# Global Constraints for Discrete Lattices

Alessandro Dal Palù[1], Agostino Dovier[2], and Enrico Pontelli[3]

[1] Dip. Matematica, Univ. di Parma, alessandro.dalpalu@unipr.it
[2] Dip. Informatica, Univ. di Udine, dovier@dimi.uniud.it
[3] Dept. Computer Science, New Mexico State University, epontell@cs.nmsu.edu

**Abstract.** Constraint solving on discrete lattices has gain momentum as a declarative and effective approach to solve complex problems such as protein folding determination. In particular, [8] presented a comprehensive constraint solving platform (COLA) dealing with primitive constraints in discrete lattices.
The purpose of this paper is to discuss some preliminary ideas on possible global contraints that can be introduced in a constraint system like COLA. The paper discusses various alternatives and provides preliminary results concerning the computational properties of the different global constraints.

## 1 Introduction

Discrete finite lattices are often used for approximated studies of 3D conformations of molecular structures. These models are used in particular to compute reasonable approximations of foldings of protein structures in 3D space [17, 10, 1]. Polymers are located in particular subsets of $\mathbb{N}^3$. Those subsets are often characterized by the vectors that specify the set of neighbors of each point. Lattice models like FCC and chess knight are among them.

The protein folding problem has been studied as a *Constraint Optimization Problem* with a simplified energy model [2] and with a more precise energy model [7] in the FCC lattice. In these approaches, each point $P$ of the lattice is identified by a triplet of finite domain variables $\langle P_x, P_y, P_z \rangle$, where each variable separately describes a coordinate of the point. Following this approach, propagation algorithms operate on points by means of projection of domain information on single coordinates. Considering each coordinate separately limits the power of propagation among points as single objects rather than triples of FD variables. In [8] we proposed the constraint solver called *COLA (COnstraint solver on LAttices)*, whose primitive domain contains lattice points as atomic values.

In this paper we develop a preliminary study targeting the problem of dealing with global constraints in the general context of constraint solvers on lattice domain—and specifically in COLA. Global constraints are proven constructs to facilitate the declarative encoding of problems; at the same time, they allow the programmer to express knowledge about $n$ary relationships between variables, that can be effectively employed by the search algorithm to prune infeasible parts of the solution search space. We introduce various global constraints, and

we study the complexity of their satisfiability and of the associated propagation process.

We hope this paper will inspire interest in this problem and promote discussion about suitable global constraints and implementation techniques.

## 2 Lattices and COLA

A *lattice* is a graph $(P, E)$, where $P$ is a set of 3D points $\langle x, y, z\rangle \in \mathbb{N}^3$, connected by undirected edges $(E)$.

Lattices contain strong symmetries and present regular patterns repeated in the space. If all nodes have the same degree $\delta$, then the lattice is said $\delta$-*connected*. Three examples of lattices are described next and depicted in Fig. 1.

A *cubic lattice* $(P, E)$ is defined by the following properties:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N}\}$;
- $E = \{(A, B) \mid A, B \in P, eucl(A, B) = 1\}$.

where $eucl(A, B) = (B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2$. The cubic lattice is 6-connected.

A `FCC` lattice $(P, E)$ is defined by the sets:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N} \wedge x + y + z \text{ is even}\}$;
- $E = \{(A, B) \mid A, B \in P, eucl(A, B) = 2\}$.

Thus, in a `FCC` lattice we consider the 3D space organized in cubes, with side of length 2, and where the central point of each face is also admitted. The practical rule to compute the points belonging to the lattice is to check whether the summation of the point's coordinates $(x, y, z)$ is even. Pairs of points at Euclidean distance $\sqrt{2}$ are linked and form the edges of the lattice; their distance is called *lattice unit*. Observe that, for lattice units, it holds that $|x_i - x_j| + |y_i - y_j| + |z_i - z_j| = 2$. The `FCC` lattice is 12-connected.

A *chess knight* lattice is defined as follows:

- $P = \{(x, y, z) \mid x, y, z \in \mathbb{N}\ \}$;
- $E = \{(A, B) \mid A, B \in P, eucl(A, B) = 5\}$.

