Enhancing the Computation of Approximate Solutions of the Protein Structure Determination Problem Through Global Constraints for Discrete Crystal Lattices*

Alessandro Dal Palù
Dip. Matematica, Univ. di Parma,
alessandro.dalpalu@unipr.it

Agostino Dovier
Dip. Informatica, Univ. di Udine,
dovier@dimi.uniud.it

Enrico Pontelli
Dept. Computer Science, New Mexico State University,
epontell@cs.nmsu.edu

Abstract

This paper investigates alternative global constraints *that can be introduced in a constraint solver over discrete crystal lattices. The objective is to enhance the efficiency of lattice solvers in dealing with the construction of approximate solutions of the protein structure determination problem. The paper discusses various alternatives and provides preliminary results concerning the computational properties of the different global constraints.*

1. Introduction

Discrete (crystal) lattices are formal and discrete models of the space [17]. They have been frequently adopted to provide formal descriptions of crystal structures of chemical compounds, and they have provided valuable insights in the study of approximated 3D conformations of molecular structures. A field of application is the approximations of foldings of protein structures in the 3D space [17, 2, 1, 12]. In this context, polymers are laid out in regularly organized subsets of $\mathbb{N}^3$. These subsets are described concisely by the vectors that specify the neighbors of each point.

The protein structure determination problem in the context of discrete lattice structures has been studied as a *Constraint Optimization Problem* in the Face Centered Cubic (FCC) lattice—using a simplified pairwise energy model in [2] and a more precise energy model in [9]. In these approaches, the position of each amino acid is described by a triplet of *finite domain (FD) variables* (P_x, P_y, P_z), where each variable describes one coordinate of the point in the 3D space. Propagation algorithms operate on points by means of projection of domain information on individual axis. Unfortunately, experimental studies have shown that applying constraints separately to the individual coordinates significantly limits the power of *propagation* among points as *individual objects*, rather than as triples of FD variables. This limits the effective propagation of domain changes performed along one axis to the other dimensions of the space. To address this problem, researchers have developed ad-hoc constraint solvers (e.g., [10]), whose underlying primitive domain views lattice points as *atomic values*.

In this paper we investigate the problem of introducing *global constraints* in the general context of constraint solving on discrete lattices. Global constraints allow the programmer to express knowledge about complex relationships between variables, that can be effectively employed by the search algorithm to prune infeasible parts of the solution search space. We introduce different global constraints, structured according to the specific properties of finite lattices, and motivated by the approximated protein structure determination problem.

Some specialized types of global constraints in lattice spaces have already been considered in [10, 2], while some global constraints in the *real 3D space ($\mathbb{R}^3$)* have been used in [13, 14]. Nevertheless, the literature does not provide a comprehensive study of discrete lattice global constraints relevant to the protein structure determination problem. In this paper we identify some relevant global constraints and analyze their computational properties, with respect to both the satisfiability problem and the propagation process. The study is also completed by some preliminary pragmatic considerations concerning the implementation of global constraints in the context of an actual lattice constraint solver.

*Work partially supported by FIRB Project RBNE03B8KK, by PRIN Project 2005015491, and by NSF grants CNS-0220590 and HRD-0420407.

The main contribution of this paper is the identification and computational characterization of *which* global constraints should be introduced in a discrete lattice constraint solver, to enhance its declarativeness and facilitate the efficient resolution of the protein structure determination problem. We expect our design to be applicable to any constraint solver on discrete lattices (e.g., [2, 10]), as well as to other systems that use constraint-based technology to address the protein structure determination problem.

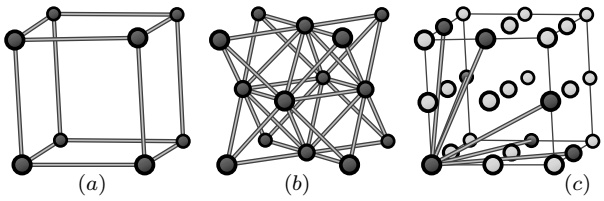

Figure 1. Components of a Cubic, FCC, and Chess Knight Lattice

2. Crystal Lattices and the Protein Structure Problem

Let us start by reviewing the formal definitions of discrete crystal lattices and their use in the protein structure determination problem.

