Chapter 1

Constraint based methods for Bioinformatics

1.1. Introduction

Bioinformatics is a challenging area of research where every serious contribution can have thousands of positive effects in medicine, agriculture, or industry. Biology, in general, is a source of extremely interesting and computationally expensive tasks. Most of the typical problems can be effectively formulated by using declarative languages and constraints. Constraints on finite domains (and on reals) are applied for predicting spatial conformation of polymers, concurrent constraint programming can be used for simulations of biological systems, and constraints on strings are employed for the analysis of DNA sequences.

The WCB06 workshop was organized with the aim of sharing new theoretical and practical results in the area and of summarizing new challenging problems for the declarative programming and constraint community. The workshop is the successor of the workshops *Constraints and Bioinformatics/Biocomputing* colocated with CP'97 and CP'98, and of the workshop WCB05 colocated with ICLP 2005.

The workshop benefited from the excellent invited talk of François Fages about *Using temporal logics with constraints to express biological properties of cell processes* (Sect. 1.2) and from the presentation of 7 contributed papers. The contribution by Bortolussi and Policriti (Sect. 1.3) belongs to the field of *Systems Biology*, as well. In the area of *Structural Prediction* we experienced four contribution: by Krippahl and Barahona; by Elisabetta De Maria et al., by Dal Palù et al., and by Will and Mann (Sect. 1.5–1.7). A work on suffix array by Zytnicki et al. (Sect. 1.8) and a paper by Prosser on Supertree Construction (Sect. 1.9) concluded the contributions to the

Chapter written by Alessandro Dal Palù, Agostino Dovier, François Fages, and Sebastian Will.

workshop. In the rest of this Chapter we report the abstract of the invited talk written by François Fages (Sect. 1.2) whom we would like to thank again, and our short summaries of the 7 contributed papers (Sect. 1.3–1.9).

The interest of the constraint community in bioinformatics and biology is witnessed by the considerable number of participants (35) although the workshop has run in parallel with other extremely interesting workshops. During the final discussion, we decided to apply for co-location of WCB07 at the next edition of ICLP07 in Porto, Portugal, where we are confident to receive another strong contribution to this research area by the Logic Programming community. Other information, the proceedings and some pictures from the workshop can be found in the WCB06 web-site `http://www.dimi.uniud.it/dovier/WCB06`. We conclude by acknowledging all the PC members, the external referees, and all the participants. A particular thank to the CP workshop chair Barry O'Sullivan, and to the two other editors of this book, Frédéric Benhamou and Narendra Jussien.

1.2. On Using Temporal Logic with Constraints to Express Biological Properties of Cell Processes—Invited talk by François Fages

One promise of systems biology is to model biochemical processes at a sufficiently large scale so that the behavior of a complex system can be predicted under various conditions in *in silico* experiments. The language approach to systems biology aims at designing formal languages for describing biochemical mechanisms, processes and systems at different levels of abstraction, and for providing automated reasoning tools to assist the biologists [FAG 04a].

The pioneering use of the π-calculus process algebra for modeling cell signaling pathways in [REG 01], has been the source of inspiration of numerous works in the line of process calculi and of their stochastic extensions. The biochemical abstract machine BIOCHAM[1] [FAG 04b] has been designed as a simplification of the process calculi approach to model biological processes, using a language of reaction rules that is both more natural to the biologists, and well suited to consider different dynamics and use model-checking techniques.

In BIOCHAM, the rule-based language is used for modeling biochemical networks at three abstraction levels:

1. BIOCHAM is a free software implemented in Prolog and distributed under the GPL license. It is downloadable on the web at `http://contraintes.inria.fr/BIOCHAM`. The BIOCHAM project is a joint work with Nathalie Chabrier-Rivier, Sylvain Soliman and Laurence Calzone, with contributions from Sakina Ayata, Loïc Fosse, Lucie Gentils, Shrivaths Rajagopalan and Nathalie Sznajder. In addition, support from the EU STREP project April-II and the EU Network of Excellence REWERSE are warmly acknowledged.

1) the *Boolean semantics*, where one associates to each object (protein, gene, etc.) a Boolean variable representing its presence or absence in the system, and the reaction rules are interpreted by a highly non-deterministic *asynchronous transition system* representing competition between reactions;

2) the *concentration semantics*, where one associates to each object a real number representing its concentration, and the reaction rules are interpreted with their kinetic expressions by a set of non-linear ordinary differential equations (ODE);

3) the *stochastic semantics*, where one associates to each BIOCHAM object an integer representing the number of molecules in the system, and the rules are interpreted as a continuous time Markov chain.

One striking feature of this multi-level approach is that in the three cases, temporal logics can be used to formalize the biological properties of the system, and verify them by different model-checking techniques. The thesis is that, to a large extend, one can make the following identifications:

$$\begin{aligned} \textit{biological model} &= \textit{transition system,} \\ \textit{biological property} &= \textit{temporal logic formula,} \\ \textit{biological validation} &= \textit{model-checking.} \end{aligned}$$

At the Boolean level, the *Computation Tree Logic* CTL [CLA 99] allows one to express *qualitative properties* about the production of some protein (reachability), the checkpoints for its production, the stability or oscillations for its presence, etc. These properties are known from biological experiments in wild-life or mutated organisms. Some of the most used CTL formulae are abbreviated in BIOCHAM as follows:

- `reachable(P)` stands for $EF(P)$;
- `steady(P)` stands for $EG(P)$;
- `stable(P)` stands for $AG(P)$;
- `checkpoint(Q,P)` stands for $!E(!Q\ U\ P)$;
- `oscil(P)` stands for $AG((P \Rightarrow EF\ !P) \wedge (!P \Rightarrow EF\ P))$.

In this setting, such properties can be checked with state-of-the-art symbolic model checkers such as NuSMV using binary decision diagrams. The performances obtained on a large model of the mammalian cell cycle control after Kohn's map [KOH 99], involving 800 rules and 500 variables, have been shown to be of the order of a few tenths of seconds to compile the model, and check simple CTL formulae.