Each edge allows a move like the knight on chess, 2 units in one direction, 1 in another direction, 0 in the third direction. The chess knight lattice is 24-connected.

In COLA, a *domain* $D$ is described by a pair of lattice points $\langle \underline{D}, \overline{D}\rangle$, where $\underline{D} = (\underline{D}_x, \underline{D}_y, \underline{D}_z)$ and $\overline{D} = (\overline{D}_x, \overline{D}_y, \overline{D}_z)$. The domain $D$ defines a set of lattice points in the 3D box identified by the two opposite verices $\underline{D}$ and $\overline{D}$. COLA handles the domain operations of *intersection*, *union*, and *dilation*, described in [8]. In modeling a constraint satisfaction problem, each variable represents an entity to be placed in a point in the lattice space. The variable is associated to a *domain* $D^V = \langle \underline{D}^V, \overline{D^V}\rangle$.

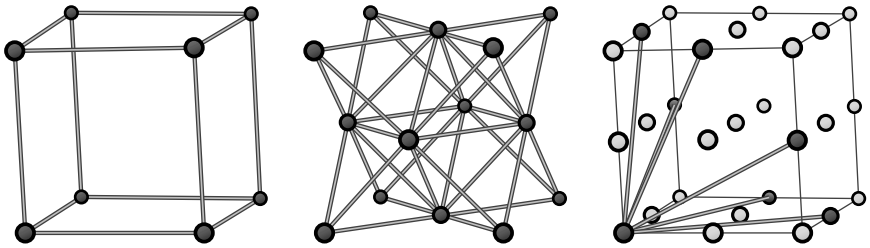

**Fig. 1.** Basic Component of a Cubic, `FCC`, and Chess Knight Lattices

Let $V_1, V_2$ denote two lattice variables, let $B_1 = Box(D^{V_1})$ and $B_2 = Box(D^{V_2})$, let $d \in \mathbb{N}$, and let $P_1, P_2$ be two lattice points. The following constraints are admitted in COLA:

$$\begin{aligned}
&\texttt{DIST_LEQ}(V_1, V_2, d) \Leftrightarrow \exists P_1 \in B_1, \exists P_2 \in B_2 \ s.t.\ norm_\infty(P_1, P_2) \leq d \\
&\texttt{EUCL}(V_1, V_2, d) \quad\ \Leftrightarrow \exists P_1 \in B_1, \exists P_2 \in B_2 \ s.t.\ eucl(P_1, P_2) = d \\
&\texttt{EUCL_LEQ}(V_1, V_2, d) \Leftrightarrow \exists P_1 \in B_1, \exists P_2 \in B_2 \ s.t.\ eucl(P_1, P_2) \leq d \\
&\texttt{EUCL_GEQ}(V_1, V_2, d) \Leftrightarrow \exists P_1 \in B_1, \exists P_2 \in B_2 \ s.t.\ eucl(P_1, P_2) \geq d
\end{aligned}$$

where $norm_\infty(A, B) = \max\{|B_x - A_x|, |B_y - A_y|, |B_z - A_z|\}$.

The work presented in [8] describes the implementation of these concepts in a concrete constraint solving system, capable of performing *bounds consistency* on the previously described constraints. The COLA solver has been applied to the problem of solving the protein folding problem in the `FCC` lattice [8], producing interesting results for proteins of length up to 100.

## 3 Global constraints

The main contribution of this paper is the identification of which *global constraints* should be introduced in COLA to enhance its declarative nature and facilitate the efficient resolution of complex problems. In particular, we expect our design to be general enough to be applicable to other constraint solvers on lattice domains. To be able to perform forms of consistency more defined than bounds consistency, we assume that variable are finite sets of lattice points (not simply boxes as in COLA).

