framework to solve the PSP problem. The high level heuristic includes two main components: a selection mechanism and an acceptance criterion. Both components are generated using grammatical evolution (GE) [22], which is a kind of genetic programming (GP) [13]. A set of experiments was designed to evaluate the approach using eleven PSP instances. The results are compared with the previous works of the literature.

This paper is organized as follows: in the next section we briefly introduce the Protein Structure Prediction problem and a review of the related works is also presented. Section III presents a background of hyper-heuristics, GP and the GE algorithm. Thereafter, in Section IV, the GEHyPSP is introduced. In Section V, the experimental benchmark and numerical results of the conducted experiments are presented. Finally, in Section VI, the conclusions of the research are given, and further work is discussed.

II. Protein Structure Prediction

Proteins are macromolecules composed by an alphabet of twenty different amino-acids, also referred to as residues. An amino-acid is formed by a peptide backbone and a distinctive side chain group. The peptide bond is defined by an amino group and a carboxyl group connected to an alpha carbon to which a hydrogen and side chain group are attached.

The protein sequence folds, under particular conditions, into a unique native 3-D structure. Each possible protein conformation/structure has an associated energy value. The thermodynamic hypothesis states that the native structure of a protein is the one for which the free energy achieves the global minimum. Based on this hypothesis, many methods [6], [12], [14], [18], [28] that search for the protein native structure define an approximation of the protein energy and use optimization algorithms that look for the protein fold that minimize the energy. These approaches mainly differ in the type of energy approximation employed and in the characteristics of the protein modeling.

A. The HP Model

Lau and Dill [17] created a model called *Hydrophobic-Hydrophilic* Model (HP Model), to represent the proteins using simplifications. The model can be used either to represent proteins in a 2D space or 3D space.

The HP model considers only two types of residues: hydrophobic (H) and hydrophilic or polar (P) residues. A protein is considered a sequence of these two types of residues, which are located in regular lattice models and must form a self-avoided walk (SAW). If a structure is not SAW it means that it contains collisions between the amino-acids and this structure is invalid. Given a pair of residues, they are considered neighbors if they are adjacent either in the chain (connected neighbors) or in the lattice but not connected in the chain (topological neighbors). In the HP context the amino-acid sequences are used as instances to the problem.

For the HP model, an energy function that measures the amount of topological neighbor residues is defined as $\epsilon_{HH} = -1$ and $\epsilon_{HP} = \epsilon_{PP} = 0$. The HP problem consists of finding the solution that minimizes the total energy. In the linear representation of the sequence, hydrophobic residues are represented with the letter H and polar ones, with P. In the graphical representation, hydrophobic proteins are represented by black beads and polar proteins, by white beads. Figure 1 presents a possible graphical conformation for the sequence $PPPPHPHHHHPH$ in a 2D space. The energy associated with this conformation is -2.

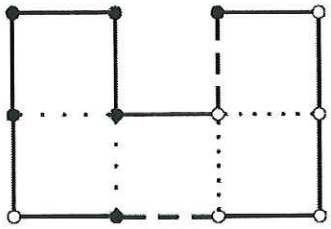

Fig. 1. One possible configuration of sequence $PPPPHPHHHHPH$ in the HP model. The white circles represents the P amino-acids and the black circles describes the H amino-acids. There is two $H-H$ (represented by a dotted line with wide spaces), one $H-P$ (represented by a dashed line) and two $P-P$ (represented by dotted lines) contacts.

Among many works that explores the PSP problem, here are some examples of the approaches that have been used to solve it.

A study presented by [14] introduces the MMA which consisted in an EA combined with a group of local search methods. For each individual in the population, the MMA, selects a local search method that is more suitable with the individual. Used first to find solutions for the functional model protein, the strategy was later improved with fuzzy-logic-based local searches, leading the algorithm to achieve improved results in the PSP problem.

In [12], Hsu et al. presents the pruned-enriched Rosenbluth method (PERM), also known as chain growth algorithm, that is based on growing the sequence conformation by adding particle by particle, aiming to increase good configurations and eliminating bad ones.

