X_1, \ldots, X_n) = \texttt{contiguous}(X_1, \ldots, X_n) \cap \texttt{alldifferent}(X_1, \ldots, X_n)$$

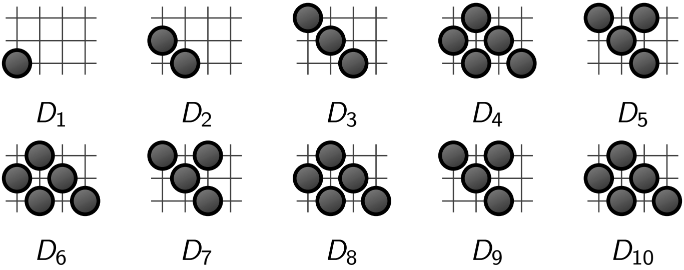
- It is equivalent to a conjunction of binary constraints (of contiguity and pairwise difference). AC works on them. AC filtering is polynomial.
- GAC of `contiguous` and of `alldifferent` (separately) is polynomial.
- You can iterate them for a fixpoint in polynomial time.

Self Avoiding Walk
example



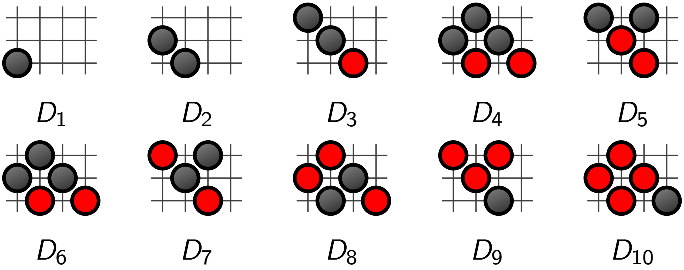
- Let $D_1 = \{\circ\}$ and $D_2 = \cdots = D_{10}$ be the set of grey points.
- Consider $\mathtt{saw}(X_1, \ldots, X_{10})$.

Self Avoiding Walk
AC propagation: 38 points



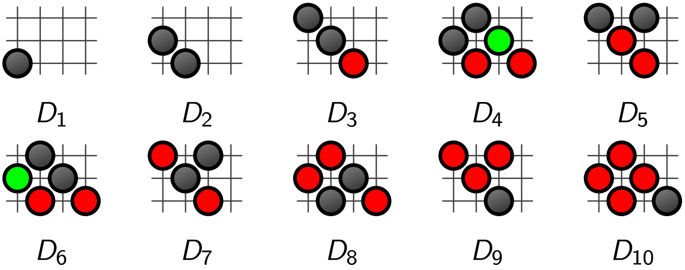
Self Avoiding Walk

Iterated contiguous and alldifferent GAC propagation — 19 points



Self Avoiding Walk

saw GAC propagation — 17 points



Self Avoiding Walk
Complexity

Theorem (NP completeness)

Determining whether a `saw` constraint is satisfiable is NP-complete. Determining whether it is GAC is NP-hard.

- Proof Sketch: reduction from Hamiltonian Circuit (HC)
- HC is NP complete on *special planar graphs*.
- We basically *map* those graphs in a lattice, stretching edges matching nodes of degree 2
- We then select two neighbor nodes α and β both of degree 2.
- Assign $D_1 = \{\alpha\}, D_n = \{\beta\}$, and introduce a suitable number of variables with domain in all the points of the lattice related to the initial graph.
- A `saw` exists iff there is a HC in the original graph.

Self Avoiding Walk
Complexity

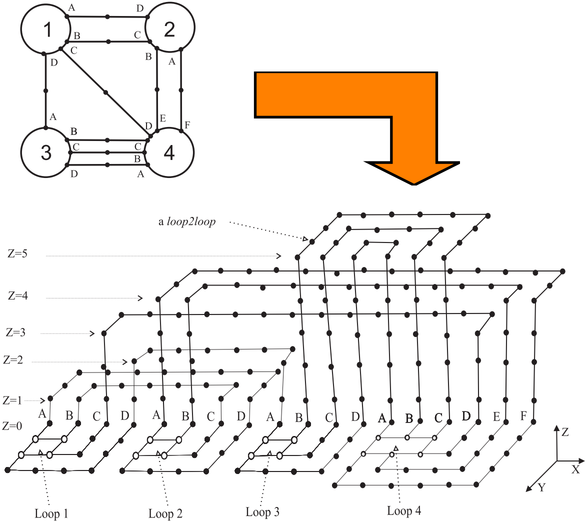
Theorem (NP completeness)

Determining whether a `saw` *constraint is satisfiable is NP-complete. Determining whether it is GAC is NP-hard.*

- Proof Sketch: reduction from Hamiltonian Circuit (HC)
- HC is NP complete on *special planar graphs*.
- We basically *map* those graphs in a lattice, stretching edges matching nodes of degree 2
- We then select two neighbor nodes α and β both of degree 2.
- Assign $D_1 = \{\alpha\}, D_n = \{\beta\}$, and introduce a suitable number of variables with domain in all the points of the lattice related to the initial graph.
- A `saw` exists iff there is a HC in the original graph.