Intuitively, a global constraint is a non-binary constraint and/or a conjunction of elementary binary (or unary) constraints. More formally, given $n$ variables $X_1, \ldots, X_n$, respectively having domains $D_1, \ldots, D_n$, a *global constraint* $C$ on $X_1, \ldots, X_n$ can be formally defined as a subset $C \subseteq D_1 \times \cdots \times D_n$. For each global constraint $C$, we are interested in verifying two properties [4]:

- *consistency (CON):* $C \neq \emptyset$

– *generalized arc consistency (GAC):*

$$\forall i \in \{1,\ldots,n\} \forall a_i \in D_i \exists a_1 \cdots \exists a_{i-1} \exists a_{i+1} \cdots \exists a_n.\ (a_1,\ldots,a_n) \in C$$

In the specific case where the constraint $C$ is binary, i.e., it involves only two variables $X_1, X_2$ with domains $D_1, D_2$, then the GAC notion is known as *arc consistency*: $C$ is arc-consistent (AC) iff $\forall a_1 \in D_1\, \exists a_2 \in D_2.\ (a_1,a_2) \in C \wedge \forall a_2 \in D_2\ \exists a_1 \in D_1.\ (a_1,a_2) \in C$.

Other properties could be of interest, e.g., generalized bounds consistency, and directional arc/bounds consistency, nevertheless, in this paper, we only deal with the two properties described above.

Related to the notion of *GAC* is the notion of *filtering*, i.e., the problem of removing values from domain variables in order to obtain an equivalent constraint which is *GAC*. If the filtering is computationally too expensive, one can run fast approximated algorithms, that eliminate some values in some domains obtaining an equivalent constraint $C'$, which is not, however, ensured to be GAC.

By definition (assuming the domains non empty), GAC implies CON. Thus, if we prove that testing GAC is polynomial, the same holds for CON. If CON is NP-complete, then GAC is NP-hard.

Let us proceed with the analysis of different global constraints in the context of lattices.

### 3.1 `alldifferent`

The `alldifferent` constraint is probably the most well-known global constraint used in constraint programming. Its semantics is given as follows: if $X_1,\ldots,X_n$ are variables with domains $D_1,\ldots,D_n$, then

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

It is well-known that testing the CON, GAC properties as well as performing GAC filtering for the `alldifferent` constraint can be performed in polynomial time. These problems can be solved, for example, by adapting algorithms for bipartite graph matching (the first contibution in this direction is [16]).

### 3.2 `close`

The `close` global constraint is used to describe the fact that a list of variables represent lattice points that are adjacent (in terms of positions in the lattice graph). Let $E$ be the set of edges in a lattice, and let $X_1,\ldots,X_n$ be a list of variables (with respectively domains $D_1,\ldots,D_n$). Then the `close` constraint can be defined as follows:

$$\texttt{close}(X_1,\ldots,X_n) = (D_1 \times \cdots \times D_n) \setminus \{(a_1,\ldots,a_n) \in (D_1 \times \cdots \times D_n) : \exists i.\ (1 \le i < n \wedge (a_i,a_{i+1}) \notin E)\}$$

Testing the GAC of `close` can be done in polynomial time. In fact, the `close` constraint is equivalent to the conjunction of the $n-1$ binary constraints of the form $C_{i,i+1}$, with $i \in \{1,\ldots,n-1\}$, such that $C_{i,i+1} = (D_i \times D_{i+1}) \setminus \{(a_i, a_{i+1}) : a_i \in D_i \ \wedge \ a_{i+1} \in D_{i+1} \ \wedge \ (a_i, a_{i+1}) \notin E\}$.

We show that arc consistency on the equivalent binary version of `close` is sufficient to imply GAC. Let us assume that for every $i \in \{1,\ldots,n-1\}$, $C_{i,i+1}$ is arc consistent. Choose $i \in \{1,\ldots,n\}$ and choose $a_i \in D_i$.

- Since $C_{i-1,i}$ is AC, then it exists $a_{i-1} \in D_{i-1}$ such that $(a_{i-1}, a_i) \in C_{i-1,i}$. Apply back the same process until $C_{1,2}$ is reached.
- Since $C_{i,i+1}$ is AC, then it exists $a_{i+1} \in D_{i+1}$ such that $(a_i, a_{i+1}) \in C_{i,i+1}$. Apply the same process forward until $C_{n-1,n}$ is reached.

