Genetics and Molecular Research

Generic framework for mining cellular automata models on protein-folding simulations

N. Diaz[1,2] and I. Tischer[2]

[1]Departamento de Sistemas, Universidad del Cauca, Popayán, Colombia
[2]Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Cali, Colombia

Corresponding author: N. Diaz
E-mail: nediaz@unicauca.edu.co

Genet. Mol. Res. 15 (2): gmr.15028654
Received March 23, 2016
Accepted April 11, 2016
Published May 13, 2016
DOI http://dx.doi.org/10.4238/gmr.15028654

ABSTRACT. Cellular automata model identification is an important way of building simplified simulation models. In this study, we describe a generic architectural framework to ease the development process of new metaheuristic-based algorithms for cellular automata model identification in protein-folding trajectories. Our framework was developed by a methodology based on design patterns that allow an improved experience for new algorithms development. The usefulness of the proposed framework is demonstrated by the implementation of four algorithms, able to obtain extremely precise cellular automata models of the protein-folding process with a protein contact map representation. Dynamic rules obtained by the proposed approach are discussed, and future use for the new tool is outlined.

Key words: Protein folding; Cellular automata; Data mining; Architectural framework; Metaheuristics; Molecular dynamics

INTRODUCTION

Protein folding is a highly complex process, understanding of which is useful in protein structure prediction. Simplified models are advantageous to shed light on complex phenomena that are difficult to understand and describe (Kolinski, 2004; Sharma et al., 2008). Furthermore, even with the use of simplified models, the modeling of a complex phenomenon such as protein folding is limited by lack of knowledge regarding the complicated dynamics of the protein-folding process.

Cellular automata (CA) models (Wolfram, 2002) are some of the most widely used simplified models in various fields, including bioinformatics (Xiao et al., 2011). CA models are characterized by a simple formalism that allows the representation of dynamic systems through a discrete space (lattice) and a set of rules that describes dynamic behavior in terms of local interactions. The utilization of CA models is wide, ranging from social phenomena to atomic processes in biochemical systems (Ganguly et al., 2001).

Complex phenomena modeling with CA can be done with existing knowledge (direct design) or by modeling based on data mining and machine learning techniques (CA mining or inverse design). In recent years, several algorithms for CA mining have been proposed (Rabino and Laghi, 2002), which can generally be divided into two categories: 1) data-driven (data mining approaches), and 2) those that explore a cellular automata search space (machine learning). The latter approach has undergone further development, because the only necessary elements are an initial configuration and the final desired configuration; the search process [usually metaheuristic-based (Luke et al., 2009)] attempts to identify an optimal cellular automata model capable of creating a simulation from the initial configuration, arriving at the desired final configuration through consecutive steps. The major disadvantage of this approach is that the success of the search rests on the selected search space; if the defined search space does not contain high quality models, the search process will fail.

Data-driven CA identification attempts to identify a fairly accurate model by a search process that includes the addition of knowledge derived from known trajectories. This approach is used when the intermediate global states are known, in additional to the initial and final configurations. The knowledge of dynamic transitions subjacent to the intermediate steps is translated into rules for the CA. A limitation of this approach is the rarity of known intermediate configurations. For protein folding, this kind of data is taken from other simulation techniques, such as molecular dynamics (Karplus and Kuriyan, 2005; Wang et al., 2014).

Contact map (CM) prediction has been acknowledged as a useful tool in protein structure prediction (Gupta et al., 2005). CMs represent the overall structure of the three-dimensional (3D) conformation of a protein through a binary matrix indicating close positioning of each pair of amino acids.

In this paper, we propose that with discrete representation of the protein structure provided by contact maps, it is possible to establish dynamic rules of the conformational transitions in a molecular dynamic protein-folding simulation. A CM is similar to a global configuration of a CA; therefore, a simulated protein-folding trajectory in a CM representation will be similar to a CA simulation trajectory. With this in mind, the focus of this work was to obtain a CA simulation of the protein-folding process without the computational cost of molecular dynamics.

Our approach is innovative in two main aspects. First, the generality of the proposed framework allows experimentation with diverse metaheuristics and novel approaches,

uncommon in CA identification. Secondly, the devised use of CA models as a representation of CMs supports the possibility of a protein-folding simulation in a CM space.

The document is organized as follows: section 1 describes the architecture of the framework; section 2 displays the utilization of the framework in the design of a genetic algorithm (GA), a co-evolutive algorithm, a tabu search (TS), and a hybrid algorithm of genetic algorithm and tabu search; and details the CA obtained by applications developed by the framework. Finally, the results are reviewed and further work is proposed.

Description of the framework

The proposed framework was developed under the Unified Modeling Language (UML) profile, UML-F (Fontoura et al., 2000). UML-F includes stereotypes and tagged values that enhance the capabilities of UML with new notational elements, enhancing the expressiveness of the architectural diagrams.

UML-F is complemented by detailed guidelines that describe the steps a developer will follow to implement an application based on a generic framework. This documentation includes a "cookbook" with step-by-step recipes for the design and definition of the specific variation points.

