Protein Folding Simulation in Concurrent Constraint Programming

Luca Bortolussi [a,2] Alessandro Dal Palù [a,2] Agostino Dovier [a,2]
Federico Fogolari [b,3]

[a] *Dip. di Matematica e Informatica, Univ. di Udine.*
Via delle Scienze 206, 33100 Udine (Italy).

[b] *Dip. Scientifico-Tecnologico, Univ. di Verona.*
Strada Le Grazie 15, 37134 Verona (Italy).

Key words: *Computational Biology, Concurrent Constraint Programming.*

Abstract

A protein is identified by a finite sequence of amino acids, each of them chosen from a set of 20 elements. The Protein Structure Prediction Problem is the problem of predicting the 3D native conformation of a protein, when its sequence of amino acids is known. This problem is fundamental for biological and pharmaceutical research. All current mathematical models of the problem are affected by intrinsic computational limits. In particular, simulation-based techniques that handle every chemical interaction between all atoms in the amino acids (and the solvent) are not feasible due to the huge amount of computations involved. These programs are typically written in imperative languages and hard to be parallelized. Moreover, each approach is based on a particular energy function. In the literature there are various proposals and there is no common agreement on which is the most reliable.

   In this paper we present a novel framework for ab-initio simulations using Concurrent Constraint Programming. We are not aware of any other similar proposals in the literature. Each amino acid of an input protein is viewed as an independent process that communicates with the others. The framework allows a modular representation of the problem and it is easily extensible for further refinements. Simulations at this level of abstraction allow faster calculation. We provide a first preliminary working example in Mozart, to show the feasibility and the power of the method. The code is intrinsically concurrent and thus easy to be parallelized.

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

1 Introduction

The Protein Structure Prediction Problem (PSP) is the problem of predicting the 3D *native* conformation of a protein, when the sequence made of 20 kinds of amino acids (or *residues*) is known. The process for reaching this state is known as the protein *folding*. This problem is fundamental for biological and pharmaceutical research. Currently, the native conformations of more than 26000 proteins are available in the Protein Data Bank (PDB) [3]. Restricting to proteins of typical length (i.e. less than 500), there are more than 20^{500} possible sequences. Moreover, for a single protein even admitting only two allowed positions for an amino acid given the positions of the previous, the number of possible conformations is of the same order. These numbers highlight the need for a general computational tool.

The PSP problem can be modelled as an optimization problem involving energy functions to be minimized and constraints on the amino acids' positions. Even simple abstractions are NP-complete (see, e.g., [8]). Nevertheless, in the last thirty years, the global optimization for the PSP problem has been tackled with different classes of methods (*ab-initio modelling*): simulated annealing [14], genetic algorithms [13], smoothing methods [25], branch and bound [18] and constraints [2]. When some additional knowledge is available (e.g., a database of already annotated proteins), it is possible to employ a different class of methods (*homology modelling*): the protein is matched against very similar sequences and the conformation prediction exploits this valuable information.

In this work we concentrate on *ab-initio* modelling. In this case, an all-atom computer simulation is typically unpractical. Even with strong simplifications it would require many CPU hours per nanosecond for a small protein on a PC. To overcome this limit, most of the approaches use *database fragment assembly* and/or *simplified models* to determine an approximate and faster solution.

Ab-initio methods are based on the *Anfinsen thermodynamic hypothesis* [1]: the (*native*) conformation adopted by a protein is the most stable one, i.e. the one with minimum free energy. A fundamental role in the design of a predictive method is played by the spatial representation of the protein and the static energy function, which is to be at a minimum for native conformations.

Simplified models of proteins are attractive in many respects: they allow clear derivation of kinetic and thermodynamic properties; the simplified representation of the actual protein, often by one or two centers of interaction per residue, allows a much faster computation and generates also smoother energy hyper-surfaces which implies faster dynamics. Unfortunately, there is no general agreement on the potential that should be used with these models, and several different energy functions can be found in literature [24].

In this paper we present a new high-level framework for ab-initio simu-

2

lation in Concurrent Constraint Programming [22]. CCP is a programming framework where computations are based on communication (using ask and tell primitives) between independent processes/agents. Each agent is capable to check satisfiability and/or impose new constraints. Each amino acid in the protein is modelled as an independent and parallel process, which reacts to modifications of spatial positions of other amino acids. Each process evolves in a Montecarlo simulation framework, exploiting the most recent information available about the surrounding objects. Every time a process updates its position, it also tells the changes to the others with a communication based on the instantiation of logical variables.