At the concentration level, it is used a first-order fragment of Linear Time Logic (LTL) with *arithmetic constraints* containing equality, inequality and arithmetic operators ranging over the real values of concentrations and of their derivatives. For instance `F([A]>10)` expresses that the concentration of `A` eventually gets above the threshold value 10. `G([A]+[B]<[C])` expresses that the concentration of C is always

greater than the sum of the concentrations of A and B. Oscillation properties, abbreviated as `oscil(M,K)`, are defined as a change of sign of the derivative of M at least K times in the time horizon:

```
F((d[M]/dt > 0) & F((d[M]/dt < 0) & F((d[M]/dt > 0)...)))
```

The abbreviated formula `oscil(M,K,V)` adds the constraint that the maximum concentration of M must be above the threshold V in at least K oscillations.

Under the hypothesis that the initial state is completely defined, numerical integration methods (such as Runge-Kutta or Rosenbrock methods) provide a discrete simulation trace. This trace constitutes a linear Kripke structure in which LTL formulae with constraints can be interpreted and model-checked [CAL 06]. Since constraints refer not only to concentrations, but also to their derivatives, we consider traces of the form $(\langle t_0, x_0, dx_0/dt\rangle, \langle t_1, x_1, dx_1/dt\rangle, \dots)$ where at each time point, t_i, the trace associates the concentration values of the x_i's and the values of their derivatives dx_i/dt.

Beyond making simulations, and checking properties of the models, the temporal properties can also be turned into specifications and temporal logic constraints for automatically searching and learning modifications or refinements of the model when incorporating new biological knowledge. This is implemented in BIOCHAM by a combination of model-checking, search and machine learning techniques in the three abstraction levels.

For instance, in a simple continuous model of the cell cycle after Tyson [TYS 91], the search of parameter values for kinetic parameters k_3 and k_4, so that the concentration of the cyclin `Cdc2-Cyclin p1` oscillates three times in the time horizon 150, can be formalized as follows:

```
biocham: learn_parameters([k3, k4], [(0, 200), (0, 200)], 20,
                          oscil(Cdc2-Cyclin~{p1},3),150).
First values found that make oscil(Cdc2-Cyclin~{p1},3) true:
parameter(k3,_).
parameter(k4,_).
```

The system finds the parameter values $k_3 = 10$ and $k_4 = 70$ satisfying the specification. However, the corresponding curve depicted in **Fig. 1.1** on the left exhibits damped oscillations. The specification can be further refined by imposing a constraint of period equal to 35 time units, `period(Cdc2-Cyclin~{p1},35)`. This produces the curve depicted in **Fig. 1.1** on the right which is close to the original model.

These first results implemented in BIOCHAM are quite encouraging and motivate further research in the direction of the formal specification of biological systems and in the improvement of the search algorithms. A coupled model of the cell cycle and the circadian cycle is under development along these lines in BIOCHAM with applications to cancer chronotherapies.

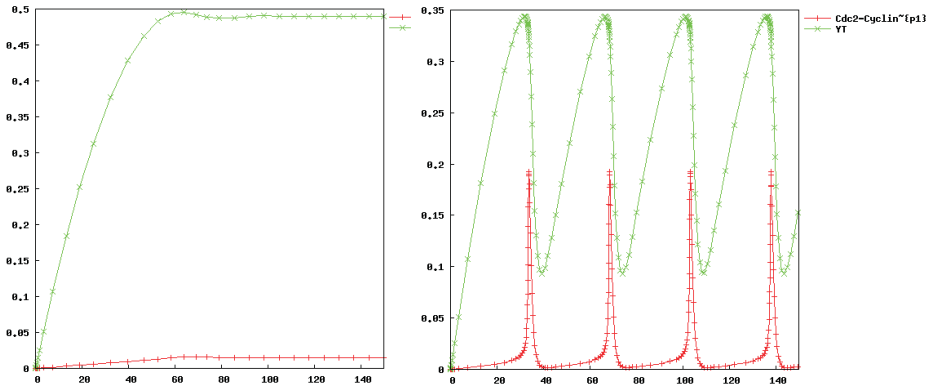


Figure 1.1. *Concentration experimental results.*

1.3. Modeling Biological Systems in Stochastic Concurrent Constraint Programming—by Luca Bortolussi and Alberto Policriti

In this work the authors show how stochastic Concurrent Constraint Programming (sCCP—[BOR 06]) can be used for modeling biological systems. sCCP is based on CCP [SAR 93], a process algebra where agents interact by posting constraints on the variables of the system in the constraint store.

Computational Systems Biology is a field in which different modeling techniques are used to capture the intrinsic dynamics of biological systems. Some of them are based on *Differential Equations*, mostly ordinary, and therefore they represent phenomena as *continuous and deterministic*. On the other side there are *stochastic and discrete* models, that are usually simulated with *Gillespie's algorithm* [GIL 77]. In the middle, there are hybrid approaches like the *Chemical Langevin Equation*, a stochastic differential equation that bridges partially these two opposite formalisms.

In the last few years *stochastic process algebras* (SPA) has emerged [PRI 01]. It is based on the parallel between molecules and reactions on one side and processes and communications on the other side. SPA have been used to model biological systems (e.g., biochemical reactions and genetic regulatory networks). Stochastic modeling of biological systems works by associating a rate to each active reaction (or, in general, interaction); rates are real numbers representing the frequency or propensity of interactions. All active reactions then undergo a (stochastic) race condition, and the fastest one is executed. These rates encode all the quantitative information of the system, and simulations produce discrete temporal traces with variable delay between events.

In the author's opinion, the advantages of using sCCP are twofold: the presence of both quantitative information and computational capabilities at the level of the constraint systems and the presence of functional rates. This second feature, in particular,

allows to encode in the system different forms of dynamical behaviors, in a very flexible way. Quantitative information, on the other hand, allows a more compact representation of models, as part of the details can be described in relations at the level of the store.

At high level, biological systems are composed of two ingredients: (biological) entities and interactions among those entities. For instance, in biochemical reaction networks, the molecules are the entities and the chemical reactions are the possible interactions. In gene regulatory networks, instead, the entities are genes and regulatory proteins, while the interactions are production and degradation of proteins, and repression and enhancement of gene's expression. In addition, entities fall into two separate classes: measurable and logical. Measurable entities are those present in a certain quantity in the system, like proteins or other molecules. Logical entities, instead, have a control function, hence they are neither produced nor degraded. Note that logical entities are not real world entities, but rather they are part of the models.

