# Simulazione del processo di ripiegamento di una proteina utilizzando un sistema ad agenti
# Agent-based Protein Folding Simulation

Luca Bortolussi    Alessandro Dal Palù    Agostino Dovier    Federico Fogolari

## SOMMARIO/*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 each approach is based on a particular energy function. There is no common agreement on which is the most reliable energy function to be used.

In this paper we present a novel agent-based framework for ab-initio simulations. Each amino acid of an input protein is viewed as an independent agent that communicates with the others. The framework allows a modular representation of the problem and it is easily extensible for further refinements and for different energy functions. Simulations at this level of abstraction allow fast calculation, distributed on each agent. We provide an implementation using the Linda package of SICStus Prolog, to show the feasibility and the power of the method. The code is intrinsically concurrent and thus natural to be parallelized.

**Keywords:** Computational Biology, Agent-Based Technologies, Concurrent Constraint Programming.

## 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, say of length 500, even with the unrealistic simplifying assumption of two allowed positions for an amino acid given the positions of the previous, the number of possible conformations is $2^{500}$. 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., [9]). Nevertheless, in the last thirty years, the global optimization for the PSP problem has been tackled with different classes of methods: simulated annealing [15], genetic algorithms [13], smoothing methods [26], branch and bound [19], and constraints [2]. These methods are all *ab-initio methods*, namely, approaches that are not based on similarities to already known proteins. 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 for each simulated 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 [25].

In this paper we present a new high-level framework for ab-initio simulation using Agent-based technologies. Each amino acid in the protein is modelled as an independent agent, which reacts to modifications of spatial positions of other amino acids. Each process evolves in a Monte Carlo 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 Linda tuple space (see, e.g., [7]). The system is implemented using Concurrent Constraint Logic Programming [24] in SICStus Prolog [27].

The framework is modular, being independent from the protein spatial model and energy function. As preliminary test, we adopt a basic energy function description, based on a statistical analysis of known native configurations of some proteins. 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]. Moreover, to model properly local interactions in an off-lattice energy field, we also include a bond length term, a bend angle term and a torsion angle term, similar to [28]. Note that in [22] 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-tested reduced model, as, for instance, that presented in [11]. Being intrinsically concurrent, the framework is well-suited to be parallelized.

To test our framework, we simulate a poly-alanine sequence, which has an high tendency to fold into an $\alpha$-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 Section 2 we briefly discuss the main results related to the solution of the PSP problem. The problem is then formalized in Section 3. In Section 4 we present the Agent-based framework. In Section 5 we describe the energy model employed. In Section 6 we provide some details of the implementation and in Section 7 we show our preliminary results. In Section 8 we draw some conclusions.

## 2 Related Work

We refer to [25] for a detailed review. We focus here on *ab-initio* approaches only. All-atoms ab-initio simulations by means of molecular dynamics (e.g. [23, 5, 17]), 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 simulation through either rigid constraints (as done in [10]) 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 [18]. Thirumalai and coworkers [28] have designed a forcefield suited for representing a protein through its $C_\alpha$-chain, which includes bonds, bend, and torsion angle energy terms. Scheraga and coworkers [16] 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], 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

| | | | | | |
|---|---|---|---|---|---|
| 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**so**le**ucine | 13 |
| K | **Lys**ine | 15 | L | **Leu**cine | 13 |
| M | **Met**hionine | 11 | N | **As**paragi**n**e | 8 |
| P | **Pro**line | 8 | Q | **Gl**utami**n**e | 11 |
| R | **Arg**inine | 17 | S | **Ser**ine | 5 |
| T | **Thr**eonine | 9 | V | **Val**ine | 10 |
| W | **Tr**y**p**tophan | 18 | Y | **Tyr**osine | 15 |

Figure 1: Amino acids: abbreviations, full names, and number of atoms in the side chain

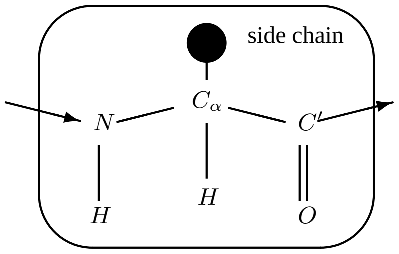

Figure 2: Amino acids: Overall structure

no common agreement on which energy function should model correctly the phenomenon.

