Applied Soft Computing 45 (2016) 61–70

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Genetic algorithm with advanced mechanisms applied to the protein structure prediction in a hydrophobic-polar model and cubic lattice

Borko Bošković*, Janez Brest

Faculty of Electrical Engineering and Computer Science, University of Maribor, SI-2000 Maribor, Slovenia

ARTICLE INFO

Article history:
Received 16 October 2015
Received in revised form 16 February 2016
Accepted 5 April 2016
Available online 14 April 2016

Keywords:
Protein structure prediction
Hydrophobic-polar model
Cubic lattice
Genetic algorithm
Clustering
Crowding
Repair mechanism
Local movements
Opposition-based mechanism

ABSTRACT

This paper presents a genetic algorithm applied to the protein structure prediction in a hydrophobic-polar model on a cubic lattice. The proposed genetic algorithm is extended with crowding, clustering, repair, local search and opposition-based mechanisms. The crowding is responsible for maintaining the good solutions to the end of the evolutionary process while the clustering is used to divide a whole population into a number of subpopulations that can locate different good solutions. The repair mechanism transforms infeasible solutions to feasible solutions that do not occupy the lattice point for more than one monomer. In order to improve convergence speed the algorithm uses local search. This mechanism improves the quality of conformations with the local movement of one or two consecutive monomers through the entire conformation. The opposition-based mechanism is introduced to transform conformations to the opposite direction. In this way the algorithm easily improves good solutions on both sides of the sequence. The proposed algorithm was tested on a number of well-known hydrophobic-polar sequences. The obtained results show that the mechanisms employed improve the algorithm's performance and that our algorithm is superior to other state-of-the-art evolutionary and swarm algorithms.

1. Introduction

The problem of protein structure prediction (PSP) represents the computational problem of how to predict the native structure of a protein from its amino acid sequence [1]. Protein function is determined by its structure. With the wrong structure, a protein cannot correctly fulfill its function [2]. With current algorithms and computational resources it is possible to predict the native structures of relatively small proteins and to help in drug design and proteomics [1]. The PSP problem is one of the more important challenges of this century [3] and because of its nature it attracts scientists from different fields such as physics, chemistry, biology, mathematics, and computer science. However, despite large research efforts in recent decades, further improvements are still possible for this problem [2].

The PSP problem is a hard combinatorial optimization problem, even under an hydrophobic-polar (HP) simplified model [4] that has been proved to be NP-complete [5]. Researchers have developed a great variety of simplified protein models [6–8]. These models reduce the number of degrees of freedom, take into account only specific interactions and therefore speed up the calculations. Although these models are incomplete, they allow for the development, testing, and comparison of various search algorithms and they offer a global perspective of protein structures. They can be helpful in confirming or questioning important theories [9].

This paper presents the genetic algorithm (GA_{PSP}) for constructing the native tridimensional structure of a given amino acid sequence under an HP model and cubic lattice. This algorithm belongs to the *ab initio* protein prediction type of methods which predict structures from scratch and do not require any information about related protein structures. The proposed GA is extended with crowding and clustering mechanisms, because the PSP problem is multimodal by nature. The crowding is employed for maintaining the diversity of the population and for preserving good solutions to the end of the evolutionary process. The clustering is used to divide a whole population into more subpopulations that can locate different good conformations [10]. Additionally, our algorithm includes a repair mechanism which transforms infeasible conformations to feasible ones. Feasible conformations do not occupy a lattice point with more than one monomer. Thus, the repair mechanism detects collision (a lattice point that is occupied with two

* Corresponding author.
E-mail addresses: borko.boskovic@um.si (B. Bošković), janez.brest@um.si (J. Brest).

http://dx.doi.org/10.1016/j.asoc.2016.04.001
1568-4946/© 2016 Elsevier B.V. All rights reserved.

or more monomers) and tries to repair it or form a contiguous self-avoiding path within the lattice. A backtracking algorithm is used for this purpose, as proposed in [11]. For every detected collision, a repair mechanism tries to move a monomer from an already occupied point to an unoccupied neighborhood point. If there are no such points available, then it repeats this process for the previous monomer.

In order to speed up the convergence of the algorithm and to improve the energies of the generated feasible conformations, the algorithm uses local search that includes two operators of local movement. Local movement is a transformation of protein conformation whereby only a few monomers are locally moved from one point to another while the remaining monomers remain on their points. The introduced operators try to improve the energy of the conformations with local movements of one or two consecutive monomers. Local movements that have been described in the literature are used either as genetic operators that are applied on randomly chosen positions [12] or as macromutations within local searches [9]. In contrast to these operators, our operators try to locally move only one or two consecutive monomers, where at least one of them is an H–monomer, through the entire protein conformation. Used mechanisms inside an algorithm change conformation in a specific direction. Therefore an opposition-based mechanism is introduced to transform conformations of the next generation into opposite directions using inverse amino acid sequence. This mechanism is based on the assumption that inverse sequence allows the algorithm to easily improve conformation of the monomers at the beginning or at the end of the sequence.

The main contributions of this paper are: (1) the proposed new GA algorithm for the PSP problem, (2) the introduced local search or local movement operators that try to improve the energy of the conformations with local movements of one or two consecutive monomers throughout the entire conformation, and (3) a proposed opposition-based mechanism that transforms conformations into the opposite direction and the corresponding amino acid sequence into its inverse sequence. The proposed algorithm is tested on different well-known HP sequences and compared with the state-of-the-art evolutionary and swarm algorithms. Experimental results show that the proposed algorithm is superior to its competitors.

The remainder of this paper is organized as follows. The existing evolutionary and swarm algorithms for the PSP problem are briefly described in Section 2. Sections 3 and 4 describe the hydrophobic-polar model and the proposed algorithm in sufficient detail. The experimental setup and results are presented in Sections 5 and 6. Section 7 concludes this paper.

2. Related work

Evolutionary and swarm algorithms have previously been successfully applied to the HP model on the cubic lattice. In [13] the authors developed a GA algorithm that has a population of conformations. Conformations are changed using mutation, in the forms of conventional Monte Carlo steps, and crossovers, in which parts of the sequence of directions are interchanged between conformations. Both genetic operators iterate until feasible conformations are created. The results show that the GA algorithm is superior to conventional Monte Carlo methods. This work was upgraded in [14] where the authors showed that GA is effective for determining protein structure. In this work, infeasible conformations were used within the evolutionary process and their energies were calculated using a penalty function. Additionally, a crowding mechanism was used for maintaining population diversity.

The study [15] considered that the two operators of multi-point crossover and a local perturbation are required for any GA to be fully effective. A systematic crossover search is proposed in [16] that uses information about differences between two individuals. This mechanism in combination with GA significantly increased the search effectiveness. Another study [17] compared relative and absolute encoding of conformations. As a result of encoding, the search space is different and this is the reason why the GA described in [14] is superior to the GA described in [13]. The authors in this study also proposed a modified energy potential that facilitated the GA search and identified weaknesses in the constraint management strategies.

A repair procedure and evolutionary operators whose functioning is closed in feasible space are used within the evolutionary algorithm with an embedded backtracking algorithm [11]. A memetic algorithm with a self-adaptive local search mechanism is introduced in [9]. The results of that study confirm that the local search improves the algorithm's performance. In [18] GA is used with a modification of the scoring system for improving the model's capacity and to generate more natural-like structures. In this paper an islands' algorithm was also analyzed and it improved performance of the GA algorithm. The same authors in [12] developed a robust and efficient GA that employs a phenotype based crowding mechanism for the maintenance of useful diversity within the population. The GA algorithm that is extended with heuristic repair methods for dealing with infeasible intermediate candidate solutions is analyzed in [19].

A memetic algorithm with the following novel features is introduced in [20]: a modified fitness function, a systematic generation of population that automatically prevents infeasible conformations, a generalized non-isomorphic encoding scheme that implicitly eliminates a generation of symmetrical conformations, population clustering and the identification of a meme according to the genotype, and a 2-stage mechanism for migrating domain knowledge between different basins of attraction. The multi-objective approach to constraint-handling was investigated in [21]. For that purpose the HP model was reformulated as an unconstrained multi-objective problem by treating constraints as an additional objective function. From the obtained results, the authors showed that the multi-objective strategy can be incorporated within state-of-the-art algorithms and can improve their performances.