Measurable entities are associated exactly to stream variables (unbounded tail lists of time varying variables). Logical entities, instead, are represented as processes actively performing control activities. In addition, they can use variables of the constraint store either as control variables or to exchange information. Finally, each interaction is associated to a process modifying the value of certain measurable stream variables of the system. Associating variables to measurable entities means that they are represented as part of the environment, while the active agents are associated to the different actions capabilities of the system. These actions have a certain duration and a certain propensity to happen: a fact represented here in the standard way, i.e. associating to each action a stochastic rate.

Constraints maintain information about the biological entities. This leads to the definition of a general purpose library of processes that can be used in the modeling phase. However, this is only a part of the general picture, as there are more complex classes of biological systems that need to be modeled, like transport networks and membranes. In addition, all these systems are strongly interconnected, and they must be modeled altogether in order to extract deep information about living beings. The authors believe that the flexibility of constraints makes sCCP a powerful general purpose language that can be simply programmed, extended with libraries, and used to model all these different classes of systems in a compact way.

Biochemical reactions can be challenging to be modeled, because proteins can form very big complexes that are built incrementally. Therefore, the cell is populated by a huge number of sub-complexes. Usually, these networks are described by biologists with diagrams, like Kohn maps,that are very compact, because they represent complexes and sub-complexes implicitly. Constraints can be used to encode the calculus elegantly, by representing complexes implicitly, i.e. as lists of basic constituents.

Functional rates can be used in enzymatic reactions to represent more complex kinetic dynamics, allowing a more compact description of the networks. In this direction, the authors need to make deeper analysis of the relation between these different kinetics in the context of stochastic simulation, in order to characterize the cases where these different kinetics can be used equivalently. Notice that the use of complex rates can be seen as an operation on the Markov Chain, replacing a subgraph with a smaller one, hiding part of its complexity in the expression of rates. Finally, the authors plan to implement a more powerful and fast interpreter for the language, using also all available tricks to increase the speed of stochastic simulations. Moreover, the authors plan to tackle the problem of distributing efficiently the stochastic simulations of programs written in sCCP.

1.4. Chemera: Constraints in Protein Structural Problems—by Pedro Barahona and Ludwig Krippahl.

Chemera is a molecular modelling software package that includes the algorithms BiGGER (Bimolecular complex Generation with Global Evaluation and Ranking), for modelling protein interactions and protein complex structures [KRI 05], and PSICO (Processing Structural Information with Constraint programming and Optimisation), to integrate experimental and theoretical data to solve protein structures [KRI 02]. Authors' contribution to the workshop focuses on the constraint programming aspects of Chemera, namely *constrained docking*, which allows the user to restrict the search for protein-protein complex models in a manner consistent with the ambiguity of some experimental data, and the processing of structural constraints to generate approximate models of protein structures from heterogeneous data (e.g. spectroscopy, site-directed mutagenesis, homology models, secondary structure prediction, reaction mechanisms).

Protein-protein interactions play a central role in biochemical reactions. Modelling software provides useful tools to help researchers elucidate protein interaction mechanisms. A common trend in these approaches is to try to model interactions using only knowledge derived from the structure and physico-chemical properties of the proteins involved.

In modelling the structure of a protein, the common approaches have been either theoretical, to try to predict the structure from the physical properties of the amino acid sequence in the protein, possibly using homologies with other known structures, or experimental, specializing on the processing of data from specific techniques like Nuclear Magnetic Resonance (NMR) spectroscopy. PSICO aims at bringing the two approaches together by providing a flexible framework for processing geometrical constraints and thus integrate information from all relevant sources in the modelling of a protein structure. NMR data can be modelled as distance constraints [KRI 02] or as torsion-angle constraints [KRI 05], homology or secondary structure prediction data can be modelled as rigid-group constraints [KRI 05], energy functions can be

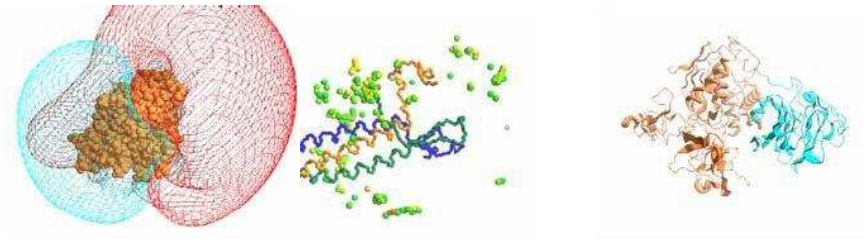
included in the local-search optimization stage, and amino acid properties relevant for protein folding, such as hydrophobicity, can be part of the enumeration heuristics during constraint processing.

The core of protein docking algorithm is the representation of the protein shapes and the measure of surface contact. The former is a straightforward representation using a regular cubic lattice of cells. In BiGGER the cells do not correspond to numerical values, but each cell can be either an empty cell, a surface cell, or a core cell. The surface cells define the surface of the structure, and the overlap of surface cells measures the surface of contact. BiGGER also models side-chain flexibility implicitly by adjusting the core grid representation and allows for hard or soft docking simulations depending on the nature of the interaction to model. Furthermore, this representation and the search algorithm can take advantage of information about the interaction to simultaneously improve the results and speed up the calculations.

Grids are composed of lists of intervals specifying the segments of similar cells along the x coordinate. The fact that core cells can not overlap induces a powerful constraint that is able to prune the relative shifts between the two structures. Moreover, a branch and bound search is applied in order to optimize the overlap of surface cells, and restrict the search to those regions where this overlap can be higher than that of the lowest ranking model to be kept.

In some cases there is information about distances between points in the structures, information that can be used to restrict the search region. The most common situation is to have a set of likely distance constraints of which not all necessarily hold. To cope with this, the program supports the constraint of the form: *At least K atoms of set A must be within R of at least one atom of set B*, where set A is on one protein and set B on the other, and R a distance value.

There are several sources of information that can help model the structure of a protein. First of all, the amino acid sequences of the protein chains determines most chemical bonds, restricting interatomic distances in many atom pairs, angles formed by atom triplets, of even larger groups of atoms that are effectively rigidly bound together by the chemical bonds. NMR data provides several types of distance constraints by showing that two atoms must be close enough, by limiting the angles of rotation around some chemical bonds, by suggesting limits for relative special orientations of groups of atoms. Furthermore, homology with known structures or modelling secondary structure can provide detailed information of the structure of parts of the protein being modelled. This information identifies three types of constraints implemented in the program: distance constraints between two atoms, group constraints that fix the relative positions of a group of atoms in a rigid configuration, and torsion angle constraints that restrict the relative orientation of two groups joined together by a chemical bond.



