Exploring the high selectivity of 3-D protein structures using distributed memetic algorithms

Mario Inostroza-Ponta[a,*], Márcio Dorn[d], Iván Escobar[a], Leonardo de Lima Correa[d], Erika Rosas[c], Nicolás Hidalgo[b], Mauricio Marin[a]

[a] *Universidad de Santiago de Chile, Estación Central, Santiago, Chile.*
[b] *Universidad Diego Portales*
[c] *Universidad Técnica Federico Santa María*
[d] *Institute of Informatics, Federal University of Rio Grande do Sul*
Av. Bento Goncalves 9500, 91501-970, Porto Alegre, RS, Brazil.

Abstract

This paper addresses the problem of predicting the tertiary structure of a protein given its amino acid sequence, which has been reported to belong to the NP-Complete class of problems. We design an ad-hoc distributed memetic algorithm (DMA) and evaluate several algorithm configurations in terms of different distributed population structures, ad-hoc local search strategies and the combination of two energy functions. The algorithm uses an asynchronous hierarchical population of agents that exchange solutions along the execution of the algorithm. Extensive computational experiments were carried out in order to test: (1) the impact of the communication on different population structures, (2) the combination of the energy functions used for fitness calculations, (3) the scalability of the algorithm for structures with a larger number of agents, (4) the performance of the different approaches proposed for local search and diversity calculations, (5) the biological significance of the predicted structures and (6) to compare the best performing configuration of the DMA with other algorithms from the literature. The algorithm was tested on 20 sequences of different size, and the analysis was performed regarding both computational quality and biological significance of the predicted structures. Results show that the combination of energy functions and the proposed Distributed Memetic Algorithm allows the prediction of structures that are similar to the experimental ones. Performance analysis shows that increasing parallelism improves the execution times, without worsening the quality of the solutions.

Keywords: protein structure prediction, memetic algorithm, distributed memetic algorithms, structural bioinformatics.

*Corresponding author
Email address: mario.inostroza@usach.cl (Mario Inostroza-Ponta)

1. Introduction

It is well known that proteins are present in all living systems, and are responsible for performing a large set of functions in the cell. Knowing the structure of a protein allows the investigation of the biological processes in which the protein could be involved more directly, with higher resolution and finer detail. The sequence-protein-structure paradigm (also knows as the "lock-and-key" hypothesis) says that the protein can achieve its biological function only by reaching a given tertiary structure determined by its amino acid sequence. The Protein Structure Prediction (PSP) problem is one of the most important problems in Structural Bioinformatics. Each protein, or polypeptide, is defined by an unique sequence of chained amino acids that under some physiological conditions reach a particular 3-D shape. *In silico* prediction of the three-dimensional (3-D) structure of proteins has been the *'holy grail'* of computational biologists for many years [41].

The PSP problem has challenged Biochemists, Biologists, Computer Scientists, and Mathematicians, comprising the domain of high-dimensional continuous optimization problems [9]. Nowadays, a wide range of optimization algorithms and metaheuristics are being applied in an attempt to find approximate solutions to predict the 3-D structure of protein only from its linear sequence of amino acid residues [10]. Despite the number and the variety of methods developed for structure prediction over the past 40 years, no truly accurate methods exist to predict the correct structure of polypeptides. Most of the existing challenging optimization problems, such as the PSP, cannot be optimally solved by any known computational method due to the high dimensionality and complexity of the search space [1].

In this paper we propose a knowledge-based *Distributed Memetic Algorithm* (DMA) [30] to deal with the 3D PSP problem. The proposed algorithm takes advantage of the experimental knowledge stored in the *Protein Data Bank* (PDB). The main goal is to decrease the PSP complexity, reducing the size of the search space and increasing the method effectiveness. To incorporate the structural information of protein conformations, we use the *Angle Probability List* (APL) [5, 4]. APL analyses the conformational preferences of amino acids in proteins according to its secondary structure. The DMA proposed in this work also uses a *Local Search* (LS) technique to intensify the search around the most favourable regions.

Parallelism is intrinsic for memetic algorithms. Harris et al. [19] and Blocho et al. [3] use memetic algorithms with a parallel local search. The first solves the quadratic assignment problem using a ternary tree of 13 agents. The second solves the vehicle routing problem with time windows using a ring structure of N agents. This approach significantly improves execution times. However, the synchronization among the agents makes it difficult to scale over 32 agents [3]. Our proposal also explores different structures for cooperation between agents. We used tree-based structures to calibrate knowledge sharing and analyze the scalability of the approach.

This paper is organized as follows. Section 2 presents a brief review about

the problem of PSP and the current techniques that exist to use distributed metaheuristics. Section 3 describes in detail the proposed method and their variants. In Section 4 we present the computational experiments and discussion of results. Finally, the last section present the conclusions and points out some future works.

2. Background

2.1. The 3D Protein Structure Prediction Problem

The prediction of protein structure is one of the most important and challenging problems in structural bioinformatics. Traditionally, protein's structure can be obtained through X-ray crystallography and multi-dimensional magnetic resonance in the laboratory. Although this method is expensive, it provides an accurate protein structure. However, the high costs of the method, its extreme difficulty,and the time required to carry out the study cannot cope with the increasing number of proteins data sequences. In this scenario, computational methods are a more suitable solution to deal with this problem due to its low costs and fast results [27]. These methods have the potential to correlate and predict the structure of a protein from its primary sequence to overcome the difficulties associated with the experimental approaches [21].

In an attempt to solve this problem many methods and algorithms have been proposed and they can be classify over four categories: (a) methods without database information; (b) methods with database information; (c) fold recognition methods, and (d) comparative modelling methods.

Methods without database information are founded on thermodynamics and based on the fact that the structure of a protein corresponds to the global minimum energy conformation. Thus, a stable structure of a protein always consumes minimum energy among all the conformations of the protein. These methods are supported by two main components: a mathematical model for the protein-energy function and an optimization algorithm to find a global minimum of the potential energy [42] [7] [16] [15] [35].

The methods with database information follow a set of general rules extracted from protein databases to establish a starting protein structure. These methods compare fragments of the protein with fragments of the structure of reference proteins [49] [37] [45] [11] [28] [44] [8].

Fold recognition methods assume that the structure is more likely to be preserved in the protein evolution rather than the sequence. i.e., proteins with no apparent sequence similarity could present similar folds. These methods' goal is to predict the protein structure by fitting a protein sequence correctly against a structural model [2] [22] [43] [17].

Finally, the comparative modelling employs a target sequence of amino acid is aligned against the amino acid sequence of another protein with known structure (template protein) and stored in the protein data bank (pdb) [29] [38] [13] [43].

Predicting the 3D structure of a protein molecule *in silico* is a NP-hard problem. Metaheuristic optimization algorithms have potential ability to solve

complex problems efficiently providing near optimal solutions. Our proposal follows this optimization strategy as well as many of the before cited solutions.

2.2. Distributed Memetic Algorithms

Memetic algorithms have been distributed in literature given their collaborative nature and the complexity of the real-world problems they are used to solve. In Gong et al. [18] the authors review the models of the distributed evolutionary algorithms. Different authors have proposed different memetic algorithms that use distributed models that include the well-know island model [14, 33, 39], the cellular model [26, 34], and the hierarchical architecture [19]. In this work we study the hierarchical model of organisation among the agents, which has been previously shown good results for our particular problem in [8].

The different cooperation topology inside the components of each model has influence on the algorithm search capabilities in terms of increasing exploration or exploitation of the search space. Cooperation is to define: what is to be transferred, what is the cooperation topology, how often the processes communicate, and what is the strategy to handle immigrant and emigrants. Nalepa et al. [33] [32] studied the impact of this cooperation for the VRPTW and the PDTW problem. The authors conclude that if the convergence time is to be as minimum as possible, frequent cooperation intervals are better. Moreover, if the running time is a restriction, then rarer co-operation intervals benefit a broader exploration of the search space. Our proposal also explores different structures for cooperation between agents. We used tree-based structures in order to calibrate knowledge sharing and analyse the scalability of the approach.

3. DMA-3DPSP: A Distributed Memetic Algorithm for the PSP problem

We propose a Distributed Memetic Algorithm to tackle the `3-D` Protein Structure Prediction problem (`DMA- 3DPSP`). The distributed model follows a hierarchical population structure based on the *Island Model* [18], with a set of agents working independently in a particular search space and cooperating by exchanging information according to a hierarchical population structure. The hierarchical structure used will be determined from the computational experiments. In the `DMA-3DPSP`, agents share the information about local solutions found and preserve the most promissory ones. The goal of this distributed strategy is to improve both the execution times of the search process and the quality of the solutions obtained.

Algorithm 1 shows the pseudocode for each agent of the `DMA-3DPSP`. Agents are organised under a hierarchical structure or *n-ary* tree. The structure used defines the concept of sub-populations composed by a *leader agent* and *supporter agents*. *Leaders* are responsible for receiving the best solutions generated by their *supporters* and also to spread good quality solutions among their *supporters*. Information updates are performed asynchronously each time an agent finds a better solution. Each agent keeps their best solutions on a local memory

called *pockets* (line 5) that keeps track of the best solutions found so far by the agent. In each generation, the agent selects one of its pocket solutions and performs a *crossover* with the solution received from its leader. Then, on the *offspring* generated it applies a local search algorithm to improve the quality of the solution. After that, if a new best solution is found, the agent sends this solution to its leader, and it starts a new generation. In the following sections we described the main components of the proposed algorithm.

3.1. Agent Structure

Figure 1 shows a scheme of the main components of the agent. Each agent has two structures to store and to manage the progress of the search: (1) the *current*, that is a temporary solution on which evolutionary operators are being applied, and (2) a set of *pockets*, a local memory structure that stores the best solutions found by the agent and received by its supporters. The pockets enable the agent to perform a better exploration of the search space by focusing on the sectors where these solutions are located; their number is variable, and it is a parameter of the algorithm.

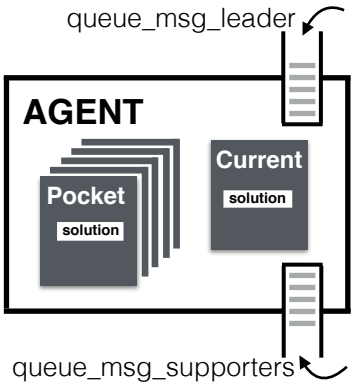


Figure 1: Structure of each agent of `DMA-3DPSP`. Agents work in the *current* solution, save the best solutions found in the *pockets* and send them to *leader* and supporters through message queues.

Each *solution* (*current* and *pockets*) describes a possible `3-D` structure for the peptide and is composed by: (1) a vector of size $2 \times n$ (for a sequence of n amino acids) with the value of the ϕ and ψ torsion angles for each residue of the sequences and (2) the free energy of the structure given by the energy fitness function. For communication, the agents use a distributed message queueing system. Each agent has two queues, one dedicated to receiving the solutions from its leader and the other to receive them for its supporters (sub-agents). This asynchronous model allows the agents to send solutions to other agents without interrupting the search process.