A *discrete crystal lattice* (or, simply, a *lattice*) is a graph (P, E), where P is a set of triples $(x, y, z) \in \mathbb{N}^3$, connected by undirected edges (E). Given $A = (x, y, z) \in P$, we will denote x, y, z with A_x, A_y, A_z, respectively. Lattices contain strong symmetries and regular patterns repeated in the space. If all the nodes have the same degree δ, then the lattice is said to be δ-*connected*. Let us introduce the following preliminary definitions. The *square Euclidean distance (sqeucl)* between two 3D points is

$$sqeucl(A, B) = (B_x - A_x)^2 + (B_y - A_y)^2 + (B_z - A_z)^2$$

while the *norm infinity distance (norm$_\infty$)* is

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

Three examples of lattices are described next.

A *cubic lattice* (P, E) (Fig. 1(a)) is defined as

$$\begin{aligned} P &= \{(x, y, z) \mid x, y, z \in \mathbb{N}\} \\ E &= \{(A, B) \mid A, B \in P, sqeucl(A, B) = 1\}. \end{aligned}$$

The cubic lattice is 6-connected.

A FCC lattice (P, E) is characterized by

$$\begin{aligned} P &= \{(x, y, z) \mid x, y, z \in \mathbb{N} \wedge x + y + z \text{ is even}\} \\ E &= \{(A, B) \mid A, B \in P, sqeucl(A, B) = 2\}. \end{aligned}$$

The FCC lattice is organized in cubes, each side having length 2, and where the center point of each face is also admitted. The FCC lattice is 12-connected—see Fig. 1(b).

A *chess knight* (Fig. 1(c)) lattice is characterized by

$$\begin{aligned} P &= \{(x, y, z) \mid x, y, z \in \mathbb{N}\} \\ E &= \{(A, B) \mid A, B \in P, sqeucl(A, B) = 5\}. \end{aligned}$$

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

The lattice can be used as the underlying structure for a constraint domain; let us consider the approach explored in *COLA* [10]. In COLA, a *domain* D is described by a pair of lattice points $\langle low(D), up(D)\rangle$. The domain D defines a set of lattice points in the 3D *box*: $\{A \in P : low(D)_x \leq A_x \leq up(D)_x, low(D)_y \leq A_y \leq up(D)_y, low(D)_z \leq A_z \leq up(D)_z\}$. Each variable V represents an amino acid to be placed in a point in the lattice space, and it is associated to a domain $D^V = \langle low(D^V), up(D^V)\rangle$. Various primitive constraints have been provided, such as:

DIST_LEQ$(V_1, V_2, d) \Leftrightarrow$
$\exists P_1 \in B_1, \exists P_2 \in B_2$ *s.t.* $norm_\infty(P_1, P_2) \leq d$
EUCL_LEQ$(V_1, V_2, d) \Leftrightarrow$
$\exists P_1 \in B_1, \exists P_2 \in B_2$ *s.t.* $sqeucl(P_1, P_2) \leq d$

(V_1, V_2 are variables and B_1, B_2 are their boxes).

These lattice structures and constraints have been used to model the protein structure determination problem. Let us illustrate, for example, the basic encoding of this problem in the FCC lattice. Let $S = s_1 \ldots s_k$ be the primary sequence of a protein. We wish to determine a placement of the amino acids in the lattice; the position of the amino acid s_i is represented by a constraint variable V_i. The modeling leads to the following constraints:

- For each $1 \leq i \leq k - 1$, we have that EUCL_LEQ$(V_i, V_{i+1}, 2)$ and EUCL_GEQ$(V_i, V_{i+1}, 2)$: adjacent amino acids in the primary sequence are mapped to lattice points connected by one lattice unit;
- For each $2 \leq i \leq k - 1$, we have that EUCL_LEQ$(V_{i-1}, V_{i+1}, 7)$: three adjacent amino acids may not form an angle of $180°$ in the lattice;
- For each $1 \leq i, j \leq k$ and $|i - j| \geq 2$, we have that EUCL_GEQ$(V_i, V_j, 4)$: two non-consecutive amino acids must be separated by more than one lattice unit, and $60°$ angles are disallowed for three consecutive amino acids;
- For each known ssbond present between amino acids s_i and s_j, we have that DIST_LEQ$(V_i, V_j, 4)$.

