Agent-based Protein Structure Prediction

Luca Bortolussi Agostino Dovier

Dept. of Mathematics and Computer Science, University of Udine

`bortolussi|dovier@dimi.uniud.it`

Federico Fogolari

Dept. of Biomedical Science and Technology, University of Udine

`ffogolari@makek.dstb.uniud.it`

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. Although it is accepted that the native state minimizes the free energy of the protein, all current mathematical models of the problem are affected by intrinsic computational limits, and moreover there is no common agreement on which is the most reliable energy function to be used.

In this paper we present an agent-based framework for ab-initio simulations, composed by different levels of agents. Each amino acid of an input protein is viewed as an independent agent that communicates with the others. Then we have also strategic agents and cooperative ones. 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 have written a multi-thread implementation, and tested the feasibility of the engine with two energy functions.

1 Introduction

The Protein Structure Prediction Problem (PSP), fundamental for biological and pharmaceutical research, 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.

Amino acids are molecules composed of a number of atoms ranging from 7 to 24. The process for reaching the native state is known as the protein *folding*. Currently, the native conformations of more than 30000 proteins are available in the Protein Data Bank (PDB) [4].

In this work we concentrate on *ab-initio* modelling. These methods are based on the *Anfinsen thermodynamic hypothesis* [2]: the (*native*) conformation adopted by a protein is the one with minimum free energy, i.e. the most stable state. 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. All-atom computer simulations are typically unpractical, because they are extremely expensive. In fact, each simulated nanosecond for a small protein requires 2 to 4 CPU days on a supercomputer at Berkeley Lab. To overcome this limit, amino acids are modeled in a coarser way, usually as a single sphere. These simplified models of proteins are attractive in many respects, as they have smoother energy hyper-surfaces, and the dynamics is faster. 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 [32, 31].

In this paper we present a new high-level framework for ab-initio simulation using Agent-based technologies, which extends the one presented in [6]. It is developed by following the architecture for agent-based optimization systems presented by Milano and Roli in [23]. This framework stratifies the agents in different levels, according to their knowledge and their power. Here we have three layers: one containing agents designed to explore the state space, one dealing with agents implementing global strategies and the last one containing cooperation agents.

Each amino acid in the protein is modeled as an independent agent, which has the task of exploring the configuration space. This is accomplished mainly by letting these agents interact and exchange information. These processes operate within a simulated annealing scheme, and their moves are guided by the knowledge of the position of surrounding objects. The communication network changes dynamically during the simulation, as agents interact more often with their spatial neighbours. The strategic agents govern the environmental properties and they also coordinate the basic agents activity in order to obtain a more effective exploration strategy of the state space. The cooperative agent, instead, exploits some external knowledge, related to local configurations attainable by a protein, to improve the folding process. The communication between agents is based on Linda tuple space (see, e.g., [11]), and the program is implemented in SICStus Prolog [14]. Moreover, we have also implemented an equivalent multithread version, written in C, which is much faster (cf. Section 6).

Here we test the system using two different energy models. The first one has been developed by Micheletti in [13], and used there to generate coarse grained structures to be then processed by molecular dynamics. The second has been used in a previous version of this work [6], as a first benchmark, but never tested in detail. However, other energy models can be used as well, as the framework is independent from it.

The program produces quite stable outcomes, in the sense that the solutions found in different runs have similar energies. Unfortunately, the potentials used are still too coarse to produce good predictions in terms of RMSD from known native structures. Moreover, the introduction of the strategic level improves very much the quality of the solutions (in terms of energetic values), while the cooperative agent decreases the RMSD, but increases the energy. This shows the need of using a potential with a higher resolution.

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 5 we present the Agent-based framework. In Section 4 we describe the energy models employed. In Section 6 we provide some details of the implementation and in Section 7 we show the results. Finally, in Section 8 we draw some conclusions.

2 Related Work

The field of *ab-initio* protein structure prediction is an highly active area of research, and we refer to [32] for a detailed review.

All-atoms ab-initio simulations by means of molecular dynamics (e.g. [28, 10, 20]) are not feasible, due to the high intrinsic complexity of the needed operations. In particular, two to four CPU-days of a supercomputer at Berkeley Labs are needed to simulate a single nanosecond in the evolution of a medium length protein, while the typical folding time is of the order of milliseconds or seconds.

More efficient methods are offered by simplified models, where each amino acid is described in a coarser way, for instance, as a single center of interaction. The corresponding energy models are built around by taking into account the local propensity to adopt well-defined secondary structures, as well as intramolecular interactions. The secondary structure propensity is usually encoded in energy terms derived from statistical database analysis, depending on the type of amino acids involved. Another possibility is that of imposing rigid constraints (as done in [12]) directly in the simulation. Interactions between amino acids, on the other end, 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, a table which has been proven to be rather accurate, when tested on several decoys' sets, is derived in [5]. Similar tables have been provided based on different criteria by other authors [24]. Thirumalai [33] has designed a forcefield suited for representing a protein through its C_α-chain, which includes bonds, bend, and torsion angle energy terms. Scheraga [19] has used a similar model including side-chain centroids. For an up-to-date discussion on reduced potentials, see [31]. Since the focus of this paper is the development of a computational approach to PSP based on MAS, we will not go in detail into most recent results. It is worth however noticing that the world-wide prediction CASP experiment [25] has seen a steadily increase in the performance of many prediction algorithms and procedures. Among most successful ab-initio predictors (i.e. predictors which do not start from homology with a structurally characterized sequence), the programs Rosetta [8], Tasser [34], and Fragfold [16] have been performing very well in the recent CASP rounds. The accuracy of the methods used in the group of Baker enabled the design (and later experimental verification) of a novel protein fold [18].

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 [3] for proteins of length up to 160. A constraint-based solution to the general problem (with the 20 amino acids) is proposed instead in [12], 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 [14].

At present time, we are not aware of any other approach modeling amino acids as concurrent processes. We believe therefore that it is worth to explore further this modeling approach. We wish to remark, however, that the simple energetic models used impair the predictive power of the procedures described here, especially when compared with the much more established methods mentioned above (see for a recent perspective [30]).