Note that the framework is independent from the protein spatial model and energy function. As preliminary test, we adopt a basic energy function description, used to implement in Mozart [26] a first version of the simulation exploiting the CCP techniques. In this model, each aminoacid is represented by an off-lattice, single center of interaction. The energy function is composed by the empirical contact term developed in [4] and used in constraint-based approach to the problem [10,9]. Moreover, to model properly an off-lattice energy field, we also include a bond length term, a bend angle term and a torsion angle term, similar to [27]. Note that in [20] the correlation between torsion angles and related bend angles is thoroughly studied. In this paper, we do not provide many details on this specific energy function and tuning procedure, since in the next future we plan to test our framework replacing this naive energy model with a well-known reduced model (i.e. UNRES [15]).

To test our framework, we simulate a poly-alanine sequence, which has an high tendency to fold into an α-helix. Even with a so coarse model (in terms of amino acid representation and energy function), we obtain a proper helical structure.

The paper is organized as follows. In Sect. 2 we briefly discuss the main results related to the solution of the PSP problem. The problem is then formalized in Section 3. In Sect. 4 we present our encoding in CCP. In Sect. 5 we describe the energy model employed. In Sect. 6 we provide some details of the Mozart implementation and in Sect. 7 we show our preliminary results. In Sect. 8 we draw some conclusions.

2 Related Work

We refer to [24] for a detailed review. We wish to focus here on the *ab-initio* approaches, namely, approaches that are not based on similarities to already known proteins. All-atoms ab-initio simulations by means of molecular dynamics (e.g. [21,5,16]), are precluded by the intrinsic complexity of the needed operations. More efficient methods are offered by simplified models. It is accepted that important features of protein sequences are the local propensity to adopt well-defined secondary structures, as well as the polar and hydrophobic interactions. The secondary structure propensity may be included in simula-

A	**Ala**nine	4	C	**Cys**teine	4
D	**Asp**artic Acid	16	E	**Glu**tamic Acid	10
F	**Phe**nylalanine	14	G	**Gly**cine	1
H	**His**tidine	11	I	**I**soleucine	13
K	**Lys**ine	15	L	**Leu**cine	13
M	**Met**hionine	11	N	**A**sparagi**n**e	8
P	**Pro**line	8	Q	**Gl**utamine	11
R	**Arg**inine	17	S	**Ser**ine	5
T	**Thr**eonine	9	V	**Val**ine	10
W	**Tryp**tophan	18	Y	**Tyr**osine	15

Fig. 1. Amino acids: abbreviations, full names, number of atoms in side chain and structure

tion through either rigid constraints (as done in [10,9]) or energy terms which depend on the type of amino acids involved, derived from statistical database analysis. Interactions between amino acids may be treated either considering their chemical and physical properties or using a statistical approach. One relevant problem is the correlation between the different propensities and interactions singled out: as far as empirical contact energies are concerned, in [4] it is compiled a table which has been proven to be rather accurate, when tested on several decoys' sets. Similar tables have been provided based on different criteria by other authors [17]. Thirumalai and coworkers [27] have designed a forcefield suited for representing a protein through its C_α-chain, which includes bonds, bend, and torsion angle energy terms. Scheraga and coworkers [15] have used a similar model including side-chain centroids.

As we describe in Sect. 3, the problem can also be formulated as a non-linear minimization problem, where the spatial domain for the amino acids is a discrete lattice. A constraint-based approach to this problem on the so-called *Face Centered Cubic lattice*, with a further abstraction on amino acids (they are split into two families H and P), is successfully solved in [2] for proteins of length up to 160. A constraint-based solution to the general problem (with the 20 amino acids) is proposed instead in [10,9], where proteins of length up to 50 are solved. In the latter approach, the solution search is based on the constraint solver for finite domains of SICStus Prolog [6].

As said above, we are not currently aware of any other approach modelling amino acids as concurrent processes.

3 Proteins and the PSP Problem

A protein is a sequence of 20 kinds of linked units, called amino acids. This sequence is called the *primary structure* of a protein.

Each protein always reaches a peculiar 3D conformation, called native conformation or *tertiary structure*, which determines its function. The *protein structure prediction problem* is the problem of predicting the tertiary structure of a protein given its primary structure. It is accepted that the primary structure uniquely determines the tertiary structure. Due to entropic considerations [1], it is also accepted that the tertiary structure is the one that

minimizes the global energy of the protein. Though, there is no common agreement on which energy function should model correctly the phenomenon.