Algorithm 1 Distributed Memetic Algorithm for the PSP problem - DMA-3DPSP

```
1: Input: seq ← Aminoacid sequence, ss ← Secondary structure, histograms
   ← APL Histograms
2: procedure DMA-3DPSP
3:     for all Agent do
                                              ▷ Initialise agent population
4:         pockets ← ∅
5:         pockets_leader ← None
6:         APL ← LoadHistograms(seq, histograms)
7:         current ← initSolution(seq, APL)
8:         updatePockets(current)
                                                    ▷ Perform search
9:         while stop_criteria has not been reached do
10:            if Agent is root then
11:                current ← GetRandom(pockets)
12:            else
13:                current ← Crossover(ss, GetRandom(pockets),
   Roulette(pockets_leader))
14:            end if
15:            current ← SimulatedAnnealing(current, seq, APL)
16:            updatePockets(current)
                                              ▷ Perform communication
17:            while queue_supporters ≠ ∅ do
18:                updatePockets(queue_supporters.pop())
19:            end while
20:            while queue_leader ≠ ∅ do
21:                pockets_leader ← queue_leader.pop()
22:            end while
23:            if pockets were updated then
24:                for each Supporter do
25:                    send pockets to Supporter
26:                end for
27:                if pockets[0] was updated then
28:                    send pockets[0] to Leader
29:                end if
30:            end if
31:            restartControl()
32:        end while                                ▷ End search process
33:        if Agent is root then
34:            return pockets[0]
35:        end if
36:    end for
37: end procedure
```

3.2. Population Structure and Communication

Agents in our approach are organised in a *n-ary* tree, where the father of an agent is called its *leader* and the children of the agent are part of its *supporters*. Each group of a leader and its supporters defines a subpopulation of the system. *Leaders* keep in their *pockets* the best solutions found in the subpopulation, and feed the search process of their *supporters* with these good solutions. Supporters, on the other hand, are responsible for sending their best solutions to their leaders as soon as they found a new one. Agents work independently and communicate with them only to share solutions and major events. In order to avoid interruption and the loss of solutions, a message queueing system is deployed at each agent. It enables the agent to send solutions without interrupt the receiver's search process nor wait for an acknowledgment.

We evaluate three population structures: (1) a binary tree, (2) a ternary tree, and (3) a 12-ary tree, all of them composed by 13 agents. In the 12-ary tree, the structure has one level, following a master-slave approach where twelve of the thirteen agents have the same leader agent. The ternary tree builds a structure with two levels and the binary tree a three level structure. A structure with more agents should reach a better exploration of the search space and therefore obtain better results regarding processing time and solution quality, however, it also impacts communication. Communication used for sharing solutions between the agents is bi-directional: each time an agent has found a new best solution it sends the solution to its leader and also to its supporters, then a structure composed with a larger number of agents requires a more significant amount of communication. It is necessary to explore the balance between the convergence time and the exploration of the search space to reach high-quality solutions. Section 4.1 presents a complete description of each one of these structures.

3.3. Fitness Function

In order to evaluate the quality of the predicted structures and to guide the search process of the algorithm, a fitness function is required. Theoretically, the native structure of a protein corresponds to the global minimum of its free energy [6]. We propose a fitness function compose by two different energy approaches: (1) the *Rosetta all-atom high-resolution* energy function, using "*Talaris*" weights for the energy terms, and (2) the Solvent Accessible Surface Area calculation (`SASA`) [36] with an atomic radius of 1.5Å, both of them implemented in the *PyRosetta toolkit*. The former seeks to improve the stability of the structure, looking for solutions with low values of energy among all interactions and that fulfil the restrictions of atoms clashes, and the latter aims to improve the packing of the `3-D` structure by reducing the area exposed to solvent.

For the combination of the energy functions terms `Talaris` and `SASA`, we used the fitness function described in equation 1:

$$fitness(S) = (1 - \alpha) \times Talaris + \alpha \times SASA \tag{1}$$

where α determines the weight of the energy functions. In the computational experiments we will show the results of testing the fitness function using

$\alpha \in \{0.0; 0.1; \ldots; 0.9; 1.0\}$, where an $\alpha = 0.2$ means that the fitness function is composed by 80% of Talaris and 20% of $SASA$.

3.4. Crossover Operator

The `DMA-3DPSP` combines population operators and a *Simulated Annealing* local search strategy. We used the same operators as those used in [20], but we adapt them to work asynchronously. At the beginning of each generation the algorithm generates new *offspring* using a *crossover* operator. The *crossover* process first selects two solutions as parents, one of them randomly selected from the pockets of the agent, and the other one selected from its leader's pockets through a roulette wheel selection, so the best solutions are more likely to be selected. The new *offspring* is generated by taking the angles of each residue of the sequence from the residues of the parents, with a probability of 0.6 for the parent taken of the agent's pockets and 0.4 for the one selected from its leader's pockets. In order to preserve the formation of secondary structures, we implement a *crossover* operator, where the amino acid chain is separated into segments based on its secondary structure (received as a parameter of the algorithm). Sequential amino acids that share the same secondary structure will be treated as a whole section for effects of *crossover*, and the angles for the residues of the section will be taken from the same parent. The only agent that does not perform crossover is the leader of the population or root agent. Instead, this agent only replaces its current with one of its pocket solutions randomly chosen.

3.5. Local Search Algorithm

The new solution generated in the *crossover* will go through the local search procedure. For each residue of the sequence, the algorithm decides whether it goes to local search with a probability of 0.9. A residue selected for local search first determines whether to perform a jump to another sector in the search space based on the preferences of the APL [5, 4]. Then, the jump is performed with a probability of 0.3, and it will add a value $< jump$, which will be decreased in each generation at a rate of 0.85. After that, a *Simulated Annealing* algorithm is performed on the angles ϕ and ψ of the residue. For each angle, the algorithm makes small modifications such that it is always within the same angle cell. Each change will be followed by an evaluation of the globally free energy of the structure. Then, the solution is accepted if it improves the fitness value, otherwise the algorithm applies the Boltzmann probability function to decide if the new solution is accepted. The temperature function controls the acceptance function and the number of iteration of the *Simulated Annealing* algorithm:

$$T(iteration) = T_{initial} \times \alpha^{iteration} \tag{2}$$

where $T_{initial}$ is the initial temperature and α is the cooling factor. We use $T_{initial} = 4000$ for the root agent and $T_{initial} = 2000$ for the rest of the agents, $\alpha = 0.9$ and a limit of $T(iteration) < 0.1$, using the results of the parametrisation tests performed in [20].

3.6. Pockets Updating Policy

After memetic operators are applied, the agent evaluates whether it should update its *pockets* with the new solution. To determine if a new solution is worth to keep in the *pockets*, the agent compares its free energy (through the $E(sol)$ function) and its diversity (through the $Div(sol_a, sol_b)$ function) with each one of its *pockets*. To be saved in the *pockets*, the new solution must fulfill at least one of the following conditions:

- $\exists j$ / (Div($current$, $pocket_j$) $< Div_{min}$) $\wedge$ (E($current$) $<$ E($pocket_j$))

  So $current$ will replace $pocket_i$ such that Div($current$, $pocket_i$) is minimum, $\forall i$ where E($current$) $<$ E($pocket_i$).

- (Div($current$, $pocket_j$) $\geq Div_{min}$, $\forall j$) $\wedge$ ($pockets \neq$ full)

  So $current$ will replace the first empty pocket.

- (Div($current$, $pocket_j$) $\geq Div_{min}$, $\forall j$) $\wedge$ ($pockets =$ full) $\wedge$ ($\exists k$ such that E($current$) $<$ E($pocket_k$))

  So $current$ will replace $pocket_i$ such that E($pocket_i$) is maximum, $\forall i$.

The agent also uses these rules to determine whether it should save the solutions put on its queue from its supporters. Then, if there was an update in the pockets, the agent proceeds to the communication process. The best solution is sent to the agent's leader and the list of *pockets* is forwarded to each supporter to perform the crossovers of the next generations. Therefore, good solutions go towards the top of the hierarchical structure.

3.7. Local Search Approaches

We evaluate a set of modifications on the original Simulating Annealing local search implementation in order to improve the quality of its results. In the following we give a detail description of each one of these approaches:

- **LS1**: in the original local search algorithm, before the execution of the Simulating Annealing on the angle cell, the residues could perform a change of neighborhood. Then, with a probability of 0.3, the angles ϕ and ψ of the residue will relocate to a different position in the search space, based on the preferences for these angles on the APL. This was meant to improve the exploration of the search space, but there is the risk of losing a good solution when the changed is performed. We add an additional restriction: the residue will only maintain the new position if it generates an improvement in the fitness value.

- **LS2**: the original local search performs the Simulating Annealing on the dihedral angles independently. Therefore, the algorithm first applies small modifications in the ϕ angle while ψ remains the same, and then it does the same for ψ with ϕ remaining static. This implementation limits the exploration around the angle cell. In order to avoid the above, this local search modifies both angles in each iteration of the Simulated Annealing.

- **LS3**: in the original local search algorithm, the angle cell that limits the range of the small random modifications is determined at the start of the Simulated Annealing, and it remains static throughout the iterations of the algorithm. The angle cell with the size of one degree is determined by taking the two integer values that surround each dihedral angle as the limit, and every modification must be kept in this area. We modify this approach by implementing a dynamic angle cell that is recalculated every time the algorithm applies an adjustment in the dihedral angles, allowing the algorithm to focus the search towards the sectors that show improvements in the results. The new cell will be the area delimited by a range of $\pm 0.5°$ from the new ϕ and ψ angles.

- **LS4**: this local search has two phases: (1) during the first iterations, the algorithm uses the `APL` to determine the new positions for the ϕ and ψ angles. The size of the modification is delimited by a range not greater than *jump*, defined before, and it is only performed if the new position improves the global free energy or it is accepted by the acceptance function of the Simulated Annealing. When *jump* value has become less that one degree, (2) for the rest of the iterations the algorithm will use the original approach applying small random modifications in the angle cell. This local search algorithm allows searching good sections of the search space in the first iterations and focus the search around them in the last iterations.

3.8. Diversity Approaches

We implement two different techniques to determine the diversity among the solutions in the pockets of the agents:

- **Div1**: the first approach calculates the average difference among the dihedral angles from the residues of the two solutions, according to the equation 3.

$$Div(a,b) = \frac{\sum_{i=1}^{n}(\phi_i^a - \phi_i^b) + \sum_{i=1}^{n}(\psi_i^a - \psi_i^b)}{n} \tag{3}$$

  where ϕ_i^a and ψ_i^a represent the dihedral angles of the residue i in the structure a, and ϕ_i^b and ψ_i^b do the same for the structure b.

- **Div2**: the second approach uses the calculation of the `RMSD` (CA atoms) of the structures as a measure of the difference between the solutions. The `RMSD` is calculated by superimposing the atoms of the structures and measuring their average distance, as it is shown in the equation 4.

$$RMSD(a,b) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}||a_i - b_i||^2} \tag{4}$$

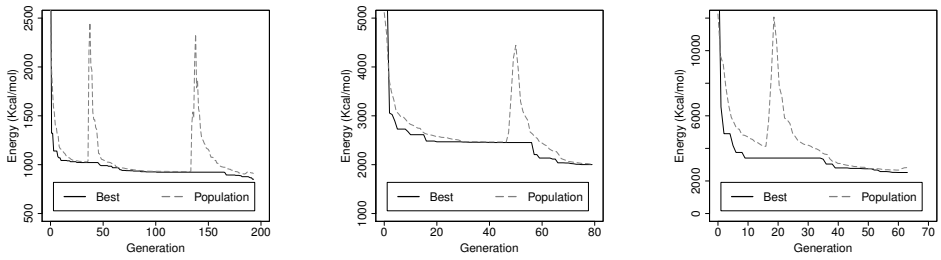
  where a_i and b_i are the vector of positions for the atom i of the structures a and b when they are optimally superimposed.

3.9. Restart Procedure

The search space is continuous, which makes easy for the algorithm usually to fall on local optima, so the algorithms implements a restart procedure to escape from these situations. If during a g_{limit} number of generations the best solution of the leader agent of the population does not change, the algorithm restarts the whole population. To do it so, it deletes the solutions stored in the pockets of every agent, keeping only the best solution found so far by the entire population. The value of g_{limit} is a parameter of the algorithm, but it dynamically changes during the execution according to the number of generations that took to reach the last best solution. This gives time to the algorithm to converge in advanced stages of the evolution process, where usually is more difficult to reach better solutions. The g_{limit} for the n restart is calculated following the equation:

$$f(n) = \begin{cases} g_{min} & n = 1 \\ g_{min} + (g_{current} - g_{last}) - f(n-1) & n > 1 \end{cases} \tag{5}$$

where g_{min} is given as parameter for the first g_{limit}, $g_{current}$ is the current generation in which the algorithm is working and g_{last} is the generation in which the last restart was made. The root agent is in charge of the restart procedure, evaluating the number of generations since the last time its best pocket was updated, and communicating and synchronizing the rest of agents when it is necessary to make a restart.