Each amino acid is made by several atoms (cf. Fig. 2); there is a part *common* to all amino acids, the N-$C_\alpha$-$C'$ 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. 2. 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 [21].

A more abstract view of amino acids considers each of them as a single *sphere* centered in the $C_\alpha$ atom. The distance between two consecutive $C_\alpha$ 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 [18, 4].

Just as an example, we show how it is possible to formalize the protein folding problem as an optimization problem. Given a sequence $S = s_1 \cdots s_n$, with $s_i$ amino acids, a *folding* of $S$ is a function $\omega : \{1, \dots, n\} \longrightarrow \mathbb{R}^3$ 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 amino acids 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 $\omega$ of $S$ such that the following energy is minimized [8, 10]:

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

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

## 4 The Simulation Framework

In this section we describe the novel abstract framework of simulation, which 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 agent to each amino acid involved. These processes react to changes of position of other processes. We used Linda [7] as concurrent paradigm: all the communications between agents are performed through writing and reading logical atoms in the Linda tuple space.

### 4.1 The main program

We describe, using a Prolog pseudo-code, the structure of the program used. Let `S = [s1,...,sn]` be the list of amino acids, which is the input of our program.

```
simulation( S ) :-
    out(pos(1,initpos_1)), ...,
    out(pos(n,initpos_n)),
    out(trigger(1)), ...,
    out(trigger(n)),
    amino(1,S ) || ... || amino(n,S).

amino(i,S) :-
    in(trigger(i)),
    get_pos([pos(1,Pos_1),...,
             pos(n,Pos_n)]),
    update_pos(i,S,[pos(1,Pos_1),...,
                    pos(n,Pos_n)], Newpos),
    out(pos(i,Newpos)),
    out(trigger(1)),...,
    out(trigger(i-1)),
    out(trigger(i+1)),...,
    out(trigger(n)),
    amino(i,S).
```

The main procedure is the predicate `simulation`. Here we first put in the tuple space the initial positions for the different amino acids. Then we put the $n$ atoms `trigger(i)`, which are the switches that regulate agent's activation. Finally, we launch in parallel $n$ executions of the procedure `amino(i,a)`, one for each of the different indexes $i$ that identify the correspondent amino acids in the chain.

The core of the simulation is the predicate `amino(i,S)`, which governs the behaviour of each agent. The first instruction is a blocking `in`, which removes the switch `trigger(i)` from the tuple space. If there isn't such an atom, the `in` instruction suspends the execution of process $i$ and waits for some other agent to put this atom in the tuple space. When the execution continues, the process retrieves the most recent position of all other amino acids (`get_pos`), which is stored in the tuple space in terms of the kind `pos(i,position)`. Successively, the current position of each agent is updated by `update_pos` through a mechanism described in section 4.2, and this new position is put in the tuple space by the following `out` instruction. Finally, the switches of all other processes are turned on by the $i-1$ instructions `trigger(j)`, with $j \neq i$, and then the process recursively calls itself. Actually, in the real implementation, triggers are added only if they are not already present. The initial positions can be chosen between three different situations: straight line, zig-zag, and, for known proteins, the deposited structure can be used. The program is not terminating (the idea is that simulations run forever) and the sequences of computed positions are stored in an auxiliary file. Termination can be forced from the operating system level.

## 4.2 Simulating moves

The procedure `amino` computes a new position for the corresponding amino acid using a Monte Carlo-like simulation (cf., e.g., [8, 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 `update_pos` is invoked, the amino acid $a_i$ estimates its *current* potential $P_c$, 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 local minimum. The Monte Carlo technique is that of accepting the position $p'$ if a randomly generated value $\mathsf{r} \in [0,1]$ is less than a value depending on $P_n - P_c$:

$$\text{newPosition} = \begin{cases} p' & \textbf{if } P_n < P_c \textbf{ or } \mathsf{r} < e^{-\frac{P_n - P_c}{\mathsf{Temp}}} \\ p & \textbf{else} \end{cases}$$