3 Proteins and the PSP Problem

A protein is made by amino acids, present in nature in 20 different types that can be identified by a letter in the set $\mathcal{A} = \{A, \ldots, Z\} \setminus \{B, J, O, U, X, Z\}$. The *primary structure* of a protein is its amino acid sequence $s_1 \cdots s_n$, where $s_i \in \mathcal{A}$. Each protein assumes a particular 3D conformation, called native conformation or *tertiary structure*, determined uniquely by its primary structure. It is precisely this specific spatial structure that determines the function of a protein. Thermodynamical consideration justify the hypothesis that the native structure is the 3D conformation that minimizes the global free energy [2]. A well-defined energy function should consider all possible interactions between all atoms of every amino acid composing the protein, though there is no common agreement on which energy function should model correctly the phenomenon. A review of various forces and potentials can be found in [26]. The *protein structure prediction (PSP) problem* is the problem of predicting the tertiary structure of a protein given its primary structure.

The general structure of an amino acid is reported in Figure 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 by a peptide bond, represented in the diagram by incoming and outgoing arrows.

A more abstract view of amino acids considers each of them as a single *sphere* centered in the C_α atom (see Figure 8). It is reasonable to assume the distance between two consecutive C_α atoms to be 3.8 Å, measure chosen as unitary. At this level of abstraction, different potentials can be found in literature [19], usually composed by statistical terms derived from analysis of known native structures from PDB. In the following section, we present in more detail two energy functions we used for testing the system.

4 Energy function

The problem of identifying an accurate energy for a simplified representation of the amino acids is considered very difficult, and there is no general accordance relatively to which one reflects better the physical reality. Consequently, in literature there are many different energy functions one can choose from [31, 32]. All these models share a common feature: the more accurate are the results, the more complex are the calculations involved.

As the focus of this paper is to present the agent-based framework, the two energy models that we used will be presented briefly hereafter. The interested reader can find more details in the referenced papers. Both energy models represents amino acids as single spheres centered in the C_α-atoms; at this level of description the local geometry is described by bend angles (the angles formed between two bonds linking three consecutive carbon atoms, see Figure 3) and torsion angles (the angles between the planes spanned by the first three and the last three C_α-atoms in a run of four consecutive ones, see Figure 4).

4.1 First Energy Model

The first energy model has been developed by Micheletti in [22]. The energy comprises three terms, devoted to interaction, cooperation and chirality. In addition, there are some penalty terms that penalize non-physical configurations by giving them a high energy.

If we indicate with $\mathbf{x}$ the spatial disposition of the amino acid's chain and with $\mathbf{t}$ their type, the energy can be expressed as

$$\begin{aligned} E(\mathbf{x},\mathbf{t}) = \quad & E_{coop}(\mathbf{x},\mathbf{t}) + E_{pairwise}(\mathbf{x},\mathbf{t}) + \\ & E_{chiral}(\mathbf{x},\mathbf{t}) + E_{constr}(\mathbf{x}) \end{aligned} \tag{1}$$

The *pairwise term* ($E_{pairwise}$) captures the interactions that occur between two amino acids that are close enough, and it is defined using a contact interaction matrix developed by Kolinsky in [17]. The *cooperative term* (E_{coop}) involves four different amino acids and it tries to improve the packing of secondary motifs, favoring the formation of close hydrogen-bonds. The *chiral term* (E_{chiral}), instead, is used to favor the formation of helices for some putative segments, identified using knowledge extracted from PDB database. Finally, there are three kinds of *penalty terms*: one imposing steric constraints on the position of C_α and C_β atoms,[1] one forbidding unrealistic torsional angles and one keeping fixed the distance between two consecutive C_α atoms.

[1]The position of C_β is calculated from the chain of alpha carbon atoms using the Park and Levitt rule (cf. [27]).

4.2 Second Energy Model

The second energy model has been developed in [6] to perform the tests on the first version of the simulation engine. It is composed by four separate terms, related to *bond distance* (E_b), *bend angle* (E_a), *torsional angle* (E_t), and *contact interaction* (E_c):

$$E(\mathbf{x},\mathbf{t}) = E_b(\mathbf{x},\mathbf{t}) + E_a(\mathbf{x},\mathbf{t}) + E_t(\mathbf{x},\mathbf{t}) + E_c(\mathbf{x},\mathbf{t}) \qquad (2)$$

The *bond distance term* (E_b) is designed to keep fixed to 3.8Å the distance of two consecutive C_α atoms. The *bend angle term* (E_a) and the *torsional angle term* (E_t), instead, are statistical potentials trying to induce a good local geometry in the folded chain, e.g. favoring the formation of secondary structure elements. Finally, the *contact interaction term* captures long range interactions using the statistical contact matrix developed in [5].

5 The Simulation Framework

In this section we describe the abstract framework of the simulation, following the line of [7]. This scheme is independent both on the spatial model of the protein and on the energy model employed. Therefore, it can be instantiated using different representations. Here we test the system with the two different energy functions of Section 4.

Each amino acid is associated to an independent agent, which moves in the space and communicates with others in order to minimize the energy function. Moreover, we also introduce other agents at different hierarchical levels, which have the objective of coordinating and improving the overall performance of the system.

Milano and Roli in [23] have devised a general scheme to encode agent–based minimization, and our framework can be seen as an instantiation of that model. In particular, they identify four levels of agents, which interact in order to perform the optimization task. Level 0 deals with generation of an initial solution, level 1 is focused on the stochastic search in the state space, level 3 performs global strategic tasks and level 4 is concerned with cooperation strategies.

According to this scheme, here we have agents of level 1, 2 and 3 (level 0 ones are trivial). We deal separately with them in next subsections.

We present the basic function of those agents using Linda [11] as concurrent paradigm. In this language, all agents interact and communicate through writing and reading logical atoms in the Linda tuple space.

5.1 Level 1 – searching agents

We associate to each amino acid an agent, which has the capability to communicate its current position to other processes, and to move in the search space, guided by its knowledge of the position of other agents. The general behaviour of these agents can be easily described declaratively by the predicate `amino(i,S)`:

```
amino(i,S) :-
    read(authorized(i)),
    in(trigger(i)),
    in(resting(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)),
    out(resting(i)),
    amino(i,S).
```

The first instruction is a blocking `read`, which tests if the term `authorized(i)` is present in the tuple space. This term is used to coordinate the activity with higher level agents: these processes can remove it and thus blocking the activity of the amino agents. Also the term `resting(i)` is used in this coordination task, and it allows an amino agent to tell other processes if he is moving or not. In fact, it is removed from the tuple space at the beginning of a computation, and it is added again at the end.