(a) 1L2Y (b) 3P7K (c) 2JUC

Figure 2: Analysis of energy convergence of the agent pockets of the `DMA-3DPSP` on three different protein sequences. The solid line shows the energy value of the best solution found so far by the entire population, and the dashed line shows the average energy of the best solutions stored in each agent of the population.

In Figure 2 we show the convergence of the free energy on the `DMA-3DPSP` for three different sequences. When a restart is performed, the energy of the pockets of the population (dashed line) increases rapidly, because the relocation of the solutions in the search space, but after these solution converge it is possible to see how the relocation allows the algorithm to escape from local minima.

4. Computational Experiments

We performed several computational experiments to evaluate the performance of the alternative configurations of the proposed DMA. The tests aim to evaluate: (1) the impact of the communication on different population structures, (2) the combination of the energy functions used for fitness calculations, (3) the scalability of the algorithm for structures with a greater number of agents, (4) the performance of the different approaches proposed for local search and diversity calculations, (5) the biological significance of the predicted structures and (6) compare the best performing configuration of the DMA with other algorithms from the literature.

We select 20 sequences from known peptide structures as cases of study for our experiments. Table 1 shows a list of the chosen peptides, its number of amino acids and secondary structures. The selection was made aiming at using sequences that have been used on other works for comparison purposes [24, 31, 8]. Also we selected those that have different secondary structures to see the performance of the proposal on different conditions. We run a small test to determine the time required to execute the first 100,000 evaluations of the energy function. Based on those results and the size of the sequences, we classified them into three groups: (1) *short* sequences, with a size lower than 40 residues; (2) *medium* sequences, with a size between 40 and 50 residues; and (3) *long* sequences, with a size over 50 residues.

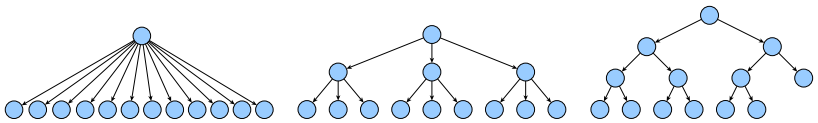
PDB ID	Size	Folding pattern	100K Test (sec)
1K43	14	2 β-sheet	35,04
1L2Y	20	1 α-helix, 1 3_{10} helix	55,75
2MTW	20	1 α-helix	66,51
*1WQC	26	2 α-helix	90,31
*1ACW	29	1 α-helix, 2 β-sheet	63,19
1DFN	30	3 β-sheet	75,81
1Q2K	31	1 α-helix, 2 β-sheet	105,94
2F4K	35	3 α-helix	125,74
2MR9	44	3 α-helix	152,79
2P81	44	2 α-helix	241,76
*3P7K	45	1 α-helix	229,34
1AB1	46	2 α-helix, 2 β-sheet	157,69
*3V1A	48	2 α-helix	225,18
2P6J	52	3 α-helix	324,41
1ENH	54	3 α-helix	272,15
*2JUC	59	3 α-helix, 1 3_{10} helix	278,51
1ROP	63	2 α-helix	300,25
*2P5K	64	3 α-helix, 2 β-sheet	330,16
1AIL	73	3 α-helix	332,03
2PMR	87	3 α-helix	402,44

Table 1: List of protein sequences used in computational tests. We classified them in three groups based on its size and a short test of 100,000 evaluations of the energy function. Sequences marked with * were used for tests to define the best DMA implementation.

We also selected 2 sequences of each group (marked with a * in Table 1) that will be used in the tests to determine the best `DMA` implementation.

To validate the results we execute each computational experiment 10 times. Experiments were executed in two different environments: (1) a 4-processor 2.6GHz Intel Xeon CPU with 8 cores (32 cores) and 32GB RAM, and (2) a cluster composed by 7 machines (112 cores in total): two 4-processor Intel Xeon E-5405 with 2 cores (2×8 cores), three 8-processor Amd Opteron 6128 with 2 cores (3×16 cores) and two 12-processor Amd Opteron 6168 with 2 cores (2×24 cores).

4.1. Topology and Energy Configuration



(a) 12-ary Tree (b) Ternary tree (c) Binary tree

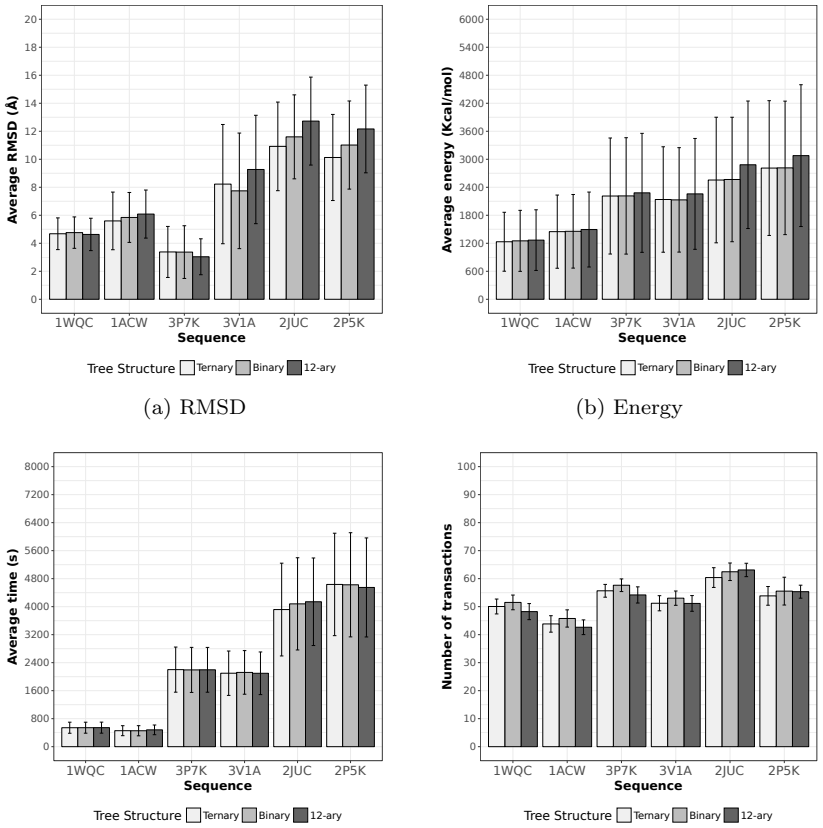
Figure 3: Tree structures evaluated in the tests. All the structures had 13 agents, which are distributed in one, two and three levels on the tree. The binary tree was left incomplete to keep the same number of agents among the structures.

The first tests were designed to evaluate the structure of agents and the combination of energy functions terms to determine the best configuration for the next experiments. We evaluate three hierarchical structures (Figure 3) to distribute the population of agents: a two-level ternary tree, an incomplete three-level binary tree and a 12-ary tree. All of them were adapted to have 13 agents to have comparable results.

We use the sequences marked with * in Table 1 to test the tree structures and energy combinations. For each sequence, we execute ten runs using each one of the three topologies and one of the eleven tested α to combine the energy functions, which gives a total of 1980 experiments. For *short* sequences we use 1×10^6 evaluations of the energy function as a stop criteria, 2×10^6 for *medium* sequences and 3×10^6 for *long* sequences.

4.2. Tree structures assessment

Figure 4 shows a comparison between the results obtained by the three population structures evaluated regarding quality of the predicted structures (`RMSD` (CA atoms), free energy) and performance of the execution (total time, number of transactions). We consider the mean of the results from each energy configuration tested using each structure. As can be seen, the quality of the results are similar between different structures. For both `RMSD` and free energy, the average and standard deviation remain in similar ranges for the structures evaluated. It reflects that there is no a great impact on the results obtained for different distributions of the same number of agents. Similarly occurs for



(a) RMSD (b) Energy

(c) Time (d) Transactions

Figure 4: Results of the experiments using different hierarchical structures. For each peptide and tree structure we show the average of the results from all the energy configurations defined by $\alpha \in \{0.0; 0.1; ...; 0.9; 1.0\}$. We show the distribution for the RMSD (CA atoms) vs experimental structures (a), the free energy of the predicted structures (b), the total time of the experiments (c) and average number of transactions between the 13 agents (d).

the performance of the executions, where the runtime and the number of transactions also do not seem to be profoundly affected by the tree structure used in the experiment. Extended information about the results can be checked in Table 1 and 2 of Supplementary Material.

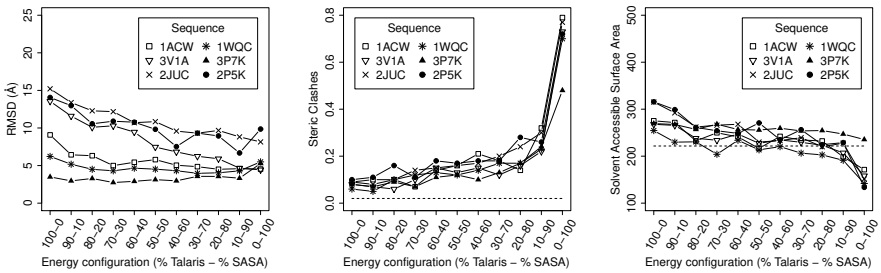
In Table 2 we show the *p-value* obtained from the comparison between the results of the different topologies using the *Wilcoxon signed-rank test*. As it is shown, there are almost no significant differences between the results of the tree structures evaluated. Except for the 12-ary tree that shows evidence of a slightly lower number of transactions, none of the structures prove to be superior respect to the free energy of the solutions, `RMSD` (CA atoms) calculation with experimental structures or executions times. We decided to use the two-level

Sequence	Comparison	p-value				
		RMSD	Energy	Time	Generations	Transactions
1WQC	Ter-Bin	0.478	0.847	0.797	0.818	0.148
	Ter-12ary	0.898	0.748	0.748	0.450	**0.047**
	Bin-12ary	0.606	0.748	0.606	0.792	**0.006**
1ACW	Ter-Bin	0.365	1.000	0.949	0.818	0.101
	Ter-12ary	0.171	0.797	0.116	0.921	0.401
	Bin-12ary	0.606	0.797	0.0879	0.742	**0.008**
3P7K	Ter-Bin	0.562	0.949	0.847	0.645	**0.040**
	Ter-12ary	0.101	0.748	0.847	0.669	0.139
	Bin-12ary	0.401	0.748	0.847	0.293	**0.004**
3V1A	Ter-Bin	0.797	0.898	0.949	1.000	0.076
	Ter-12ary	0.365	0.748	0.699	0.693	1.000
	Bin-12ary	0.193	0.699	0.606	0.767	0.076
2JUC	Ter-Bin	0.438	0.949	0.193	0.250	0.088
	Ter-12ary	0.0879	0.606	**0.040**	0.131	**0.013**
	Bin-12ary	0.193	0.652	0.847	0.038	0.606
2P5K	Ter-Bin	0.300	0.898	0.748	0.532	0.168
	Ter-12ary	**0.0334**	0.652	0.562	0.598	0.133
	Bin-12ary	0.193	0.606	0.898	0.224	0.694

Table 2: Comparison between the results from the three hierarchical structures through the *Wilcoxon signed-rank test*. We show the *p-values* obtained from the test for comparing the differences in RMSD (CA atoms), free energy, runtime, total generations and number of transactions. In **bold** we show the *p-values* smaller than 0.05.

ternary tree for the next set of testing.

4.3. Energy Function Combination



(a) RMSD (b) Clashes (c) SASA

Figure 5: Results of the experiments for different combinations of weights for Talaris and SASA functions using a ternary tree structure. For each target peptide and energy configuration we show the best RMSD obtained, and the steric clashes and solvent accessible surface area (SASA) of the best solution, calculated using *Gaia Web Tool*.