The energy of the protein is assumed to be given by the sum of the energies generated by all pairs of amino acids, and it will depend on their distances and their types. In particular, we employ the function contact to state that two amino acids s_i and s_j are sufficiently close to interact, and thus they contribute to the energy function. In the FCC we state that contact$(A, B) = 1$ iff EUCL_LEQ$(A, B, 4)$. The *protein structure determination problem (PSD)* can be modeled as the problem of finding an assignment satisfying all these constraints and which minimizes the energy cost function:

$$E(S) = \sum_{1 \leq i < k} \sum_{i+2 \leq j \leq k} \texttt{contact}(V_i, V_j) \cdot \texttt{Pot}(s_i, s_j)$$

where `Pot` denotes the energy contribution of two amino acids in contact (see e.g. [9]).

3. Global constraints

A global constraint is a relation between n variables. Global constraints are very valuable in enhancing the declarative encoding of problems and the efficiency of constraint solvers (e.g., [16, 18, 5]). In order to be more general, we assume that the finite domain associated to each variable is a generic finite set of lattice points (instead, e.g., of a box representation as in [10]).

Given n domain variables $X_1, \ldots, X_n$, with domains $D^{X_1}, \ldots, D^{X_n}$, a *global constraint* C on $X_1, \ldots, X_n$ is a relation $C \subseteq D^{X_1} \times \cdots \times D^{X_n}$. For each global constraint C, we are interested in verifying two properties [6]:

- *(CON) Consistency:* $C \neq \emptyset$
- *(GAC) Generalized Arc Consistency:* $\forall 1 \leq i \leq n$ and $\forall a_i \in D^{X_i}$: $\exists a_1 \in D^{X_1} \cdots \exists a_{i-1} \in D^{X_{i-1}} \exists a_{i+1} \in D^{X_{i+1}} \cdots \exists a_n \in D^{X_n} \ : \ (a_1, \ldots, a_n) \in C$

If the constraint C is binary, i.e., it involves only two variables X_1, X_2, then the GAC notion is known as *arc consistency (AC)*.

The notion of *GAC* is commonly associated to the notion of *filtering*, i.e., the problem of removing values from the domains of variables in order to obtain an equivalent constraint which satisfies the *GAC* property. Often, the filtering problem is computationally expensive, and approximated solutions can be considered, leading to an equivalent constraint C', not necessarily meeting the GAC condition.

Observe that, under the assumption that the domains are not empty, the definition of *GAC* implies *CON*. Thus, if we prove that establishing *GAC* is polynomial, the same will hold for *CON*. If *CON* is NP-complete, then NP-hardness will be inherited by *GAC*. Additionally, if we assume an explicit representation of the domains, then the NP-completeness of *CON* will actually imply the NP-completeness of *GAC*.

In the rest of the paper, we discuss different types of global constraints. We start with simple and general global constraints (such as `alldifferent`), and move towards constraints more closely tied to the properties of discrete lattices and the needs of the PSD problem. The typical problems encoded on discrete lattices deal with finding adequate placements of objects in the lattice space. Placements require the entity to occupy contiguous locations in the lattice (`contiguous` constraint, Sect. 3.2), two parts of the same entity cannot be in the same location (`saw` constraint, Sect. 3.3), and components of the entity must maintain a minimum distance to account for the size of the entity (`alldistant` constraint, Sect. 3.4). Combinations of these conditions lead to more specialized global constraints (`chain` constraint, Sect. 3.5, and `block` constraint, Sect. 3.6). We also discuss the constraint `density` suggested by protein density map information determined via electron cryo-microscopy (Sect. 3.7).