Each amino acid is made by several atoms (cf. Fig. 1); there is a part *common* to all amino acids, the N-C_α-C' backbone, and a *characteristic part* known as *side chain*, which consists of a number of atoms ranging from 1 to 18. Each amino acid is linked to the following with the incoming and outgoing edges represented by arrows in Fig. 1. A well-defined energy function should consider all possible interactions between all atoms of every amino acid composing the protein. A review of the various forces and potentials at this abstraction level can be found in [19].

A more abstract view of amino acids considers each of them as a single *sphere* centered in the C_α atom. The distance between two consecutive C_α atoms is assumed to be 3.8 Å, measure chosen as unitary. Recent work has been done to model energy functions with this level of abstraction. A pair of non consecutive amino acids contributes to the energy when the two amino acids are in *contact*, namely under a given distance that can be approximated by 2 units. A table that points out the energy associated to pairs of amino acids in contact has been developed [17,4].

Just as an example, we show how is possible to formalize the protein folding problem as an optimization problem. Given a sequence $S = s_1 \cdots s_n$, with s_i aminoacids, a folding *of S is a function $\omega : \{1, \ldots, n\} \longrightarrow \mathbb{R}^3$ such that: $|\omega(i) - \omega(i+1)| = 1$, and $|\omega(i) - \omega(j)| \geq 1$ for $i \neq j$. The first constraint states that consecutive aminoacids have the fixed unitary distance; the second that each aminoacid occupies a unitary sphere and that two spheres cannot overlap. In Sect. 5 we present a more refined description that deals with local interactions. In particular, the hard constraint $|\omega(i) - \omega(j)| \geq 1$ will be rewritten as a soft constraint, namely a smoother potential barrier that has the effect of repelling two colliding residues, together with other local energy interactions.*

The protein folding problem can be reduced to the optimization problem of finding the folding ω of S such that the following energy is minimized [7,9]:

$$E(\omega) = \sum_{\substack{1 \leq i < n \\ i+2 \leq j \leq n}} \mathsf{contact}(\omega(i), \omega(j))\mathsf{Pot}(s_i, s_j)$$

where $\mathsf{contact}(\omega(i), \omega(j))$ is 1 if $|\omega(i) - \omega(j)| \leq 2$, 0 otherwise. $\mathsf{Pot}(x, y)$ denotes the energy value associated to a contact between amino acids x and y.

---

4 The CCP Simulation Framework

In this section we describe the novel abstract framework of simulation. The approach is independent on the spatial model of the protein and on the energy model employed. Using a Concurrent Constraint Logic Programming language we can encode the problem associating an independent process to each amino acid involved. Processes react to changes of position of other processes.

4.1 The main program

Let us assume that the input is the list $S = [s_1, \ldots, s_n]$ of amino acids. The

```
 1 simulation(S):-                    7 run(ID, S, [P1, P2, ..., Pn]):-
 2    Init=[[I1|_],                    8   getTails([P1, ..., Pn],[T1, ..., Tn]),
           [I2|_],                     9   ask(T1=[_|_]) -> skip +
           ...,                       10   ask(T2=[_|_]) -> skip +
           [In|_]],                   11   ... +
 3    run(1,S,Init) ||               12   ask(Tn=[_|_]) -> skip,
 4    run(2,S,Init) ||               13   getLast([P1, ..., Pn],[L1, ..., Ln]),
 5    ... ||                         14   updatePosition(ID,S,[L1,..,Ln],NP),
 6    run(n,S,Init).                 15   tell(TID=[NP|_]),
                                     16   run(ID,S,[P1, ..., Pn]).
```

Fig. 2. The abstract CCP Program

main procedure is described in Fig. 2 by lines 1–6. The variable `Init` contains
n lists, each of them associated to the respective amino acid. `I1 ...   In`
are the initial *positions* of the amino acids $s_1, \ldots, s_n$, respectively. Each list
contains the history of moves done by the corresponding amino acid. This
procedure concurrently calls `n` executions of the procedure `run` each of them
instantiated by the index `i` in the input sequence passed as parameter.

The process related to the amino acid `ID` is activated when a new position
is computed by some other amino acid (lines 9–12). This control is made by
checking that unbounded variables become a list.