Temp is a parameter simulating the temperature effects. Technically, it controls the acceptance ratio of moves that increase the energy. In Monte Carlo simulations Temp remains constant and it can be proven that allowing sufficient time the thermodynamic ensemble corresponding to the selected Temp is reached. Simulating annealing methods slowly decrease the value of Temp during simulation (observe that when Temp is close to 0, the test `rand` $< e^{-\frac{P_n - P_c}{\mathsf{Temp}}}$ is typically false, and thus only moves that decrease the energy are allowed). We stress the fact that Temp has no physical meaning in our framework, as the physical temperature is "incorporated" inside the parameters of the energy function. Thus Temp works only as a modifier of the acceptance ratio of bad–moves (which increase energy). A good strategy is to keep this ratio around 5%, as an higher value tends to perform a random walk in the configuration space. With the current energy function we have experimentally found that a value of 0.1 for Temp achieves this goal.

## 4.3 Moving Strategies

As said in Sect. 2, two consecutive amino acids tend to stay at a distance of $3.8$ Å from each other. We allow to temporarily break this constraint, forcing it by means of a soft constraint based on the bond energy term.

We test two different moving strategies. The first one is normally used in Monte Carlo simulation. Every new point is selected inside a cube that is centered in the previous amino acid position, according to an uniform distribution of probability. The cube side is 0.1 Å long. We call this one the *uniform distribution strategy*.

The second strategy, the *toroidal distribution strategy*, is as follows. Let us consider the case of an internal aminoacid $a_i$ in the sequence. If the distance between

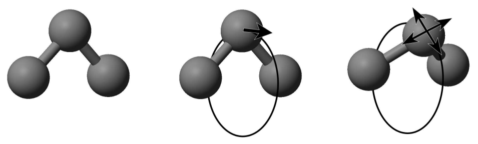

Figure 3: Moving strategy: the 3 amino acids, moving along the circumference and leaving the circumference.

$a_i$ 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)—see Fig. 3.

The case of the first and the last amino acids is slightly different, as they have two degrees of freedom, namely they are free to move on the surface of the sphere centered in their only neighbor; the consequent modification of the probability distribution is straightforward.

## 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. [28]). 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:

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

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

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

where $r(s_i, s_{i+1})$ is the distance between the $C_\alpha$ 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 $C_\alpha$s. The distribution is rather constant for every protein in the PDB and independent from the types of amino acids involved. The profile (see [12]) 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:

$$E_a(\vec{s}) \;=\; \sum_{i=1}^{n-2} -\log\left(a_1\, e^{-\beta_{i,1}^2} + a_2\, e^{-\beta_{i,2}^2}\right), \quad (3)$$

where $\beta_{i,j} = \left(\frac{\beta_i - \beta_j}{\sigma_j}\right)$ for $j = 1, 2$.

The torsion angle energy function is modelled using statistical information of torsional behavior extracted from the PDB. In detail, four $C_\alpha$ 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% identity, 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:

$$E_t(\vec{s}) \;=\; \sum_{i=1}^{n-3} -\log\left(a_1\, e^{\Phi_{i,1}} + a_2\, e^{\Phi_{i,2}}\right) \qquad (4)$$

where $\Phi_{i,j} = \frac{(\Phi_i - \phi_j)^2}{(\sigma_j + \sigma_0)^2}$ for $j = 1, 2$, and the parameters $a_1$, $a_2$, $\sigma_1$, $\sigma_2$, $\phi_1$, $\phi_2$ depend on the classes of the four residues, while $\sigma_0$ 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 [22]. 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

$$E_c(\vec{s}) \;=\; \sum_{i=1}^{n-3} \sum_{j=i+3}^{n} \left[|\varepsilon_{i,j}| x_{i,j}^{12} + \varepsilon_{i,j} x_{i,j}^{6}\right] \qquad (5)$$

where $x_{i,j} = \frac{r_0(s_i,s_j)}{r(s_i,s_j)}$, $\varepsilon_{i,j} = \varepsilon(s_i, s_j)$, $r(s_i, s_j)$ is the distance between the $C_\alpha$ 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 [12].