3.1. The `alldifferent` Constraint

The `alldifferent` constraint [18] is probably the best-known global constraint used in constraint programming. Its semantics is as follows: if $X_1, \ldots, X_n$ are variables with domains $D^{X_1}, \ldots, D^{X_n}$, then

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

It is well-known that testing the *CON* and *GAC* properties and performing *GAC* filtering for the `alldifferent` constraint can be done in polynomial time. These problems can be solved, for example, by adapting algorithms for bipartite graph matching (as discussed, e.g., in [16]). The `alldifferent` constraint has a significant role in the modeling of the PSD problem on discrete lattices, to express the fact that a point in the lattice cannot be used to accommodate two distinct amino acids.

3.2. The `contiguous` Constraint

The `contiguous` 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 domains $D^{X_1}, \ldots, D^{X_n}$, respectively). The `contiguous` constraint can be defined as follows:

$$\begin{aligned}\texttt{contiguous}(X_1, \ldots, X_n) = (D^{X_1} \times \cdots \times D^{X_n}) \setminus \\ \{(a_1, \ldots, a_n) \in (D^{X_1} \times \cdots \times D^{X_n}) \ : \\ \exists\, i.\ (1 \leq i < n \ \wedge \ (a_i, a_{i+1}) \notin E)\}\end{aligned}$$

Testing the *GAC* of `contiguous` can be done in polynomial time. In fact, the `contiguous` 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

$$\begin{aligned}C_{i,i+1} = (D^{X_i} \times D^{X_{i+1}}) \setminus \{(a_i, a_{i+1}) \ : \\ a_i \in D^{X_i} \ \wedge \ a_{i+1} \in D^{X_{i+1}} \ \wedge \ (a_i, a_{i+1}) \notin E\}\end{aligned}$$

The overall constraint graph induced by these binary constraints is acyclic—and under these conditions *AC* implies *GAC* [11]. Since *AC* for binary constraints can be tested in polynomial time, the same holds for *GAC*. Polynomiality of *CON* follows easily. GAC filtering is equivalent to AC filtering, which is also polynomial.

The `contiguous` constraint is useful when modeling PSD problems, as it allows us to state that the sequence of amino acids composing the primary sequence of a protein should remain contiguous in the discrete lattice.

3.3. The saw Constraint

The saw constraint requires that each assignment to the variables $X_1, \ldots, X_n$ represents a *self-avoiding walk (SAW)* in the lattice. More formally, the constraint can be defined as follows:

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

The saw constraint can be used, for example, to model the fact that the primary sequence of a protein cannot create cycles when placed in the 3D space. A similar constraint (called *SAWalk*) has been used in [2].

saw can be replaced by a number of binary constraints, and performing *AC* filtering on them can be employed as a first rough polynomial approximation of *GAC* filtering. A second polynomial filtering can be achieved by iterating the *GAC* filtering of alldifferent and contiguous. However, these represent weaker propagation filterings than a direct saw *GAC* filtering. Figure 2 compares the three types of filtering on a small example (in a 2D version of the cubic lattice). The domains are shown to the left of the arrow (D_1 contains a single point, while $D_2, \ldots, D_{10}$ include 10 points). On the right, the figure shows the results of the different forms of filtering—where *(a)* all the circles (of any color) represent points left in the domain by *AC* filtering; *(b)* light grey (yellow) circles are points that are removed by the iterated *GAC* filtering of alldifferent+contiguous; *(c)* the white circles are the additional points removed by *GAC* filtering of the saw constraint. The size of all domains (initially equal to 91) is reduced to 38, 19, and 17 in the different approaches.

Testing the *CON* property for saw is clearly in NP. We have proved that it is NP-complete by reduction of the NP-complete Hamiltonian Cycle (HC) problem on a particular class of planar graphs, called *special planar graphs* in [8]. We omit here the details of the proofs due to space limits.[1]

3.4. The alldistant Constraint

When we model biologically motivated problems (e.g., protein structure determination) on a discrete lattice, we often observe that the alldifferent constraint is not sufficiently expressive. As a matter of fact, we usually require that values assigned to a group of variables are sufficiently spread in the lattice, ensuring a minimal distance between each assigned pair of points. This is required, e.g., to address the fact that different amino acids of a protein have different volume occupancies in the 3D space. In the alldistant constraint, given n variables $X_1, \ldots, X_n$, with respective domains $D^{X_1}, \ldots, D^{X_n}$, and

[1] Details can be found at www.dimi.uniud.it/dovier/WCB06/WCB06_proceedings.pdf. pp. 59–64.