A B C

Figure 1.2. *Visualisation possibilities in Chemera. See text for details.*

Chemera is the interface to all BiGGER and PSICO calculations and includes tools for handling *Electrostatics* (**Fig. 1.2**–A), *Clustering and Scoring* (**Fig. 1.2**–B) and *Web Services* — interface with several web services, to assign secondary structure elements, identify domains, display sequence conservation along the protein structure (**Fig. 1.2**–C).

Constraint programming techniques in Chemera are seamlessly integrated into a general molecular modelling package. This is an important aspect because research and development in this area is very dependent on a close interaction with the end users in the biochemistry community. Authors experience and work currently in progress on several protein interactions (e.g. Aldehyde Oxidoreductase and Flavodoxin, Ferredoxin NADP Reductase and Ferredoxin, Fibrinogen and Gelatinase A) demonstrate this for the BiGGER docking algorithm, which is currently available in Chemera 3.0 `http://www.cqfb.fct.unl.pt/bioin/chemera/`.

1.5. Exploiting Model Checking in Constraint-based Approaches to the Protein Folding Problem—by Elisabetta De Maria, Agostino Dovier, Angelo Montanari, and Carla Piazza

In this paper the authors show how *Model Checking* could be used to drive the solution search in the protein folding problem encoded as a constraint optimization problem. The application of the model checking techniques allows the authors to distinguish between meaningful protein conformations and bad ones. This classification of conformations could be exploited by constraint solvers to significatively prune the search space of the protein folding problem. Furthermore, the approach seems promising in the study of folding/energy landscapes of proteins.

The authors consider foldings (i.e., self avoiding walks) of proteins on 2D discrete lattices. If the first two points are set (w.l.o.g., the authors set $\omega(0) = (n, n)$ and $\omega(1) = (n, n + 1)$), then a folding on this lattice can be uniquely represented by a sequence of directions with respect to the preceding one: left (*l*), forward (*f*), and right (*r*) (see **Fig. 1.3** for an example)

Figure 1.3. *The folding* frrfll *on* $\mathbb{Z}^2$ *lattice (left). Contacts and energy contributions of the string HHPHHHPH w.r.t. the same folding (right).*

The authors exploits their analysis using the HP energy model (although the method can be employed on richer energy models such as the 20×20 contact energy table used in [Dal 04]). Such a model reduces the 20-letter alphabet of amino acids to a two-letter alphabet $\{H, P\}$, where H (resp., P) represents a hydrophobic (resp., polar) amino acid. The energy function states that the energy contribution of a *contact* between two non consecutive amino acids is -1 if both of them are H amino acids, 0 otherwise (see **Fig. 1.3**—right).

The authors then introduce the notion of valid transformations among foldings. Roughly speaking, a valid transformation of a given folding f consists in selecting at random a position in f and performing a rotation of the part of f between this position and the ending position (*pivot move*). Precisely, let $f = f_1 \dots f_n$, with $f_i \in \{l, f, r\}$ for all $2 \leq i \leq n$, be a folding of a sequence s. A folding f' of s is obtained from f through a *pivot move* with pivot k if $f'_i = f_i$ for all $i \neq k$ and $f'_k \neq f_k$. As an example, consider the 6 pivot moves from folding *lfl* in **Fig. 1.4**. It is possible to show that pivot moves are ergodic, namely, they allow to cover the entire folding space.

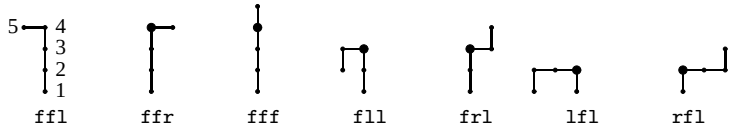


Figure 1.4. *The 6 pivot moves from string* `fll`*. Large bullets are the pivots.*

The authors then define the notion of *2D Protein Transition System* of a string P of length n over $\{H, P\}$ as a tuple $M_P = (Q, T, L)$, where

– Q is the set of all foldings of length n on the $2n \times 2n$ 2D lattice;

– $T \subseteq Q \times Q$ is the set of pairs of states (q_1, q_2) such that q_2 can be obtained from q_1 by a pivot move;

– $L : Q \rightarrow 2^{AP}$ is a labeling function over the set AP of atomic propositions which consists of the following predicates: $2nd\_l, 2nd\_f, 2nd\_r, \dots, nth\_l, nth\_f, nth\_r$, plus the following three predicates: min_en, $inter\_en$, max_en, where for all $2 \leq i \leq n$, the predicate ith_l (resp., ith_f, ith_r) holds at a state q if the i-th segment of q has a *left* (resp., *forward*, *right*) orientation and min_en (resp., $inter\_en$, max_en) holds at a state q if the energy of q is minimum (resp., intermediate, 0).

Given a 2D Protein Transition System $M_P = (Q, T, L)$ and a temporal logic formula f expressing some desirable property of the system, the *model checking problem* consists in finding the set of all states in Q satisfying f [CLA 99]. When a state does not satisfy a formula, model checking algorithms produce a counterexample that falsifies it, thus providing an insight to understand failure causes and important clues for fixing the problem. The authors restrict their attention to two well-known fragments of the *computation tree logic* CTL*, namely, the *branching time* logic CTL and the *linear time* logic LTL.

The authors then show how meaningful properties of 2D Protein Transition Systems can be encoded in both CTL and LTL. Here two of them are reported:

F1: Does it exist a path of length at most k that reaches a state with minimum energy?
CTL: $\bigvee_{i=0}^{k} E_1 X_1 \ldots E_i X_i min\_en.$
LTL (actually, it expresses ¬**F1**): $A(\bigwedge_{i=0}^{k} X_1 \ldots X_i \neg min\_en).$
F2: Is energy the minimum one? Alternatively, if energy is the maximum one, is it possible to reach a state with minimum energy without passing through states with intermediate energy?
CTL, LTL: $A(max\_en\ U\ min\_en).$

The authors finally show some results of the tests of the properties described using a model checker written in SICStus Prolog. A faster implementation of the method using on *on-the-fly* model checking is under analysis.