The second instruction is a blocking `in`, which removes the switch `trigger(i)` from the tuple space. This is a mechanism used to guarantee (a week form of) fairness to the system: each agent must wait for the movement of another process before performing its own move. In this way we avoid that a single agent takes the system resources all for itself. Clearly, at the beginning of the simulation the switches for all the amino processes are turned on, in order to let them move.

When the guards are satisfied, 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 5.1.1, 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 position of these agents is expressed in cartesian coordinates. This choice implies that the moves performed by these processes are local, i.e. they do not affect the position of other amino acids. This is in syntony with the locality of the potential effects: the modification of the position of an amino acid influence only the nearby ones.

The initial configuration of the chain can be chosen between three different possibilities: straight line, random, and the deposited structure for known proteins.

5.1.1 Simulating moves

The amino acids move according to a simulated annealing scheme. This algorithm, which is inspired by analogy to the physical process of slowly cooling a melted metal to crystallize it (cf. [1]), uses a Montecarlo-like criterion to explore the space. Each time the procedure `update_pos` is invoked, the amino acid a_i computes a new position p' in a suitable neighborhood (see next Subsection), and then compares its *current* potential P_c with the *new* potential P_n, corresponding to p'. If $P_n < P_c$, the amino acid updates its position to p', otherwise it accepts the move with probability $e^{-\frac{P_n - P_c}{\mathsf{Temp}}}$. This hill-climbing strategy is performed to escape from a local minimum.

Temp is a parameter simulating the temperature effects. Technically, it controls the acceptance ratio of moves that increase the energy. In simulated annealing algorithms, it is initially high, and then it is slowly cooled to 0 (note that if $Temp$ is very low, the probability of accepting moves which increase the energy is practically 0). It can be shown that simulated annealing converges to the global optimum of the energy, if the temperature is lowered sufficiently slowly, i.e. in an exponential time (cf. [1]). We discuss the cooling schedule in Section 5.2.2.

5.1.2 Moving Strategy

In the energy model adopted (cf. Section 4), we have several constraints on the position of the amino acids, which are implemented via energy barriers. This means that we may perform moves in the space that violate these constraints, but they will be less and less probable as the temperature of the system decreases. We have chosen this "soft constrained" approach because the Montecarlo-like methods require an unbiased exploration of the search space, i.e. they require that the underlying Markov Chain model is irreducible and ergodic (cf. [1]). Now, because all the search space is virtually accessible, the moving strategy is very simple, and guarantees the previous two properties: we choose with uniform probability a point in a cube centered at the current position of the amino acid. The length of the side of the cube is set experimentally to 1 Å.

5.1.3 Communication Scheme

When a single agent selects a new position in the space, it calculates the variation of energy corresponding to this spatial shift. However, in almost all the reduced models, the energy relative to a single amino acid depends only from the adjacent amino acids in the polymer chain, and from other amino acids that are close enough to trigger the contact interactions. Thus, if an agent is very far from the current one, it won't bring any contribution to the energy evaluation, at least as long as it remains distant.

This observation suggests a strategy that reduces considerably the communication overhead. Agent i first identifies its neighbors $\mathcal{N}_i$, i.e. all the amino acids in the chain that are at distance less than a certain threshold, fixed here to 14 Å(more or less four times the distance of two consecutive amino acids). Then, for an user-defined number of moves M, it communicates just with the agents in $\mathcal{N}_i$, ignoring all the others. When the specified number M of interactions is reached,amino acid i retrieves the position of all the amino acids and then refresh its neighbour's list $\mathcal{N}_i$.

In our LINDA framework, this means that the process i will turn on the switches `trigger(j)` just for the agents $j \in \mathcal{N}_i$. Moreover, it will read the current positions only of those amino acids, before performing the move.

The refresh frequency must not be too low, otherwise a far amino acid could, in principle, come very close and even collide with the current one, without any awareness of what is happening. We experimentally set it to 100.

This new communication strategy impose a redefinition of the agent behaviour, which now has to update also the neighbor list, thus leading to the code shown below.

```
amino(i,S,Curr,Neigh_List) :-
    read(authorized(i)),
    in(trigger(i)),
    in(resting(i)),
    update_neigh(Curr,Neigh_List,New_NList)
    get_pos(New_NList,Pos_list),
    update_pos(i,S,New_NList,Pos_List,Newpos),
    out(pos(i,Newpos)),
    signal_move(New_NList),
    C1 is Curr + 1,
    out(resting(i)),
    amino(i,S,C,New_NList).
```

It is quite similar to the one presented in section 5.1, with few obvious modifications in `get_pos` and `update_pos` clauses, and with the presence of a new function for updating the neighbour's list, i.e. `update_neigh`. This predicate performs the update only if $Curr \bmod \mathrm{M} \equiv 0$, otherwise copies `Neigh_List` into `New_NList`.

5.2 Level 2 – strategy

The first layer of agents is designed to explore the search space, using a simulated annealing strategy. However, the neighborhood explored by each agent is small and simple, while the energy landscape is very complex. This means that the simulation is unwilling to produce good solutions in acceptable time periods. This fact points clearly to the need of a more coordinated and efficient search of the solution space, which can be achieved by a global coordination of the agents. This task can be performed by a higher level agent, which has a global knowledge of the current configuration, and it is able to control the activity of the single agents. Details are provided in next subsection.

At the same time, the simulated annealing scheme is based on the gradual lowering of the temperature, which is not a property of the amino agents, but it is rather a feature of the environment where they are endowed. This means that the cooling strategy for temperature must be governed by a higher level agent, which is presented in subsection 5.2.2.

5.2.1 Enhanced exploration of state space

As said in the previous paragraphs, the way the solution space is searched influences very much the performance at finite of stochastic optimization algorithms. Our choice of using an agent based optimization scheme results in an algorithm where subset of variables of the system are updated independently by different processes. However, the neighborhood of each agent is quite restricted, to avoid problems arising from delayed communication (cf. Section 8 for further comments). This choice implies that the algorithm may take a very long time to find a good solution, and consequently the temperature must be cooled very slowly.

To improve the efficiency, we designed a higher level agent, called the "*orchestra director*", which essentially moves the amino acids in the state space according to a different, global strategy. It can move the chain using two different kind of moves: crankshaft and pivot (cf. Figure 6). Crankshaft moves essentially fix two points in the chain (usually at distance 3), and rotate the inner amino acids along the axis identified by these two extremes by a randomly chosen angle. Pivot moves, instead, select a point in the chain (the pivot), and rotate a branch of the chain around this hub, again by a random angle. These global moves keep fixed the distance between two consecutive C_α carbon atoms, and are able to overcome the energy barriers introduced by the distance penalty term.