Figure 5 shows a comparison between the results from the 11 weighted combinations of the `Talaris` and `SASA` energy functions using a ternary tree. Figure 5a shows the `RMSD` values of the best solution found by each energy configuration for the six sequences tested, while Figures 5b and 5c show the packing quality of the best solutions from the test of *Gaia Web Tool* [25]. The last analysis also extends to the binary and 12-ary trees (see Figures 1 and 2 from the Supplementary Material).

In Figure 5a shows that by increasing the weighting of the `SASA` function over Talaris function, the solutions tend to be slightly better regarding `RMSD` values. It can be explained by the packing properties of the `SASA` function, leading the structure to a form in which fewer atoms are exposed to the solvent environment. However, solutions also start to show a high amount of steric clashes in their structures as shown in Figure 5b. Talaris function seeks for structures with low levels of energy among all their atom interactions. It means that if Talaris weight is lower than `SASA` weight, the algorithm will give priority to compact solutions even if these are highly unstable. Structures in nature rarely have clashes in their structure, therefore it is important to maintain this value as lower as possible to generate biologically significant solutions.

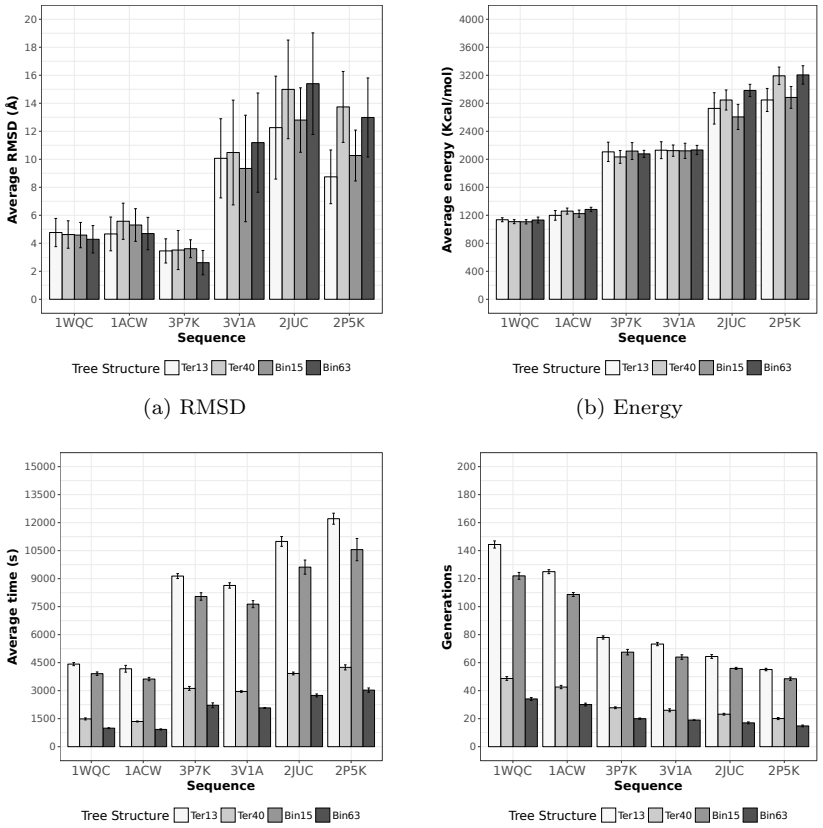
An extended analysis of the combination of the energy terms in a ternary tree structure can be reviewed in [12]. Based on the above, we decided to use for the next test a value of $\alpha = 0.4$, which means that the fitness function will be made of 60% of Talaris energy function and 40% of `SASA`.

4.4. Scalability

We perform scalability tests to evaluate the performance of the distributed implementation of the proposed memetic algorithm. These tests aim to assess the performance of the execution and quality of the solutions when a structure with a high number of agents is used. We evaluate two hierarchical structures, a binary and a ternary tree. The ternary tree was tested using two and three levels, for a total of 13 and 40 agents, while the binary tree was tested using three and five levels. We execute 10 runs for each one of these structures using the same testing environment: 20 peptides from Table 1, $\alpha = 0.4$ to weighing the energy functions based on equation 1 and a stop criteria of 5×10^6 evaluations of the energy function among all the agents of the structure. We expect that, since all algorithms evaluate the same number of solutions throughout the execution, the results obtained should be similar, but the execution times should be reduced proportionally to the number of agents used by each structure.

Figure 6 shows the average results for 6 of the 20 sequences tested using each one of the structures evaluated. The following conclusion also applies to the others sequences evaluated in the scalability tests. Tables 3 and 4 of Supplementary Material have detailed information of all the peptide target sequences evaluated in these tests.

As we can observe in Figure 6c, the execution time drops almost proportionally to the number of agents used. The three-levels ternary tree, using around three times as many agents, reached the 5×10^6 evaluations of the energy function in about a third of the time used for the two-levels ternary tree, as same



(a) RMSD (b) Energy

(c) Time (d) Generations

Figure 6: Results of the scalability tests using a ternary and binary trees. The ternary tree was tested using two (13 agents) and three (40 agents) levels and the binary tree using three (15 agents) and five (63 agents). For each peptide and tree structure we show the average results of 10 experiments in terms of `RMSD` vs experimental structures (a), free energy of the predicted structures (b), total time of the experiments (c) and average generations executed (d).

as the five-levels binary tree, which with 4 times more agents reached the end criteria in about a quarter of the time used for the three-level binary tree. These results apply to all the sequences, being slightly better in the shorter sequences. On the other hand, the number of generations reached for the structures was reduced proportionally to the increase in the number of agents. This outcome is expected, as structures with more agents perform a higher number of evaluations of the energy function in each generation.

Regarding the quality of the results, Figures 6a and 6b show that the solutions predicted by all the structures remain in similar ranges, as it would be expected for using the same stopping criteria. It can be seen that for *short* and

medium sequences the differences between all the structures is small, in particular regarding free energy. For *long* sequences the solutions prove to be slightly better for both RMSD and free energy in the structures with a fewer number of agents. However, this difference could be more associated with the efficiency of the smaller trees in the convergence to better solutions, than the efficiency of the distributed implementation. This explains why the difference is reduced in smaller sequences, where all the trees have enough evaluations of the energy function to reach optimal values. Even considering the latter, the distributed implementation shows to be highly scalable, which allows it to reduce the execution times almost proportionally to the number of agents used and to maintain to a large degree the effectiveness of the algorithm.

4.5. Local Search and Diversity Assessment

In Section 3.7 we proposed several modifications of the local search algorithm to improve its efficiency and the quality of the predicted solutions. Two different approaches were proposed to calculate the diversity of the solutions generated throughout the algorithm. We tested all these approaches to determine the best implementation of the DMA-3DPSP. Our goal is to verify which are the best local search strategies to be employed in 3D protein structure prediction methods based on memetic algorithms. We seek to keep the diversity of the population as a way to better explore the protein conformational search space. The comparison is made in terms of solution quality and execution time. It is important to mention that we are identifying the algorithm configuration that is able to find the best solution using a fixed number of evaluations. In order to deal with the problem, larger runs will be later performed and compared with the state of the art algorithms.

For the testing environment we use the six target sequences marked with * in Table 1, a two-level ternary tree structure with 13 agents (Figure 3b) and an energy fitness function (Equation 1) with $\alpha = 0.4$ for weighting 60% of Talaris function and 40% of SASA function. Each experiment was executed ten times using an end criteria of 1×10^6 evaluations of the energy function.

The first set of tests compare the original local search algorithm with the first three modifications proposed in Section 3.7. Unlike LS4, which redefines the simulating annealing used in the local search, the first three approaches apply specific changes in the local search algorithm that could generate improvements in its results. Table 3 shows the results of the tests for the original local search and the first three new approaches.

From the results, all the proposed modifications shown slightly improve the energy convergence, especially for *long* target sequences (2JUC and 2P5K). The improvement varies between the different sequences, and none of the three implementations show to be particularly superior to the others. Regarding RMSD, in four of the six sequences tested (1WQC, 3V1A, 2JUC and 2P5K), all the new implementations reduced the average RMSD values of the original local search. The target sequence 3P7K was the only exception where the original local search shows better solutions regarding RMSD.

Sequence	Structure	Quality of solutions		Execution	
		RMSD (Å)	Energy (Kcal/mol)	Total time (s)	Generations
1WQC	LS-Original	4.91 (±1.14)	1150.7 (±48.2)	956.9 (±19.0)	38.8 (±1.0)
	LS1	4.75 (±1.03)	1159.0 (±48.1)	993.7 (±19.2)	75.9 (±1.5)
	LS2	4.79 (±0.84)	1131.6 (±41.1)	1055.6 (±21.8)	156.6 (±2.5)
	LS3	4.38 (±0.90)	1108.4 (±31.8)	975.8 (±20.6)	20.4 (±0.5)
1ACW	LS-Original	5.23 (±1.94)	1298.3 (±76.8)	871.0 (±31.7)	34.5 (±0.7)
	LS1	4.81 (±1.45)	1260.7 (±57.5)	925.0 (±19.5)	68.3 (±1.3)
	LS2	5.31 (±1.31)	1251.4 (±36.9)	995.3 (±20.5)	138.2 (±3.7)
	LS3	4.76 (±1.46)	1297.9 (±42.9)	887.4 (±27.2)	18.0 (±0.5)
3P7K	LS-Original	2.83 (±1.16)	2173.8 (±125.7)	2001.1 (±51.9)	22.0 (±0.5)
	LS1	2.88 (±1.11)	2147.3 (±170.6)	1981.9 (±23.6)	41.5 (±0.8)
	LS2	3.54 (±0.98)	2096.8 (±125.9)	2063.0 (±30.8)	84.5 (±1.8)
	LS3	3.18 (±1.39)	2053.7 (±75.9)	2076.5 (±61.8)	12.1 (±0.3)
3V1A	LS-Original	12.62 (±3.11)	2292.5 (±110.5)	1909.5 (±75.8)	21.3 (±0.7)
	LS1	8.53 (±5.25)	2144.4 (±188.4)	1890.7 (±46.5)	39.5 (±0.8)
	LS2	10.36 (±3.66)	2118.8 (±159.8)	1957.3 (±23.5)	78.3 (±1.3)
	LS3	10.32 (±4.00)	2107.1 (±85.2)	2013.2 (±81.0)	11.6 (±0.5)
2JUC	LS-Original	15.05 (±2.18)	3260.8 (±169.1)	2494.8 (±89.3)	19.2 (±0.8)
	LS1	13.26 (±3.40)	2805.1 (±188.2)	2419.8 (±58.2)	35.3 (±0.9)
	LS2	11.30 (±2.53)	2755.9 (±233.7)	2491.6 (±75.9)	69.4 (±1.4)
	LS3	12.36 (±2.00)	2671.1 (±78.7)	2629.3 (±93.3)	10.6 (±0.5)
2P5K	LS-Original	13.49 (±4.33)	3497.6 (±320.7)	2754.3 (±92.7)	16.8 (±0.6)
	LS1	12.15 (±2.24)	3084.6 (±113.8)	2595.4 (±53.9)	30.1 (±0.7)
	LS2	8.96 (±2.33)	3005.9 (±227.4)	2713.4 (±48.4)	60.6 (±1.3)
	LS3	12.01 (±2.28)	3078.2 (±108.4)	2813.0 (±147.6)	9.3 (±0.5)

Table 3: Results of the experiments for different local search approaches in six peptide target sequences. LS-Original uses the original local search implementation described in Section 3.5, while LS1, LS2 and LS3 apply the modifications in the local search described in Section 3.7. For each amino acid and local search implementation, we show the average results of 10 experiments in terms of quality of the solutions (RMSD and free energy) and performance of the execution (runtime and generations performed).