1.6. Global Constraints for Discrete Lattices—by Alessandro Dal Palù, Agostino Dovier, and Enrico Pontelli

Constraint solving on discrete lattices has gained momentum as a declarative and effective approach to solve complex problems such as protein folding determination. In particular, [Dal 05] presented a comprehensive constraint solving platform (COLA) dealing with primitive constraints in discrete lattices. The authors discuss some preliminary ideas on possible global constraints that can be introduced in a constraint system like COLA. Various alternatives are presented and preliminary results concerning the computational properties of the different global constraints are reported.

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 [SKO 04]. Polymers are laid out in particular subsets of $\mathbb{N}^3$. These subsets are often described by the vectors that specify the set of neighbors of each point. Lattice models like FCC and chess knight are among them.

The protein structure prediction in the context of discrete lattice structures has been studied as a *Constraint Optimization Problem* in the FCC lattice, using simplified energy models [BAC 06, Dal 04]. In these approaches, each point P of the

lattice is identified by a triplet of *finite domain variables* (P_x, P_y, P_z). It is proved that mantaining an independent variable for each point coordinate limits the power of *propagation* w.r.t. an approach (as in COLA) where a point is considered as a whole.

The authors propose a study targeting the problem of dealing with *global* constraints in the general context of constraint solvers on lattice domain. Global constraints are proven constructs that facilitate the declarative encoding of problems; at the same time, they allow the programmer to express knowledge about relationships between variables, that can be effectively employed by the search algorithm to prune infeasible parts of the solution search space. The authors introduce different global constraints, and they study the complexity of their satisfiability and of the associated propagation process. For each global constraint C (with variables $X_1, \ldots, X_n$) analyzed, the authors are interested in verifying two properties:

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

– *generalized arc consistency (GAC):* $\forall i \in \{1, \ldots, n\}\, \forall a_i \in D^{X_i}$
$\exists a_1 \in D^{X_1} \cdots \exists a_{i-1} \in D^{X_{i-1}} \exists a_{i+1} \in D^{X_{i+1}} \cdots \exists a_n \in D^{X_n}\ (a_1, \ldots, a_n) \in C$

• The `alldifferent` global constraint is used to describe that all the variables must assume different points. It is well known that consistency and propagation of `alldifferent` is polynomial.

• 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) and has the form:

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

where E is the set of edges in a lattice, and $X_1, \ldots, X_n$ is a list of variables (respectively, with domains $D^{X_1}, \ldots, D^{X_n}$).

The authors prove that verifying CON and GAC are in P.

• The `saw` constraint is used to require that each assignment to the variables $X_1, \ldots, X_n$ represents a self-avoiding walk (SAW) in the lattice and has the form:

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

The `saw` constraint can be used, for example, to model the fact that the primary sequence of a protein can not create cycles when placed in the 3D space. The authors prove that CON of `saw` global constraint is NP-complete, and, consequently, GAC is NP-hard. `saw` can be replaced by a set of binary constraints. AC filtering on them is a trivial polynomial approximation for GAC filtering. Iterating `alldifferent` and `contiguous` GAC filtering is a second polynomial filtering. However, these polynomial filterings are weaker propagation than `saw` GAC filtering.

• The `alldistant` constraint formalizes the fact that different amino acids of a protein have a specific volume occupancy. Given n variables $X_1, \ldots, X_n$, with respective domains $D^{X_1}, \ldots, D^{X_n}$, and n numbers $c_1, \ldots, c_n$, admissible solutions $X_1 = p_1, \ldots, X_n = p_n$ are searched such that p_i and p_j are located at distance at least $c_i + c_j$, with $1 \leq i, j \leq n$. More formally:

$$\texttt{alldistant}(X_1, \ldots, X_n, c_1, \ldots, c_n) = (D^{X_1} \times \cdots \times D^{X_n}) \setminus \{(a_1, \ldots, a_n) : \exists i, j.\ 1 \leq i < j \leq n \wedge ||a_i - a_j||_2 < (c_i + c_j)\}$$

Note that `alldistant` with $c_1 = \frac{1}{2}, \ldots, c_n = \frac{1}{2}$, is equivalent to `alldifferent`. The authors prove that the CON and GAC test are both NP-complete.

The authors also introduce and study the *rigid block constraint* and prove that CON and GAC are in P. Future work is needed for fast implementation of the polynomial time algorithms and of efficient approximations of the NP-complete tests and filtering.

1.7. Counting Protein Structures by DFS with Dynamic Decomposition—by Sebastian Will and Martin Mann

The authors introduce depth-first search with dynamic decomposition for counting all solutions of a binary CSP. In particular, they use their method for computing the number of minimal energy structures for a discrete protein model.

The number of minimal energy structures of proteins in a discrete model is an important measure, which is strongly related to protein stability. The enumeration of optimal and suboptimal structures has applications in the study of protein evolution and kinetics [REN 97, WOL 06]. Even the prediction of protein structures in simplified protein models is a complex, NP-completecombinatorial optimization problem that received lots of interest in the past. Importantly for the presented work, it can be successfully modeled as Constraint Satisfaction Problem (CSP) [BAC 01, BAC 06]. Recently, counting solutions of a CSP in general and related problems gained a lot of interest over considering only satisfiability [ROT 96]. This is partly due to the increased complexity of counting compared to deciding on satisfiability [PES 05]. For general CSPs and in particular for protein structure prediction, solving is NP-complete. However, the counting of CSP solutions is an even harder problem in the complexity class #P. This class is defined as the class of counting problems associated with nondeterministic polynomial time computations.

Standard solving methods in constraint programming like Depth-First Search (DFS) combined with constraint propagation are well suited for determining one solution, but leave room for saving redundant work when counting all solutions. Here, the authors present a method that is especially tailored for this case. Applied to the CSP formulation of structure prediction, it improves exhaustive counting and enumeration of optimal protein structures.