New positions are appended to the tail, ensuring that the last element
is always a new unbounded variable. For example, if the element `P4` has to
be added to the list $[P1, P2, P3|Tail]$, the information is updated posting the
constraint `Tail` $= [P4|NewVar]$, where `Tail` and `NewVar` are variables (line 15),
obtaining the new list $[P1, P2, P3, P4|NewVar]$.

Note that many concurrent updates can occur while a process is monitoring
the changes. When at least one modification happened, the predicate `getLast`
retrieves the latest information available and `updatePosition` calculates the
new position as explained in the next Subsection. When the new position is
computed, it is added to the list by means of `tell` and the process performs
another loop.

4.2 Simulating moves

The procedure `run` (lines 7–16) computes a new position for the correspond-
ing amino acid implementing a Montecarlo-like simulation (cf., e.g., [7, page
44]). Each amino acid can move in the space guided by its evaluation of the
energy function. The knowledge of other's positions and amino acids' type
is sufficient to completely evaluate the function. Each time the procedure
`updatePosition` is invoked, the amino acid a_i estimates its *current* poten-
tial P_c, according to its position p. Moreover, a new position p' is devised,
according to a moving strategy (see next Subsection).

The amino acid evaluates then the *new* potential P_n associated to p' (P_c and P_n are computed using the formula (1)). If $P_n < P_c$, the amino acid updates its position to p'.

If $P_n \geq P_c$, the position p' seems not suitable to improve the local potential. However, it is possible that moving the aminoacid in worse positions allows to exit from a position of a local minimum. The Montecarlo technique is that of accepting the position p' if a randomly generated value rand $\in [0,1]$ is less than a value depending on $P_n - P_c$:

$$\text{newPosition}(\vec{p}, P_c, \vec{p'}, P_n, \text{Temp}) = \begin{cases} \vec{p'} & \text{if } P_n < P_c \text{ or rand} < e^{-\frac{P_n - P_c}{\text{Temp}}} \\ \vec{p} & \text{else} \end{cases}$$

Temp is a parameter simulating the temperature effects. Technically, it controls the acceptance ratio of moves that increase the energy. In Montecarlo algorithms Temp remains constant and it can be proven that allowing sufficient time the native conformation depending on the energy functions is reached. Simulating annealing methods slowly decrease the value of Temp during simulation (observe that when Temp is close to 0, the test rand $< e^{-\frac{P_n - P_c}{\text{Temp}}}$ is typically false, and thus only moves that decrease the energy are allowed).

4.3 Moving strategy

As said in Sect. 2, two consecutive amino acids tend to stay at a distance of 3.8 Å from each other. Instead of stating this as a hard constraint, we preferred to use a soft constraint (the bond energy term).

Moving the residues blindly in the space leads very often to a great increase of this energy contribution. Therefore, we implemented a more clever moving strategy, that selects a new position according to a specific spatial probability distribution.
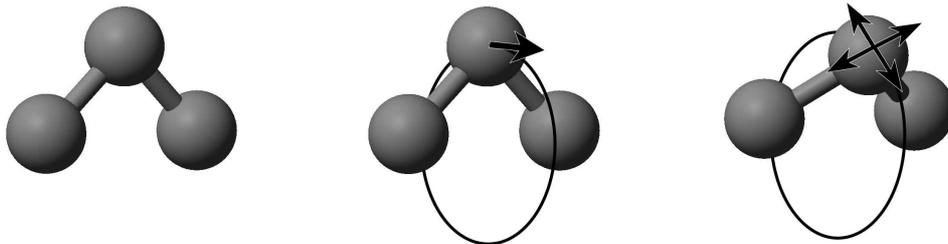

Fig. 3. Moving strategy: the 3 amino acids, moving along the circumference and leaving the circumference.

Let us consider the case of an internal aminoacid a_i in the sequence (cf. Fig. 3). If the distance between a_i and its neighbors is fixed, this residue has only one degree of freedom of movement, i.e. it can move only on the circumference which is the intersection of the two spheres centered in its neighbors and with radius equal to the respective distances.

Hence, we first randomly select a point P on this circumference according to a gaussian distribution, with the maximal probability assigned to the current position of a_i. Then, allowing for little variations of the distance, we select a point in the plane orthogonal to the circumference and passing for P, according to a 2–dimensional gaussian distribution centered in P (the resulting distribution is a non–uniform toroidal distribution in the space, with its peak centered in the current position of the residue).