Regarding the execution performance, the new implementations do not seem to generate great differences in the runtimes. Only for *short* target peptide sequences, we can see a slight increase in the total time by the LS1 and LS2 implementations. On the other hand, the number of generations reached by the algorithm changes with the modifications in the local search, where LS1 and especially LS2 perform a larger number of generations and LS3 considerably less. These results are justified by each implementation. LS1 favours better solutions in the neighbourhood changes, which naturally reduces the number of iterations of simulated annealing by having more optimal solutions. LS2 combines the small modifications on angles phi and psi in one cycle of iterations, so the simulated annealing naturally consumes less time and allows to reach a larger number of generations. LS3 improves the search range of the angles, which

generates a more efficient simulated annealing that performs more number of iterations per generation.

Sequence	Structure	Quality of solutions		Execution	
		RMSD (Å)	Energy (Kcal/mol)	Total time (s)	Generations
1WQC	Div1	3.83 (±1.15)	1060.5 (±34.6)	1004.8 (±17.2)	26.6 (±0.5)
	Div2-2.0	3.86 (±0.75)	1061.0 (±19.3)	1070.4 (±27.5)	23.6 (±0.7)
	Div2-4.0	4.09 (±1.04)	1065.0 (±42.7)	1037.4 (±28.5)	25.1 (±0.9
1ACW	Div1	4.68 (±1.54)	1161.2 (±47.2)	934.6 (±26.0)	23.8 (±0.6)
	Div2-2.0	5.04 (±0.91)	1211.5 (±43.1)	961.8 (±30.5)	20.4 (±0.5)
	Div2-4.0	4.26 (±1.64)	1206.1 (±58.2)	971.0 (±28.4)	21.2 (±0.6)
3P7K	Div1	2.99 (±1.33)	2009.2 (±71.9)	2048.6 (±50.5)	19.0 (±0.5)
	Div2-2.0	3.04 (±0.84)	1988.6 (±52.5)	2171.7 (±92.3)	17.8 (±0.9)
	Div2-4.0	3.86 (±1.36)	2038.7 (±141.7)	2064.1 (±51.0)	18.6 (±0.5)
3V1A	Div1	9.21 (±3.66)	2080.6 (±69.4)	1916.0 (±52.7)	17.6 (±0.5)
	Div2-2.0	8.18 (±4.48)	2059.2 (±113.0)	2029.5 (±36.5)	16.0 (±0.0)
	Div2-4.0	8.84 (±4.26)	2035.3 (±86.1)	2063.0 (±73.4)	17.0 (±0.7)
2JUC	Div1	13.48 (±2.54)	2642.1 (±87.9)	2503.4 (±96.1)	16.0 (±0.5)
	Div2-2.0	14.42 (±2.55)	2762.4 (±143.0)	2776.3 (±79.1)	14.9 (±0.3)
	Div2-4.0	12.71 (±2.54)	2737.1 (±115.2)	2787.6 (±83.9)	15.1 (±0.6)
2P5K	Div1	13.29 (±3.95)	2929.8 (±151.5)	2756.5 (±87.7)	14.6 (±0.5)
	Div2-2.0	12.88 (±2.96)	3039.4 (±106.3)	2967.2 (±164.6)	13.8 (±0.6)
	Div2-4.0	13.74 (±1.29)	3041.9 (±134.5)	2938.6 (±174.2)	13.6 (±0.5)

Table 4: Results of the experiments for different diversity approaches in six target sequences. Div1 and Div2 follow the diversity implementations described in Section 3.8. Div2 was tested using a RMSD cutoff of 2.0Å and 4.0Å. For each amino acid and diversity implementation, we show the average results of 10 experiments in terms of quality of the solutions (RMSD and free energy) and performance of the execution (runtime and generations performed). For the RMSD and GDT measurements we show in bold the best results between the two implementations

We also tested the two diversity approaches proposed in Section 3.8: Div1 that compares the average difference of the angles of two solutions, and Div2 that calculates the RMSD between their structures. We tested Div2 using 2.0 and 4.0 as RMSD cutoff to determine if a solution is diverse, while Div1 limit was calculated based on the size of the peptide (six times). Table 4 shows the results of the tests using the two diversity implementations. Both, the quality of the solutions and the execution performance reached similar ranges and do not show significant differences between the three implementations evaluated.

The first group of experiments shows that all of the local search approaches evaluated improve the predicted solutions. These three approaches are not mutually exclusive, therefore, for the last set of tests, we compare a new implementation of the local search that combines the approaches of LS1, LS2, and LS3 with the original local search implementation and with LS4, the last approach proposed in Section 3.7. LS4 defines a new local search algorithm, combining the change of neighborhood with the simulating annealing process, therefore approaches as LS1 does not work in this implementation.

Sequence	Structure	Quality of solutions		Execution	
		RMSD (Å)	Energy (Kcal/mol)	Total time (s)	Generations
1WQC	LS-Original	4.33 (±1.10)	1180.2 (±59.1)	1040.3 (±26.1)	23.1 (±0.3)
	LS1-2-3	3.86 (±0.75)	1061.0 (±19.3)	1070.4 (±27.5)	23.6 (±0.7)
	LS4	3.55 (±0.53)	1009.6 (±26.4)	1106.5 (±25.1)	25.4 (±0.5)
1ACW	LS-Original	5.70 (±0.61)	1348.3 (±58.2)	966.7 (±27.5)	20.7 (±0.5)
	LS1-2-3	5.04 (±0.91)	1211.5 (±43.1)	961.8 (±30.5)	20.4 (±0.5)
	LS4	4.43 (±1.17)	1102.3 (±42.7)	1065.6 (±28.2)	22.6 (±0.5)
3P7K	LS-Original	2.70 (±1.06)	2302.3 (±166.9)	2176.5 (±79.2)	18.0 (±0.7)
	LS1-2-3	3.04 (±0.84)	1988.6 (±52.5)	2171.7 (±92.3)	17.8 (±0.9)
	LS4	2.96 (±1.00)	1879.9 (±36.3)	2187.8 (±59.6)	19.7 (±0.7)
3V1A	LS-Original	11.73 (±1.97)	2475.5 (±225.6)	2078.0 (±63.0)	16.4 (±0.5)
	LS1-2-3	8.18 (±4.48)	2059.2 (±113.0)	2029.5 (±36.5)	16.0 (±0.0)
	LS4	4.88 (±2.16)	1829.2 (±47.8)	2102.0 (±78.1)	17.7 (±0.5)
2JUC	LS-Original	13.08 (±2.23)	3621.7 (±422.0)	2851.2 (±132.2)	15.5 (±0.7)
	LS1-2-3	14.42 (±2.55)	2762.4 (±143.0)	2776.3 (±79.1)	14.9 (±0.3)
	LS4	9.94 (±1.82)	2214.1 (±65.8)	2655.9 (±77.5)	15.9 (±0.6)
2P5K	LS-Original	13.22 (±2.94)	3624.9 (±305.9)	2987.5 (±72.8)	13.9 (±0.3)
	LS1-2-3	12.88 (±2.96)	3039.4 (±106.3)	2967.2 (±164.6)	13.8 (±0.6)
	LS4	9.73 (±1.98)	2492.8 (±147.9)	2966.3 (±80.3)	14.8 (±0.4)

Table 5: Results of the experiments for different local search approaches in six peptides. LS-Original uses the original local search implementation described in Section 3.5. LS1-2-3 apply the modifications LS1, LS2 and LS3 described in Section 3.7 at the same time in the local search. LS4 use the last local search implementation described in Section 3.7. For each amino acid and local search implementation, we show the average results of 10 experiments in terms of quality of the solutions (RMSD and free energy) and performance of the execution (runtime and generations performed).

Table 5 shows the results of the three local search implementations tested. LS1-2-3 and LS4 reached energy values significantly better than the original local search implementation. Regarding RMSD, LS1-2-3 obtained better solutions than the original implementation in four of the six target peptide sequences (1WQC, 1ACW, 3V1A, and 2P5K), while LS4 got better solutions than the other two implementations in all the peptides tested, except for the sequence 3P7K where the original algorithm was slightly better. Again, the case for the 3P7K sequence can be explained by an imprecision in the energy function, since the energy values of LS1-2-3 and LS4 were lower than those obtained by the original implementation. The performance of the execution remained in similar ranges for all implementations; therefore there does not seem to be a drawback in using the newly proposed implementations.

4.6. Best implementation of the DMA-3DPSP

Using the above results, we measure the performance and effectiveness of the best performing configuration: LS4 local search, and Div2 with a cutoff of 2.0.

Table 6 shows the results of the last group of experiments. The DMA-Original corresponds to the original simulating annealing algorithm described in Section 3.5 using Div1 as diversity strategy in updating pockets. DMA-Final uses the latest local search implementation, the LS4 strategy updated with LS2 and LS3 approaches in its second phase and it uses Div2 for diversity calculations. The different methods were tested using the two-level ternary tree topology for agents distribution, an energy fitness function weighting 60% of Talaris and 40% of SASA, and 5×10^6 evaluations of the energy functions as the stop criteria.

Results in Table 6 were assessed in terms of structural quality of the solutions by comparing the three-dimensional structures with the experimental ones using two measurements: RMSD and the *Global Distance Test* total score (GDT_TS [48]), metric that gives a more accurate comparison by measuring the percentage of atoms of the structures within the distance cutoff using different superpositions. Equation 6 describes the calculation of the GDT total score. Small values of $RMSD$ and large values of GDT means good quality solutions.

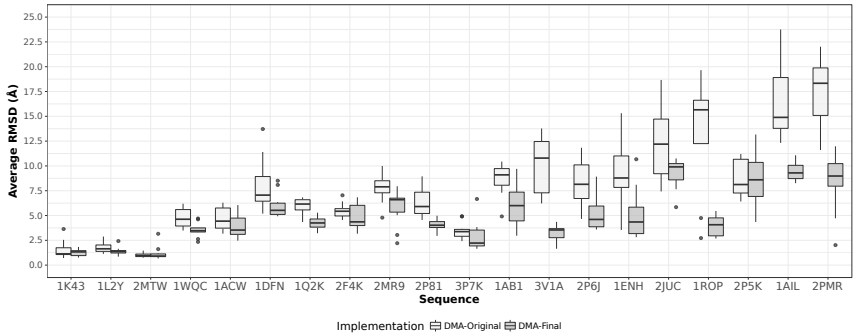
$$GDT_{TS} = \frac{GDT_{P1} + GDT_{P2} + GDT_{P4} + GDT_{P8}}{4} \tag{6}$$

where $GDT_{Pn)}$ is the percentage of the residues under the cutoff $\leq n$Å. We also show the average values of free energy, total execution time and number of generations reached by each implementation.

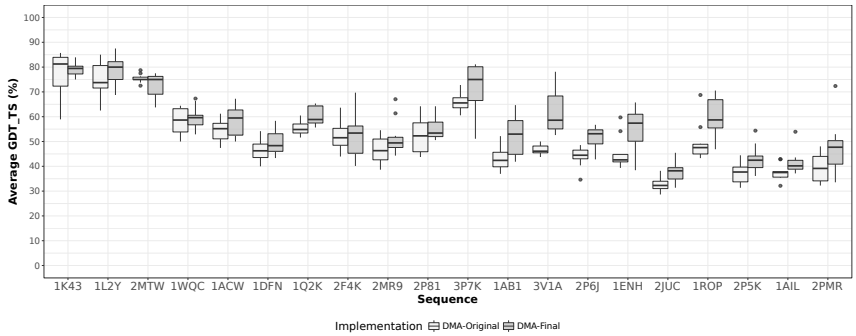
Figure 7 shows a comparison between the distribution of the results from both implementations regarding RMSD and GDT. We observed that the solutions predicted by the DMA-Final show improvements in both metrics over the original implementation. In 18 out of 20 peptides evaluated the DMA-Final solutions obtained lower average RMSD values, especially for *long* sequences. Also, in 19 out of 20 target peptides, the best solutions found were predicted by the DMA-Final implementation. The same applies concerning the GDT_TS percentages. In 18 out of 20 peptides the average GDT_TS of the DMA-Final solutions were higher than the ones predicted by DMA-Original, and in 17 out of 20 of these sequences the best solutions in terms of GDT_TS were found by DMA-Final.