These are essentially the two moves executed by our director, which is described by the following code:

```
director(S,N) :-
    read(orchestra_authorized),
    in(orchestra_resting),
    get_position(Pos_List),
    move_chain(Pos_List,New_Pos_List,S,N),
    put_position(New_Pos_List),
    out(orchestra_resting),
    director(S,N).
```

The first two lines implement a synchronization mechanism similar to the one used for amino agents. The agents waits for the ground atom `orchestra_authorized` to be in the tuple space, then it start moving and it signals its movement by removing the predicate `orchestra_resting`. Once the director has got the position of the amino acids (`get_position`), it performs some moves using the previously described strategies, calling the `move_chain` clause. This predicate calls itself recursively a predefined number `N` of times (depending on the protein's length), and each time it selects what move to perform (i.e. crankshaft or pivot), the pivot points and an angle. Then it computes the energy associated to the old and the new configuration, and applies a Montecarlo criterion to accept the move (cf. Section 5.1.1). Finally, the end of the movement is signaled by putting back the atom `orchestra_resting`.

In the real implementation, we decided to activate this agent only if the temperature of the system is high enough, that is to say, only when this moves guarantee an easy overcome of energy barriers, hence an effective exploration of the search space. Thus we have an additional condition at its beginning, like `Temperature > Threshold`.

5.2.2 Environment

The environmental variables of the simulation are managed by a dedicated agent. In this case, the environment simply controls the temperature, which is a feature of the simulated annealing algorithm. This temperature must not be conceived as a physical quantity, but rather as a control value which governs the acceptance ratio in the choice of moves increasing the energy.

From the theory of simulated annealing (cf. [1]), we know that the way the temperature is lowered is crucial for the performance of the algorithm. In fact, this can be seen as a sequence of Markov chain processes, every one with its own stationary distribution. These distributions converge in the limit to a distribution which assigns probability one to the points of global minimum for the energy. For this to happen, however, one must lower the temperature logarithmically towards zero (giving rise to an exponential algorithm). Anyway, to reach good approximations, one has just to let the simulation perform a sufficient number of steps at each value of the control parameter, in order to stabilize the corresponding Markovian process.

We applied a simple and very used strategy: at each step the temperature is decreased according to $T_{k+1} = \alpha T_k$, where $\alpha = 0.98$. The starting temperature T_0 must allow a high acceptance ratio, usually around 60%, and we experimentally set T_0 as to attain this acceptance ratio at the beginning of the simulation. This value guarantees also that the system does not accept moves with too high energy penalties (cf. Section 7). Regarding the number of iterations at each value of T, we set it in such a way that the average number of moves per amino acid is around 200, in order to change the value to all the variables a suitable number of times. The code for the environment agent is straightforward.

5.3 Level 3 — cooperation

In this section we present a dynamic cooperation strategy between agents, which is designed to improve the folding process and try to reach sooner the configuration of minimum energy. The main idea behind is to combine concurrency and some external knowledge to force the agents to assume a particular configuration, which is supposed to be favorable. More specifically, this additional information can be extracted from a database, from statistical observations or from external tools, such as secondary structure predictors.

The cooperation is governed by a high level agent, which has access to the whole status of the simulation and to some suitable external knowledge. The basic additional information we use to induce cooperation is related to the secondary structure. In particular, we try to favor from the beginning the formation of local patterns supposed to appear in the protein (see below).

To coordinate the action of single agents and let a particular configuration emerge from their interaction, we adopt a strategy which is very similar in spirit to the "computational fields" technique introduced by Mamei in [21]. The idea is to create a virtual force field that can drive the movement of the single agents towards the desired configuration. In our setting we deal with energy, not with forces, so we find more convenient to introduce a biasing term modifying the potential energy calculated by a single agent. In this way, we can impose a particular configuration by giving an energy penalty to distant ones (in terms of RMSD).

In addition, the cooperation agent controls the activity of all lower level agents, in particular, it decides the scheduling between the amino agents and the orchestra director. Moreover, it controls also the termination of the simulation. The stopping condition is simple: it is triggered when the system reaches a frozen configuration at zero temperature[2], meaning that a (local) minimum has been reached and no further improvement is possible. The declarative code of the agent is the following:

```
cooperator(Moves,Old_pos_list, Sec_info) :-
    get_position(Pos_List),
    secondary_cooperation(Pos_List, Sec_info),
    check_termination(Pos_list,Old_pos_list),
    out(authorized(1)),...,out(authorized(n)),
    wait_for_amino_moves(Moves),
    in(authorized(1)),...,in(authorized(n)),
    read(resting(1)),...,read(resting(n),
    out(orchestra_authorized),
    in(orchestra_authorized),
    read(orchestra_resting),
    cooperator(Moves,Pos_list, Sec_info).
```

The first actions of the agent are reading the current configuration of the protein (`get_position(Pos_List)`), activating the secondary structure cooperation mechanism (`secondary_cooperation(Pos_List, Sec_info)`) and checking the termination conditions (`check_termination`). In particular, the activation

[2]When the temperature falls below a predefined threshold, it is quenched to zero.

of the cooperation mechanism is obtained by posting some atoms that trigger the corresponding energy term in the computations performed by lower level agents. The following predicates are devoted to the synchronization between the amino agents and the orchestra director. The amino agents are authorized to move by putting in the tuple space the predicate `authorized(i)`. When the cooperator decides that amino agents should stop, it removes such predicate and waits for the agents to stop moving (by a blocking read on `resting(i)`). Things work similarly for the orchestra director. `wait_for_amino_moves(Moves)` waits for the amino acids to perform a certain amount of moves specified by `Moves`.

In the previous code, we showed a strategy that alternates between amino agents and orchestra director, by a mutually exclusive activation. Another possible strategy is to run both amino agents and the orchestra director simultaneously, though this may introduce more errors in the simulated annealing scheme due to higher asynchrony. A comparison of the two strategies can be found in Section 7.

5.3.1 Cooperation via Secondary Structure

In the literature it is recognized that the formation of local patterns, like α-helices and β-sheets, is one of the most important aspects of the folding process (cf. [26]). Actually the combined usage of alignment profiles, statistical machine learning algorithms and consensus methods has resulted in location of these local structures with a three state accuracy close to 80% (see for a discussion [29] and references cited therein). We plan to use the information extracted from them to enhance the simulation. For the moment, however, we introduced a preliminary version of cooperation via secondary structure, which identifies the location of secondary structure directly from pdb files (when using information coming from secondary structure predictors, we should include in the cofields also the accuracy level of the prediction).