The ant colony optimization (ACO), in [26], [27], was applied to the PSP problem using the HP-2D model. This strategy, uses artificial ants to build conformations for a given HP instance (sequence of amino-acids). A local search method is then applied to further improve the solutions and also maintain the quality of the solutions.

The study of Santana et al. [24] applies EDAs as an efficient evolutionary algorithm that learns and exploits the search space in the form of probabilistic dependencies. They introduced new ideas for the application of EDAs to 2D and 3D simplified protein folding problem. The obtained results showed that EDAs can achieve superior solutions compared with other well-known population based optimization algorithms.

The present paper proposes the use of a grammatical evolution GE to generate high level heuristics for a hyper heuristic framework that will be applied to the PSP problem.

$$
\begin{array}{lll}
\langle expr \rangle & ::= \langle expr \rangle\ \langle op \rangle\ \langle expr \rangle & (0) \\
& \mid (\langle expr \rangle\ \langle op \rangle\ \langle expr \rangle) & (1) \\
& \mid \langle pre\text{-}op \rangle\ (\langle expr \rangle) & (2) \\
& \mid \langle var \rangle & (3) \\
\\
\langle op \rangle & ::= + & (0) \\
& \mid - & (1) \\
& \mid / & (2) \\
& \mid * & (3) \\
\\
\langle pre\text{-}op \rangle & ::= \text{Sin} & (0) \\
& \mid \text{Cos} & (1) \\
& \mid \text{Tan} & (2) \\
\\
\langle var \rangle & ::= \text{X} & (0)
\end{array}
$$

Grammar 1. Sample grammar to demonstrate how to decode integer vectors in computer programs

programs using the GF (Grammar File). Those programs are evaluated and a fitness value is assigned for each individual in the population. Line 7 represents the main loop of the algorithm and the evolution process occurs within this loop. Line 8 represents the parent selection that will be used in the crossover in Line 9 which will generate an offspring (integer vectors). The offspring is then used as input to prune and duplication operators in Lines 10, 11. The mutation operator is applied into the offspring in Line 12. The mapping between the integer vectors to programs is represented in Line 13. Next the execution and fitness assignment to the individuals from the population are presented in Line 14 and 15. The population replacement with the fittest individuals is shown in Line 16. Finally, line 18 return the program with the best fitness found.

Algorithm 1: Pseudo code from the Grammatical Evolution

```
1  Input: GF – Grammar File
2  begin
3      population ← Create population;
4      programs ← Maps the population to programs using GF;
5      Execute the program;
6      Assign fitness value to the solutions of population according with
       the output obtained by the respective decoded program;
7      while Stop condition not reached do
8          parents ← Select individuals for crossover;
9          offspring ← Crossover(parents);
10         Apply the Prune operator to the offspring;
11         Apply the Duplicate operator to the offspring;
12         Apply the Mutation operator to the offspring;
13         programs ← Maps offspring to programs using GF;
14         Execute programs;
15         Assign fitness value to solutions offspring according with
           the output obtained by the respective decoded program;
16         population ← Replacement;
17     end
18     return Best program from the population;
19 end
```

For the sake of comprehension the chromosome mapping process will be illustrated using the Grammar 1. The Algorithm 2 presents the general template of the generated programs. The expression $\langle expr \rangle$ presented on line 2 is replaced by mathematical expressions coded by the chromosomes (integer vectors).

Algorithm 2: General template for the generated algorithms

```
1 float symb(float x)
2 a = ⟨expr⟩;
3 return a;
```

Now suppose the following integer vector:

$$[220, 203, 17, 6, 108, 215, 104, 30]$$

This vector will be used to decode the chromosome (genotype) into a piece of code (phenotype) using the Grammar 1. Table I summarizes the number of choices associated within each production rule from Grammar 1.