4.7. Biological Significance Assessment

Even with good RMSD and GDT scores, the solutions predicted by the implementations must prove to have biological significance. We examine the tertiary structure of the best solutions found by both implementations using PyMOL [40]. Cartoon representation of the solutions with the lowest RMSD obtained by each DMA-3DPSP implementation are shown in Figure 8. We observed that the α-helix secondary structures are formed and positioned correctly in almost all structures. The β-sheet secondary structures, on the other hand, although well positioned prove to be difficult to form correctly in several of the predicted structures (1ACW, 1DFN, 1Q2K and 2P5K are good examples). This can be explained because the search space of these structures tends to overlap much more with one of turns and coils than the α-helix structures, which makes it much harder to predict even with experimental knowledge.



(a) `RMSD`

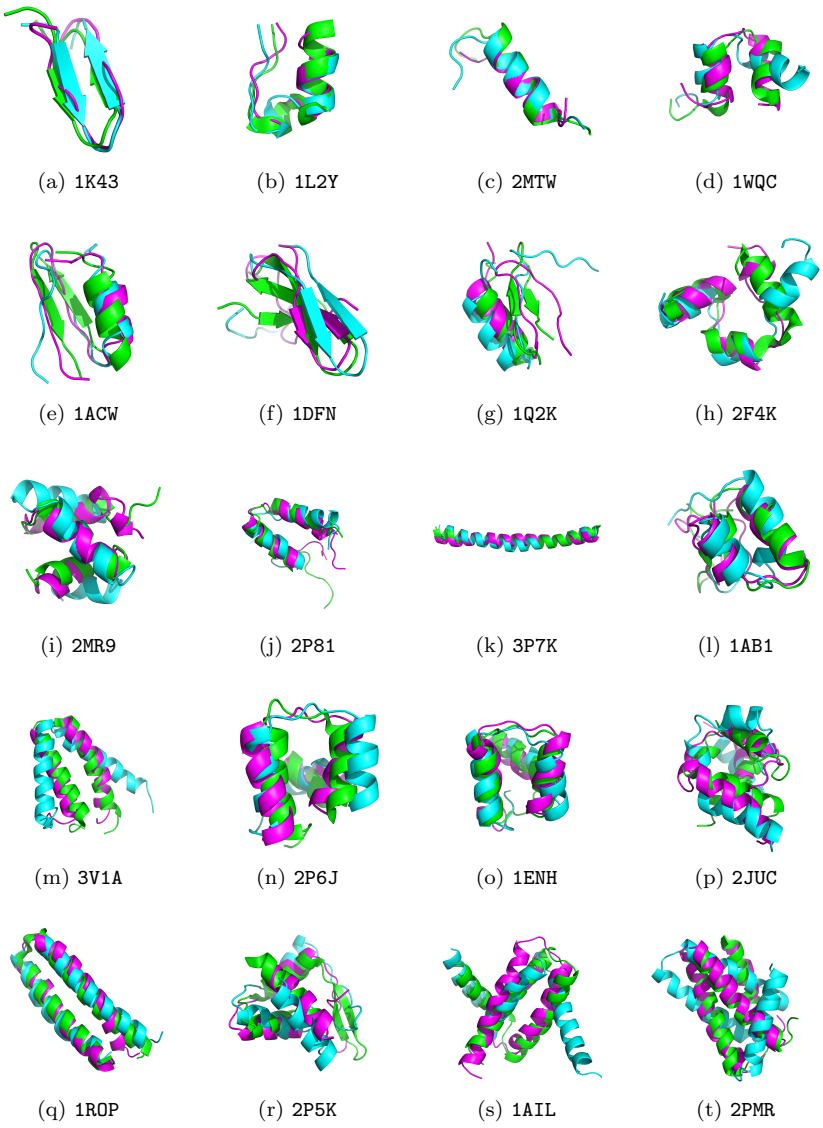


(b) `GDT_TS`

Figure 7: Comparison of the quality of the solutions from the original and the latest implementation. For each peptide we show the distribution of the results in terms of `RMSD` values and `GDT` percentage vs experimental structures.

We also performed quality assessment of the best solutions regarding `RMSD` using *Gaia Web Tool* [25]. *Gaia* measures the packing quality of a given protein structure using several metrics, between them, the number of steric clashes, the percentage of unsatisfied hydrogen bonds, total void volume and the scaling of the accessible surface area with protein length.

Table 7 shows the results of the *Gaia* reports for the predicted structures with the lowest `RMSD` of both `DMA` implementations. It can be observed that, in all the peptide sequences evaluated in the tests, the `DMA-Final` implementation reduced both the number of steric clashes and the area exposed to the solvent of the predicted structures over the original `DMA` implementation. It proves that the solutions of the new implementation not only have better `RMSD` and `GDT` but also their structures are more biologically significant. There is no much impact in the percentage of unsatisfied hydrogen bonds, and these remain in good margin for both implementations, but there was an excess of void volume in several of the structures predicted by the `DMA-Final` implementation. It is also



(a) 1K43 (b) 1L2Y (c) 2MTW (d) 1WQC

(e) 1ACW (f) 1DFN (g) 1Q2K (h) 2F4K

(i) 2MR9 (j) 2P81 (k) 3P7K (l) 1AB1

(m) 3V1A (n) 2P6J (o) 1ENH (p) 2JUC

(q) 1ROP (r) 2P5K (s) 1AIL (t) 2PMR

Figure 8: Cartoon representation of the best solution found for each peptide tested in the final experiments. We show in green the experimental protein, in cyan the structure predicted by the original DMA-3DPSP and in magenta the structure predicted by the final update of the DMA-3DPSP. The C_α atoms of the experimental and predicted structures are fitted.

noticeable that the results of both implementations show a much higher level of steric clashes than the target expected by Gaia. In experimental structures, the number of clashes rarely exceeds 1% of total contacts, so there is still room

to improve the results of the DMA-3DPSP.

4.8. Comparison with State of the Art Methods

We also test the best approach using peptides that have been tested in the CASP-2016 experiments [31] and compare our results with some of the "reference methods" in the PSP area. We chose five of the protein targets from CASP 11 experiments [24] with sizes from 70 to 128 residues and different folding patterns. Since the sequences evaluated are larger and structurally more complex than our previous case studies, we set 1×10^7 evaluations of the energy functions as the stopping criteria of the DMA-3DPSP. We performed 10 runs of the algorithm using a two-levels ternary tree structure to distribute the agents and an energy configuration weighting 60% of Talaris function and 40% fo SASA function.

Table 9 shows the results of the DMA-3DPSP for the five protein targets from CASP 11 experiments. We show the results obtained by two of the methods with the best performance over the last CASP experiments, Zhang-Server [46], which combine the *ab initio* strategy of QUARK [45] and the threading modeling of I-TASSER [47], and Robetta [23], the Rosetta PSP web server of Baker Labs.

We see that the results of the DMA-3DPSP are still far from the top methods of the last CASP experiments in both RMSD and GDT values. Even the best solutions found in all the sequences are still not close to the Zhang-Server and Robetta results. The proposed strategy requires a large amount of time and resource for the predictions of sequences over 100 residues, therefore the 1×10^7 evaluations of the energy function proves to be insufficient to predict the sequences selected in the tests. Further testing is still needed to determine the maximum scope of the proposed strategy.

5. Conclusion

In this work, we presented a Distributed Memetic Algorithm (DMA-3DPSP) to deal with the Tertiary Protein Structure Prediction Problem (3D-PSP problem). The 3D-PSP problem is one of the hardest problem to solve and it requires the developed of smart and efficient techniques. In this work we evaluated different configurations of a Distributed Memetic Algorithm in order to measure which one shows a better contribution to improve the quality of solutions.

The algorithm uses an asynchronous hierarchical population of agents that shares the solution's information along with the execution of the algorithm. We also used a combination of well-known energy scoring function as a fitness value for the optimisation process. The DMA-3DPSP algorithm uses information extracted from the PDB to guide the search process. We evaluated three different hierarchical population structures and several alternatives to local search and population operators.

We perform a thorough testing process in order to correctly identify the best performing configuration in terms of computational results and biological significance. All the approaches developed were tested in order to elucidate different ways to improve the current protein structure prediction methods based

on memetic algorithms. We investigated different strategies to be considered in the design of prediction methods for the 3D structure of proteins, showing which mechanisms bring advantages in terms of computational time and biological significance. Results show that the selection of the combination of energies allows the generation of solutions that have a good trade-off between similarity with the experimental protein and biological significance. The use of a distributed strategy improved the time of the solutions as expected, and they show slights improvements in terms of solution quality. In our experiments, communications strategies, and solution sharing in the distributed algorithm, did not show a significant improvement effect, so this is an area that needs to be further explored.

We were able to identify a configuration for the `DMA-3DPSP` that can find solutions with good computational quality, and also with good biological significance, which is a must in bioinformatics. Further work will be focused on the improvement of the implementation to reach similar quality solutions on larger sequences, and also in the incorporation of new machine learning techniques to extract new information from the PDB. Also, the incorporation of different local search strategies that can be combined during the evolution process, in order to exploit better the search space.

Acknowledgments

This work was partially funded by grants `STIC-AMSUD` (88887.135130 /2017-01 CAPES-Brazil and 17-STIC-05 Conicyt-Chile), `FAPERGS` (16/2551-0000520-6; 19/2551-0001906-8), `MCT/CNPq` (311611/2018-4), and `DICYT-VRIDEI USACH` (061919IP).

References

[1] R. M. Abaskharon and F. Gai. Meandering down the energy landscape of protein folding: Are we there yet? *Biophysical Journal*, 110(9):1924 – 1932, 2016.

[2] Nickolai Alexandrov and Ilya Shindyalov. PDP: protein domain parser. *Bioinformatics*, 19(3):429–430, 02 2003.

[3] M. Blocho and Z.J. Czech. A parallel memetic algorithm for the vehicle routing problem with time windows. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 144–151, Oct 2013.

[4] B Borguesan, M.B. e Silva, B. Grisci, M. Inostroza-Ponta, and M. Dorn. Apl: An angle probability list to improve knowledge-based metaheuristics for the three-dimensional protein structure prediction. *Comput. Biol. Chem.*, 59:142–157, 2015.

[5] B. Borguesan, M. Inostroza-Ponta, and M. Dorn. NIAS-Server: Neighbors influence of amino acids and secondary structures in proteins. *Journal of Computational Biology*, 24(3):255–265, 2017.

[6] Joseph D Bryngelson, Jose Nelson Onuchic, Nicholas D Socci, and Peter G Wolynes. Funnels, pathways, and the energy landscape of protein folding: a synthesis. *Proteins: Struct., Funct., Bioinf.*, 21(3):167–195, 1995.

[7] David Case, Thomas Cheatham, Tom Darden, Holger Gohlke, Ray Luo, Kenneth Merz, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert Woods. The amber biomolecular simulation programs. *Journal of computational chemistry*, 26:1668–88, 12 2005.

[8] Leonardo de Lima Correa, Bruno Borguesan, Camilo Farfán, Mario Inostroza-Ponta, and Márcio Dorn. A memetic algorithm for 3d protein structure prediction problem. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 15(3):690–704, 2018.

[9] K. A. Dill, S. B. Ozkan, T. R. Weikl, J. D. Chodera, and V. A. Voelz. The protein folding problem: when will it be solved? *Current Opinion in Structural Biology*, 17 3:342–6, 2007.

[10] M Dorn, MB e Silva, L Buriol, and L Lamb. Three-dimensional protein structure prediction: methods and computational strategies. *Comput. Biol. Chem.*, 53:251–276, 2014.

[11] Márcio Dorn and Osmar Norberto de Souza. Cref: A central-residue-fragment-based method for predicting approximate 3-d polypeptides structures. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 1261–1267, New York, NY, USA, 2008. ACM.