Once the cooperation agent possesses this knowledge, it activates a computational field that forces amino acids to adopt the corresponding local structure. The mathematical form of this new potential penalizes all configurations having a high RMSD from a “typical” helix or sheet.[3] Of course, this energy regards only the amino acids supposed to form a secondary structure, and it is activated from the beginning of the simulation, as it should be able to drive the folding process, at least locally.

[3]There are different kinds of α-helices and β-sheets, but we omit here further details.

6 Implementation

In this section we describe some details of the implementation of the simulation technique in the language SICStus Prolog [14]. 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, cf. Sec. 5) 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 5.1, and its length is less than 150 lines. 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.

Actually, LINDA communication is not very efficient: each communicative act takes about 100 milliseconds, thus leading to a very slow simulation. Therefore, we have also written a multithreading version in pure **C**, which can run both under Windows and Linux. This version reproduces the communication mechanisms of LINDA using the shared memory, so it is equivalent to the program presented in the paper, though much more efficient. All the codes can be found in `http://www.dimi.uniud.it/dovier/PF`.

7 Experimental results

In this section we present the results of some tests of our program. We are mainly interested in two different aspects: seeing if and how the novel features of the framework (parallelism, strategy, cooperation) improve the simulation and checking how good are the predicted structures with respect to the resolution of the energy functions used. Note, however, that these potentials are structurally very simple, so we are not expecting outstanding results out of them.

We ran the simulation on different proteins of quite small size, taken from PDB [4]. This choice is forced by the low resolution of the potential. We also had to tune a lot of parameters of the program, especially the cooling schedule for the temperature, the weights of the penalty terms and the scheduling of the strategic and the cooperative agents.

All the tests were performed on a bi-processor machine, mounting two Opteron dual core CPU at 2 GHz. Therefore, we have a small degree of effective parallelism, which can give a hint on the parallel performances of the system.

In the following, we comment separately on the agent-based framework and on the energy models.

7.1 Analysis of the Agent-Based Framework

In order to analyze the different enhancements introduced in the system, we performed several tests on a single protein, 1VII, which is composed of 36 amino acids. In the following we focus our attention on the orchestra director, on the cooperation features and on the effects of parallelism on the underlying simulated annealing engine.

Exploration Strategy To evaluate the enhancements introduced by the orchestra director, we ran several simulation with and without it, comparing the results, in terms of energetic values and RMSD from the native state of the solutions found. As expected, the amino acid agents alone are not able to explore exhaustively the state space, and the minimum values found in this case are very poor (see Table 1). This depends essentially from the fact that most of their moves violate the distance constraint, especially at high temperatures. At low temperatures, instead, the system seems driven more by the task of minimizing this penalty term, than by the optimization of the “real” components of energy. Therefore, the simulation gets stuck very easily in bad local minima, and reaches good solutions just by chance. If the “orchestra director” agent is active, instead, much better energy minima are obtained (see again Table 1). Note that combining the amino acid agents and the strategic one corresponds to having a mixed strategy for exploring the state space, where two different neighborhoods are used: the first one, local and compact, is searched by the amino acid agents, while the second one, which links configurations quite far away, is searched by the strategic agent.

We observe also that the poor values of the energy and of RMSD for the simulation with amino agents alone depend from the fact that this exploration scheme is not able to compact and close the structure by itself. In fact, the chain remains open, and we can see only some local structure emerging. This is evident in Figure 7.

Effects of Parallelism It is well-known that asynchronous parallel forms of simulated annealing can suffer from a deterioration of results with respect to sequential versions (cf. [15]), due to the use of outdated information in the calculation of the potential. Therefore, we ran some sequential simulations, using essentially the moves performed by the orchestra director, that is crankshaft and pivot. The energy values of solutions obtained with the sequential simulation are, on average, a little bit higher than those of the multi-agent simulation (see Table 2). Hence, the introduction of parallelism not only does not worsen the quality of solutions, but it also slightly improves it. This may depend on the fact that amino agents are able to locally intensify the exploration of the search space. In addition, we compared the execution time of the parallel and the sequential simulations, for the same total number of moves per amino acid. As we can see, multi-agent simulation is three times quicker using 4 processors, showing therefore an almost optimal parallel speed-up.

Cooperation To test the cooperative field, we compared separately runs with and without cooperation via secondary structure. It comes out that the quality of the solutions in terms of energy are generally worse with the cooperative agent active (cf. Tables 4 and 5, and 6 and 7). On the other hand, the RMSD is improved in most of the cases (cf. again Tables 4 and 5, and 6 and 7). This is quite remarkable, as the information relative to secondary structure is still local. We are also pondering the introduction of a better form of cooperation, with some capability of driving globally the folding process, cf. Section 8 for further comments.

7.2 Energy Comparison

In this section we compare the performances of the two energy models presented in Section 4. The quantities used to estimate the goodness of the results are the value of the energy and the root mean square deviation (RMSD) from the known native structure.

In the following, we presents the results of tests ran on a set of 9 proteins, extracted from the Protein Data Bank. The complete list of tested proteins, together with their length and the type of secondary structure present, is shown in Table 3. Therefore, knowing their native structure, we are able to estimate the distance of the predicted structure from the real one, thus assessing the quality of the potentials. In the tables following, we present the simulation results with and without cooperation.

First Energy Model In Table 4, we show the best results obtained in terms of RMSD and energy without cooperation, while in Table 5 cooperation was active. For a safe comparison, the energy value does not take into account the contributions of cooperative computational fields, present in experiments of Table 5 only. The numbers shown are an average of 10 runs; standard deviation is shown in brackets.

From Table 4, we can see that, without cooperation, the simulation is quite stable: most of the runs produce solutions with energy varying in a very small range of values. On the contrary, the RMSD is quite high. This depends mostly on the low resolution of the potential. In fact, this energy function has terms which compact the chain, but no term imposing a good local structure. Therefore, the simulation maximizes the number of contacts between amino acids, giving rise to a heavily non-physical shape. The situation, however, changes with the introduction of the cooperative effects. In this case, the maximization of contacts goes together with potential terms imposing good local shapes, creating better structures but worsening the energy (less contacts are formed). In particular, we can see a remarkable improvement of RMSD for proteins 1PG1 and 2GP8; a visual comparison of the outcome for 2GP8 is shown in Figure 8.