Going backward and forward we have collected a set of elements such that $(a_1,\ldots,a_i,\ldots,a_n) \in C$, thus proving GAC.

Observe also that, if it exists $C_{i,i+1}$ that is not AC, then $C$ is not GAC. □

Since AC can be enforced in polynomial time, it follows that GAC can be obtained in polynomial time as well. Polynomiality of CON follows.

### 3.3 `saw`

The `saw` constraint is used to require that each assignment to the variables $X_1,\ldots,X_n$ represents a self-avoiding walk in the lattice. More formally, the constraint can be defined as follows:

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

Testing the CON property for `saw` is NP-complete—and, thus GAC is NP-hard. The proof is a reduction from the NP-complete Hamiltonian Cycle (HC) problem on special planar graphs [6]. The proof consists in two steps. First we embed the special planar graph on a 2D lattice. Then we use this encoding to define variable and domains to define an equivalent `saw` problem. We sketch here the proof.

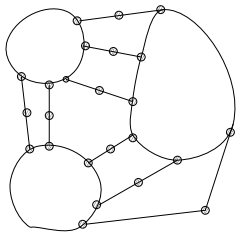

**Fig. 2.** A special planar graph (picture from [6])

Let $G = (N, E)$ be a special planar graph, where $|N| = n$. The graph $G$ is characterized by *loops* formed by nodes with degree 3 and *paths* of length 2

connecting pairs of distinct loops. It can be proved that if the number of nodes forming a loop is odd, then no HC exists for $G$. W.l.o.g. we restrict our attention to graphs with even nodes for each loop.

Planar graphs with degree at most 3 can be drawn in a $\mathbb{N}^2$ lattice (see, e.g., [11]). However, for our purposes, we need a specific mapping that allows to encode the problem with a saw constraint.

We define a new graph $G' = (N', E')$, $N' \subseteq \mathbb{N} \times \mathbb{N}$ and $E' \subseteq N' \times N'$ that maintains the properties of $G$. Let $m = |N'|$. Let us define a mapping $M : N \longrightarrow N'$ such that each node $v \in N$ is mapped to a point $M(v) = (v_x, v_y)$. The embedding preserves the planar property of $G$: each edge $e = (u, v) \in E$ is mapped into a set of edges $(p_1, p_2), (p_2, p_3) \ldots, (p_{k-1}, p_k) \in E'$ such that:

- $\forall 1 \leq i < k, eucl(p_i, p_{i+1}) = 1$,
- $M(u) = p_1$ and $M(v) = p_k$ and
- the points $p_2 \ldots p_{k-1}$ do not belong to any mapping of edges $f \neq e \in E$

For each loop $i$ made of $2n_i$ nodes we generate a gadget of $2n_i$ nodes as follows. The gadget is a rectangle $2 \times n_i$, in which every lattice point on the perimeter forms a new node and every lattice edge connected by the rectangle is included in the edges of $E'$ (see Figure 3). We associate each new node to a node of the corresponding loop maintaining a clockwise ordering.

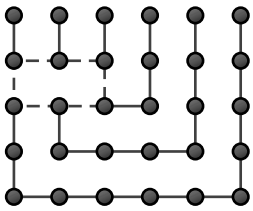

**Fig. 3.** An example of gadget for a cycle of dimension 6 (cycle is dashed)

We align the gadgets horizontally. Note that part of the linking paths are contained in the gadget (top row in Figure). To complete paths between gadgets it is sufficient to create the connections between the free pins, preserving the topology of the planar graph.

Given $G$, $G'$ always exists using a grid of size $O(n^2) \times O(n)$. Since the graph $G'$ maintains the topology (and at most introduces new nodes with degree 2 on paths), it holds that the Hamiltonian Cycle problem on $G'$ has solution iff $G$ does.