The following have also been applied to the HP model on a cubic lattice: an ant colony optimization algorithm with local search [22], a particle swarm optimization based algorithm [23], an estimation of distribution algorithm [24], a differential evolution algorithm [25], and an immune algorithm [26].

Unlike those in the literature [12,18], the proposed algorithm does not use elitism for fast convergence. The PSP problem is multimodal by nature and without elitism the algorithm has a greater chance of avoiding local optima. Nevertheless, our algorithm shows fast convergence. The local search is used for this purpose. In contrast to those in the literature [12,22,9], our local search uses local movements that are applied to the whole sequence and not only to the one randomly chosen position within the sequence, and it is not limited to local movements of only one monomer. It also performs local movements of two consecutive monomers. The proposed algorithm additionally includes the clustering mechanism that is designed according to [10] and the repair mechanism according to [11]. The clustering mechanism improves the convergence rate and the diversity of the population while the repair mechanism forms feasible solutions. Thus the algorithm does not need to compute an unnecessary evaluation of infeasible solutions. The local search and repair mechanisms perform their tasks from the beginning to the end of the sequence. After each generation, the introduced opposition-based mechanism transforms all individuals into their opposite directions. The result of this transformation is that the local search and repair mechanisms perform their tasks

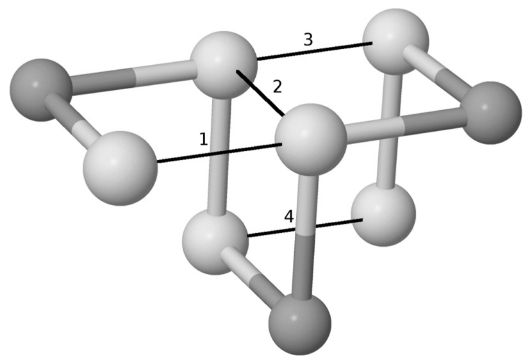

Fig. 1. The protein energy value $E(\boldsymbol{s}, \boldsymbol{c}) = -4$ of sequence $\boldsymbol{s} = \{H, P, H, H, P, H, P, H, H\}$ in the cubic lattice with the following conformation: $c = \{(0, 0, 0), (0, 0, -1), (1, 0, -1), (1, -1, -1), (1, -1, 0), (1, 0, 0), (2, 0, 0), (2, 0, -1), (2, -1, -1)\}$. The H monomers are shown in white and the P monomers in dark. Hydrophobic-hydrophobic contacts between two non-consecutive monomers that occupy adjacent lattice points are shown with numbered black lines.

in both directions and consequently facilitates the algorithm to find better conformations.

3. Hydrophobic-polar model

In the hydrophobic-polar (HP) model, an amino acid sequence of length L is represented as a string $\boldsymbol{s} = \{s_1, s_2, \ldots, s_L\}$, $s_i \in \{H, P\}$, where H represents a hydrophobic and P a hydrophilic amino acid or monomer. Different lattices can be used to reduce the number of degrees of freedom and they define the conformation of monomers [7]. Each point in the lattice can be occupied by only one monomer and the energy of a protein is calculated as a negation of the number of hydrophobic-hydrophobic contacts between two non-consecutive monomers which occupy adjacent points in the lattice, as shown in Fig. 1 and in the following equation:

$$
\begin{aligned}
E(\boldsymbol{s}, \boldsymbol{c}) &= -1 * \sum_{i=1}^{L-2} \sum_{j=i+2}^{L} f_{HH}(s_i, c_i, s_j, c_j) \\
f_{HH}(s_i, c_i, s_j, c_j) &= \begin{cases} 1, & \text{if } s_i = s_j = H \text{ and} \\ & c_i \text{ and } c_j \text{ are adjacent points} \\ 0, & \textbf{otherwise}, \end{cases} \\
\boldsymbol{c} &= \{c_1, c_2, \ldots, c_L\}, \quad \boldsymbol{c} \in C
\end{aligned} \tag{1}
$$

where c_i and c_j are lattice points and C is the set of all valid conformations (contiguous self-avoiding path) for the specific lattice. According to the thermodynamic hypothesis [27], simplified computational models of PSP are typically formulated to find a native structure or conformation of amino acids, which has the lowest energy value (native energy) on a specific lattice as shown in the following equation:

$$
\boldsymbol{c}_N = \arg\min_{\boldsymbol{c} \in C} E(\boldsymbol{s}, \boldsymbol{c}). \tag{2}
$$

The energy function described in Eq. (1) has a discrete property and this additionally increases the difficulty of the problem. In some cases this function cannot give direction to an algorithm for locating better conformation. Therefore, the energy function can be transformed to a continuous one, as was performed in [8].

4. Proposed algorithm

The proposed algorithm (GA$_{\text{PSP}}$) extends and adapts GA which is a stochastic, population-based optimization algorithm. The population (P) of our algorithm contains *popSize* individuals $\boldsymbol{x_i}$, $i = 1, 2, \ldots, popSize$ that are encoded with absolute encoding [13,12]. Thus each individual is defined as a vector of $L-1$ absolute directions: $x_{i,j} \in \{L, R, U, D, F, B\}$; $j = 1, 2, \ldots, L-1$. The following six possible directions are used: left (L), right (R), up (U), down (D), forward (F), and backward (B).

The GA$_{\text{PSP}}$ algorithm is shown in Fig. 2a. Initial population P is selected uniform randomly and improved with local searches at

```
procedure GA_PSP($s$, $popSize$, $subPopSize$, $pm$)
    Initialize a population $P = \{x_1, x_2, ..., x_{popSize}\}$
    $x_i$ = randomConformation();
    $\{x_i, e_i\}$ = localSearch($s$, $x_i$)
    while stopping criteria is not met do
        $C$ = clustering($P$, $popSize$, $subPopSize$)
        for $k = 1$ to sizeof($C$) do
            for $i = 1$ to $subPopSize$ do
                do { $j = rand()\%subPopSize$ } while($i = j$)
                $\{u_1, u_2\}$ = crossover($c_{k,i}$, $c_{k,j}$)
                if $u_1 = c_{k,i}$ or $u_{i,1} = c_{k,j}$ or rand(0,1) $< pm$ then
                mutation($u_1$) end if
                if $u_2 = c_{k,i}$ or $u_{i,2} = c_{k,j}$ or rand(0,1) $< pm$ then
                mutation($u_2$) end if
                $\{u_1, e_{u1}\}$ = localSearch($s$, $u_i$)
                $\{u_2, e_{u2}\}$ = localSearch($s$, $u_i$)
                $x_n$ = nearest($P$, $u_1$)
                if $e_{u1} \geq e_n$ then $\{x_n, e_n\} = \{u_1, e_{u1}\}$ end if
                $x_n$ = nearest($P$, $u_2$)
                if $e_{u2} \geq e_n$ then $\{x_n, e_n\} = \{u_2, e_{u2}\}$ end if
            end for
        end for
        $\{s, P\}$ = oppositeDirections($s$, $P$)
    end while
end procedure
```

(a) GA$_{\text{PSP}}$

```
procedure clustering($P$, $popSize$, $subPopSize$)
    $T = P$
    $C = \emptyset$
    for $i = 1$ to $\frac{popSize}{subPopSize}$ do
        $t_r$ = random($T$)
        $N$ = nearest($T$, $t_r$, $subPopSize - 1$)
        $S = \{t_r\} \cup N$
        $T = T \setminus S$
        $C = C \cup S$
    end for
    return $C$
end procedure
```

(b) Clustering

```
procedure localSearch($s$, $u$)
    $\{u, e_u\}$ = repairAndEvaluate($s$, $u$)
    while true do
        $\{u_{lm}, e_{lm}\}$ = twoMonomersMovements($s$, $u$, $e_u$)
        $\{u_{lm}, e_{lm}\}$ = oneMonomerMovements($s$, $u_{lm}$, $e_{lm}$)
        if $e_u = e_{lm})$ then break end if
        $\{u, e_u\} = \{u_{lm}, e_{lm}\}$
    end while
    return $\{u, e_u\}$
end procedure
```