On the other hand, for several proteins, among which 1VII, cooperation does not bring any sensible improvement on the RMSD value. This probably depends from the fact that, in these cases, the constraints on the secondary structure are not so strong to force a good global shape. In particular, the predominant terms of the energy function try to maximize the number of favorable contacts, thus creating a structure where some areas are extremely compact and others are left open. If the local effect induced by secondary structure cofields is concentrated in the compressed area, then no essential improvement arises. This is a witness of the low resolution of the potential.

Second Energy model This energy model was never tested in all its potentiality before. Test in [6] were performed in a simulation with only amino agents, which are not able to overcome the energy barriers blocking the compactification of the chain.

In Tables 6 and 7, we can see the results for the set of testing proteins listed in Table 3. We can see that also the resolution of this potential is not very accurate. Generally, the values in terms of RMSD are slightly worse than for the first energy model. This happens despite the presence of stronger local terms, that should, at least in principle, generate better shapes. Specifically, those terms seem to enter in conflict with the contact energy, forcing a shape with good local properties but poor global structure. For instance, for 1ZDD without cooperation, the chain remains open, though some form of helices emerge, cf. Figure 10.

Secondary structure cooperation follow a pattern similar to the first energy model, improving the RMSD (and worsening the energy) for a slightly broader set of proteins. For example, 1ZDD gets closed with secondary structure cooperation, see Figure 10.

Comparing the first and the second energy model, we can observe that there is no real winner. The first potential has generally solutions of slightly better quality, but it is much heavier from a computational point of view (the simulation using the second potential is approximately 10 times faster, cf. Table 2). This increase in time depends from the fact that the first model needs continuously the computation positions of C_β atoms. Moreover, also the single terms in the energy are more expensive to calculate.

The previous discussion implies that we need a more refined potential, taking into account all the correlations involved and modeling not only the C_α atoms, but also a coarse-grained representation of the whole side chain.

8 Conclusions

In this paper we presented a multi-agent based framework to predict the tertiary structure of a protein, designed according to the MAGMA scheme [23]. This approach is independent from the energy model used, and can be easily adapted to more complex spatial representations and potential functions. We basically identify every amino acid with a concurrent agent, and we introduced also other agents, aimed at coordinating the activity of the basic processes and inducing some basic form of cooperation.

Our long term goal is to provide a powerful tool for folding proteins, however, at this point, we focused more in the analysis of the improvements that can arise from the introduction of a multi-layer architecture. In fact, the energy functions used here are too coarse to provide good biological models, as confirmed also from out tests. In the future, we plan to develop and use a more reliable energy, possibly encapsulated in an iterative process using more and more detailed —and computationally expensive— potentials to refine the previous solutions.

The agent-based framework presented here shows interesting potentials, though the energy functions used are too coarse to compete with up-to-date ab-initio predictors. In particular, it’s remarkable the gain in speed and in quality of solutions with respect to a sequential version of the algorithm (cf. Section 7).

The cooperation level is a powerful feature of the framework, though it is not exploited yet. Specifically, it offers the possibility of designing complex heuristics depending on external information and on the search history [23]. Up to now, we use only local information related to secondary structure. One possible direction to improve it is to introduce information about the real physical dynamic of the folding, adding, for instance, a computational field mimicking the hydrophobic force, therefore compactifying the structure at the beginning of the simulation. Another possibility is to identify the so called folding core [9], which is a set of key contacts between some amino acids. These contacts have a stabilizing effect, and their formation is thought to be one of the most important steps in the folding process. These contacts can be forced during the simulation by means of a computational field. In general, to fully exploit the cooperativity of agents, we need to integrate this information with exploration-dependant knowledge.

Finally, we plan to integrate this simulation engine in a global schema for protein structure prediction, combining together multi-agent simulation (using a potential modeling also the side-chain), lattice minimization [12] and molecular dynamics.

Acknowledgements

We thank FIRB RBNE03B8KK and PRIN 2005015491 for partially supporting this research. A special thank to Alessandro Dal Palù for his precious work on the second energy model. We also wish to thank the anonymous referee for the detailed and precise comments.

References

[1] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann machines*. John Wiley and sons, 1989.

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

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

[4] H. M. Berman et al. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000. `http://www.rcsb.org/pdb/`.

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

[6] L. Bortolussi, A. Dal Palù, A. Dovier, and F. Fogolari. Protein folding simulation in CCP. In *Proceedings of BioConcur2004*, 2004.

[7] L. Bortolussi, A. Dovier, and F. Fogolari. Multi-agent simulation of protein folding. In *Proceedings of* MAS-BIOMED 2005, 2005.

[8] P. Bradley, L. Malstrom, B. Qian, J. Schonbrun, D. Chivian, D. E. Kim, J. Meiler, K. Misura, and D. Baker. Free modeling with Rosetta in CASP6. *Proteins*, 7S:128–134, 2005.

[9] R. A. Broglia and G. Tiana. Reading the three-dimensional structure of lattice model-designed proteins from their amino acid sequence. *Proteins: Structure, Functions and Genetics*, 45:421–427, 2001.

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

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

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

[13] 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.

[14] Swedish Institute for Computer Science. Sicstus prolog home page. `http://www.sics.se/sicstus/`.

[15] D. R. Greening. Parallel simulated annealing techniques. *Physica D*, 42:293–306, 1990.

[16] D. T. Jones. Predicting novel protein folds by using FRAGFOLD. *Proteins*, 5S:127–131, 2001.

[17] A. Kolinsky, A. Godzik, and J. Skolnick. A general method for the prediction of the three dimensional structure and folding pathway of globular proteins: application to designed helical proteins. *J. Chem. Phys.*, 98:7420–7433, 1993.

[18] B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, and D. Baker. Design of a novel globular protein fold with atomic-level accuracy. *Science*, 302:1364–1368, 2003.

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

[20] 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.

[21] M. Mamei, F. Zambonelli, and L. Leonardi. A physically grounded approach to coordinate movements in a team. In *Proceedings of* ICDCS, 2002.

[22] C. Micheletti, F. Seno, and A. Maritan. Recurrent oligomers in proteins - an optimal scheme reconciling accurate and concise backbone representations in automated folding and design studies. *Proteins: Structure, Function and*, 40:662–674, 2000.