The case of the first and the last aminoacids is slightly different, as they have two degrees of freedom, namely the surface of the sphere centered in their only neighbors. Hence, we select a point on this surface, according to a 2–dimensional gaussian distribution centered in the current position of the residue, and then we shift it along the radius, always according to a gaussian distribution centered on the surface.

Our moving strategy is able to explore the whole space of configurations, although we give less chances to those moves that provide high energy contribution.

5 Energy function for folding simulation

In this section we briefly describe the energy function we use to implement our preliminary simulation of the folding process dynamics. However, as already said, our framework is parametric w.r.t. a given energy function. The four energy contributions are related to *bond distance* (E_b), *bend angle* (E_a), *torsion angle* (E_t), and *contact interaction* (E_c) (cf. [27]). For the sake of simplicity, assume that $\vec{s} = s_1, s_2, \ldots, s_n$ contain the names of the n amino acids as well as their positions. The total energy E depends on $\vec{s}$ as follows:

(1) $E(\vec{s}) = \eta_b\, E_b(\vec{s}) + \eta_a\, E_a(\vec{s}) + \eta_t\, E_t(\vec{s}) + \eta_c\, E_c(\vec{s})$

where $\eta_b, \eta_a, \eta_t, \eta_c$ are set for blending the energy contributions.

For each pair of *consecutive* amino acids s_i, s_{i+1}, we have a quadratic term

(2) $E_b(\vec{s}) = \sum_{1 \le i \le n-1} \left(r(s_i, s_{i+1}) - r_0 \right)^2$

where $r(s_i, s_{i+1})$ is the distance between the C_α of the two amino acids and r_0 is the typical amino acid distance of 3.8 Å.

The bend energy is associated to the bend angle formed by a triplet of consecutive C_αs. The distribution is rather constant for every protein in the PDB and independent from the types of amino acids involved. The profile (see [11]) can be approximated by a combination of two Gaussian distributions, one around 120 degrees and the other, sharper, around 90 degrees. The energy is obtained applying the opposite logarithm to the distribution function:

(3) $E_a(\vec{s}) = \sum_{i=1}^{n-2} -\log \left(a_1\, e^{-\left(\frac{\beta_i - \beta_1}{\sigma_1} \right)^2} + a_2\, e^{-\left(\frac{\beta_i - \beta_2}{\sigma_2} \right)^2} \right).$

The torsion angle energy function is modelled using statistical information

of torsional behavior extracted from the PDB. In detail, four C_α atoms ($C_{\alpha i}$, $C_{\alpha i+1}$, $C_{\alpha i+2}$ and $C_{\alpha i+3}$) form a specific torsion angle: it is the angle between the normal to the plane spanned by $C_{\alpha i}, C_{\alpha i+1}$ and $C_{\alpha i+2}$ and the normal to the plane spanned by $C_{\alpha i+1}, C_{\alpha i+2}$ and $C_{\alpha i+3}$. The angle is positive when, looking along $\vec{r}_{23} = C_{\alpha i+2} - C_{\alpha i+1}$, the atom $C_{\alpha i+3}$ rotates clockwise. This angle is influenced both by the type of the amino acids involved and by their position in the protein. The information available in the PDB does not allow one to reconstruct a sharp distribution profile of each combination of amino acids (there are only 2.000 proteins with less than 25% homology, i.e. that carry non redundant information). Consequently, we first identify four classes of amino acids which share the same torsional behavior and we calculate the distribution profile for every sequence of 4 consecutive classes. This profile is approximated by the sum of two Gaussians. The function has the form:

$$(4) \quad E_t(\vec{s}) = \sum_{i=1}^{n-3} - \log \left(a_1\, e^{\frac{(\Phi_i - \phi_1)^2}{(\sigma_1 + \sigma_0)^2}} + a_2\, e^{\frac{(\Phi_i - \phi_2)^2}{(\sigma_2 + \sigma_0)^2}} \right)$$

where the parameters a_1, a_2, σ_1, σ_2, ϕ_1, ϕ_2 depend on the classes of the four residues, while σ_0 is used to adapt the distribution variance to an effective energy function. Actually, there is a well-known correlation between the bend and the torsion angles [20]. We have not used it in this paper.