(c) Local search

Fig. 2. The proposed algorithm is divided into three procedures: GA$_{\text{PSP}}$, clustering and local search. The GA$_{\text{PSP}}$ procedure: evolutionary process of the proposed algorithm or how all used mechanisms work together. The clustering procedure: dividing the whole population P into the set C of subpopulations. The local search procedure: try to improve the individual using repair and local movement mechanisms.

the beginning of the evolutionary process. Within every generation clustering is used (line 4) to divide a whole population into a number of subpopulations that can locate different good conformations. For every generated subpopulation C_k the algorithm generates new individuals using multi-point crossover and segment mutation. The multi-point crossover randomly performs one-, two- or three-point crossover. As already shown in literature [12,18], a multi-point crossover is an important genetic operator for protein structure prediction. Therefore this operator is performed *subPopSize* (size of subpopulation) times between *i*th and *j*th individuals for one population, as shown in line 8. The segment mutation randomly selects directions of one, two or three consecutive directions. This is performed with the probability of 0.01 or when the crossover generates the same crossover individuals (lines 9 and 10). In such a way, the mutation ensures that the generated individuals are different and also increases the diversity of the generated individuals. The local search improves generated individuals (lines 11, 12) with the repair mechanism and local movements. Improved individuals are then compared with the nearest individual from population *P* and the nearest individual is calculated using the Hamming distance. This comparison with the nearest individual actually represents the crowding mechanism (lines 13–16). Mechanisms within the local search change conformations from the beginning to the end of the sequence. Therefore, the opposition-based mechanism is performed at the end of each generation (line 19) which transforms individuals into the opposite directions. This allows the algorithm to easily improve good conformations on both sides of the sequence.

4.1. Clustering

The main goal of the used clustering mechanism [10] is to divide a whole population into a number of subpopulations which are assigned to different promising subregions or subpopulations within the search space. As shown in Fig. 2b it takes the population *P* of *popSize* individuals and a parameter *subPopSize* that determines the size of subpopulations. Firstly, the population *P* is copied to a temporary population *T* and a set of subpopulations *C* is initialized to an empty set. Then inside the *for* loop, population *T* is divided into subpopulations. The individual $\boldsymbol{t_r}$ is randomly selected from *T* (line 5). This individual and its *subPopSize* − 1 nearest individuals from *T* are stored within population *S* and removed from population *T*. One individual can belong to only one subpopulation and the average distance between individuals is greater for subsequent subpopulations. The nearest individual is calculated using the Hamming distance that is defined by the number of directions at which the two corresponding individuals are different.

4.2. Local search

Local search is used within the $\mathrm{GA_{PSP}}$ algorithm to improve the generated individuals. It tries to improve individuals with a repair mechanism and local movements of one or two consecutive monomers through the entire conformation. The repair mechanism transforms infeasible to feasible individuals which do not occupy lattice points with more than one monomer using the backtracking algorithm. Local movement is a transformation of protein conformation where only a few monomers, in our case only one or two consecutive monomers where one of them is an H monomer, are locally moved from one point to another while the remaining monomers stay on their lattice points. Fig. 3 shows an example of one monomer local movement which improves the energy value of a selected individual. The second monomer is only moved while all the remaining monomers stay on their lattice points. As we can see, the replacement of two consecutive directions, in our case the first and second directions, allows local movements of one monomer

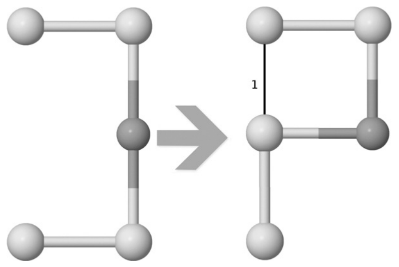

Fig. 3. The local movement of one monomer for sequence $\boldsymbol{s} = \{H, H, P, H, H\}$. The individual on the left $\boldsymbol{x_l} = \{R, U, U, L\}$ has energy value $E(\boldsymbol{s}, \boldsymbol{x_l}) = 0$, while the individual on the right $\boldsymbol{x_r} = \{U, R, U, L\}$ is obtained after local movement of the second monomer and has energy value $E(\boldsymbol{s}, \boldsymbol{x_r}) = -1$.

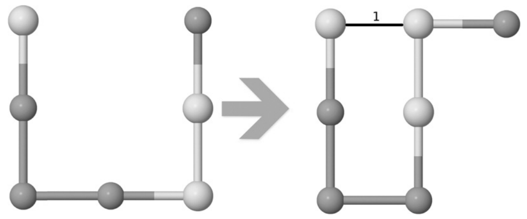

Fig. 4. The local movement of two monomers for sequence $\boldsymbol{s} = \{H, P, P, P, H, H, P\}$. The individual on the left $\boldsymbol{x_l} = \{D, D, R, R, U, U\}$ has energy value $E(\boldsymbol{s}, \boldsymbol{x_l}) = 0$, while the individual on the right $\boldsymbol{x_r} = \{D, D, R, U, U, R\}$ is obtained after local movement of the fifth and sixth monomers and has energy value $E(\boldsymbol{s}, \boldsymbol{x_r}) = -1$.

with only one exception. For the last monomer, the last direction is replaced using all the remaining directions. An example of the local movement of two monomers is shown in Fig. 4. We can see that two consecutive monomers (fifth and sixth) have moved while all the remaining monomers remain on their lattice points and the individual energy is improved. Here, we observe that the replacement of the direction before the first moved monomer and the direction after the second moved monomer allows the local movement of the two consecutive monomers.

The proposed local search is performed on one individual, as shown in Fig. 2c. First, the individual is repaired and evaluated. Feasible and evaluated individuals are required for efficient local movements. Note that local movements do not re-evaluate an individual, they only update the individual's directions and its energy value (see Figs. 5 and 6). Within the *while* loop local movements are performed and the local search is finished when there are no improvements in the energy value during the one loop iteration.

The one-monomer local movements algorithm is shown in Fig. 5. It has two *for* loops. In the first loop it tries to locally move one H monomer by swapping two consequent directions. If improvement is achieved then the new conformation is accepted and the energy value is updated. In the second loop, the last monomer is moving. In other words the last direction of the individual is swapped with the remaining directions (d_i) and the best conformation is selected. The algorithm of two monomer local movements is shown in Fig. 6. It tries to locally move two consequent monomers where one of them is an H monomer. If improvement is achieved then the energy value is updated and the new conformation is accepted. Note that the two-monomer local movements allow additional improvements compared to the one monomer local movements. For example, the conformation shown in Fig. 4 cannot be improved with one-monomer local movements.

4.3. Opposition-based mechanism

The opposition-based mechanism transforms an individual's conformation of the next generation into the opposite directions for an inverse amino acid sequence. This mechanism is based on

```
procedure oneMonomerMovements($s$, $u$, $e$)
    for $i = 2$ to $L - 1$ do
        if $s_i = H$ and $u_{i-1} \neq u_i$ then
            swapDirections($u_{i-1}, u_i$)
            if moved monomer occupies free lattice point and has larger # of HH contacts than before then
                e = e + Δ number of HH contacts
            else
                swapDirections($u_{i-1}, u_i$)
            end if
        end if
    end for
    if $s_L = H$ then
        for $i = 1$ to 6 do
            swapDirections($u_{L-1}, d_i$)
            if moved monomer occupies free lattice point and has larger # of HH contacts than before then
                e = e + Δ number of HH contacts
            else
                swapDirections($u_{L-1}, d_i$)
            end if
        end for
    end if
end procedure
```

Fig. 5. One-monomer local movements.

```
procedure twoMonomersMovements($s$, $u$, $e$)
    for $i = 2$ to $L - 2$ do
        if $s_i = H$ or $s_{i+1} = H$ then
            if $u_{i-1} \neq u_{i+1}$ then
                if $u_{i-1} \neq$ opp($u_{i+1}$) and $u_{i-1} \neq$ opp($u_{i+2}$) then
                    swapDirections($u_{i-1}, u_{i+1}$)
                    if moved monomers occupie free lattice points and have larger # of HH contacts than before then
                        e = e + Δ number of HH contacts
                    else
                        swapDirections($u_{i-1}, u_{i+1}$)
                    end if
                end if
            end if
        end if
    end for
end procedure
```