The graph $G'$ is defined over the $\mathbb{N}^2$ lattice. However, to encode a problem using `saw` constraints, we need to 'enlarge' the graph as follows: let $G''(N'', E'')$, where $N'' = \{(2x, 2y).(x, y) \in N'\} \cup \{(x + x', y + y').((x, y), (x', y')) \in E'\}$ and

$$E'' = \bigcup_{((x,y),(x\pm1,y))\in E'} \{((2x, 2y), (2x \pm 1, 2y)), ((2x \pm 1, 2y), (2x \pm 2, 2y))\} \cup \bigcup_{((x,y),(x,y\pm1))\in E'} \{((2x, 2y), (2x, 2y \pm 1)), ((2x, 2y \pm 1), (2x, 2y \pm 2))\}$$

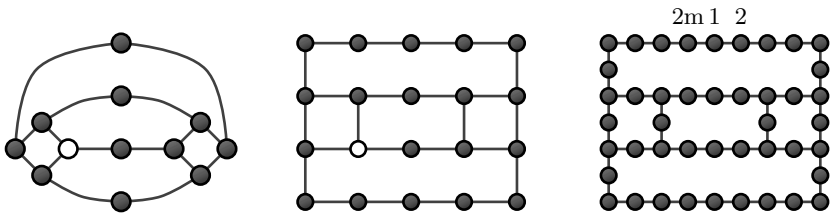


**Fig. 4.** Example of graph construction: $G$ ($n = 12$), $G'$ ($m = 20$), and $G''$ (44 nodes)

Basically the new graph $G''$ is the copy of $G'$, in which each edge is substituted by a gadget formed by (edge, new node, edge). Observe that the number of nodes of $G'$ is not $2m$. As a matter of fact, for each pair of edges outgoing from a cycle of $G$, we introduce a node that could not correspond to any Hamiton Cycle of $G$ (or $G'$).

We are now ready to reduce the HC problem for the special planar graph $G$ to the CON problem for `saw` constraints. Define the variables $X_1 \dots X_{2m}$, with domains $D(X_i) = N''$ for $3 \leq i < 2m - 1$. For variables $1, 2, 2m$, identify in $G'$ a node $v$ of degree 2. If $v$ does not exists, then $G'$ is made of disconnected cycles and thus the possible HC is trivial to detect. We identify the unique path $(u, v)(v, u')$ in $G'$, with $u, v, u' \in N'$ and the corresponding version in $G''$: $(a, b)(b, c)(c, d)(d, e)$, with $a, b, c, d, e \in N''$ and where $u$ corresponds to $a$, $v$ to $c$ and $u'$ to $e$. We define $D(X_1) = c, D(X_2) = d, D(X_{2m}) = b$. The CSP is completely defined adding the constraint $\texttt{saw}(X_1 \dots X_{2m})$.

It is now easy to prove that the CSP defined above is consistent iff the original special planar graph admits a Hamilton Cycle.

Since the reduction is polynomial, CON of `saw` global constraint is NP-complete.

### 3.4 `alldistant`

When we model biological motivated problems (e.g., protein folding) on a discrete lattice, we often observe that the `alldifferent` global constraints is not sufficiently expressive. In particular, we often require that values assigned to a group of variables are sufficiently spread in the lattice, ensuring a minimal distance between each pair of points assigned to the variables. This is required, for example, to address the fact that different aminoacids of a protein have different volume occupancy.

In the `alldistant` constraint, given $n$ variables $X_1, \dots, X_n$, with respective domains $D_1, \dots, D_n$, and $n$ numbers $c_1, \dots, c_n$, we are looking for a solution $X_1 = p_1, \dots, X_n = p_n$ such that for each pair $i, j$ we have that $p_i$ and $p_j$ are

located at distance at least $c_i + c_j$. More formally:

$$\texttt{alldistant}(X_1, \ldots, X_n, c_1, \ldots, c_n) = (D_1 \times \cdots \times D_n) \setminus \\ \left\{(a_1, \ldots, a_n) \in (D_1 \times \cdots \times D_n) : \exists i, j.\ 1 \leq i < j \leq n \wedge eucl(a_i, a_j) < (c_i + c_j)^2\right\}$$

Note that if we consider the `alldistant` with $c_1 = \frac{1}{2}, \ldots, c_n = \frac{1}{2}$ then we achieve the same effect as `alldifferent`.