Self Avoiding Walk

Example of reduction



alldistant

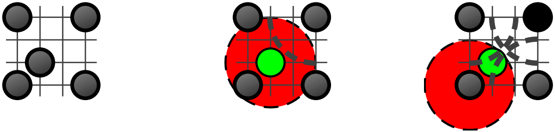
- This global constraints aims at considering the volume of the ojects to be placed in the lattice.
- Given n variables $X_1, \ldots, X_n$, with domains $D_1, \ldots, D_n$, and n numbers $c_1, \ldots, c_n$:

$$\begin{array}{l}\texttt{alldistant}(X_1, \ldots, X_n, c_1, \ldots, c_n) = (D_1 \times \cdots \times D_n) \setminus \\ \quad \{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) \ : \\ \quad\quad \exists i, j.\ 1 \leq i < j \leq n \ \wedge \ Eucl(a_i, a_j) < (c_i + c_j)\}\end{array}$$

- Namely, we are looking for a solution $X_1 = p_1, \ldots, X_n = p_n$ such that, for each pair $1 \leq i, j \leq n$, we have that p_i and p_j are located at distance at least $c_i + c_j$.

alldistant

`alldistant`$(X_1, X_2, X_3, 2, 2, 2)$: **AC vs GAC**



$D_1 = D_2 = D_3$ | AC removes one node | GAC detects unsat

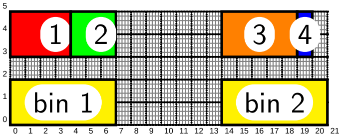
Theorem (NP completeness)

Determining whether `alldistant` *constraint is satisfiable is NP-complete. Determining whether it is GAC is NP-hard.*

Proof: reduction from Bin Packing.

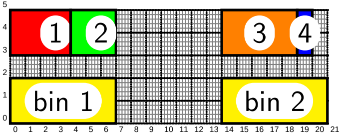
alldistant

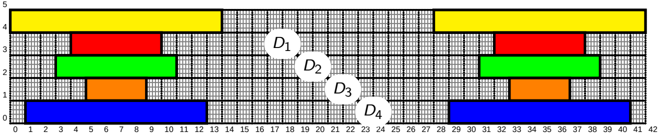
From bin packing to $\texttt{alldistant}(X_1, X_2, X_3, X_4, 4, 3, 5, 1)$



alldistant

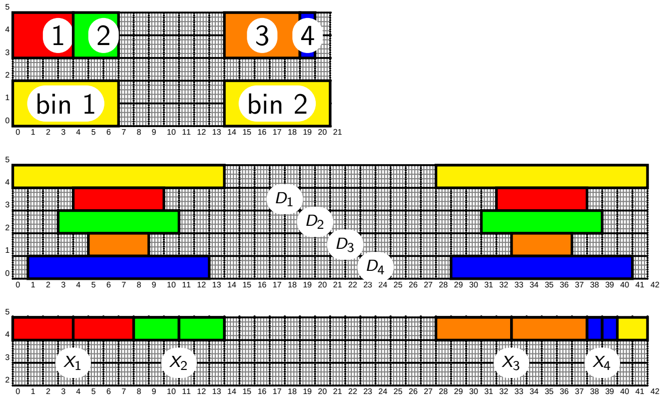
From bin packing to $\texttt{alldistant}(X_1, X_2, X_3, X_4, 4, 3, 5, 1)$





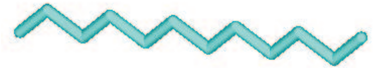
alldistant

From bin packing to $\texttt{alldistant}(X_1, X_2, X_3, X_4, 4, 3, 5, 1)$



Rigid block constraints
Basics

- In structure prediction it is helpful to use information from known substructures of the protein/molecule
- For instance, secondary structure elements of a protein (α-elices, β sheets)



- or more complex parts predicted by homology

Rigid block constraints
Definition

- Given variables $X_1, \ldots, X_n$ with domains $D_1, \ldots, D_n$ and a list $\vec{B} = B_1, \ldots, B_n$ of lattice points
- block$(X_1, \ldots, X_n, \vec{B})$ is a n-ary constraint, whose solutions are assignments of lattice points to the variables $X_1, \ldots, X_n$, that can be obtained from $\vec{B}$ modulo *translations* and *rotations*.
- Given a lattice, only few rotations r are admissible (e.g., in the cubic lattice, we have that $r = 16$, and in the $\mathtt{FCC}$ we have that $r = 24$).
- This finiteness allows us to prove that CON and GAC are polynomial.

Conclusions

- We have studied some reasonable global constraints on lattices
- `alldifferent` and `continuos` have polynomial time consistency test and GAC filtering
- `saw` and `alldistant` instead have NP complete consistency test and NP hard filtering
- Some approximations for `saw` can be obtained by AC filtering, by iterating GAC for `alldifferent` and `contiguous`, or by using the 3-saw, 4-saw, etc. of Backofen-Will
- Some approximations for `alldistant` can be obtained by AC filtering, by `alldifferent`, or sweeping (Beldiceanu and Carlsson).

Future Work

- Implementing the global constraints presented, and their complete/approximated GAC algorithms in COLA
- Studying other global constraints *between blocks* (parallelism of secondary structures, angles between them, or other already used by Krippahl and Barahona for Docking)
- Hopefully, integrating the tool in a wider prediction tool that makes use, e.g., of homology for detecting known rigid blocks (actually, we are currently doing similar things by hands).

Summary

Please don't miss the final discussion.

Don't go away.

Great things will happen!

Summary

Please don't miss the final discussion.

Don't go away.

Great things will happen!