Fig. 6. Two-monomer local movements.

the assumption that opposite individuals and an inverse amino acid sequence allow the algorithm to easily improve conformation at the beginning or at the end of the sequence. For this purpose, opposite individual $\boldsymbol{x_i^{opp}}$ and inverse sequence $\boldsymbol{s^{inv}}$ are calculated as follows:

$$x_{i,j}^{opp} = oppositeDirection(x_{i,(L-1)-j})$$
$$s_j^{inv} = s_{L-j} \tag{3}$$

For example, the opposite individual and the inverse sequence of the protein from Fig. 1 are shown in the following equations:

$$\boldsymbol{x_i} = \{B, R, D, F, U, R, B, D\}$$
$$\boldsymbol{x_i^{opp}} = \{U, F, L, D, B, U, L, F\}$$
$$\boldsymbol{s} = \{H, P, H, H, P, H, P, H, H\} \tag{4}$$
$$\boldsymbol{s^{inv}} = \{H, H, P, H, P, H, H, P, H\}$$

5. Experimental setup

The proposed GA$_{PSP}$ algorithm was implemented in C++ programming language with a GNU C++ compiler 4.6.3 and executed using an Intel Core i7 computer with 2.93 GHz CPU and 8 GB RAM under Linux Mint 13 Maya and grid environment (Slovenian Initiative for National Grid). In order to evaluate the performance of the proposed algorithm, we used four sets of sequences as shown in Table 1. These sequences have been frequently used in the literature and they enable comparisons with different algorithms. The first set contains 10 sequences with 48 monomers [28], the second set contains 10 sequences with 64 monomers [14], the third set contains 8 sequences with 20–64 monomers [20], and the fourth set contains 5 sequences with 46–135 monomers [29]. The last two sets are useful in demonstrating the efficiencies of algorithms on sequences with a different number of monomers.

In the experiments we used the following settings for our algorithm: *popSize* = 500, *subPopSize* = 10, and *pm* = 0.01. These values were selected according to the analysis of the algorithm (see Section 6.1). In order to provide a fair comparison, the stopping conditions were set according to the literature and the reported results were calculated through 50 independent runs for each sequence. In order to compare the performances of algorithms from literature with the proposed algorithm, we measured the following criteria:

Table 1
Test sequences.

Label	Sequence
T48.1	HPH2P2H4PH3P2H2P2HPH3PHPH2P2H2P3HP8H2
T48.2	H4PH2PH5P2HP2H2P2HP6HP2HP3HP2H2P2H3PH
T48.3	PHPH2PH6P2HPHP2HPH2PHPHP3HP2H2P2H2P2HPHP2HP
T48.4	PHPH2P2HPH3P2H2PH2P3H5P2HPH2PHPHP4HP2HPHP
T48.5	P2HP3HPH4P2H4PH2PH3P2HPHPHP2HP6H2PH2PH
T48.6	H3P3H2PHPH2PH2PH2PHP7HPHP2HP3HP2H6PH
T48.7	PHP4HPH3PHPH4PH2PH2P3HPHP3H3P2H2P2H2P3H
T48.8	PH2PH3PH4P2H3P6HPH2P2H2PHP3H2PHPHPH2P3
T48.9	PHPHP4HPHPHP2HPH6P2H3PHP2HPH2P2HPH3P4H
T48.10	PH2P6H2P3H3PHP2HPH2P2HP2HP2H2P2H7P2H2
T64.1	P2H5P3H2P5H2P3HP6HPHP3HP2HP2HP5HP4H2PH2P2HP2 HP
T64.2	P2HPHP2HP2H3PH4P2H3P4HPHP3HPHP3HPHP5HPHP2HP HP3HP2HP2
T64.3	HPH2P2H2PHP5H3PH4P2HP2HPH2P3HPHP2H3PH2PHP5 H8P3
T64.4	HP2H2P2HP2HPHP2HP4HP6HPHPH3P2HPHP3HPHP2H2P2 HP2HP2HPH3PH
T64.5	HP3H2P2HPHP3HP3HPH2P3H2PHPH2PHP2HP3HP2HPH3 P2HP2HP2H3PH4
T64.6	HP2H2PH4P6H2P2HP4H2P3HP2HPH2PHP4H2P4HP5HP4HP H2
T64.7	P4HP3HP3H4PH2P5HP2HPH2PHPHP5HP10H4P4H2P2H
T64.8	P3H3P2HPHP2HP2H2P3HP2HP2H2PHP3HP7HPH3PH5P2H2 P3HP2H
T64.9	HP2HP2H3P4HPHP3HPH2PH5P4HPHPHP4HPHP3H2PHP4 HP2H2PHP
T64.10	P2HP2HP2H3P3HPHP2HP2HP6HP2H3P2HP2HP2HPHP6H3 P5HPHP
T20	HPHP2H2PHP2HPH2P2HPH
T24	H2P2HP2HP2HP2HP2HP2HP2H2
T25	P2HP2H2P4H2P4H2P4H2
T36	P3H2P2H2P5H7P2H2P4H2P2HP2
T48	P2HP2H2P2H2P5H10P6H2P2H2P2HP2H5
T50	H2PHPHPHPH4PHP3HP3HP4HP3HP3HPH4PHPHPHPH2
T60	P2H3PH8P3H10PHP3H12P4H6PH2PHP
T64	H12PHPHP2H2P2H2P2HP2H2P2H2P2HP2H2P2H2P2HPHP H12
T46	P2H3PH3P3HPH2PH2P2HPH4PHP2H5PHPH2P2H2P
T58	PHPH3PH3P2H2PHPH2PH3PHPHPH2P2H3P2HPHP4HP2H P2H2P2HP2H
T103	P2H2P5H2P2H2PHP2HP7HP3H2PH2P6HP2HPHP2HP5H3P4 H2PH2P5H2P4H4PHP8H5P2HP2
T124	P3H3PHP4HP5H2P4H2P2H2P4HP4HP2HP2H2P3H2PHPH3 P4H3P6H2P2HP2HPHP2HP7HP2H3P4HP3H5P4H2PHPHPH PH
T136	HP5HP4HPH2PH2P4HPH3P4HPHPH4P11HP2HP3HPH2P3 H2P2HP2HPHPHP8HP3H6P3H2P2H3P3H2PH5P9HP4HPH P4

Table 2
Influence of the control parameters. Stopping condition was $MAX\_NSEs = 4 \times 10^6$ and number of runs was 50.

Seq.	GA$_{PSP}$		*popSize* = 200		*popSize* = 1000		*subPopSize* = 5		*subPopSize* = 20		*pm* = 0.1		*pm* = 0.001	
	E_{mean}	E_{std}	E_{mean}	E_{std}	E_{mean}	E_{std}	E_{mean}	E_{std}	E_{mean}	E_{std}	E_{mean}	E_{std}	E_{mean}	E_{std}
T46	34.42	0.49	34.30	0.46	34.32	0.51	34.30	0.46	34.36	0.48	34.42	0.53	**34.44**	0.50
T58	41.92	0.59	41.80	0.66	41.76	0.47	41.58	0.60	41.88	0.68	**42.02**	0.65	41.78	0.54
T103	50.80	0.98	50.70	0.88	50.60	0.87	50.66	0.93	**50.88**	1.05	50.72	0.92	50.66	1.03
T124	**64.78**	1.25	64.24	1.45	64.20	0.96	64.06	1.42	64.42	1.02	64.50	1.22	64.50	1.28
T136	68.54	1.20	**68.76**	1.52	68.26	1.37	68.10	1.28	68.10	1.33	68.22	1.06	68.52	1.32
Mean	**52.09**	**0.90**	51.96	0.99	51.83	0.84	51.74	0.94	51.93	0.91	51.98	0.88	51.98	0.93

best obtained energy E_{best}, mean energy E_{mean}, standard deviation of energies E_{std}, *runtime*, final population *diversity*, the number of sequence evaluations (line 2 in Fig. 2c) per second (*speed*), and success rate *SR*. Success rate is calculated as shown in the following equation:

$$SR = \frac{NSR}{NR} \tag{5}$$

where *NSR* denotes the number of runs during which the best-known energy was found and *NR* the number of all runs. The mean diversity of final populations for *NR* runs is calculated with the following equation:

$$div = \frac{2}{NR * n * popSize} \sum_{r=1}^{NR} \sum_{i=1}^{n} \sum_{j=i+1}^{popSize} distance_r(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{6}$$
$$n = popSize - 1$$

where $distance_r(\boldsymbol{x_i}, \boldsymbol{x_j})$ represents the Hamming distance between the two individuals from the final population of the *r*th run. All energy values reported in this section are multiplied by -1. This means that all reported energies have positive values and higher values are better. Using the aforementioned criteria, the proposed GA$_{PSP}$ algorithm was analyzed and compared with the following seven algorithms from the literature:

- GAHP (2014) [12]: the adaptive genetic algorithm with phenotypical crowding,
- MA (2004) [9]: a memetic algorithm with self-adaptive local search,
- ACO (2005) [22]: an ant colony optimization algorithm,
- MO + FR (2015) [21]: a multi-objective genetic algorithm with feasibility rules,
- HGA (1999) [16]: a genetic algorithm with systematic crossover,
- EDA (2008) [24]: an estimation of distribution algorithm, and
- CMA (2013) [20]: a clustered memetic algorithm with local heuristics.

6. Experimental results

In this section, we analyzed the influence of the control parameters and the used mechanisms to the algorithm performances. Then we compared the GA$_{PSP}$ algorithm with the existing algorithms listed in the previous section.

6.1. Control parameters

The influence of the control parameters on performance was analyzed on the third test set of sequences because this set contains sequences with different lengths, including long sequences. The stopping condition was the maximum number of sequence evaluations $MAX\_NSEs = 4 \times 10^6$ or number of *repairAndEvaluate* method invocations. As we already mentioned our algorithm has three control parameters: *popSize*, *subPopSize*, and *pm*. Their default values are 500, 10 and 0.01. In this analysis for each of these parameters we selected three different values: $popSize \in \{200, 500, 1000\}$, $subPopSize \in \{5, 10, 20\}$ and $pm \in \{0.1, 0.01, 0.001\}$. As shown in Table 2, the default parameter values (column GA$_{PSP}$) are compared with those where only one value is different from the default ones. For example, the results for *popSize* = 200 include default parameter values of *subPopSize* = 10 and *pm* = 0.01. From the results we can see that the best results (bold mean values) for different sequences are obtained with different settings. Therefore, in the last row we show mean values for all sequences. From this we can see that the default parameter values obtained the best result.

6.2. Influence of the used methods

The influence of the used mechanisms on performance was analyzed on the third test set sequences with $MAX\_NSEs = 4 \times 10^6$ as in the previous subsection. Four variants of our algorithm were analyzed compared to the used mechanisms:

- GA^1_{PSP} – the GA$_{PSP}$ algorithm without the local search, opposition-based, and clustering method,
- GA^2_{PSP} – the GA$_{PSP}$ algorithm without the opposition-based, and clustering method,
- GA^3_{PSP} – the GA$_{PSP}$ algorithm without the clustering method,
- GA$_{PSP}$ – the proposed GA$_{PSP}$ algorithm that includes clustering method, opposition-based method and local search.

The algorithms were ordered based on a number of the used mechanisms, i.e. GA^1_{PSP} was the simplest one, GA^2_{PSP} included the local search, etc. Note that the local search was replaced with the *repairAndEvaluate* method within the GA^1_{PSP} algorithm. The results obtained from these four algorithms are shown in Table 3. In order to determine the statistical significance of energy and

Table 3
Influence of the used mechanisms. Stopping condition was $MAX\_NSEs = 4 \times 10^6$ and number of runs was 50. The underline mean energy and italic speed values show that there is no significant differences between the corresponding values according to the Wilcoxon's signed rank test at the 5% significance level.

Seq.	GA^1_{PSP}					GA^2_{PSP}					GA^3_{PSP}					GA$_{PSP}$				
	E_{best}	E_{mean}	E_{std}	*speed*	*div*	E_{best}	E_{mean}	E_{std}	*speed*	*div*	E_{best}	E_{mean}	E_{std}	*speed*	*div*	E_{best}	E_{mean}	E_{std}	*speed*	*div*
T46	**35**	$\underline{33.42}$	0.70	**36,846**	34.46	**35**	$\underline{33.46}$	0.54	26,409	36.52	**35**	34.00	0.53	26,201	36.34	**35**	**34.42**	0.49	23,812	36.78
T58	42	40.80	0.60	**30,335**	40.32	43	$\underline{41.50}$	0.54	*21,852*	43.71	43	$\underline{41.68}$	0.54	*21,957*	43.24	**44**	**41.92**	0.59	20,993	45.34
T103	51	48.00	1.08	**16,113**	54.99	52	$\underline{49.88}$	0.86	13,346	63.44	52	$\underline{50.20}$	0.96	14,293	64.98	**53**	**50.80**	0.98	14,754	77.44
T124	64	61.14	1.70	**13,610**	52.67	67	63.94	1.45	10,802	62.92	67	$\underline{64.52}$	1.24	11,305	60.81	**68**	$\underline{\mathbf{64.78}}$	1.25	12,267	85.69
T136	70	64.90	1.53	***11,921***	63.74	**73**	67.42	1.50	9882	70.63	72	$\underline{68.12}$	1.75	10,665	69.13	71	$\underline{\mathbf{68.54}}$	1.20	*11,579*	96.93

Table 4
Influence of the used mechanisms. Stopping condition was *MAX_NSEs* = 4×10^7 and number of runs was 50.

Seq.	GA^2_{PSP}			GA_{PSP}		
	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{std}
T136	74	69.7	1.80	**75**	**71.7**	1.32

speed values, Wilcoxon's signed rank test [30,31] was performed between all algorithms at the 5% significance level. In the table, the underlined mean energy and italic speed values indicate that there is no significant difference between the corresponding values. For example, there is no significant difference between GA_{PSP} and GA^3_{PSP} for sequences T124 and T136 in terms of energy values. We can also see that for almost all speed values there exists a significant difference. There is no significant difference between GA^2_{PSP} and GA^3_{PSP} for sequence T58 and between GA^1_{PSP} and GA_{PSP} for sequence T136.

From the obtained results we can derive the following observations. As expected, all the analyzed mechanisms improved the mean energy values E_{mean} for all sequences. The values of E_{best} were also improved with more included mechanisms, with the exception of sequence T136. This might be a result of the stopping condition and the population diversity. The results concerning speed are also interesting. However, the GA_{PSP} with the most included mechanisms obtained higher speeds than GA^2_{PSP} and GA^3_{PSP} for longer sequences and similar speed to GA^1_{PSP} for T136. We argue that GA_{PSP} increased in speed over longer sequences compared to all remaining algorithms as a result of the population diversity.

Table 3 shows the mean diversity values (Eq. (6)) of the final populations with the largest diversity being obtained by GA_{PSP}.

Table 7
Mean runtime required for obtaining native energy in minutes. 50 independent runs were performed for GA_{PSP} and 50–100 runs for ACO. Intel Core i7 with 2.93 GHz CPU was used for GA_{PSP} and 2.4 GHz Pentium IV CPU was used for ACO.

Seq.	E_n	*runtime* [minutes]		Speed up
		GA_{PSP}	ACO [22]	
T48.1	32	**1.75**	30	17.14
T48.2	34	**5.37**	420	78.21
T48.3	34	**7.52**	120	15.96
T48.4	33	**7.26**	300	41.32
T48.5	32	**2.64**	15	5.68
T48.6	32	**9.16**	720	78.60
T48.7	32	**37.45**	720	19.22
T48.8	31	**4.82**	120	24.90
T48.9	34	**14.74**	450	30.53
T48.10	33	**5.54**	60	10.83

The value of *MAX_NSEs* = 4×10^6 was relatively small for T136 and therefore in the case of a good attractor GA^2_{PSP} obtained the best energy value. When we increased *MAX_NSEs* to 4×10^7, as shown in Table 4, all statistics of GA_{PSP} were better than those of GA^2_{PSP} and an improvement in E_{mean} occurred, increasing from 1.12 to 2. It is likely that GA_{PSP} obtained good *speed* for longer sequences as a result of the population diversity and the efficient calculation of the distance between the two individuals. When the nearest individual is calculated and the distance between the two individuals is greater than the minimal distance found so far, then the algorithm does not need to calculate an exact distance. In such a way, the distance calculation is more efficient for populations with higher diversity and because of this GA_{PSP} has an improved speed in comparison to other versions of our algorithm.