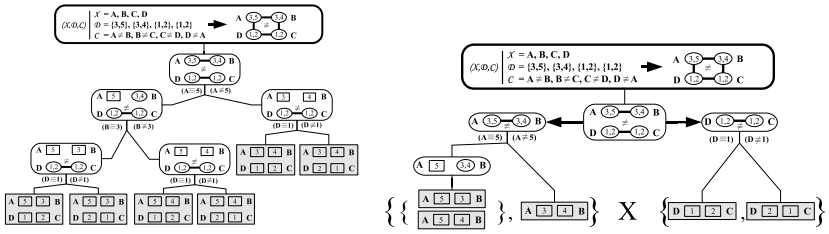


Figure 1.5. *Search tree traversed by DFS (left) and DDS (right) search.*

Basically, the new method *dynamically* decomposes the constraint (sub-)problems that emerge during the search into independent partial problems along connected components of the problem's associated constraint graph. Separate counting in the partial problems still allows to infer the number of solutions of the complete problem. **Fig. 1.5** illustrates the new decomposing strategy. For a simple example CSP, the standard non-decomposing search yields a search tree **Fig. 1.5**(left). For the same problem, dynamically decomposing search yields the search tree in **Fig. 1.5**(right). Even for the small example the tree is much reduced. Note that at the same time it yields a sort of compressed representation of the solution space.

Instead of *statically* exploiting only properties of the initial constraint graph, dynamic strategies analyze the emerging constraint graphs during the search and employ their features. The authors believe this is a major advantage in many constraint problems. In particular, if the initial constraint network is very dense (as in the structure prediction problem), static methods don't make an impact.

Decomposing into connected components and, more generally, utilizing the special structure of the constraint graph is discussed already for a long time. As their main contribution, Will and Mann demonstrate that the ideas of employing the graph structure dynamically are applicable to binary CSPs, even including certain global constraints, and are useful for constraint programming. In particular, this allows to use the strategy in the complex problem of protein structure counting. Furthermore, they discuss several ideas going beyond previous approaches. For example, dynamic decomposition can yield a more compact representation of the solution space.

The paper shows that the introduced method can be generalized such that even global constraints can be used. As shown the strategy of dynamically decomposing the (sub-)problems into partial problems reduces the search tree significantly. Since partial problems can be efficiently detected using well established graph algorithms, this results in a speed up of the search. Beyond this, the authors discussed how the graph structure can guide the variable and value selection in order to achieve many balanced decompositions, e.g. by the identification of articulation points. Such considerations go beyond previous work on constraint graph decomposition.

The application of dynamically decomposing search (DDS) to the CPSP problem shows the large capabilities of the method. First results with a prototypic implementation already show a significant speedup. Improving the ability for counting and enumerating optimal structures has important implications for the investigation of protein evolution and the folding process.

The paper gives evidence that the more general approach of dynamically analyzing the constraint graph during the search and employing its special structure has a large potential for solution counting in constraint programming. To the authors conviction, exploring these possibilities even further is an interesting field for future research.

1.8. Suffix Array and Weighted CSPs—by Matthias Zytnicki, Christine Gaspin, and Thomas Schiex

The authors describe a new constraint that uses the data structure suffix array, well-known in pattern matching. They show how it helps answering the question of non-coding RNA detection (ncRNA), and more precisely, finding the best hybrid in a duplex constraint. A ncRNA is usually represented by a sequel of letters, or *nucleotides*: A, C, G and T and it also contains *interactions*—mainly A–T and C–G—that are essential to its biological function.

The authors work under the assumption that the *structure* (namely, the set of information located on a ncRNA that discriminate for a given biological function) is known. The aim of the work is to answer the following question: how can I get all the candidates matching a given structure in a sequence that may contain several billions of nucleotides?

One of the main approaches to solve this problem uses statistical information in a context-free grammar that describes this structure [EDD 94]. However, some complex ncRNA families cannot be described within this formalism and [VIA 04] showed that only NP-hard formalisms may correctly describe them. This favors a CSP model of the problem.

However, usual queries give hundred of thousands of solutions and, in practice, it is impossible to exploit this huge amount of solutions.

This is why the authors use the weighted CSP (WCSP) [LAR 04] formalism to solve the ncRNA detection problem In WCSP a cost can be associated to each domain value in order to express preferences. A *valuation structure* $\mathcal{S} = \langle E, \oplus, \leq \rangle$ specifies the costs, where: $E = [0..k] \subseteq \mathbb{N}$ is the *set of costs*. The highest cost k can possibly be ∞, and it represents an *inconsistency*. $\leq$ is the usual operator on $\mathbb{N}$ and $\oplus$, the *addition* on E, is defined by $\forall (a, b) \in \mathbb{N}^2, a \oplus b = \min\{a + b, k\}$. A WCSP is a tuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where: $\mathcal{S}$ is the valuation structure; $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of n *variables*; $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$ is the set of possible *values* of each variable,

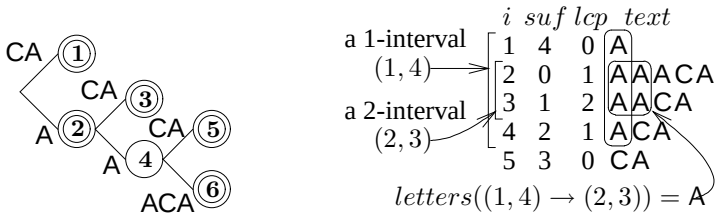
or *domains*, and the size of the largest one is d; $\mathcal{C} = \{c_1, \ldots, c_e\}$ is the set of e soft constraints. Assignments, which are defined as usual, can now not only be permitted or forbidden by a (soft) constraint but also be admissible with a certain cost. The cost of an assignments is the sum of costs over all constraints.

In the model, the variables represent the *positions* on the sequence of the elements of structure. The initial domain of the variables will therefore be equal to the size of the sequence. The constraints enforce the presence of the wished elements of structure between the specified variables. Within this model, a solution is a position for each variable, such that all the elements of structure specified by the constraints can be found. The aim is to find all the solutions of the problem, i.e. assignments with a cost less than a maximum cost k.