For each pair of amino acids s_i and s_j, such that $|i - j| \geq 3$ we consider the contact interaction term of the form

$$(5) \quad E_c(\vec{s}) = \sum_{i=1}^{n-3} \sum_{j=i+3}^{n} \left[|\mathsf{Pot}(s_i, s_j)| \left(\frac{r_0(s_i, s_j)}{r(s_i, s_j)} \right)^{12} + \mathsf{Pot}(s_i, s_j) \left(\frac{r_0(s_i, s_j)}{r(s_i, s_j)} \right)^{6} \right]$$

where $r(s_i, s_j)$ is the distance between the C_α of s_i and s_j and $r_0(s_i, s_j)$ is a parameter describing the steric hindrance between a pair of non consecutive amino acids s_i and s_j. In our model, $r_0(s_i, s_j)$ is the sum of the radii of the two spheres that represent the two specific amino acids. An approximation of them is derived in [11].

A crucial problem while dealing with energy functions is to set correctly the parameters involved, as their value changes dramatically the energy landscape and influences the behavior of the simulation. We performed an optimization using a sampling of 700 proteins from PDB database with less than 25% homology. We calculated the sum of the squares of the difference between the energy computed for their native conformation and the folding energy (i.e., the free energy required to unfold a protein, which has been roughly approximated as proportional to the number of amino acids). Then we minimized this function using a simulated annealing method, identifying an optimal value for the scaling parameters. The rationale behind this procedure is to tune parameters as to have reasonable energies computed on known native protein structures.

As said before, we do not present here the fully detailed description of the methods involved, since this energy function is considered as a simple test to drive a real implementation of our framework.

6   Mozart implementation

In this section we describe some details of the implementation of the simulation technique in the language Mozart [26]. In Mozart it is natural to model concurrent programs in a simple notation mixing classes, constraints, and logic variables. Code's length is less than 1000 lines; the code can be found in http://www.dimi.uniud.it/dovier/PF.

The core scheme described in Sect. 4.1 is expanded with the help of two classes: `Protein` and `Amino`. The first class implements the `simulation` predicate and basically coordinates each process that is associated to an amino acid, i.e. amino acid creation, launch, statistics and termination. The `Amino` class describes each single amino acid. `Protein` contains and runs a number of `Amino` classes equal to the number of amino acids in the simulated protein.

As said before, lists are used to store positions of the various amino acids during computation. The data structure is described by the class `PosList` that allows us to access the tail and last elements of the list in constant time, making use of Mozart class attributes (non logical variables).

The method `add` behaves as the `tell` operator in CCP. Each call `add(El _)` expands the list by one element and leaves the internal variable `tail` not instantiated, as described in Sect. 4.1.

The `Amino` class provides some methods for the process maintenance (e.g. `live`, `act`, `die`, `debug`) and others for the loop `act`, described in Sect. 4.1. The concurrent `ask`s (lines 9–12) check if at least one tail gets instantiated (i.e. a new position is communicated by another amino acid). This test is accomplished by the method `Wait4News` reported in Fig. 4. To implement this check it is necessary to generate n threads, one for each tail to be checked. Our solution is to nest then n threads that launch a `cond` blocking test for each `ask` (cf. CCP translation of `ask` in [12]). In fact, since n is unbounded, it is not possible to statically write an explicit piece of code to handle the test. Each `cond` branch is associated to a distinct process that terminates either when its guard associated is realized or whenever another process terminates, by means of the additional shared variable `Done`. On line 21, the notation `_|_` denotes the structure of a list, with general unassigned variables (i.e. `_`). When a tail is instantiated, this variable is unified with a term of the kind `El | _` and thus the `cond` test is verified.

We give some details about the implementation of `updatePosition` and the subsequent `tell` (lines 14–15). The `updatePosition` method in Mozart reproduces what we described in Sect. 4.2. The computation of the energy is accomplished calling the method `energy(X Y Z E)`, where X,Y and Z are the coordinates of the amino acid and E is the energy associated as described in Sect. 5. Note that we invoke twice the function: the first time with the actual coordinates and the second with the new position generated by a random shift. Once the two energies are compared, the next position is set and communicated via `add([X Y Z] _ )` to the appropriate `PosList`.

```
17 meth Wait4News(N Done)
18        if (N\=@ID) then
19         thread
20            cond
21               {{Get @LocalAAList N} getTail($)} = _|_
22            then Done=unit
23            [] Done=unit then skip
24            end
25          end
26        end
27        if (N>1) then {self Wait4News(N-1 Done)} end
28 end
```