Table 5
Results for 10 sequences of 48 monomers and for 10 sequences of 64 monomers. Stopping condition was *MAX_NSEs* = 4×10^6 and number of runs was 50 for GA_{PSP} and GAHP. MA was limited with *MAX_NSEs* = 6×10^6 over 30 runs. The symbol * indicates that GA_{PSP} obtained a new best known energy value.

Seq.	E_n	GA_{PSP}				GAHP [12]			MA [9]		Seq.	E^*	GA_{PSP}				GAHP [12]		
		E_{best}	E_{mean}	E_{std}	SR	E_{best}	E_{mean}	E_{std}	E_{best}	SR			E_{best}	E_{mean}	E_{std}	SR	E_{best}	E_{mean}	E_{std}
T48.1	32	**32**	**31.82**	0.38	**0.82**	**32**	30.72	0.67	**32**	0.23	T64.1	32	**32**	**30.86**	0.60	0.12	31	28.50	1.10
T48.2	34	**34**	**33.08**	0.77	**0.34**	**34**	31.26	0.59	**34**	0.03	T64.2	38	**37**	**35.12**	0.71	0.00	36	33.18	1.22
T48.3	34	**34**	**33.26**	0.44	**0.26**	**34**	32.08	0.80	**34**	0.03	T64.3	45	**45**	**43.54**	0.57	0.04	44	41.88	0.87
T48.4	33	**33**	**32.22**	0.54	**0.28**	**33**	31.16	0.81	**33**	0.13	T64.4	42	**41**	**39.74**	0.59	0.00	39	36.02	1.39
T48.5	32	**32**	**31.58**	0.49	**0.58**	**32**	30.52	0.73	**32**	0.23	T64.5	42	**42***	**40.62**	0.72	0.08	40	37.96	1.12
T48.6	32	**32**	**31.18**	0.38	**0.18**	**32**	29.86	0.78	**32**	0.10	T64.6	35	**34**	**33.52**	0.50	0.00	33	31.52	0.86
T48.7	32	**32**	**30.62**	0.56	**0.04**	**32**	29.82	0.56	31	0.00	T64.7	28	**28**	**28.00**	0.00	1.00	**28**	26.70	0.70
T48.8	31	**31**	**30.38**	0.48	**0.38**	**31**	29.32	0.58	**31**	0.07	T64.8	38	**38***	**36.54**	0.54	0.02	36	33.72	0.85
T48.9	34	**34**	**33.02**	0.37	**0.08**	**34**	31.92	0.66	33	0.00	T64.9	41	**40**	**38.00**	0.60	0.00	38	36.32	0.93
T48.10	33	**33**	**32.28**	0.45	**0.28**	**33**	31.08	0.56	**33**	0.07	T64.10	31	**31**	**31.00**	0.00	1.00	**31**	28.90	0.88

Table 6
Results for 8 sequences with different lengths. 50 independent runs were performed for GA_{PSP}, EDA, and HGA. 100 independent runs were performed for MO + FR, and 20 for CMA. Under the runtime limit constraint of 12*h* three different grids were used and all runs were independent. E_{ci} represents the margin of error for 10–90% confidence interval.

Seq.	E^*	*MAX_NSEs* = 10^6					*MAX_NSE* = 5×10^6									*runtime* = 12 h					
		GA_{PSP}			MO + FR [21]		GA_{PSP}			HGA [16]			EDA [24]			GA_{PSP}			CMA [20]		
		E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{ci}
T20	11	**11**	**11.00**	0.00	**11**	**11.00**	**11**	**11.00**	0.00	**11**	10.52	0.54	**11**	10.82	0.38	**11**	**11.00**	0.00	**11**	**11.00**	0.00
T24	13	**13**	**13.00**	0.00	**13**	12.96	**13**	**13.00**	0.00	**13**	11.28	0.90	**13**	12.02	0.94	**13**	**13.00**	0.00	**13**	**13.00**	0.00
T25	9	**9**	**9.00**	0.00	**9**	**9.00**	**9**	**9.00**	0.00	**9**	8.54	0.64	**9**	8.96	0.19	**9**	**9.00**	0.00	**9**	**9.00**	0.00
T36	18	**18**	**18.00**	0.00	**18**	16.84	**18**	**18.00**	0.00	**18**	15.76	1.05	**18**	16.40	0.80	**18**	**18.00**	0.00	**18**	**18.00**	0.00
T48	31	**31**	**31.00**	0.00	**31**	27.39	**31**	**31.00**	0.00	28	24.60	1.57	29	27.24	0.92	**31**	**31.00**	0.00	**31**	**31.00**	0.00
T50	34	**33**	**31.66**	0.57	32	27.40	**33**	**32.62**	0.48	26	23.02	1.48	29	25.70	1.26	**34**	**33.96**	0.14	31	31.00	0.00
T60	55	**53**	**50.84**	0.67	50	44.45	**54**	**52.40**	0.66	49	41.18	2.75	49	46.30	2.04	**55**	**54.46**	0.50	54	52.52	0.20
T64	59	**58**	**55.52**	1.00	51	45.63	**59**	**57.62**	0.60	46	40.40	2.50	52	46.78	2.28	**59**	**59.00**	0.00	58	56.30	0.35

Table 8
Results for 5 longer sequences with different lengths. 50 independent runs were performed for GA_{PSP} and GAHP. 100 independent runs were performed for MO + FR.

Seq.	E^*	$MAX\_NSEs = 4 \times 10^6$					$MAX\_NSEs = 10^6$				
		GA_{PSP}			GAHP [12]		GA_{PSP}			MO + FR [21]	
		E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}	E_{best}	E_{mean}	E_{std}	E_{best}	E_{mean}
T46	35	**35**	**34.42**	0.49	**35**	33.04	**35**	**33.40**	0.60	32	28.92
T58	44	**44**	**41.92**	0.59	42	40.04	**42**	**40.78**	0.61	41	34.52
T103	58	**53**	**50.80**	0.99	50	46.58	**52**	**48.48**	1.27	40	35.35
T124	75	**68**	**64.78**	1.25	63	58.12	**65**	**61.48**	1.15	51	43.56
T136	83	**71**	**68.54**	1.20	70	62.22	**69**	**65.63**	1.47	55	46.94

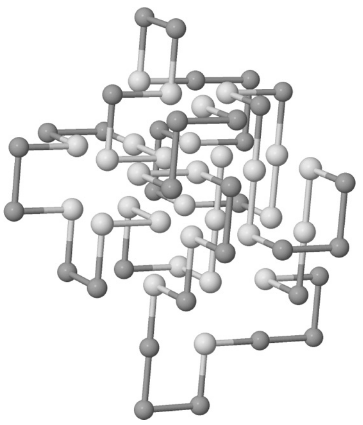
Fig. 7. New best known conformation for sequence T64.5: {*F, L, D, B, R, F, D, B, D, B, D, D, R, U, R, R, U, L, F, U, U, F, D, L, B, U, U, B, R, D, D, B, L, B, U, F, R, U, L, L, U, F, D, L, D, R, B, B, L, F, L, D, R, D, F, U, R, B, D, R, F, U, U*} with $E(s, c)^* - 1 = 42$.