There are 4 options of production rules that can be selected for the expression $\langle expr \rangle$. In order to select which option, the first value from the vector should be used. The value is 220 and its remainder divided by 4 (four options that can be selected for the expression $\langle expr \rangle$) results in 0, which means that the first option should be selected $\langle expr \rangle \langle op \rangle \langle expr \rangle$. Note that the first expression is also $\langle expr \rangle$ and following the same logic we should expand using the next value from the integer vector and applies its remainder by the division of the number of options. The remainder of 203 % 4 = 3, which indicates that we should select the fourth option: $\langle var \rangle$ which is a terminal. The $\langle var \rangle$ has just one option associated with it and is the value X. Placing this selection in the original expression we have $X \langle op \rangle \langle expr \rangle$.

Next it is necessary to decode the non-terminal expression $\langle op \rangle$. The next value from the integer vector is 17 and again we have 4 options $(+| - |/|*)$. The result is equal to 1, and indicates to select the: $-$. Re-writing the expression we got: $X - \langle expr \rangle$. This process should continue until all the non-terminals be expanded to terminals. In this example the resulting expression (phenotype) is: $X - Sin(X)$. Note that not all of the genes were necessary to obtain the phenotype. In these cases, the genes that were not used are discarded. Moreover, the opposite case can occur: if a chromosome does not contain the necessary number of genes to map to a program. In this scenario the strategy is re-utilize the genes starting from the first one.

IV. GEHyPSP

In this section will be presented the GEHyPSP: an off-line grammatical evolution application for generating high level heuristics of hyper-heuristics framework to the Protein Structure Prediction problem. The approach presented next is based on the study by Sabar et al. [23]

The high level heuristics are composed by a selection mechanism and an acceptance criterion. These high level

2) Low Level Heuristics Set: The low level heuristics set was selected from previous studies [1], [7], [6], [10] that explore the PSP. It consists of the following operators:

- *Two Points Crossover* (2X): This operator selects, randomly, two crossing points splitting the individuals in 3 pieces. The genes between the selected positions are exchanged between the parents in order to generate the offspring. [1].
- *Multiple Points Crossover* (MPX): Similar with 2X although with c points of crossing. We use $c = int(n*0.1)$, ..., where n is the sequence length. The MPX is useful to promote a wide structural diversity [23].
- *Segment Mutation* (SMUT): It changes a random number (5 to 7) of consecutive genes to distinct values. This heuristic introduces large changes in the protein conformation, and it has a great probability of creating collisions (cases in which the protein conformation are not self avoiding).
- *Exhaustive Search Mutation* (EMUT): This heuristic selects a random gene and modifies it to other possible value. All possible values are tried and the one whom achieved the higher improvement will be kept. This operator demands four fitness evaluations instead of just one. However, this heuristic has great potential of improving the input solution.
- *Local Move Operator* (LM): This heuristic exchanges directions between two consecutive random selected genes. This heuristic introduces small local changes that can reduce the distance between pairs of H-H.
- *Loop Move Operator* (LPM): Similar with LM, this heuristic exchanges directions between two genes that are five genes of distance between each other.
- *Opposite Mutation* (OM): This heuristic exchanges multiple between two genes (i, j) to the respective opposite. In the HP-2d this means that genes that code L are changed to R or vice and versa. The F has no opposite so, the operator is not applied.

3) Backtrack Repair: The low level heuristics have a great potential of creating solutions with collisions [1]. Those solutions are submitted to repair attempt using a backtrack strategy. The solutions that can be repaired are kept and the others are penalized.

4) Memory Mechanism: Sabar et al. [23] suggested that the use of a memory of solutions to the PSP would be more effective than relying on a single solution and may restrict the ability of dealing with large and heavily constrained search spaces. Also, Blum et al. [2] mention that single solution based methods are widely known to not cope with large search spaces and heavily constrained problems.

V. Experiments

In this section, we will present and discuss the conducted experiments in order to evaluate the GEHyPSP proposed by this paper. Three groups of experiments were designed/executed and will be described next. The first group was only concerned on generating selection mechanisms. The second group was designed only to generate acceptance criteria. The third group was developed to generate both selection and acceptance mechanisms. All experiments were executed 30 times because of the stochastic behavior of the GEHyPSP. The results obtained by the groups were compared with each other. Also a comparison with a state-of-art hyper-heuristic will be presented. The results were compared with the Generic Intelligent Hyper-heuristic (GIHH) presented by Misir el al [19].