We show how to reduce BIN-Packing to the consistency problem for the `alldistant` constraint. Let us consider $n$ items of size $c_1, \ldots, c_n$ and $k$ bins (bin 0, ..., bin $k-1$) of capacity $B$. W.l.o.g., assume that for all $i \in \{1, \ldots, n\}$ it holds that $c_i \leq B$ (otherwise the problem is trivially unsatisfiable).

We reduce the problem using only one dimension of the lattice (assume, e.g. that all $y$ and $z$ coordinates are fixed to 0). We consider consecutive lattice collinear points. For the sake of simplicity, we consider lattice points $(0, 0, 0)$, $(1, 0, 0)$,$(2, 0, 0), \ldots$ and we refer to them simply as $0, 1, 2, \ldots$.

For some lattice structures, it may be necessary to choose a different subset of points. The proof can be adapted by choosing, e.g., a colinear set of lattice points (some scaling of coefficients may be needed).

The reduction is defined as follows. Let us introduce $n$ lattice variables $X_1, \ldots, X_n$. For $i \in \{1, \ldots, n\}$ the domain $D_i$ is defined as

$$D_i = \bigcup_{j=0}^{k-1} [4jB + c_i..4jB + 2B - c_i]$$

Just to fix the ideas, consider the instance: $c_1 = 4, c_2 = 3, c_3 = 5, c_4 = 1, B = 7, k = 2$. Then $D_1 = [4..10] \cup [32..38]$, $D_2 = [3..11] \cup [31..39]$, $D_3 = [5..9] \cup [33..37]$, $D_4 = [1..13] \cup [29..41]$.

Intuitively, each of intervals $[0..2B], [4B..6B], [8B..10B], \ldots$ corresponds to a bin. Each assignment of the variable $X_i$ in $D_i$ is such that all values $[X_i - c_i, X_i + c_i]$ are included in exactly one of the above intervals (intuitively, the item $i$ is assigned to the bin corresponding to such interval). If the values of $X_i$ and $X_j$ are in two different intervals, then $|X_i - X_j| > 2B \geq c_i + c_j$.

We show that there is a solution for the instance of the BIN-packing problem if and only if there is a solution for the CSP

$$X_1 \in D_1, \ldots, X_n \in D_n, \texttt{alldistant}(X_1, \ldots, X_n, c_1, \ldots, c_n)$$

(in the above example, a solution of the CSP is $X_1 = 4, X_2 = 11, X_3 = 33, X_4 = 40$ from which we could deduce to put items 1 and 2 in bin 0 and items 3 and 4 in bin 1).

For one direction, assume that the CSP admits a solution $\sigma$. Consider all the variables taking values in $\sigma(X_i) \in [4Bj..4Bj + 2B]$. Assume that those variables are $X^j_1, \ldots, X^j_{m_j}$, and assume that $\sigma(X^j_1) < \cdots < \sigma(X^j_{m_j})$. This means that

- $\sigma(X^j_1) \geq 4Bj + c^j_1$ (constraint on the domain),
- $\sigma(X^j_2) \geq 4Bj + c^j_1 + (c^j_1 + c^j_2) = 4Bj + 2c^j_1 + c^j_2$ (`alldistant` constraint),

– $\sigma(X_3^j) \geq 4Bj + c_1^j + (c_1^j + c_2^j) + (c_2^j + c_3^j) = 4Bj + 2c_1^j + 2c_2^j + c_3^j$ (`alldistant` constraint),
– and so on, until $\sigma(X_{m_j}^j) \geq 4Bj + 2(c_1^j + c_2^j + c_{m_j-1}^j) + c_{m_j}^j$

Moreover, for the constraint on the domain, it holds that $\sigma(X_{m_j}^j) \leq 4Bj + 2B - c_{m_j}^j$. This means that $2(c_1^j + c_2^j + \cdots + c_{m_j}^j) \leq 2B$. Thus, put all items associated to the considered variables to bin $j$ to obtain a solution of the bin packing.