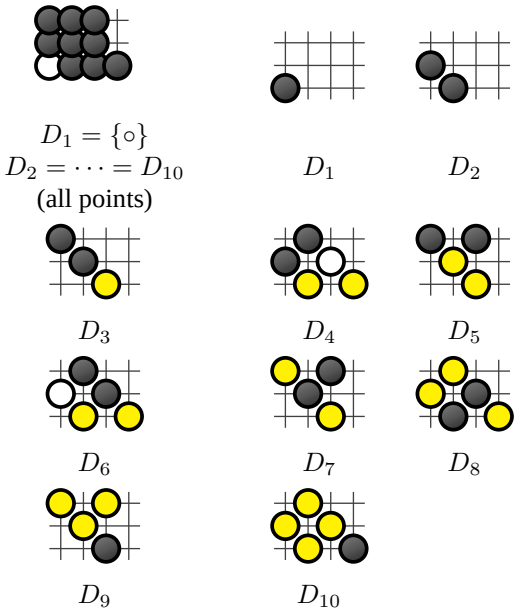

Figure 2. Propagation based on *AC* (all points), iterated alldifferent + continuous *GAC* (dark grey and white), and saw *GAC* (dark grey).

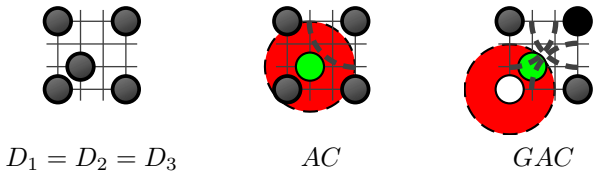

Figure 3. AC and GAC propagation on alldistant$(X_1, X_2, X_3, 2, 2, 2)$

n numbers $d_1, \ldots, d_n$—representing minimal distances derived from the "size" of each object—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 $d_i + d_j$. More formally:

$$\texttt{alldistant}(X_1, \ldots, X_n, d_1, \ldots, d_n) = (D^{X_1} \times \cdots \times D^{X_n}) \setminus \{(a_1, \ldots, a_n) \in D^{X_1} \times \cdots \times D^{X_n} : \exists 1 \leq i < j \leq n.\ sqeucl(a_i, a_j) < (d_i + d_j)^2\}$$

Note that if we consider the alldistant with $d_1 = \frac{1}{2}, \ldots, d_n = \frac{1}{2}$ then we achieve the same effect as alldifferent. Fig. 3 shows a simple example of application of *GAC* for alldistant. Let the domains of all the three variables be the set of grey points in the left-most picture. In the center picture, the light grey point is at distance less than $2 + 2$ from all other points, and it is removed by AC. For every other point, there is always a point

at distance greater than 4 (the point at the opposite corner). Finally, consider the rightmost picture. If the white point is selected for D_1, only the black point is available in D_2. No point remains for D_3. Thus, the white point must be removed. The same will happen for the other points, and GAC filtering detects unsatisfiability.

It is possible to reduce the BIN-Packing problem to the consistency problem for the `alldistant` constraint, thus proving its NP-completeness (observe that the NP-membership trivially holds). We skip the proof due to space limits. The problem of fast approximated filtering is open and could be investigated, e.g., by adapting the *sweep algorithms* of [4].