TABLE II
Instances and the respective size of each one

Inst	1	2	3	4	5	6	7	8	9	10	11
Size	20	24	25	36	48	50	60	64	85	100	100

In the first group of experiments (GEHyPSP-1) only selection mechanisms are generated. The acceptance criterion was fixed with a "better or equal" acceptance [4]. The goal of this group is to evaluate the ability of our approachx to generate selection mechanisms using a fixed acceptance criterion.

The second group of experiments, GEHyPSP-2, consisted on generating only acceptance criteria. The selection mechanism was fixed using the best selection mechanism found in the first group. Consequently, this group of experiments depends on the output from the first group. The goal of this experiment was to evaluate the generation of acceptance criteria separately from the selection mechanism using a fixed one.

The third group of experiments GEHyPSP-3 was designed to generate both selection mechanisms and acceptance criteria. The goal of this group was evaluate the ability of generating selecting mechanisms along with acceptance criteria. Differently from GEHyPSP-2, this group of experiment does not depend on any output from previous experiments since it generates both mechanisms without fixing any component.

For each group of experiments eleven instances (amino-acid sequences from the HP model) selected from the previous studies with the PSP problem [6], [?], [18], [24], [7]. The sizes (number of residues) of the HP instances used in the experiments are presented in table II. For the sake of space, the sequences are not reproduced here. They can be obtained from [24]. Also, for each group of experiments the training phase consisted on executing the GE process with three ramdonly selected instances from the eleven available instances. In the validation phase the best generated selection mechanisms and acceptance criteria were executed together against all eleven instances.

A. Results from GEHyPSP-1

The best individual found in the GEHyPSP-1 was the following selection mechanism: $RC * Ccurrent * Cava - Cr$ and it was executed 30 times with a time limit of 10 minutes against the eleven instances. Table III presents the average, standard deviation, minimum and maximum of 30 executions of the best individual found in the GEHyPSP-1 experiment. The last row denoted with O(x*) is the best known value for

were always accepting worst solutions just like the "all moves" described by Burke et al. [3]. Finally, one individual was never accepting any solution.

These experiments showed that the GE managed, several times, to find different acceptance criteria with the same behavior of human-designed strategies.

VI. Conclusion

In this work the GEHyPSP, an automatic way of generating high level heuristics to a hyper-heuristic framework for the PSP problem, was presented and evaluated. The PSP is a very challenging problem with a high number of local optima and a very complex landscape. Many authors explored the PSP problem with heuristic methods. However, very often the proposed heuristic approaches are unable to find the best known results when executing against longer sequences. Usually, the hyper-heuristic frameworks fits well in this kind of complex scenario. Hence, the goal of this paper was to generate, using a grammatical evolution strategy, selection mechanisms and acceptance criteria to a hyper-heuristic framework and evaluate its performance and behavior with a set of eleven HP instances. Three groups of experiments were executed, using three randomly selected HP instances, in order to generate the high level heuristics and later the best individuals found in the experiments were executed using all the eleven HP instances.

Three groups of experiments were executed: first generating only selection mechanisms within a fixed acceptance criterion; second generating only acceptance criteria using the best selection mechanism found in the first; finally both high level heuristics were generated in parallel. The results showed that better high level heuristics were found when generating them separately. Unfortunately, when analyzing the behavior of the generated high level heuristics against the eleven instances it was possible to see that they were not able to achieve the best known results for the longer sequences. However, when comparing with a good state-of-art human-designed hyper-heuristic framework (GIHH) [19] the results are slightly close. This fact shows that it is possible to automate the creation of high level heuristics and obtain results close to the state-of-art hyper-heuristics frameworks.

Another finding of this work was the behavior of the best generated acceptance criteria. It was noticed that it behaves just like "a better or equal" human-designed move acceptance strategy. Also some of the generated acceptance criteria were always accepting worst solution and this fact impacted in the individual fitness. This fact demonstrates that the GEHyPSP was able to generate acceptance criteria with the same behavior of simple human-designed move acceptance strategies. However, in order to obtain better results it might be necessary to improve the GEHyPSP to generate more complex selection mechanisms and acceptance criteria to couple with the landscape complexity of the PSP problem.