The authors focus on the duplex constraint. This constraint ensures that there exists a set of interactions between one sequence (the *main sequence*) and another given sequence (the *target sequence*). It has two parameters: the target sequence and the maximum number of errors in the interaction set. Similarly to the edit distance, the number of errors of a hybridization is the number of nucleotides that do not interact with any other nucleotide, plus the number of pairs of nucleotides associated through a non-allowed interaction. This will be the cost given by the constraint. The duplex constraint involves four variables: x_i, x_j, y_k and y_l. x_i represents the start position of the main stem, x_j represents its end position, whereas y_k and y_l represent the start and end positions of the target stem. To solve the problem, the authors use a depth-first branch-and-bound algorithm that maintains an extension of 2B-consistency adapted to soft constraints, called *bound arc consistency* (BAC*, [ZYT 06]).

The main contribution of the paper is to develop an algorithm for maintaining bound arc consistency for the duplex constraint. The algorithm uses the data structure of suffix arrays. Suffix arrays have several advantages over the more widely known suffix trees. The *suffix tree* is a tree with edges labeled with words. This data structure has been widely used in pattern matching algorithms. Given a text, the paths from the root node of its suffix tree and its terminal nodes enumerate all the suffixes of this text (cf. **Fig. 1.6**(a)) for the string AAACA). Basically, a *suffix array* is an array where all the suffixes of a text are sorted through lexicographic order (cf. **Fig. 1.6**(b)), which can be used to simulate a suffix tree. Both data structures allow fast lookup of sub-sequences.

The algorithm takes as an input the suffix array S, a word w of size n and a maximum edit distance $maxErr$. It returns the minimum distance between w and any subsequence of T, or $maxErr + 1$ if this distance is greater than $maxErr$. It uses a hybridization cost matrix c_{hyb}, that, given two nucleotides, returns the hybridization penalty (0 being a perfect hybridization). c_{ins} is the penalty cost for a non-hybridized nucleotide.



(a) suffix tree (b) suffix array

Figure 1.6. *Two representations of the suffixes of AAACA*

After introducing a version of their algorithm the authors discuss several optimizations, which save redundant work and take advantage of information in the WCSP about already reduced domains. In the future, the authors are going to compare their method with other existing ones, and provide for an empirical evaluation of their approach. An implementation is available at `carlit.toulouse.inra.fr/Darn/index.php`.

1.9. Supertree Construction with Constraint Programming: recent progress and new challenges—by Patrick Prosser

One goal of biology is to build the *Tree of Life* (ToL), a representation of the evolutionary history of every living thing. To date, biologists have catalogued about 1.7 million species, yet estimates of the total number of species ranges from 4 to 100 million. Of the 1.7 million species identified only about 80,000 species have been placed in the ToL [PEN 03]. There are applications for the ToL: to help understand how pathogens become more virulent over time, how new diseases emerge, and how recognizing species at risk of extinction. One approach to building the ToL is to combine smaller trees into "supertrees". Phylogenetic trees have been created for relatively small sets of species (see `www.treebase.org`). These trees are then combined together into supertrees.

The problem of supertree construction is to combine leaf labelled species trees, where there is an intersection in the leaf labels of those trees. The trees must be combined whilst respecting all the arboreal relationships in each tree. One of the first techniques for supertree construction is the OneTree algorithm [NG 96]. Using the same terminology, in **Fig. 1.7** (1), (2) and (3) there are the triples $(ab)c$, $(ac)b$, and $(bc)a$[2] and in (4) the fan (abc).

The constraint programming model for this problem [GEN 03] is based on the observation that any rooted species-tree is *ultrametric*. Ultrametric trees can be uniquely constructed from ultrametric matrices. In such matrices M, the ultrametric condition

2. where $(xy)z$ can be read as "x is closer to y than z"

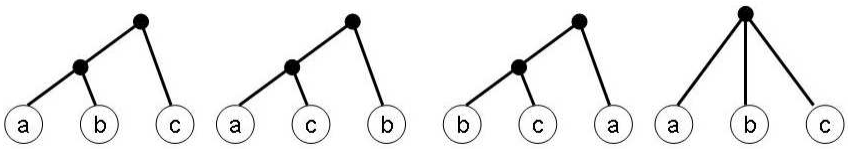


Figure 1.7. *The four possible relationships between three leaf nodes in a tree.*

holds, i.e. for any triplet i, j, k the distances $M_{i,j}$, $M_{i,k}$, and $M_{k,j}$ are either all equal or two of them are equal and the remaining one is smaller [GUS 97]. Given an ultrametric tree T and its ultrametric matrix M, it holds that the most recent common ancestor $mrca(i, j)$ of two leaf nodes i and j in T has depth $M_{i,j}$.

The constraint encoding starts by producing an $n \times n$ matrix M of constrained integer variables, each with a domain $1..n-1$. Amongst the trees to be combined there are exactly n species and each specie is mapped to an integer. The array M is symmetric such that $M_{i,j}$ is the same constrained integer variable as $M_{j,i}$ and all diagonal elements $M_{i,i}$ are preset to zero. An ultrametric constraint is blanketed across the array. This means that for all i, j, k where $1 \leq i < j < k \leq n$ the following constraint is posted

$$M_{i,j} < M_{i,k} = M_{j,k} \vee M_{i,k} < M_{i,j} = M_{j,k} \vee$$
$$M_{j,k} < M_{i,j} = M_{i,k} \vee M_{i,j} = M_{i,k} = M_{j,k}$$

The species trees are then broken up, using the BreakUp algorithm [NG 96], into triples and fans. These triples and 3-fans are then used to break disjunctions in the above constraint. The M variables are used as decision variables for finding a solution.

After describing the basics, Prosser applies the model for constructing a supertree of sea birds. Further, the author incorporates the ancestral divergence dates into the into the constraint model. The idea follows the RANKEDTREE algorithm [BIN 04], which takes as input two species trees where interior nodes are assigned integer values such that if the divergence of species A and B predates the divergence of species X and Y then the most recent ancestor of A and B will be assigned a value less than the most recent common ancestor of species X and Y.

The model is self limiting by its cubic size. There are $O(n^3)$ ternary constraints and the same number of variables for the the optimization problem (minimizing fans). The largest trees that were built have about 70 species. A next step is to make this model more compact, and this might be done by implementing a specialized ultrametric constraint that involves three variables. This constraint might propagate more efficiently than as at present (using toolkit primitives) and each of the constraints might take less space. To reduce the number of constraints, an n-ary ultrametric constraint that takes as arguments the $n \times n$ array M could be introduced.