3.5. The `chain` Constraint

This global constraint states that n variables represent a self-avoiding walk and, moreover, a certain distance between variables must be respected. In addition, variables representing consecutive amino acids are required to be placed at distance equal to 1. This last property represents the main difference with `alldistant`. More formally, given n variables $X_1, \ldots, X_n$, with respective domains $D^{X_1}, \ldots, D^{X_n}$, and n numbers $d_1, \ldots, d_n$:

$$\begin{array}{l}\texttt{chain}(X_1, \ldots, X_n, c_1, \ldots, c_n) = (D^{X_1} \times \cdots \times D^{X_n}) \setminus \\ (\,\{(a_1, \ldots, a_n) \in (D^{X_1} \times \cdots \times D^{X_n}) : \exists i, j. \\ \quad 1 \leq i+1 < j \leq n \wedge sqeucl(a_i, a_j) < (d_i + d_j)^2\}) \\ \quad \cup \{(a_1, \ldots, a_n) \in (D^{X_1} \times \cdots \times D^{X_n}) : \exists i. \\ \quad\quad 1 \leq i < n \wedge (a_i, a_{i+1}) \notin E\}\,)\end{array}$$

Note that if we consider the `chain` with $d_1 = \frac{1}{2}, \ldots, d_n = \frac{1}{2}$ then we achieve the same effect as `saw`. Being therefore a generalization of `saw`, *CON* is NP-complete and *GAC* is NP-hard for `chain`.

3.6. The `rigid block` Constraint

It is common, when dealing with protein structure determination, to have knowledge of local features of the structure, e.g., presence of secondary structure components (such as α-helices and β-strands); thus, we want 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 domains $D^{X_1}, \ldots, D^{X_n}$. Let us also consider $\vec{B} = B_1, \ldots, B_n$ to be a list of lattice points—that, intuitively, describe the desired layout of the rigid block. Then, $\mathsf{block}(X_1, \ldots, X_n, \vec{B})$ 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*. More precisely, we define a *rotation* of a lattice point $p = (p_x, p_y, p_z)$ as $rot(\phi, \theta, \psi)(p) = X \cdot Y \cdot Z \cdot p^T$, 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 ϕ, θ, ψ 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 the cubic lattice, we have that $r = 16$, and in FCC we have $r = 24$. We extend the definition of rotation to the case of lists of lattice points, $rot(\phi, \theta, \psi)(\vec{B})$, where $\vec{B}$ is a list of points and the result is a list in which every element of $\vec{B}$ is rotated according to the previous definition.

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

$$\mathsf{Templ}(\vec{B}) = \{rot(\phi, \theta, \psi)(\vec{B}) \,:\, \exists \phi, \theta, \psi.\ rot(\phi, \theta, \psi)(\vec{B}) \text{ is an automorphism on the lattice}\}$$

which contains the distinct 3-dimensional rotations of the points $\vec{B}$ in the lattice. Note that, for a given list of points $(\vec{B})$, the cardinality of $\mathsf{Templ}(\vec{B})$ is at most r. We say that $\vec{\ell} = (\ell_x, \ell_y, \ell_z)$ is a *lattice vector* if the translation by $\vec{\ell}$ 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 $\vec{\ell}$ be a lattice vector; with $\mathsf{Shift}[\vec{\ell}]$ we denote a mapping that translates a rigid block according to the vector $\vec{\ell}$. Formally, for each $1 \leq i \leq k$, we have $\mathsf{Shift}[\vec{\ell}](\vec{B})[i] = B_i + \vec{\ell}$. Shift is used to place a template into the lattice space, preserving the orientation and the distances between points. A rigid block constraint $\mathsf{block}(X_1, \ldots, X_n, \vec{B})$ is then defined as the set:

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

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 constraints has been studied in a wider context in [13]. Moreover, the idea of considering rigid blocks to model substructures of proteins is also discussed in [3].

3.7. The density Constraint

Electron cryo-microscopy is an experimental technique that has the potential to allow structure determination for large and membrane proteins [19, 15]. An electron microscope is able to produce a *density map* D that represents the electron density of a molecule in a given portion of space; these maps usually provide a resolution ranging from 6Å to 12Å. The density map is sampled at a uniform rate R in the 3D space, and generates a partition of the space into cubes with edges of length R. Each cube contains a certain amount of density, described by a real number, representing a sample measurement. Let us indicate with $D(x, y, z)$ the value of the density map sampled at location $(x, y, z) \in \mathbb{N}^3$.