References

[1] César Manuel Vargas Benítez. Um algoritmo genético paralelo para o problema de dobramento de proteínas utilizando o modelo 3dhp com cadeia lateral. 2010.

[2] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011.

[3] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[4] Edmund K Burke, Mathew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence*, pages 177–201. Springer, 2009.

[5] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.

[6] Fábio L Custódio, Hélio JC Barbosa, and Laurent E Dardenne. Investigation of the three-dimensional lattice HP protein folding model using a genetic algorithm. *Genetics and Molecular Biology*, 27(4):611–615, 2004.

[7] Fábio Lima Custódio, Helio JC Barbosa, and Laurent Emmanuel Dardenne. A multiple minima genetic algorithm for protein structure prediction. *Applied Soft Computing*, 15:88–99, 2014.

[8] Ian W Davis and David Baker. Rosettaligand docking with full ligand and receptor flexibility. *Journal of molecular biology*, 385(2):381–392, 2009.

[9] Candida Ferreira and U Gepsoft. What is gene expression programming, 2008.

[10] Mario Garza-Fabre, Eduardo Rodriguez-Tello, and Gregorio Toscano-Pulido. Multiobjectivizing the hp model for protein structure prediction. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 182–193. Springer, 2012.

[11] Mario Garza-Fabre, Gregorio Toscano-Pulido, and Eduardo Rodriguez-Tello. Locality-based multiobjectivization for the hp model of protein structure prediction. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 473–480. ACM, 2012.

[12] Hsiao-Ping Hsu, Vishal Mehra, Walter Nadler, and Peter Grassberger. Growth algorithms for lattice heteropolymers at low temperatures. *The Journal of chemical physics*, 118(1):444–451, 2003.

[13] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[14] Natalio Krasnogor, BP Blackburne, Edmund K Burke, and Jonathan D Hirst. Multimeme algorithms for protein structure prediction. In *Parallel Problem Solving from NaturePPSN VII*, pages 769–778. Springer, 2002.

[15] Natalio Krasnogor, W Hart, Jim Smith, and D Pelta. Protein structure prediction with evolutionary algorithms. 1999.

[16] Elmar Krieger, Keehyoung Joo, Jinwoo Lee, Jooyoung Lee, Srivatsan Raman, James Thompson, Mike Tyka, David Baker, and Kevin Karplus. Improving physical realism, stereochemistry, and side-chain accuracy in homology modeling: four approaches that performed well in casp8. *Proteins: Structure, Function, and Bioinformatics*, 77(S9):114–122, 2009.

[17] Kit Fun Lau and Ken A Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22(10):3986–3997, 1989.

[18] Cheng-Jian Lin and Shih-Chieh Su. Protein 3 d hp model folding simulation using a hybrid of genetic algorithm and particle swarm optimization. *International Journal of Fuzzy Systems*, 13(2):140–147, 2011.

[19] Mustafa Misir. Intelligent hyper-heuristics: a tool for solving generic optimisation problems. 2012.

[20] Gabriela Ochoa, Matthew Hyde, Tim Curtois, Jose A Vazquez-Rodriguez, James Walker, Michel Gendreau, Graham Kendall, Barry McCollum, Andrew J Parkes, Sanja Petrovic, et al. Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary computation in combinatorial optimization*, pages 136–147. Springer, 2012.

[21] Bin Qian, Angel R Ortiz, and David Baker. Improvement of comparative model accuracy by free-energy optimization along principal components of natural structural variation. *Proceedings of the National Academy of Sciences of the United States of America*, 101(43):15346–15351, 2004.

[22] Conor Ryan, JJ Collins, and Michael O Neill. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming*, pages 83–96. Springer, 1998.

[23] Nasser R Sabar, Masri Ayob, Graham Kendall, and Rong Qu. Automatic design of a hyper-heuristic framework with gene expression program-

Is this PhD or Master thesis.

3dHP

Add institution