The vice versa is similar. Given a solution of the bin packing, for each bin $j$, consider the items $\texttt{item}_1^j, \texttt{item}_2^j, \ldots, \texttt{item}_{m_j}^j$ assigned in to the bin $j$. Then set $\sigma(X_1^j) = 4Bj + c_1^j, \sigma(X_2^k) = 4Bj + 2c_1^j + c_2^j, \ldots, \sigma(X_{m_j}^j) = 4Bj + 2(c_1^j + c_2^j + c_{m_j-1}^j) + c_{m_j}^j$.

NP completeness of GAC follows, as usual. For filtering it could be possible to investigate how to adapt sweep algorithms used e.g. in [3].

### 3.5 rigid block

Often local biological structures (such as helices and strands) are known, and we may wish to express the fact that a collection of points have to be located in the discrete lattice according to a predefined pattern.

This notion can be represented using another type of global constraint, called *rigid block constraint*. A rigid block defines a layout of points in the space that has to be respected by all admissible solutions. Let $X_1, \ldots, X_n$ be a list of variables (having, respectively, domains $D_1, \ldots, D_n$), and let $\boldsymbol{B} = B_1, \ldots, B_n$ a list of lattice points—that, intuitively, describe the desired layout of the rigid block. $\mathsf{block}(X_1, \ldots, X_n, \boldsymbol{B})$ is a $k$-ary constraint, whose solutions are assignments of lattice points to the variables $X_1, \ldots, X_n$, that can be obtained from $\boldsymbol{B}$ modulo translations and rotations.

More precisely, we define a *rotation* of a lattice point $p = (p_x, p_y, p_z)$ as the formula $rot(\phi, \theta, \psi)(p) = X \cdot Y \cdot Z \cdot p^T$, where

$$X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \; Y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \; Z = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Although the rotation angles $\phi, \theta, \psi$ are real valued, only few combinations of them define automorphisms on the lattice in use. The total numbers of distinct automorphisms $r$ depends on the lattice—e.g., in CUBE, we have that $r = 16$, and in the `FCC` we have that $r = 24$. We extend the definition of rotation to lists of values, $rot(\phi, \theta, \psi)(\boldsymbol{B})$, where $\boldsymbol{B}$ is a list of points and the result is a list in which every element of $\boldsymbol{B}$ is rotated according to the previous definition.

Given a list of points $\boldsymbol{B}$, we define the concept of *templates* as the set:

$$\mathsf{Templ}(\boldsymbol{B}) = \left\{ rot(\phi, \theta, \psi)(\boldsymbol{B}) \; : \; \begin{array}{c} \exists \phi, \theta, \psi. \; rot(\phi, \theta, \psi)(\boldsymbol{B}) \text{ is an} \\ \text{automorphism on the lattice} \end{array} \right\}$$

which contains the distinct 3-dimensional rotations of the points $\boldsymbol{B}$ in the lattice. Note that, for a given list of points ($\boldsymbol{B}$), the cardinality of $\mathsf{Templ}(\boldsymbol{B})$ is at most

$r$. We say that $\boldsymbol{\ell} = (\ell_x, \ell_y, \ell_z)$ is a *lattice vector* if the translation by $\boldsymbol{\ell}$ of lattice points generates an automorphism on the lattice. Note that, for some asymmetric lattices, it is possible that lattice vectors do not exist.

Let $\boldsymbol{\ell}$ be a lattice vector; with $\mathsf{Shift}[\boldsymbol{\ell}]$ we denote a mapping that translates a rigid block according to the vector $\ell$. Formally, for each $i = 1, \ldots, k$, $\mathsf{Shift}[\boldsymbol{\ell}](\boldsymbol{B})[i] = B_i + \boldsymbol{\ell}$. Shifts are used to place a template into the lattice space, preserving the orientation and the distances between points.