Each molecule component (e.g., amino acid) placed in a point $\vec{p}$ in the space generates a density value that affects $\vec{p}$ and the neighboring points according to a given function $\mathcal{F}$—for instance $\mathcal{F}$ can be approximated by a *Gaussian* contribution:

$$\mathcal{F}(\vec{x}, \vec{p}) = \mathcal{G}_{a,\sigma}(\vec{x}, \vec{p}) = ae^{-\frac{|\vec{x}-\vec{p}|^2}{2\sigma^2}}$$

where $a \in \mathbb{R}, \vec{p} \in \mathbb{N}^3, \sigma \in \mathbb{R}$ are respectively the intensity of the map, the reference point for the center of the object, and the decay control parameter. The parameters a and σ can be estimated according to the type of the component, by first generating density maps for the single components and then performing a least square approximation.

Given a molecule chemical description, it is possible to decompose the molecule into n components (e.g., a protein can be decomposed into the set of composing amino acids). Each component $1 \leq i \leq n$ can be placed in the space in the position $\vec{p}_i$ and provides a specific contribution function $\mathcal{F}_i(\vec{x}, \vec{p}_i)$ which can be pre-computed.

The ultimate goal is to find the possible placements $[\vec{p}_1, \ldots, \vec{p}_n]$ of the components so that their combination produces a density map "similar" to D—e.g., for each $\vec{x} \in \mathbb{N}^3$, $\sum_{i=1}^{n} \mathcal{F}_i(\vec{x}, \vec{p}_i) \approx D(\vec{x})$.

We have identified some global constraints (e.g., `average_density`, `point_density`, ...) that abstract this concept and proved the NP-completeness of their consistency check, even when using a very simple $\mathcal{F}$ function. We obtained some preliminary results on the integration of this constraint with the other global constraints discussed in this paper—in particular `saw` and `chain`. The density constraint allows us to effectively prune the search space, and it becomes useful when using a high resolution lattice (`FCC` with 3.8Å between neighbors contains very few placements for consecutive amino acids); however, the use of a highly connected lattices hampers the propagation of other constraints (e.g., `saw`). The plan is to integrate propagation algorithms in order to retain the benefits of density information in a more refined context.

4. Tool and Experiments

We implemented the global constraints presented above in our system *COLA 3.0* [10, 7]. The system seamlessly integrates the new global constraints into the previous constraint system, thanks to the compositionality of constraint programming techniques. In particular, we now provide the possibility to model a protein chain structural properties using specific constraints and to superimpose on the structure known patterns taken from the Protein Data Bank. Moreover, multiple chain definitions are supported.

We briefly describe the concrete syntax of the new constraints (for more information, c.f. [7]).

- `cstore_add_next(`v_1`, `v_2`)` indicates that amino acids v_1 and v_2 are at one unit lattice distance. Used to model two amino acids that are contiguous in the primary sequence;
- `cstore_add_ang(`v_i`)` indicates that the amino acid v_i cannot form a $180°$ bend angle with amino acids v_{i-1} and v_{i+1};
- `cstore_add_ssbond(`v_1`, `v_2`)` states that amino acids v_1 and v_2 are close and form a disulfide bridge;
- `cstore_add_alldifferent(`v_i`, `v_j`)` states that amino acids from v_i to v_j cannot overlap;
- `cstore_add_contiguous(`v_i`, `v_j`)` states that the amino acids from v_i to v_j are connected and contiguous in space;
- `cstore_add_saw(`v_i`, `v_j`)` states that amino acids from v_i to v_j form a self avoiding walk;
- `cstore_add_alldistant(`v_i`, `v_j`, d)` states that amino acids from v_i to v_j cannot lie at distance smaller than $\sqrt{d}$. The constraint is not applied to each pair v_t and v_{t+1} for $i \leq t < j$, in order to avoid conflicts with `next` constraint;
- `cstore_add_chain(`v_1`, `v_2`)` states that the amino acids in the range from v_i to v_j satisfy both the `saw` constraint and the `alldistant` constraint.

Finally, the *rigid block constraint* is defined by means of the protein description file, which provides the PDB pattern and the range of primary sequence of application.