The model is available at `www.dcs.gla.ac.uk/~pat/superTrees`. The author shows the versatility of the constraint programming technology, by taking a model that essentially does the same as OneTree. Then he modified it to take a forest as input, dealt with ancestral divergence dates, managed to produce all solutions compactly, and addressed an optimization problem (although this might not be biologically sound). However, the model is limited in what it can do by its sheer size, and this should be addressed soon. The author believes that constraint programming will be the technology to retrieve the common information that is carried in all these supertrees.

1.10. Bibliography

[BAC 01] BACKOFEN R., WILL S., "Fast, Constraint-based Threading of HP-Sequences to Hydrophobic Cores", *CP2001*, vol. 2239 of *LNCS*, p. 494–508, 2001.

[BAC 06] BACKOFEN R., WILL S., "A Constraint-Based Approach to Fast and Exact Structure Prediction in Three-Dimensional Protein Models", *Constraints*, vol. 11, num. 1, p. 5–30, 2006.

[BIN 04] BININDA-EMONDS O., *Phylogenetic supertrees: Combining information to reveal the Tree of Life*, Springer, 2004.

[BOR 06] BORTOLUSSI L., "Stochastic Concurrent Constraint Programming", *4th International Workshop on Quantitative Aspects of Programming Languages*, 2006.

[CAL 06] CALZONE L., CHABRIER-RIVIER N., FAGES F., SOLIMAN S., "Machine learning biochemical networks from temporal logic properties", *Transactions on Computational Systems Biology*, vol. 4220 of *LNCS*, 2006.

[CLA 99] CLARKE E. M., GRUMBERG O., PELED D. A., *Model Checking*, The MIT Press, 1999.

[Dal 04] DAL PALÙ A., DOVIER A., FOGOLARI F., "Constraint Logic Programming approach to protein structure prediction", *BMC Bioinformatics*, vol. 5, num. 186, 2004.

[Dal 05] DAL PALÙ A., DOVIER A., PONTELLI E., "A New Constraint Solver for 3D Lattices and Its Application to the Protein Folding Problem", *LPAR 2005*, vol. 3835 of *LNCS*, p. 48–63, 2005.

[EDD 94] EDDY S., DURBIN R., "RNA sequence analysis using covariance models", *Nucleic Acids Research*, vol. 22, p. 2079–2088, 1994.

[FAG 04a] FAGES F., "From syntax to semantics in systems biology - towards automated reasoning tools", *Converging Sciences*, vol. 3939 of *LNCS*, 2004.

[FAG 04b] FAGES F., SOLIMAN S., CHABRIER-RIVIER N., "Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM", *Journal of Biological Physics and Chemistry*, vol. 4, p. 64–73, 2004.

[GEN 03] GENT I. P., PROSSER P., SMITH B. M., WEI C. W., "Supertree Construction with Constraint Programming", *CP2003*, vol. 2833 of *LNCS*, p. 837–841, 2003.

[GIL 77] GILLESPIE D., "Exact Stochastic Simulation of Coupled Chemical Reactions", *J. of Physical Chemistry*, vol. 81, num. 25, 1977.

[GUS 97] GUSFIELD D., *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, 1997.

[KOH 99] KOHN K. W., "Molecular interaction map of the mammalian cell cycle control and DNA repair systems", *Molecular Biology of the Cell*, vol. 10, p. 2703–2734, 1999.

[KRI 02] KRIPPAHL L., BARAHONA P., "PSICO: Solving Protein Structures with Constraint Programming and Optimization.", *Constraints*, vol. 7, num. 3–4, p. 317–331, 2002.

[KRI 05] KRIPPAHL L., BARAHONA P., "Applying Constraint Programming to Rigid Body Protein Docking.", *CP2005*, vol. 3079 of *LNCS*, p. 373–387, 2005.

[LAR 04] LARROSA J., SCHIEX T., "Solving weighted CSP by maintaining arc-consistency", *Artificial Intelligence*, vol. 159, p. 1–26, 2004.

[NG 96] NG M. P., WORMALD N. C., "Reconstruction of rooted trees from subtrees", *Discrete Applied Mathematics*, vol. 69, p. 19–31, 1996.

[PEN 03] PENNISI E., "Modernizing the Tree of Life", *Science*, vol. 300, p. 1692–1697, 2003.

[PES 05] PESANT G., "Counting Solutions of CSPs: A Structural Approach", *IJCAI2005*, p. 260–265, 2005.

[PRI 01] PRIAMI C., REGEV A., SHAPIRO E. Y., SILVERMAN W., "Application of a stochastic name-passing calculus to representation and simulation of molecular processes", *Inf. Process. Lett.*, vol. 80, num. 1, p. 25–31, 2001.

[REG 01] REGEV A., SILVERMAN W., SHAPIRO E. Y., "Representation and simulation of biochemical processes using the pi-calculus process algebra", *Proceedings of the sixth Pacific Symposium of Biocomputing*, p. 459–470, 2001.

[REN 97] RENNER A., BORNBERG-BAUER E., "Exploring the Fitness landscapes of lattice proteins", *2nd. Pacif. Symp. Biocomp.*, Singapore, 1997.

[ROT 96] ROTH D., "On the Hardness of Approximate Reasoning", *Artif. Intelligence*, vol. 82, num. 1–2, p. 273-302, 1996.

[SAR 93] SARASWAT V. A., *Concurrent Constraint Programming*, MIT press, 1993.

[SKO 04] SKOLNICK J., KOLINSKI A., "Reduced models of proteins and their applications", *Polymer*, vol. 45, p. 511–524, 2004.

[TYS 91] TYSON J. J., "Modeling the cell division cycle: cdc2 and cyclin interactions", *Proceedings of the National Academy of Sciences*, vol. 88, p. 7328–7332, 1991.

[VIA 04] VIALETTE S., "On the computational complexity of 2-interval pattern matching problems", *Theoretical Computer Science*, vol. 312, p. 223–249, 2004.

[WOL 06] WOLFINGER M., WILL S., HOFACKER I., BACKOFEN R., STADLER P., "Exploring the lower part of discrete polymer model energy landscapes", *Europhysics Letters*, vol. 74, num. 4, p. 725–732, 2006.

[ZYT 06] ZYTNICKI M., SCHIEX T., GASPIN C., "A new local consistency for weighted CSP dedicated to long domains", *SAC2006*, p. 394–398, 2006.