A rigid block constraint $\mathsf{block}(X_1, \ldots, X_n, \boldsymbol{B})$ is defined as the set:

$$\left\{(a_1, \ldots, a_n) \in D_1 \times \ldots \times D_n \,:\, \exists \boldsymbol{\ell} \exists P. \left( \begin{array}{r} P \in \mathsf{Templ}(\boldsymbol{B}) \wedge \\ \mathsf{Shift}[\boldsymbol{\ell}](P) = (a_1, \ldots, a_n) \end{array} \right) \right\}$$

With a fixed rotation of the block, CON is linear in the size of the smallest variable domain (a simple intersection of possible translations for each domain has to be performed). GAC is polynomial as well, since it is sufficient to repeat the CON test for each domain.

Propagation of this kind of constraint is studied in a wider context in [14]. Moreover, the idea of considering rigid blocks to model sub structures of proteins is also introduced in [9].

## 4 Conclusions and future work

In this paper we presented a preliminary study of various global constraints that can be used to provide declarative encoding of problems in discrete lattices. The introduction of global constraints have been motivated by problems derived from the use of constraint solving in discrete lattices to solve the protein folding determination problem.

The content of the work is highly preliminary, and a number of issues are open and deserve consideration (we hope to know more at the time of the workshop). First of all, it is interesting to investigate the relative expressive power of the different constraints—e.g., to understand the importance of having one type of global constraints vs. the others.

It is important to gain a clear understanding of the computational properties of the different global constraints, with particular attention to the complexity of verifying the properties CON and GAC and the cost of performing filtering. Throughout the paper we hinted at the high complexity (e.g., NP-completeness) of some of these problems; in such cases, it would be important to detect approximated polynomial filtering algorithms that can be effectively introduced in a constraint solver like COLA.

## References

1. R. Backofen. The protein structure prediction problem: A constraint optimization approach using a new lower bound. *Constraints*, 6(2–3):223–255, 2001.

2. R. Backofen and S. Will. A Constraint-Based Approach to Structure Prediction for SimplifiedProtein Models that Outperforms Other Existing Methods. *Proc. of ICLP 2003*, Springer Verlag, 2003.
3. N. Beldiceanu and M. Carlsson. Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint. LNCS 2239, 2001.
4. C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In Proceedings AAAI'04, San Jose CA, 2004.
5. P. Clote and R. Backofen.Computational Molecular Biology: An Introduction.John Wiley & Sons, 2001.
6. P. Crescenzi et al. On the Complexity of Protein Folding. *Journal of Computational Biology*, 5:3 423–466, 1998.
7. A. Dal Palù, A. Dovier and F. Fogolari.Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5(186), 2004.
8. A. Dal Palù, A. Dovier and E. Pontelli. A Constraint Logic Programming Approach to 3D Structure Determination of Large Protein Complexes. *Proc. of LPAR 2005*.
9. A. Dal Palù, S. Will, R. Backofen and A. Dovier. Constraint Based Protein Structure Prediction Exploiting Secondary Structure Information. *Proc. of Italian Conference on Computational Logic CILC 2004*.
10. W. E. Hart and A. Newman. The Computational Complexity of Protein Structure Prediction in Simple Lattice Models. *Handbook on Algorithms in Bioinformatics*, CRC Press, 2003.
11. G. Kant. Drawing Planar Graphs using the Canonical Ordering. Tech. Rep. RUU-CS-92-33, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands, 1992
12. L. Krippahl and P. Barahona. Applying Constraint Propagation to Protein Structure Determination. *Proc. of CP 2003*, LNCS 1713:289–302, 1999.
13. L. Krippahl and P. Barahona. PSICO: Solving Protein Structures with Constraint Programming and Optimisation. *Constraints* 7:317–331, 2002.
14. L. Krippahl and P. Barahona. Propagating N-Ary Rigid-Body Constraints. *Proc. of CP 2003*, LNCS 2833:452–465, 2003.
15. L. Krippahl and P. Barahona. Applying Constraint Programming to Rigid Body Protein Docking. *Proc. of CP 2005*, LNCS 3709:373–387, 2005.
16. J.-C. Regin. A filtering algorithm for constraints of difference in CSPs. R.R. LIRMM 93–068, 1993.
17. J. Skolnick and A. Kolinski. Reduced models of proteins and their applications. *Polymer*, 45:511–524, 2004.