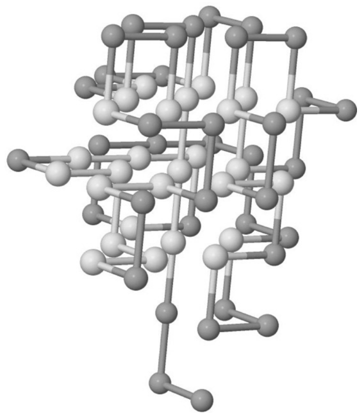
Fig. 8. New best known conformation for sequence T64.8: {*B, U, U, U, U, U, L, D, B, R, B, R, F, U, F, D, D, R, U, F, L, U, L, D, F, D, B, R, D, L, D, R, B, U, R, B, U, B, L, L, D, F, R, U, L, L, L, F, R, D, F, L, F, U, B, R, F, U, L, B, U, R, D*} with $E(s, c)^* - 1 = 38$.

6.3. Comparison with algorithms from the literature

Comparisons between the proposed algorithm and other state-of-the-art evolutionary and swarm algorithms are presented in this subsection. Note that in this section different statistics are reported for each algorithm because they are taken from the literature and not all statistics were reported for all algorithms.

First, we will compare our algorithm with those that have reported results for 10 sequences of 48 monomers (first test set) and 10 sequences of 64 monomers (second test set). The results are given in Table 5. The proposed algorithm successfully found all native energies (E_n) as well as GAHP for all sequences of 48 monomers. It outperforms GAHP with regards to mean energy values E_{mean} and MA with regards to *SR* for all sequences. We also analyzed our algorithm according to the *runtime* that is needed for obtaining native energies, as shown in Table 7. However, even though different CPUs were used in this comparison, it is evident that the proposed algorithm significantly outperforms ACO. Based on the observed results, we can conclude that GA_{PSP} always performed better on E_{mean}, *SR* and *runtime* when compared to other competitors on sequences of the first test set.

Similarly as seen in the previous comparison GA_{PSP} obtained the best results for sequences of 64 monomers, as shown in Table 5. In this comparison, our algorithm obtained not only better E_{mean} but also better E_{best}. Further more, although the algorithm was limited to only 4×10^6 sequence evaluations, it found the new best-known energy values (E^*) for T64.5 and T64.8 sequences. Conformations of these two sequences are shown in Figs. 7 and 8. From this we can also observe that in the four cases GA_{PSP} obtained better E_{mean} than the E_{best} obtained by GAHP. From the described results we can see that our algorithm is not only superior with regards to E_{mean} and *runtime* but also with regards to E_{best}.

In order to confirm efficiency of our algorithm, we compared its results with more algorithms on sequences with varying numbers of monomers. Table 6 shows the results of sequences with 20–64 monomers and three different stopping conditions. The first two stopping conditions were *MAX_NSEs* with values of 10^6 and 5×10^6 while the third was *runtime*, set to 12 hours. The shown statistics are based on 50 independent runs for GA_{PSP}, EDA, and HGA, on 100 independent runs for MO + FR, and 20 independent runs for CMA. Two different grids, Victorian Partnership for Advanced Computing[1] and Monash Campus, were used for CMA and Slovenian Initiative for National Grid[2] was used for GA_{PSP}. We can see from the obtained results that GA_{PSP} outperforms all competitors. Its superiority is particularly evident for higher sequences because it was able to find not only a better E_{mean} but also a better E_{best}.

The next comparison is based on higher sequences with different numbers of monomers, as shown in Table 8. Stopping conditions were *MAX_NSEs* with values of 10^6 and 4×10^6. Results show that GA_{PSP} obtained the best results. It is also interesting that with a four-times smaller number of sequence evaluations, the algorithm obtained better values of E_{mean} than GAHP. Further more, GA^1_{PSP} (see Table 3) without local search, opposition-based and clustering method also obtained better results than GAHP. This indicates that the way the genetic operators and the repair mechanism were used is also important for the algorithm's performance.

From the reported results of all four sequence sets we can conclude that GA_{PSP} outperformed all state-of-the-art evolutionary and swarm algorithms for protein structure prediction for HP sequences within a 3D cubic lattice.

1 Available at http://www.vpac.org.
2 Available at http://www.sling.si/sling/.

7. Conclusion

This paper proposed a genetic algorithm extended with advanced mechanisms and applied to a hydrophobic-polar (HP) model and cubic lattice for protein structure prediction. The proposed algorithm uses crowding, repair, clustering, local search, and opposition-based mechanisms. Crowding is responsible for maintaining the diversity of the population and for retaining the good conformations until the end of the evolutionary process. The clustering mechanism is used for dividing a whole population into a number of subpopulations that can locate different good conformations. The local search is used to improve convergence speed by trying to improve the quality of individuals with local movements of one or two consecutive monomers through the entire conformation. We define local movement as a transformation of protein conformation where only a few monomers, in our case one or two consecutive monomers, are locally moved from one lattice point to another while the remaining monomers remain on their lattice points. The mechanisms used improve individuals (conformations) from the beginning to the end of the HP sequence and with these mechanisms it is hard to improve good conformation at the beginning. For this purpose, our algorithm includes opposition-based mechanism that transforms individuals into their opposite directions and consequently facilitates the algorithm to find better conformations.

The introduced algorithm was analyzed and compared with state-of-the-art evolutionary and swarm algorithms on sequences that have been frequently used in the literature. The analyses of local search, clustering and opposition-based mechanisms show that they all successfully improved the efficiency of the algorithm. We also observe that the algorithm which includes all mechanisms obtained the best mean energy values and also had the highest diversity of final population. The reason for the higher diversity is because of the clustering method. However, clustering mechanism, which demands more distance calculations between individuals, increased the speed or number of sequences evaluation per second, which was not expected. This was likely to be because the distance calculation was more efficient in the case of the higher population diversity. It is not necessary to calculate the exact distance between two individuals when the distance is greater than the minimal distance found so far.

The proposed algorithm was compared to seven state-of-the-art evolutionary and swarm algorithms on the four sequence sets with varying numbers of monomers. The results show that the proposed algorithm outperformed all competitors. It superiority is especially evident for higher sequences. It improved not only the mean energy value but also the best obtained energy value and the required runtime to achieve the native energy.

Our future work will be directed towards analyzing the performance of the algorithm according to different protein structure prediction models and also to analyze the efficiency of different optimization algorithms. For example, we will analyze GA_{PSP}, binary PSO [32,33], angle modulated DE [34] and a novel variant of binary DE [35] for the HP model on cubic lattice, face centered-cubic lattice and on the 3D-HP side chain model [36].

Acknowledgements

Our work was supported by the Slovenian Research Agency under project P2-0041 (Computer Systems, Methodologies, and Intelligent Services).