[23] M. Milano and A. Roli. Magma: A multiagent architecture for metaheuristics. *IEEE Trans. on Systems, Man and Cybernetics - Part B*, 34(2), 2004.

[24] 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.

[25] J. Moult. A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Curr. Op. Struct. Biol.*, 15:285–289, 2005.

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

[27] B. Park and M. Levitt. Energy functions that discriminate x-ray and near-native folds from well-constructed decoy. *Proteins: Structure Function and Genetics*, 258:367–392, 1996.

[28] 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.

[29] B. Rost. Protein Secondary Structure Prediction Continues to Rise. *J. Struct. Biol.*, 134:204–218, 2001.

[30] O. Schueler-Furman, C. Wang, P. Bradley, K. Misura, and D. Baker. Progress in modeling of protein structures and interactions. *Science*, 310:638–642, 2005.

[31] J. Skolnick. In quest of an empirical potential for protein structure prediction. *Current Opinion in Structural Biology*, 16:166–171, 2006.

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

[33] 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.

[34] Y. Zhang, A. K. Akaraki, and J. Skolnick. TASSER: an automated nethod for the prediction of protein tertiary structures in CASP6. *Proteins*, 7S:91–98, 2005.

Enhancements introduced by orchestra director

First Energy Model	Energy	RMSD
Without orchestra director	-6.688 (1.483)	25.174 (0.863)
With orchestra director	-56.685 (2.518)	7.644 (0.951)

Second Energy Model	Energy	RMSD
Without orchestra director	28.944 (5.500)	12.892 (0.825)
With orchestra director	11.325 (3.882)	8.404 (1.078)

Table 1: Comparison of performances of the framework with and without the introduction of the orchestra director agent, for the first energy model (top table) and the second one (bottom table). Values are averages over ten runs, variance is shown in brackets.

Sequential vs Parallel simulation

First Energy Model	Energy	RMSD	time (min)
Sequential	-54.068 (1.949)	10.728 (0.793)	280
Multi-Agent	-56.685 (2.518)	7.644 (0.951)	85

Second Energy Model	Energy	RMSD	time (min)
Sequential	39.599 (5.072)	11.255 (0.837)	28
Multi-Agent	11.325 (3.882)	8.404 (1.078)	10

Table 2: Comparison of the multi agent scheme with a sequential simulated annealing performing pivot and crankshaft moves. Top table compares results for the first energy model (for protein 1VII), while bottom table deals with the second energy model. Both simulations execute the same number of moves per amino acid. Multi-agent simulation performs better in all aspects, and it is notably three times quicker, using 4 processors. Values are averages over ten runs, variance is shown in brackets.

List of tested proteins

Protein	# amino	helix	sheet
1LE0	12		x
1KVG	12		x
1LE3	16		x
1EDP	17	x	
1PG1	18		x
1ZDD	34	x	
1VII	36	x	
2GP8	40	x	
1ED0	46	x	x

Table 3: List of tested proteins, with number of amino acids and type of secondary structure present.

Results for the first potential, without cooperation

Protein	Energy	RMSD
1LE0	-11.5984 (0.8323)	4.0324 (0.7783)
1KVG	-19.8048 (0.7838)	3.8898 (0.6206)
1LE3	-29.5412 (1.1330)	6.0749 (0.7525)
1EDP	-62.9978 (1.5487)	5.0703 (0.7261)
1PG1	-52.7987 (1.5356)	7.1490 (0.8791)
1ZDD	-45.4133 (1.9326)	8.4255 (1.2578)
1VII	-56.6850 (2.5189)	7.6447 (0.9509)
2GP8	-10.5285 (1.7073)	8.7200 (1.1488)
1ED0	-50.8481 (2.5158)	9.3103 (1.1701)

Table 4: Results for the first energy model without cooperation. Values are averages over ten runs, variance is shown in brackets.

Results for the first potential, with cooperation

Protein	Energy	RMSD
1LE0	-10.5248 (0.9756)	4.2291 (0.7025)
1KVG	-16.7263 (0.9267)	3.8555 (0.5476)
1LE3	-27.1925 (0.9693)	5.5847 (0.5870)
1EDP	-49.0921 (1.6160)	3.7743 (0.5743)
1PG1	-33.7463 (1.3319)	4.3540 (0.6856)
1ZDD	-49.6080 (2.2137)	8.2052 (1.1388)
1VII	-56.2129 (1.5908)	7.7737 (0.8158)
2GP8	-2.6426 (1.8351)	4.1871 (0.9585)
1ED0	-57.4477 (2.2198)	8.8542 (1.0181)

Table 5: Results for the first energy model with cooperation. Values are averages over ten runs, variance is shown in brackets.

Results for the second potential, without cooperation

Protein	Energy	RMSD
1LE0	2.8798 (1.4315)	5.4120 (0.6963)
1KVG	-1.3340 (1.1783)	5.0596 (0.9254)
1LE3	1.8239 (1.7636)	7.2068 (0.9228)
1EDP	-3.5488 (0.7719)	6.1642 (0.7851)
1PG1	23.7913 (1.3656)	9.1229 (1.3274)
1ZDD	14.1963 (3.1172)	8.5113 (1.2836)
1VII	11.3256 (3.8826)	8.4040 (1.0787)
2GP8	48.6291 (4.5198)	7.9520 (1.5060)
1ED0	16.7202 (2.1936)	10.7979 (0.8692)

Table 6: Results for the second energy model without cooperation. Values are averages over ten runs, variance is shown in brackets.

Results for the second potential, with cooperation

Protein	Energy	RMSD
1LE0	4.8766 (2.7584)	5.2671 (0.8173)
1KVG	1.7643 (1.4065)	4.9455 (0.8481)
1LE3	1.8885 (1.6943)	6.6555 (1.1844)
1EDP	-4.7164 (0.8152)	6.0124 (0.4821)
1PG1	39.3512 (2.5879)	12.9420 (1.0104)
1ZDD	-10.7000 (0.8261)	6.8331 (1.2393)
1VII	-9.0183 (1.2882)	8.0266 (1.1269)
2GP8	-7.7530 (1.0468)	5.3764 (1.4519)
1ED0	-9.6144 (1.4974)	8.4351 (0.8369)

Table 7: Results for the second energy model with cooperation. Values are averages over ten runs, variance is shown in brackets.