[12] I. Escobar, N. Hidalgo, M. Inostroza-Ponta, M. Marín, E. Rosas, and M. Dorn. Evaluation of a combined energy fitness function for a distributed memetic algorithm to tackle the 3d protein structure prediction problem. In *2016 35th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–10, Oct 2016.

[13] Narayanan Eswar, David Eramian, Ben Webb, Min-Yi Shen, and Andrej Sali. *Protein Structure Modeling with MODELLER*, pages 145–159. Humana Press, Totowa, NJ, 2008.

[14] Antonio J. Fernández and Álvaro Gutiérrez-Rodríguez. On distributed user-centric memetic algorithms. *Soft Comput.*, 23(12):4019–4039, 2019.

[15] Rasmus Fonseca, Martin Paluszewski, and Pawel Winter. Protein structure prediction using bee colony optimization metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 9(2):181–194, 2010.

[16] Nick Gibbs, Anthony R. Clarke, and Richard B. Sessions. Ab initio protein structure prediction using physicochemical potentials and a simplified off-lattice model. *Proteins: Structure, Function, and Bioinformatics*, 43(2):186–202, 2001.

[17] Pawel Gniewek, Andrzej Kolinski, Andrzej Kloczkowski, and Dominik Gront. Bioshell-threading: versatile monte carlo package for protein 3d threading. *BMC bioinformatics*, 15(1):22, 2014.

[18] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286 – 300, 2015.

[19] M. Harris, R. Berretta, M. Inostroza-Ponta, and P. Moscato. A memetic algorithm for the quadratic assignment problem with parallel local search. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 838–845, 2015.

[20] Mario Inostroza-Ponta, Camilo Farfán, and Marcio Dorn. A memetic algorithm for protein structure prediction based on conformational preferences of aminoacid residues. In *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, pages 1403–1404, 2015.

[21] Nanda Dulal Jana, Swagatam Das, and Jaya Sil. *Backgrounds on Protein Structure Prediction and Metaheuristics*, pages 1–28. Springer International Publishing, Cham, 2018.

[22] Morten Källberg, Gohar Margaryan, Sheng Wang, Jianzhu Ma, and Jinbo Xu. Raptorx server: a resource for template-based protein structure modeling. In *Protein Structure Prediction*, pages 17–27. Springer, 2014.

[23] David E Kim, Dylan Chivian, and David Baker. Protein structure prediction and analysis using the robetta server. *Nucleic acids research*, 32(suppl 2):W526–W531, 2004.

[24] Lisa N Kinch, Wenlin Li, R Dustin Schaeffer, Roland L Dunbrack, Bohdan Monastyrskyy, Andriy Kryshtafovych, and Nick V Grishin. Casp 11 target classification. *Proteins: Structure, Function, and Bioinformatics*, 2016.

[25] Pradeep Kota, Feng Ding, Srinivas Ramachandran, and Nikolay V. Dokholyan. Gaia: automated quality assessment of protein structure models. *Bioinformatics (Oxford, England)*, 27(16):2209–2215, Aug 2011.

[26] Nuno Leite, Carlos M. Fernandes, Fernando Melicio, and Agostinho C. Rosa. A cellular memetic algorithm for the examination timetabling problem. *Computers & OR*, 94:118–138, 2018.

[27] Dapeng Li, Tonghua Li, Peisheng Cong, Wenwei Xiong, and Jiangming Sun. A novel structural position-specific scoring matrix for the prediction of protein secondary structures. *Bioinformatics*, 28(1):32–39, January 2012.

[28] Cheng-Jian Lin and Shih-Chieh Su. Protein 3d hp model folding simulation using a hybrid of genetic algorithm and particle swarm optimization. *International Journal of Fuzzy Systems*, 13:140–147, 06 2011.

[29] Dimitrios P. Lyras and Dirk Metzler. Reformalign: improved multiple sequence alignments using a profile-based meta-alignment approach. *BMC Bioinformatics*, 15(1):265, 2014.

[30] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, CALTECH, Pasadena, California, USA, 1989.

[31] John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Anna Tramontano. Critical assessment of methods of protein structure prediction: Progress and new directions in round xi. *Proteins: Structure, Function, and Bioinformatics*, 84(S1):4–14, 2016.

[32] Jakub Nalepa and Miroslaw Blocho. Co-operation in the parallel memetic algorithm. *International Journal of Parallel Programming*, 43(5):812–839, 2015.

[33] Jakub Nalepa and Miroslaw Blocho. Adaptive cooperation in parallel memetic algorithms for rich vehicle routing problems. *IJGUC*, 9(2):179–192, 2018.

[34] Quang Huy Nguyen, Yew-Soon Ong, Meng-Hiot Lim, and Natalio Krasnogor. Adaptive cellular memetic algorithms. *Evolutionary Computation*, 17(2):231–256, 2009.

[35] M. A. Rashid, M. A. H. Newton, M. T. Hoque, and A. Sattar. A local search embedded genetic algorithm for simplified protein structure prediction. In *2013 IEEE Congress on Evolutionary Computation*, pages 1091–1098, June 2013.

[36] Timothy J Richmond. Solvent accessible surface area and excluded volume in proteins: Analytical equations for overlapping spheres and implications for the hydrophobic effect. *J. Mol. Biol.*, 178(1):63–89, 1984.

[37] Carol A. Rohl, Charlie E.M. Strauss, Kira M.S. Misura, and David Baker. Protein structure prediction using rosetta. In *Numerical Computer Methods, Part D*, volume 383 of *Methods in Enzymology*, pages 66 – 93. Academic Press, 2004.

[38] Ruslan Sadreyev and Nick Grishin. Compass: a tool for comparison of multiple protein alignments with assessment of statistical significance. *Journal of molecular biology*, 326(1):317–336, 2003.

[39] Claudio Sanhueza, Francia Jimenez, Regina Berretta, and Pablo Moscato. Pasmoqap: A parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*, pages 1103–1110. IEEE, 2017.

[40] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.7, Nov 2015.

[41] N. Siew and D. Fischer. Convergent evolution of protein structure prediction and computer chess tournaments: Casp, kasparov, and cafasp. *IBM Systems Journal*, 40(2):410–425, 2001.

[42] J. E. Smith. *The Co-Evolution of Memetic Algorithms for Protein Structure Prediction*, pages 105–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[43] Johannes Söding, Andreas Biegert, and Andrei N Lupas. The hhpred interactive server for protein homology detection and structure prediction. *Nucleic acids research*, 33(suppl_2):W244–W248, 2005.

[44] Ching-Wai Tan and David Jones. Using neural networks and evolutionary information in decoy discrimination for protein tertiary structure prediction. *BMC bioinformatics*, 9:94, 02 2008.

[45] Dong Xu and Yang Zhang. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins: Structure, Function, and Bioinformatics*, 80(7):1715–1735, 2012.

[46] Wenxuan Zhang, Jianyi Yang, Baoji He, Sara Elizabeth Walker, Hongjiu Zhang, Brandon Govindarajoo, Jouko Virtanen, Zhidong Xue, Hong-Bin Shen, and Yang Zhang. Integration of quark and i-tasser for ab initio protein structure prediction in casp11. *Proteins: Structure, Function, and Bioinformatics*, 2015.

[47] Yang Zhang. I-tasser server for protein 3d structure prediction. *BMC bioinformatics*, 9(1):1, 2008.

[48] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

[49] Wei Zheng, Chengxin Zhang, Eric W. Bell, and Yang Zhang. I-tasser gateway: A protein structure and function prediction server powered by xsede. *Future Generation Computer Systems*, 99:73 – 85, 2019.