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 SICStus implementation

In this section we describe some details of the implementation of the simulation technique in the language SICStus Prolog [27]. We have interfaced SICStus with C++ where energy functions are computed. In particular, the whole mechanism for updating the positions (i.e. the `update_pos` predicate—Sec. 4) is implemented in C++ and dynamically linked into Prolog code. This guarantees a more efficient handling of the considerable amount of operations needed to calculate the potentials. The Prolog code is not very different from its abstract version presented in section 4.1. The only add-on is a control performed by the processes before putting the atom `trigger(i)` in the tuple space: they insert it only if it is not already present. In addition, to save memory, every process removes from the tuple space its previous position before writing its new one.

Prolog Code's length is less than 150 lines, while C++ code is based on a previous implementation of the energy function, used for setting and testing parameters. Codes can be found in `http://www.dimi.uniud.it/dovier/PF`. We have also written a C++ manager which launches SICStus Linda processes, visualizes the protein during the folding, and in general interacts with the Operative System.

## 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. This helix is folded quite rapidly, i.e. in less than 5000 Monte Carlo steps. Moreover, the folding pathway proceeds by first forming the correct torsional angles, and then compactifying the structure.

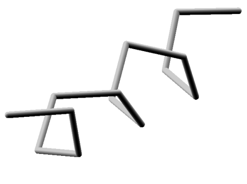

Figure 4: The simulated sequence of Alanines

We compared the toroidal distribution strategy with the uniform distribution strategy. It seems that, if we use the toroidal one, we are easily trapped in a local minimum, with no chances to escape. This probably happens due to the limited choice of moves, which forbids going through escaping trajectories. The uniform one, instead, does not suffer from these limitations: in this case we can usually see the formation of an helix from a sequence of polyalanines in very short time.

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 amino acids 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.

Finally, we tested how many processes our framework in SICStus LINDA is able to deal with. It turned out that the communication of every client process with the server, as one can expect, becomes very slow as the number of processes increases. We run a test for a protein with 54 amino acids. This chain can still be handled, but every process performs about 2 moves per second, showing the crucial need for a faster implementation.

## 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. 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 implement a stable version that implements the framework presented in this paper. In particular we want to include a more refined force field. Moreover we want to reduce the computational load: this can be accomplished by approximating and/or excluding some non relevant contributions to the energy function. The SICStus implementation of Linda protocol revealed that each communication takes about 10mS. This delay is the current bottleneck of the simulation. We would like to develop an alternative implementation that exploits the benefits of multiprocessors with shared memory architecture.

We wish 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 concept of *cooperative approach*, in which a cooperation strategy between processes is induced dynamically by the current configuration, following the approach of [20]. 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. A careful comparison with [14] and related works seems also promising for this topic.

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 [10], 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.

## 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] N. Carriero and D. Gelernter. Linda in Context. *Communications of the ACM* 32(4):444–458, 1989.

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

[9] 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.

[10] A. Dal Palù, A. Dovier, and F. Fogolari. Constraint Logic Programming approach to protein structure prediction. *BMC Bioinformatics* 5(186), November 2004.

[11] G. M. S. De Mori, C. Micheletti, and G. Colombo. All-atom folding simulations of the villin headpiece from stochastically selected coarse-grained structures. *Journal Of Physical Chemistry B* 108(33):12267–12270, 2004.

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

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

[14] S.A. Kauffman and W. Macready. Technological Evolution and Adaptive Organizations. Complexity, 26(2):26–43, 1995.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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, 1989.

[20] M. Mamei, F. Zambonelli, L. Leonardi. A Physically Grounded Approach to Coordinate Movements in a Team. Proceedings of *ICDCS*, 2002.

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

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

[23] 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.

[24] V. A. Saraswat, M. C. Rinard, and P. Panangaden. Semantic Foundations of Concurrent Constraint Programming. Proceedings of *18th ACM POPL*, pp. 333–352, 1991.

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

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

[27] Swedish Institute for Computer Science. *SICStus Prolog Home Page*. `http://www.sics.se/sicstus/`

[28] 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.