Fig. 4. Synchronization procedure

7 Experimental results

In this section we present the results of some tests of our program. First of all, note that we do not expect outstanding results, due to the naive energy function in use. In fact, we are interested in testing more the feasibility of the framework than the potential. However, we successfully run the code on a sequence of Alanines, that is known to have a high tendency to form a single helix. As initial state of the protein we set each amino acid along a line with a step of the bond distance (3.8 Å). We run the simulation for 60 seconds on a PC, 1GHz, 256MB. In Figure 7 we show the resulting proper helix for a list of 14 Alanines.
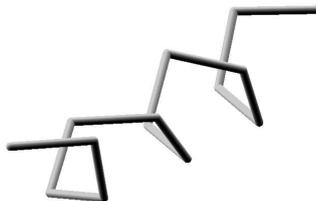

Fig. 5. The simulated sequence of Alanines

We would like to compare this result to a sequential simulation in **C** based on the same energy functions, where concurrency is simulated by a simple sequential loop on the amino acids. We noticed that the behavior of the folding pathway is different. In Mozart, some amino acids compute the energy with a partial updated information about the 3D conformation, due to concurrent communication and synchronism. In practice the amino acids consider the latest information available about the others and this fact influence the shape of the energy function. This shows that the concurrent approach is suitable to be employed in a folding process and it offers a new direction to be investigated, namely the parallel interaction between objects during the fold.

For completeness, we also tested the reliability of our energy function: we run the simulation for some PDB proteins (with known native conformation). It turned out that some aminoacids actually lie in a local minimum given our simple energy model. As expected, for some others the description does not

completely capture their properties. In fact, running a simulation for some time, we noted that the global minimum of these proteins tends to move into another conformation, which has better energy according to our model. This clearly shows that a more robust energy model is required. See the next section for other details.

8   Future work and Conclusions

In this paper we present a novel concurrent constraint programming framework to simulate a protein folding process. The approach is independent on the spatial representation and energy model of the proteins. The objective is to provide a powerful tool for obtaining the native conformation of a given protein; moreover, as a side effect, we compute all the history of the folding process. This scheme is general and it allows fast prototyping. The key idea is to identify every biological entity (i.e. amino acid, but easily extensible to atoms and molecules) with a concurrent process. We implemented a preliminary version, to show the feasibility and the power of the method. We define a simplified energy function, to test the implementation, and we are able to fold properly a helix.

In the future we plan to improve the level of approximation, i.e. to describe each residue by means of two elements: the C_α atom and the side-chain (cf. Fig. 7). In particular, we want to model the side-chains as ellipsoids and we plan to use the well-established UNRES force field [15]. This is a step towards all-atoms concurrent simulations. Our method requires simpler computations and, moreover, is easy to be parallelized.

It seems promising to design an optimized *communication framework* which adapts dynamically according to the 3D folding. For example, it could be possible to reduce the communications between non-influent pairs of entities (e.g. two distant amino acids provide a poor energy contribution, thus a lazy position update is feasible). Moreover, we want to formalize a novel concept of *cooperative approach*, in which a cooperation strategy between processes is induced dynamically by the current configuration. We want to investigate the possibility offered by our concurrent framework to represent the dynamical evolution of the system and to manage the propensity to form local regular sub-conformations (e.g. secondary structure elements), to achieve a speed up in the folding pathway.

Our concurrent constraints simulation could be combined with other approaches. Since the computational costs also depend on the complexity of the energy function, it seems reasonable to proceed by levels: a first phase (e.g. the simulation approach with the SICStus Prolog code developed in [9], which performs an on-lattice minimization of a simplified contact interaction energy) could be used to quickly generate a preliminary and coarse solution that can be used as input of our concurrent approach. The result of our simulation can be then refined again by an all-atom simulation. The output of each phase is a

folded protein with an optimal folding for the specific energy model. Refining the model and starting from a folding closer to the native state, makes the next phase more effective, since the conformational space to be searched is smaller.

Other concurrent implementations (e.g., in JCC [23] and in traditional concurrent Prolog languages, such as CIAO Prolog, CS-Prolog II, or Arity/Prolog32) will be carried on after the inclusion of UNRES energy function.

References

[1] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

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

[3] H. M. Berman et al. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000. http://www.rcsb.org/pdb/.