Sequence	Structure	RMSD (Å)		GDT_TS (%)		Energy (Kcal/mol)	Total time (s)	Generations
		Best	Average	Best	Average			
1K43	DMA-Original	**0.71**	1.52 (±0.93)	**85.71**	77.32 (±8.83)	655.8 (±35.7)	2554.1 (±77.9)	286.6 (±6.0)
	DMA-Final	0.76	**1.28 (±0.36)**	83.93	**79.11 (±2.92)**	576.9 (±17.3)	2835.6 (±54.3)	374.6 (±9.1)
	Rosetta	*0.64	*0.93 (±0.15)	85.71	*80.54 (±2.58)	116.43 (±73.2)	-	-
1L2Y	DMA-Original	1.14	1.75 (±0.54)	85.00	75.12 (±6.57)	846.2 (±56.6)	3486.4 (±160.2)	191.6 (±3.7)
	DMA-Final	**0.86**	**1.39 (±0.44)**	**87.50**	**78.75 (±6.51)**	709.6 (±27.4)	3833.0 (±51.1)	250.9 (±2.9)
	Rosetta	*0.62	1.48 (±0.45)	*96.25	*80.75 (±7.52)	-26.04 (±5.51)	-	-
2MTW	DMA-Original	0.75	**1.02 (±0.23)**	**78.75**	**75.38 (±1.77)**	914.8 (±35.6)	3813.4 (±78.5)	190.8 (±2.9)
	DMA-Final	**0.67**	1.19 (±0.74)	77.50	72.62 (±4.73)	786.4 (±19.1)	4138.1 (±57.5)	244.7 (±3.3)
	Rosetta	3.9	4.4 (±0.36)	62.5	59.75 (±2.61)	-18.28 (±2.27)	-	-
1WQC	DMA-Original	3.48	4.77 (±1.01)	64.42	58.17 (±5.48)	1136.2 (±28.3)	4423.5 (±81.3)	144.4 (±2.5)
	DMA-Final	**2.34**	**3.54 (±0.74)**	**67.31**	**59.62 (±4.51)**	917.6 (±24.4)	4904.3 (±65.3)	188.7 (±3.7)
	Rosetta	*1.82	*2.19 (±0.26)	*76.92	*72.31 (±2.85)	-27.6 (±6.58)	-	-
1ACW	DMA-Original	3.17	4.67 (±1.21)	61.21	54.40 (±4.31)	1198.7 (±69.1)	4173.0 (±182.1)	125.0 (±1.4)
	DMA-Final	**2.47**	**3.87 (±1.16)**	**67.24**	**58.02 (±6.04)**	993.5 (±28.9)	4696.9 (±94.3)	163.4 (±2.2)
	Rosetta	*1.66	*2.52 (±1.15)	*77.59	*70.95 (±6.94)	-26.0 (±3.62)	-	-
1DFN	DMA-Original	5.19	8.03 (±2.71)	54.17	46.58 (±4.66)	1417.2 (±59.5)	4527.4 (±142.2)	120.2 (±2.1)
	DMA-Final	**4.93**	**6.02 (±1.28)**	**58.33**	**49.92 (±5.31)**	1126.9 (±50.7)	5168.7 (±91.1)	158.7 (±1.5)
	Rosetta	*3.63	*5.36 (±0.79)	49.17	44.67 (±2.67)	79.04 (±233.0)	-	-
1Q2K	DMA-Original	4.34	5.89 (±0.91)	60.48	55.48 (±2.96)	1232.1 (±56.0)	4836.4 (±82.0)	116.7 (±1.8)
	DMA-Final	**3.22**	**4.25 (±0.70)**	**65.32**	**60.40 (±3.87)**	1033.1 (±38.8)	5418.7 (±82.9)	151.7 (±3.3)
	Rosetta	*0.57	*1.98 (±0.97)	*94.35	*78.79 (±10.34)	-30.73 (±5.8)	-	-
2F4K	DMA-Original	4.55	5.48 (±0.78)	63.64	**52.58 (±6.27)**	1323.1 (±67.3)	6272.4 (±117.8)	108.5 (±1.6)
	DMA-Final	**3.17**	**4.90 (±1.33)**	**69.70**	52.12 (±8.87)	1085.6 (±33.7)	6800.5 (±90.6)	142.5 (±2.9)
	Rosetta	5.02	5.79 (±0.43)	-	-	-39.42 (±5.61)	-	-
2MR9	DMA-Original	4.79	7.70 (±1.42)	54.55	46.76 (±5.52)	1738.2 (±109.1)	8075.3 (±243.7)	80.0 (±1.7)
	DMA-Final	**2.21**	**5.80 (±1.83)**	**67.05**	**51.59 (±7.13)**	1342.1 (±59.1)	8690.1 (±217.8)	104.2 (±2.2)
	Rosetta	*1.43	*2.27 (±0.65)	*83.52	*73.13 (±6.23)	-72.25 (±3.59)	-	-
2P81	DMA-Original	4.55	6.26 (±1.46)	**64.20**	52.22 (±7.15)	2110.3 (±118.1)	9196.3 (±371.9)	79.6 (±1.3)
	DMA-Final	**2.95**	**4.03 (±0.62)**	**64.20**	**55.68 (±5.10)**	1562.3 (±62.8)	10243.7 (±241.3)	106.4 (±2.5)
	Rosetta	5.61	6.85 (±0.78)	36.93	34.09 (±1.05)	-63.86 (±5.26)	-	-
3P7K	DMA-Original	2.42	3.46 (±0.86)	72.78	65.67 (±3.52)	2105.9 (±137.7)	9139.0 (±132.2)	78.0 (±1.2)
	DMA-Final	**1.65**	**2.89 (±1.54)**	**81.11**	**72.28 (±9.81)**	1848.2 (±21.3)	9585.2 (±230.4)	104.7 (±3.2)
	Rosetta	15.04	15.94 (±0.49)	54.44	48.39 (±4.68)	-65.42 (±5.58)	-	-
1AB1	DMA-Original	4.91	8.72 (±1.67)	52.17	43.15 (±4.61)	1770.7 (±63.6)	6841.7 (±142.9)	76.7 (±1.1)
	DMA-Final	**2.96**	**6.14 (±2.22)**	**64.67**	**52.66 (±8.66)**	1349.2 (±50.4)	7805.4 (±158.9)	100.9 (±1.5)
	Rosetta	3.45	*5.37 (±1.09)	*67.93	*58.1 (±5.24)	-44.21 (±5.23)	-	-
3V1A	DMA-Original	6.22	10.07 (±2.83)	50.00	46.67 (±1.88)	2129.6 (±120.7)	8635.9 (±144.3)	73.3 (±1.2)
	DMA-Final	**1.65**	**3.23 (±0.90)**	**78.12**	**61.77 (±8.47)**	1604.3 (±40.9)	9500.4 (±174.6)	97.4 (±1.9)
	Rosetta	*0.7	*2.18 (±1.82)	55.21	52.09 (±4.35)	-84.29 (±8.36)	-	-
2P6J	DMA-Original	4.66	8.41 (±2.30)	48.56	43.90 (±4.09)	2551.8 (±204.7)	10879.1 (±324.0)	66.7 (±1.5)
	DMA-Final	**3.59**	**5.40 (±2.02)**	**56.73**	**51.83 (±4.35)**	1910.1 (±84.2)	11940.8 (±266.4)	90.2 (±2.0)
	Rosetta	*2.25	*3.37 (±1.06)	*74.52	*62.93 (±5.57)	-82.89 (±7.51)	-	-
1ENH	DMA-Original	3.54	9.22 (±3.14)	59.72	45.33 (±6.45)	2549.4 (±142.2)	11434.3 (±364.4)	63.6 (±1.4)
	DMA-Final	**2.81**	**5.07 (±2.57)**	**65.74**	**55.19 (±8.87)**	1939.8 (±63.0)	12554.4 (±333.4)	87.4 (±2.0)
	Rosetta	*1.7	*2.88 (±0.94)	46.3	43.61 (±1.22)	-86.32 (±7.47)	-	-
2JUC	DMA-Original	7.43	12.26 (±3.67)	38.18	32.91 (±3.11)	2726.6 (±223.8)	10993.5 (±262.1)	64.4 (±1.3)
	DMA-Final	**5.84**	**9.27 (±1.55)**	**45.45**	**37.86 (±4.33)**	1956.0 (±89.7)	11962.2 (±309.6)	84.4 (±1.8)
	Rosetta	6.1	10.23 (±1.56)	42.8	35.97 (±3.4)	-47.09 (±7.63)	-	-
1ROP	DMA-Original	2.73	13.32 (±5.53)	68.75	49.51 (±7.62)	2447.2 (±179.1)	10423.3 (±197.5)	62.0 (±1.2)
	DMA-Final	**2.68**	**3.94 (±1.00)**	**70.54**	**60.00 (±7.43)**	1904.4 (±26.8)	11369.7 (±235.0)	81.9 (±2.2)
	Rosetta	*1.98	4.98 (±2.6)	*81.25	*65.49 (±12.42)	-88.13 (±11.25)	-	-
2P5K	DMA-Original	6.42	**8.74 (±1.92)**	44.44	37.22 (±4.10)	2847.2 (±164.7)	12212.0 (±294.5)	55.1 (±0.9)
	DMA-Final	**4.35**	8.79 (±2.68)	**54.37**	**43.13** (±5.32)	2190.3 (±101.9)	13456.7 (±260.6)	74.6 (±1.8)
	Rosetta	*1.53	*2.21 (±0.92)	53.97	*51.67 (±1.74)	-105.6 (±4.02)	-	-
1AIL	DMA-Original	12.32	16.43 (±3.64)	42.86	37.57 (±3.28)	3127.8 (±137.4)	13347.8 (±249.9)	49.8 (±0.6)
	DMA-Final	**8.26**	**9.46 (±0.94)**	**53.93**	**41.50 (±4.82)**	2319.0 (±101.2)	14720.4 (±248.9)	66.3 (±1.3)
	Rosetta	*6.85	*9.34 (±0.97)	48.93	38.96 (±4.92)	-97.26 (±6.84)	-	-
2PMR	DMA-Original	11.60	17.66 (±3.32)	48.03	39.34 (±5.93)	3575.7 (±171.3)	14426.0 (±318.3)	45.5 (±1.0)
	DMA-Final	**2.02**	**8.45 (±3.05)**	**72.37**	**47.11 (±10.98)**	2644.2 (±82.0)	16238.0 (±463.3)	62.1 (±1.3)
	Rosetta	2.46	*3.85 (±0.78)	45.39	40.2 (±3.33)	-129.86 (±6.4)	-	-

Table 6: Results of the experiments that compare the original, the latest implementation of the DMA-3DPSP, and Rosetta. DMA-Final combines the approach with the best results of the previous experiments. For each one of the 20 target peptides, we show the results of the best solution and the average of the tests for both RMSD values and GDT percentage vs experimental structures, as well as the average results of free energy, runtime and generations performed. The values in bold represent the best results between the first two implementations. The (*) denotes the case studies where Rosetta outperformed all of the others.

Sequence	Structure	Steric clashes	Unsatisfied hydrogen bonds		SASA	Void volume
			% in shell	% in core		
Gaia Target		**0.02**	**9.56**	**1.45**	**221.64**	**0.0097**
1K43	DMA-Original	0.13	11.30	0.00	208.25	0.00
	DMA-Final	0.10	18.10	0.00	203.04	0.00
1L2Y	DMA-Original	0.14	17.60	0.00	190.27	0.00
	DMA-Final	0.07	9.80	0.00	174.48	0.00
2MTW	DMA-Original	0.14	10.40	0.00	234.52	0.00
	DMA-Final	0.08	17.90	0.00	208.65	0.00
1WQC	DMA-Original	0.14	12.60	0.00	220.85	0.00
	DMA-Final	0.13	13.90	0.00	192.67	0.00
1ACW	DMA-Original	0.19	19.30	0.00	208.09	1.30
	DMA-Final	0.11	19.30	0.00	194.27	0.89
1DFN	DMA-Original	0.10	15.90	0.00	255.56	0.00
	DMA-Final	0.06	17.00	0.00	212.61	1.75
1Q2K	DMA-Original	0.11	14.90	0.00	220.23	0.00
	DMA-Final	0.07	20.60	0.00	192.60	1.58
2F4K	DMA-Original	0.12	16.40	0.00	238.06	0.00
	DMA-Final	0.08	17.50	0.00	202.65	2.48
2MR9	DMA-Original	0.14	11.80	0.00	227.07	0.00
	DMA-Final	0.09	7.60	1.60	178.10	0.51
2P81	DMA-Original	0.18	13.80	0.00	271.61	0.00
	DMA-Final	0.10	13.10	0.00	217.16	0.00
3P7K	DMA-Original	0.18	6.00	0.00	259.35	0.00
	DMA-Final	0.10	9.80	0.00	263.35	0.00
1AB1	DMA-Original	0.20	14.80	0.00	211.79	1.37
	DMA-Final	0.09	14.80	0.00	173.52	1.12
3V1A	DMA-Original	0.10	6.20	0.00	253.64	0.00
	DMA-Final	0.09	8.20	0.00	208.14	0.00
2P6J	DMA-Original	0.17	13.90	0.00	261.18	0.00
	DMA-Final	0.09	9.30	0.00	221.77	0.00
1ENH	DMA-Original	0.14	11.30	0.00	250.69	0.00
	DMA-Final	0.08	6.80	0.00	217.13	1.26
2JUC	DMA-Original	0.16	16.10	0.00	273.61	0.00
	DMA-Final	0.11	15.60	0.50	220.80	4.78
1ROP	DMA-Original	0.10	9.20	0.00	229.96	0.00
	DMA-Final	0.09	5.70	0.00	223.51	0.00
2P5K	DMA-Original	0.19	9.40	0.00	254.14	0.18
	DMA-Final	0.11	12.00	0.00	224.20	0.81
1AIL	DMA-Original	0.19	13.50	0.00	265.12	0.00
	DMA-Final	0.08	9.10	0.00	211.24	1.09
2PMR	DMA-Original	0.16	7.30	0.00	257.61	0.33
	DMA-Final	0.11	11.20	0.00	219.04	5.38

Table 7: Results of the structure quality test of *Gaia Web Tool* for the best solutions found by the two `DMA-3DPSP` implementations tested in the final experiments. For each peptide and `DMA` implementation, we show the values of steric clashes, unsatisfied hydrogen bonds, solvent accessible surface area and void volume from the Gaia report. The values that exceed the optimum margins in Gaia report are presented with red background.

CASP ID	PDB ID	Size	Folding pattern
T0769	2MQ8	112	2 α-helix, 4 β-sheet
T0773	2N2U	77	2 α-helix, 1 3_{10} helix, 4 β-sheet
T0829	4RGI	70	2 α-helix, 5 β-sheet
T0837	2MQD	128	4 α-helix, 6 β-sheet
T0855	5TF3	119	7 α-helix, 1 3_{10} helix

Table 8: List of protein sequences from CASP 11 targets used in computational tests.

Sequence	RMSD (Å)				GDT_TS (%)			
	Best	Average	Zhang	Robetta	Best	Average	Zhang	Robetta
T0769 (2MQ8)	13.503	19.286 ($\pm$4.365)	1.739	1.963	29.69	25.91 ($\pm$2.33)	59.80	53.35
T0773 (2N2U)	8.961	12.593 ($\pm$2.161)	1.180	2.268	40.26	33.44 ($\pm$3.90)	90.67	78.73
T0829 (4RGI)	11.560	12.727 ($\pm$1.131)	6.271	6.461	29.78	24.45 ($\pm$2.83)	64.92	62.31
T0837 (2MQD)	15.014	17.958 ($\pm$3.384)	7.847	8.506	22.90	19.18 ($\pm$2.02)	44.56	39.78
T0855 (5TF3)	13.723	16.749 ($\pm$2.238)	9.271	11.902	30.66	23.93 ($\pm$2.71)	38.84	25.00

Table 9: Comparison of the last version of the DMA-3DPSP method with the CASP 11 protein targets. For each peptide, we show the best solution found and the average of 10 runs regarding RMSD and GDT. We also present the results of RMSD and GDT obtained by Zhang-Server and Robetta Server in the CASP 11 experiments.