Our goal is to tackle proteins that are hundreds of amino acids long. The global constraints offer the computational tools to model and compute the conformations of such proteins in reasonable time. The knowledge of some protein domains can be included in the prediction to restrict the search. We model the protein structure exploiting the approximation of the `FCC` lattice. However, when dealing with rigid blocks, we superimpose the original non-approximated block when generating the energy evaluations and the prediction output, to provide a higher quality and reduce approximation errors.

In our preliminary tests, we make use of a potential en-

ergy model suitable for globular protein, which caused less reliability in the quality of multi-domain proteins. On the other hand, COLA is completely parametric w.r.t. the energy function used to evaluate the conformational space. The use of different energy functions does not significantly affect the efficiency of search. Thus, we expect that tighter interaction with structural biologists can provide COLA with better energetic models and higher quality results.

5. Conclusions and future work

In this paper, we presented a study of different global constraints that can be used to provide declarative encodings of problems in discrete crystal lattices. The introduction of global constraints has been motivated by problems derived from the use of constraint solving in discrete lattices to solve the PSD problem. We propose different types of constraints and investigate their computational properties.

Various aspects are still open and deserve consideration. The computational considerations indicate that *CON*, *GAC*, and filtering for a number of interesting global constraints are intractable properties. In these cases, it will be important to determine whether good approximated filtering algorithms can be devised and efficiently implemented. We also intend to extend our investigation of the rigid block constraint, allowing the expression of relationship between blocks (e.g., proximity, angles), starting from the ideas presented in [13, 14].

The global constraints contained in this paper have been implemented in different prototypes. The original implementation has been realized using constraint logic programming, and employed to compare the pruning capabilities of the different constraints. More recently, as described in this paper, we have included these global constraints in the COLA system.

We hope this paper will inspire further interest in this problem and promote discussion about suitable global constraints for discrete lattice structures and efficient implementation techniques.

References

[1] R. Agarwala et al. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. *J. Comp. Biology* 4(3):275–296, 1997.

[2] R. Backofen and S. Will. A Constraint-Based Approach to Fast and Exact Structure Prediction in 3D Protein Models. *Constraints* 11(1):5–30, 2006.

[3] R. Backofen et al. Constraint Based Protein Structure Prediction Exploiting Secondary Structure Information. *Italian Conf. on Computational Logic*, 2004.

[4] N. Beldiceanu and M. Carlsson. Sweep as a Generic Pruning Technique Applied to the Non-overlapping Rectangles Constraint. *CP*, Springer Verlag, 2001.

[5] N. Beldiceanu et al. *Global Constraint Catalog*. TR2005-08, SICS, 2005.

[6] C. Bessiere et al. The complexity of global constraints. Proceedings AAAI'04, pp. 112-117, 2004.

[7] Constraint Solver for Lattices (COLA). `www2.unipr.it/~dalpalu/COLA/`.

[8] P. Crescenzi et al. On the Complexity of Protein Folding. *J. of Computational Biology* 5(3):423–466, 1998.

[9] A. Dal Palù, A. Dovier, and F. Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics* 5(186), 2004.

[10] A. Dal Palù et al. A New Constraint Solver for 3-D Lattices and its Application to the Protein Folding Problem. *LPAR*, Springer Verlag, 2005.

[11] E. Freuder. A sufficient condition for backtrack-bounded search. *J. of ACM* 29(1):24–32, 1982.

[12] W. Hart and S. Israil. Fast protein folding in the hydrophobic-hytrophilic model within three-eighths of optimal. *J. Comp. Biology* 3(1):53–96, 1996.

[13] L. Krippahl and P. Barahona. Propagating N-Ary Rigid-Body Constraints. *CP*, Springer Verlag, 2003.

[14] L. Krippahl and P. Barahona. Applying Constraint Programming to Rigid Body Protein Docking. *CP*, Springer Verlag, 2005.

[15] E.J. Mancini et al. Cryo-electron Microscopy Reveals the Functional Organization of an Enveloped Virus. *Mol. Cell* 5:255266, 2000.

[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.

[18] W. J. van Hoeve. The Alldifferent Constraint: a Survey. CoRR cs.PL/0105015: 2001.

[19] Z.H. Zhou et al. Seeing the Herpesvirus Capsit at 8.5Å. *Science* 88:877–880, 2000.