[4] M. Berrera, H. Molinari, and F. Fogolari. Amino acid empirical contact energy definitions for fold recognition in the space of contact maps. *BMC Bioinformatics*, 4(8), 2003.

[5] B. R. Brooks et al. Charmm: A program for macromolecular energy minimization and dynamics calculations. *J. Comput. Chem.*, 4:187–217, 1983.

[6] M. Carlsson, G. Ottosson, and B. Carlson. An open-ended finite domain constraint solver. In Proc. of *PLILP'97*, vol. 1292 of *Lecture Notes in Computer Science*, pp. 191–206. Springer-Verlag, Berlin, 1997.

[7] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction.* John Wiley & Sons, 2001.

[8] P. Crescenzi, D. Goldman, C. Papadimitrou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proc. of STOC*, pp. 597–603, 1998.

[9] A. Dal Palù, A. Dovier, and F. Fogolari. Protein folding in $CLP(\mathcal{FD})$ with empirical contact energies. In *Recent Advances in Constraints*, vol. 3010 of *Lecture Notes in Artificial Intelligence*, pp. 250–265, 2004.

[10] A. Dovier, M. Burato, and F. Fogolari. Using secondary structure information for protein folding in $CLP(\mathcal{FD})$. In *Sel. papers from 11th WFLP*, vol. 76 of *Electronic Notes in Theoretical Computer Science*, 2002.

[11] F. Fogolari, G. Esposito, P. Viglino, and S. Cattarinussi. Modeling of polypeptide chains as C-α chains, C-α chains with C-β, and C-α chains with ellipsoidal lateral chains. *Biophysical Journal*, 70:1183–1197, 1996.

[12] S. Haridi, P. Brand, E. Klintskog, and T. Sjöland. Using Mozart for Modelling and Simulation of TCCP. In Proc. of *4th Int'l Workshop on Constraint programming for time critical applications—ESPRIT working group COTIC*, 2000.

[13] J. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Computing*, (2):88–105, 1973.

[14] S. Kirkpatrick, C. D. Geddat Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, (220):671–680, 1983.

[15] A. Liwo, J. Lee, D. R. Ripoll, J. Pillardy, and H. A. Scheraga. Protein structure prediction by global optimization of a potential energy function. In *Proceedings of the National Academy of Science (USA)*, vol. 96, pages 5482–5485, 1999.

[16] A. D. Jr. MacKerell, et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[17] S. Miyazawa and R. L. Jernigan. Residue-residue potentials with a favorable contact pair term and an unfavorable high packing density term, for simulation and threading. *Journal of Molecular Biology*, 256(3):623–644, 1996.

[18] G. L. Nemhauser and L. A. Wolsey. *Integer Programming, Chapter VI in Optimization*. Number 1. Handbooks in Operations Research and Management Science, North Holland, Amsterdam, 2989.

[19] A. Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39:407–460, 1997.

[20] T. J. Oldfield and R. E. Hubbard. *Analysis of C alpha geometry in protein structures. Proteins*, 18(4):324-37,1994.

[21] D. Qiu, P. Shenkin, F. Hollinger, and W. Still. The gb/sa continuum model for solvation. A fast analytical method for the calculation of approximate born radii. *J. Phys. Chem.*, 101:3005–3014, 1997.

[22] V. A. Saraswat, M. C. Rinard, and P. Panangaden. Semantic Foundations of Concurrent Constraint Programming. Proceedings of *18th ACM POPL*, 1991.

[23] V. A. Saraswat, R. Jagadeesan, and V. Gupta. JCC: Integrating timed default concurrent constraint programming into Java. In *Proceedings of EPIA 2003*, vol. 2902 of *Lecture Notes in Computer Science*, pages 156–170, 2003.

[24] J. Skolnick and A. Kolinski. Reduced models of proteins and their applications. *Polymer*, 45:511–524, 2004.

[25] F. H. Stillinger and T. A. Weber. Nonlinear optimization simplified by hypersurface deformation. *J. Stat. Phys.*, (52):1492–1445, 1988.

[26] Univ. des Saarlandes, Sweedish Inst. of Computer Science, and Univ. Catholique de Louvain. The Mozart Programming System. www.mozart-oz.org.

[27] T. Veitshans, D. Klimov, and D. Thirumalai. Protein folding kinetics: timescales, pathways and energy landscapes in terms of sequence-dependent properties. *Folding & Design*, 2:1–22, 1996.