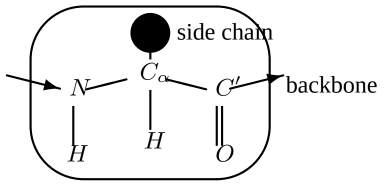

Figure 1: Amino acids: Overall structure

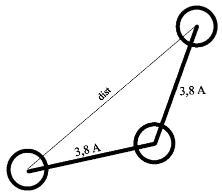

Figure 2: A simplified model of aminoacids, viewed as spheres centered in C_α atom.

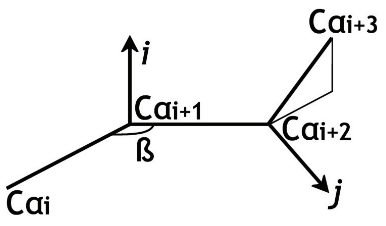

Figure 3: Bend angle β

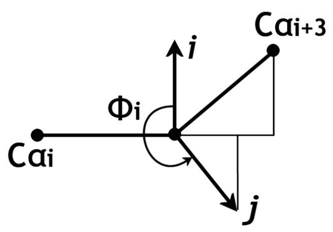

Figure 4: Torsion angle Φ_i (projection orthogonal to the axis $C\alpha_{i+1}$–$C\alpha_{i+2}$)

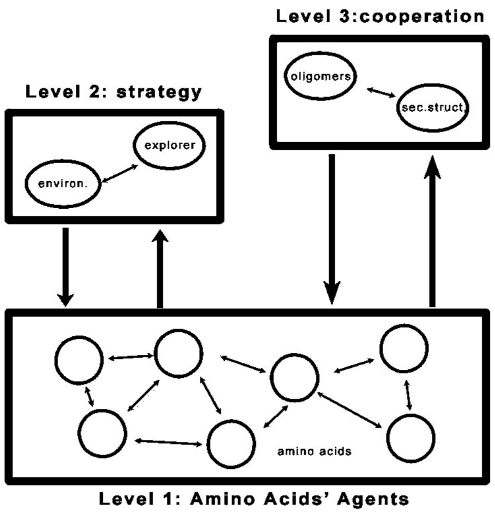

Figure 5: Structure of the multi-agent simulation. Black boxes represent the levels, blue circles the agents and red arrows the communications.

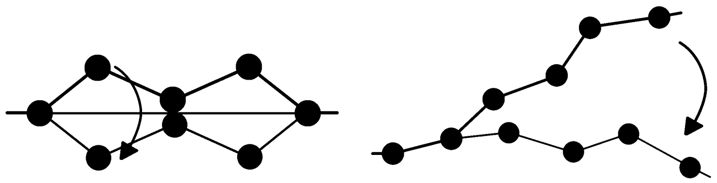

Figure 6: Crankshaft move (left) and pivot move (right)

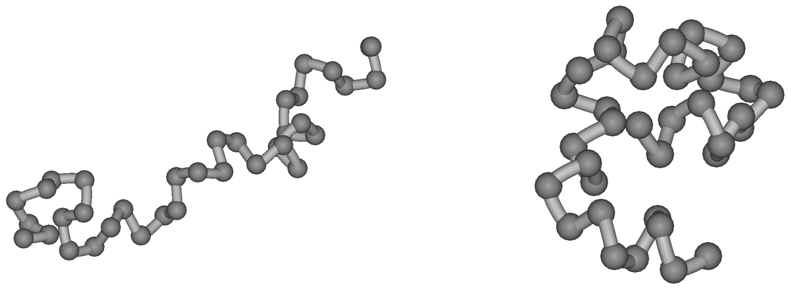

Figure 7: Two structures for 1VII generated using the second energy model, with (right) and without (left) orchestra director.

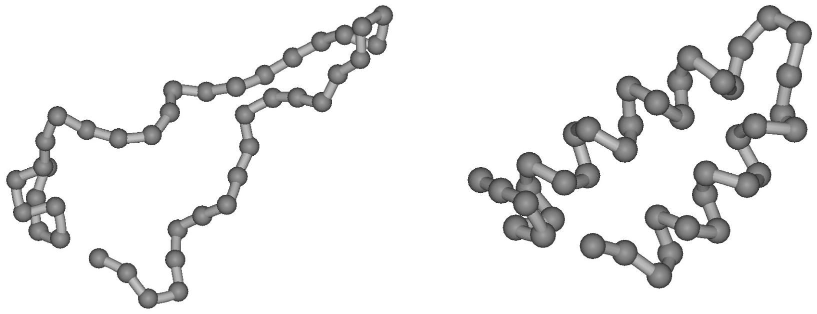

Figure 8: Protein 2GP8 predicted with the first potential. From left to right: a solution without cooperation and a solution with cooperation. Native state is depicted in Figure 9.

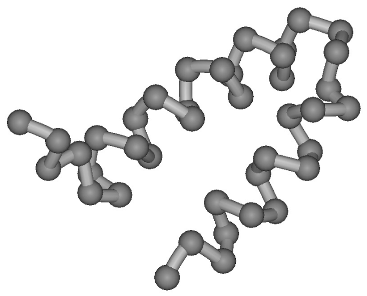

Figure 9: Native state for 2GP8.

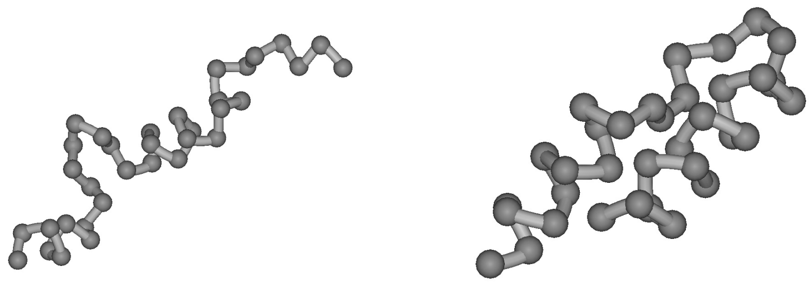

Figure 10: Protein 1ZDD predicted with the second potential. From left to right: a solution without cooperation and a solution with cooperation. Native state is depicted in Figure 11.

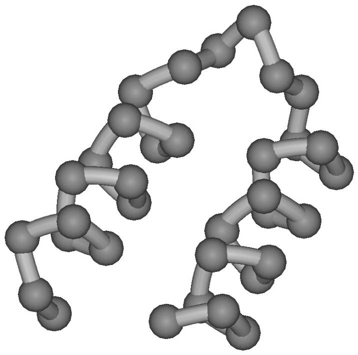

Figure 11: Native state for 1ZDD.