References

[1] K.A. Dill, S.B. Ozkan, T.R. Weikl, J.D. Chodera, V.A. Voelz, The protein folding problem: when will it be solved? Curr. Opin. Struct. Biol. 17 (3) (2007) 342–346, http://dx.doi.org/10.1016/j.sbi.2007.06.001.
[2] A.E. Márquez-Chamorroa, G. Asencio-Cortésa, C.E. Santiesteban-Tocab, J.S. Aguilar-Ruiza, Soft computing methods for the prediction of protein tertiary structures: a survey, Appl. Soft Comput. 35 (2015) 398–410, http://dx.doi.org/10.1016/j.asoc.2015.06.024.
[3] D. Kennedy, C. Norman, Editorial: so much more to know, Science 309 (2005) 78–102, http://dx.doi.org/10.1126/science.309.5731.78b.
[4] K.F. Lau, K.A. Dill, A lattice statistical mechanics model of the conformational and sequence spaces of proteins, Macromolecules 22 (10) (1989) 3986–3997, http://dx.doi.org/10.1021/ma00200a030.
[5] B. Berger, T. Leighton, Protein Folding in the Hydrophobic-hydrophilic (HP) is NP-complete, in: Proceedings of the Second Annual International Conference on Computational Molecular Biology, ACM, New York, NY, USA, 1998, pp. 30–39, http://dx.doi.org/10.1145/279069.279080.
[6] F.H. Stillinger, T. Head-Gordon, Collective aspects of protein folding illustrated by a toy model, Phys. Rev. E 52 (1995) 2872–2877, http://dx.doi.org/10.1103/PhysRevE.52.2872.
[7] W.E. Hart, A. Newman, Protein structure prediction with lattice models, in: S. Aluru (Ed.), Handbook of Computational Molecular Biology, Chapman & Hall/CRC Computer & Information Science Series, 2005, http://dx.doi.org/10.1201/9781420036275.ch30, pp. 30-1–30-24.
[8] Y. Zhang, L. Wu, S. Wang, Solving two-dimensional hp model by firefly algorithm and simplified energy function, Math. Probl. Eng. (2013), http://dx.doi.org/10.1155/2013/398141.
[9] A. Bazzoli, A.G.B. Tettamanzi, A memetic algorithm for protein structure prediction in a 3D-lattice HP model, in: G.R. Raidl, S. Cagnoni, J. Branke, D.W. Corne, R. Drechsler, Y. Jin, C.G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G.D. Smith, G. Squillero (Eds.), Applications of Evolutionary Computing, Vol. 3005 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2004, pp. 1–10, http://dx.doi.org/10.1007/978-3-540-24653-4_1.
[10] W. Gao, G.G. Yen, S. Liu, A cluster-based differential evolution with self-adaptive strategy for multimodal optimization, IEEE Trans. Cybern. 44 (8) (2014) 1314–11327, http://dx.doi.org/10.1109/TCYB.2013.2282491.
[11] C. Cotta, Protein structure prediction using evolutionary algorithms hybridized with backtracking, in: J. Mira, J.R. Álvarez (Eds.), Artificial Neural Nets Problem Solving Methods, Vol. 2687 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2003, pp. 321–328, http://dx.doi.org/10.1007/3-540-44869-1_41.
[12] F.L. Custódio, H.J. Barbosa, L.E. Dardenne, A multiple minima genetic algorithm for protein structure prediction, Appl. Soft Comput. 15 (2014) 88–99, http://dx.doi.org/10.1016/j.asoc.2013.10.029.
[13] R. Unger, J. Moult, Genetic algorithms for protein folding simulations, J. Mol. Biol. 231 (1) (1993) 75–81, http://dx.doi.org/10.1006/jmbi.1993.1258.
[14] A.L. Patton, W.F. Punch III, E.D. Goodman, A standard GA approach to native protein conformation prediction, in: Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, pp. 574–581.
[15] M.M. Khimasia, P.V. Coveney, Protein structure prediction as a hard optimization problem: the genetic algorithm approach, Mol. Simul. 19 (4) (1997) 205–226, http://dx.doi.org/10.1080/08927029708024151.
[16] R. König, T. Dandekar, Improving genetic algorithms for protein folding simulations by systematic crossover, Biosystems 50 (1) (1999) 17–25, http://dx.doi.org/10.1016/S0303-2647(98)00090-2.
[17] N. Krasnogor, W.E. Hart, J. Smith, D.A. Pelta, Protein structure prediction with evolutionary algorithms, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), in: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 2, Morgan Kaufmann, Orlando, Florida, USA, 1999, pp. 1596–1601.
[18] F.L. Custódio, H.J. Barbosa, L.E. Dardenne, Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm, Genet. Mol. Biol. 27 (2004) 611–615, http://dx.doi.org/10.1590/S1415-47572004000400023.
[19] N. Mansour, F. Kanj, H. Khachfe, Evolutionary algorithm for protein structure prediction 2010 Sixth International Conference on Natural Computation (ICNC), vol. 8, 2010, pp. 3974–3977, http://dx.doi.org/10.1109/ICNC.2010.5584796.
[20] M. Islam, M. Chetty, Clustered memetic algorithm with local heuristics for ab initio protein structure prediction, IEEE Trans. Evolut. Comput. 17 (4) (2013) 558–576, http://dx.doi.org/10.1109/TEVC.2012.2213258.
[21] M. Garza-Fabre, E. Rodriguez-Tello, G. Toscano-Pulido, Constraint-handling through multi-objective optimization: the hydrophobic-polar model for protein structure prediction, Comput. Oper. Res. 53 (2015) 128–153, http://dx.doi.org/10.1016/j.cor.2014.07.010.
[22] A. Shmygelska, H. Hoos, An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem, BMC Bioinf. 6 (1) (2005) 30, http://dx.doi.org/10.1186/1471-2105-6-30.
[23] F. Kanj, N. Mansour, H. Khachfe, F. Abu-Khzam, Protein structure prediction in the 3D HP model, in: IEEE/ACS International Conference on Computer Systems and Applications, 2009. AICCSA 2009, 2009, pp. 732–736, http://dx.doi.org/10.1109/AICCSA.2009.5069408.
[24] R. Santana, P. Larranaga, J. Lozano, Protein folding in simplified models with estimation of distribution algorithms, IEEE Trans. Evolut. Comput. 12 (4) (2008) 418–438, http://dx.doi.org/10.1109/TEVC.2007.906095.
[25] K.-C. Wong, C.-H. Wu, R.K. Mok, C. Peng, Z. Zhang, Evolutionary multimodal optimization using the principle of locality, Inf. Sci. 194 (2012) 138–170, http://dx.doi.org/10.1016/j.ins.2011.12.016.

[26] V. Cutello, G. Nicosia, M. Pavone, J. Timmis, An immune algorithm for protein structure prediction on lattice models, IEEE Trans. Evolut. Comput. 11 (1) (2007) 101–117, http://dx.doi.org/10.1109/TEVC.2006.880328.
[27] C.J. Epstein, R.F. Goldberger, C.B. Anfinsen, The genetic control of tertiary protein structure: studies with model systems, Cold Spring Harb. Symp. Quant. Biol. 28 (1963) 439–449, http://dx.doi.org/10.1101/SQB.1963.028.01.060.
[28] K. Yue, K.M. Fiebig, P.D. Thomas, H.S. Chan, E.I. Shakhnovich, K.A. Dill, A test of lattice protein folding algorithms, Proc. Natl. Acad. Sci. U. S. A. 92 (1) (1995) 325–329.
[29] L. Toma, S. Toma, Contact interactions method: a new algorithm for protein folding simulations, Protein Sci. 5 (1) (1996) 147–153, http://dx.doi.org/10.1002/pro.5560050118.
[30] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977, http://dx.doi.org/10.1007/s00500-008-0392-y.
[31] F. Wilcoxon, Individual comparisons by ranking methods, Biom. Bull. 1 (6) (1945) 80–83, http://dx.doi.org/10.2307/3001968.
[32] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Progr. Nat. Sci. 18 (9) (2008) 1161–1166, http://dx.doi.org/10.1016/j.pnsc.2008.03.018.
[33] Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary PSO with mutation operator for feature selection using decision tree applied to SPAM detection, Knowl. Based Syst (2014), http://dx.doi.org/10.1016/j.knosys.2014.03.015.
[34] G. Pampara, A. Engelbrecht, N. Franken, Binary differential evolution, in: IEEE Congress on Evolutionary Computation, 2006 (CEC 2006), 2006, pp. 1873–1879, http://dx.doi.org/10.1109/CEC.2006.1688535.
[35] Y. Chen, W. Xie, X. Zou, A binary differential evolution algorithm learning from explored solutions, Neurocomputing 149 (Part B) (2015) 1038–1047, http://dx.doi.org/10.1016/j.neucom.2014.07.030.
[36] L.F. Nunes, L.C. Galv ao, H.S. Lopes, P. Moscato, R. Berretta, An integer programming model for protein structure prediction using the 3D-HP side chain model, Discret. Appl. Math. (2016), http://dx.doi.org/10.1016/j.dam.2015.06.021.

Borko Bošković received his B.Sc. and Ph.D. in computer science from the University of Maribor, Maribor, Slovenia, in 2004 and 2010. He is currently an Assistant Professor at the Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia. He has worked in the Laboratory for Computer Architecture and Programming Languages, University of Maribor, since 2000. His research interests include evolutionary computing, optimization, natural language processing and programming languages.

Janez Brest received his B.Sc., M.Sc., and Ph.D. in computer science from the University of Maribor, Maribor, Slovenia, in 1995, 1998, and 2000, respectively. He has been with the Laboratory for Computer Architecture and Programming Languages, University of Maribor, since 1993. He is currently a Full Professor and Head of the Laboratory for Computer Architecture and Programming Languages. His research interests include evolutionary computing, artificial intelligence, and optimization. His fields of expertise embrace programming languages, and parallel and distributed computing research.