UML-F guides describe the framework's version of most common design patterns (Gamma et al., 1995). The use of design patterns allows developers to identify a standardized method to express a reusable design in a well-known design artifact.

Architecture

For design of the framework, we took into account the commonalities of the search process for metaheuristics. A metaheuristic approach implements a strategy of iterative search that explores the search space by tweaking best solutions (i.e., CA models). The best CA models were selected by a metric-based evaluation process, which depends on the results of CA simulation versus the evidences in a folding trajectory. To evaluate the behavior of the search process, a report was required to record scores for each iteration of the search process. Some metaheuristics optimize a single solution, while others diversify a set of solutions (population); it is also possible to accomplish both with the same algorithm.

Figure 1 schematizes the general generic architecture for the proposed framework in a high-level UML class diagram. The <<Application>> stereotype denotes the parts that are not provided by the framework and therefore must be defined by the developer. The <<Framework>> stereotype identifies the functionality considered in the framework. Additionally, the classes are annotated by stereotypes that indicate the selected design pattern for each framework component.

The Strategy Run class was not included in the framework; this class specifies the conditions for a run of the algorithm, therefore the developer must define several aspects that specify the details of the algorithms to be built. However, we provide some indications about how the components of the framework will be integrated.

Variation points

In the guided process for the development of the new algorithm, variation points were

determined in relation to the structural components. Logical steps were then defined to ease the implementation of classes. The variation points are defined below.

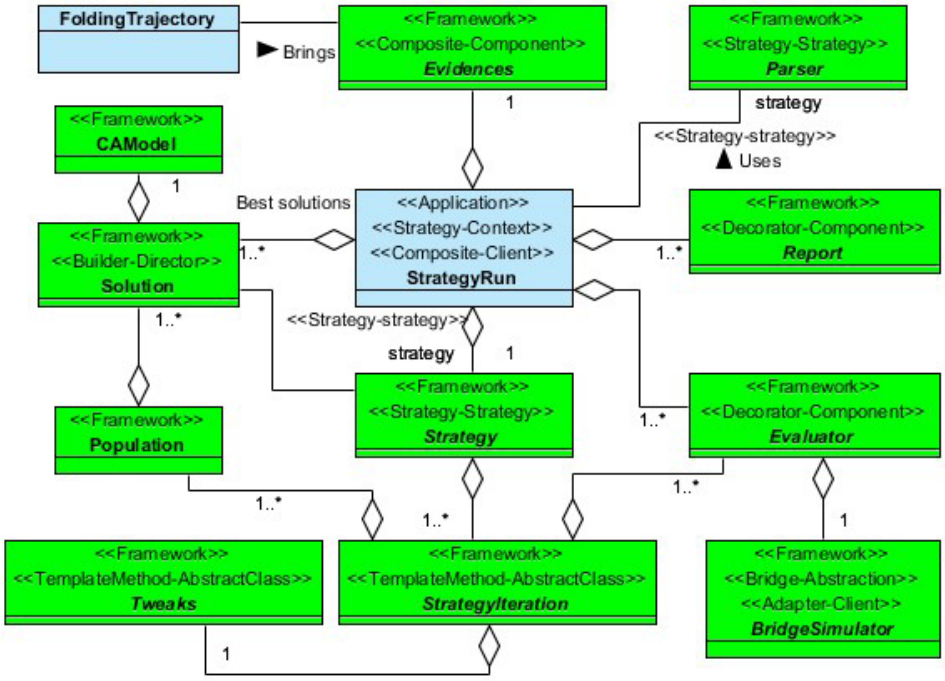

Figure 1. General architecture of the proposed framework. All classes are needed to successfully implement metaheuristics. <<Framework>> stereotype identify classes that are provided in the framework; <<Application>> stereotype identifies classes that will be defined in the new algorithm.

Strategy

Strategy was implemented with the design pattern strategy. Figure 2A illustrates the structure of the variation points. The classes at this variation point were as follows:

Strategy Run: The context of the pattern strategy.

Strategy: An abstract class that meets the strategy specifications, and hence defines the interface methods (start and stop criteria) to be implemented in the new algorithm (Please refer to the Concrete Strategy demonstrated in Figure 2A).

Strategy Iteration: Part of the Strategy class, based on the pattern design template; the template method is stepwise and was intended to perform adjustments in the solutions (CAs) of the population.

Tweaks

The design pattern template method was used to create this variation point; Figure 2B illustrates the structure. The classes at this variation point were as follows:

Tweaks: The abstract class for the pattern; the doTweaks method, a template method, was followed in the Strategy Iteration container class to invoke the primitive methods, selection

and search. In these methods, a selection process must be implemented to determine the best CAs for the diversification or intensification in the algorithm search space. The search method uses the selected population (by method selection) and performs variations and combinations, obtaining the new population to be utilized in the next iteration of the algorithm.

Strategy Iteration: Contains an instance of the Tweaks class; it calls the doTweaks template method (class Tweaks) in its step method.

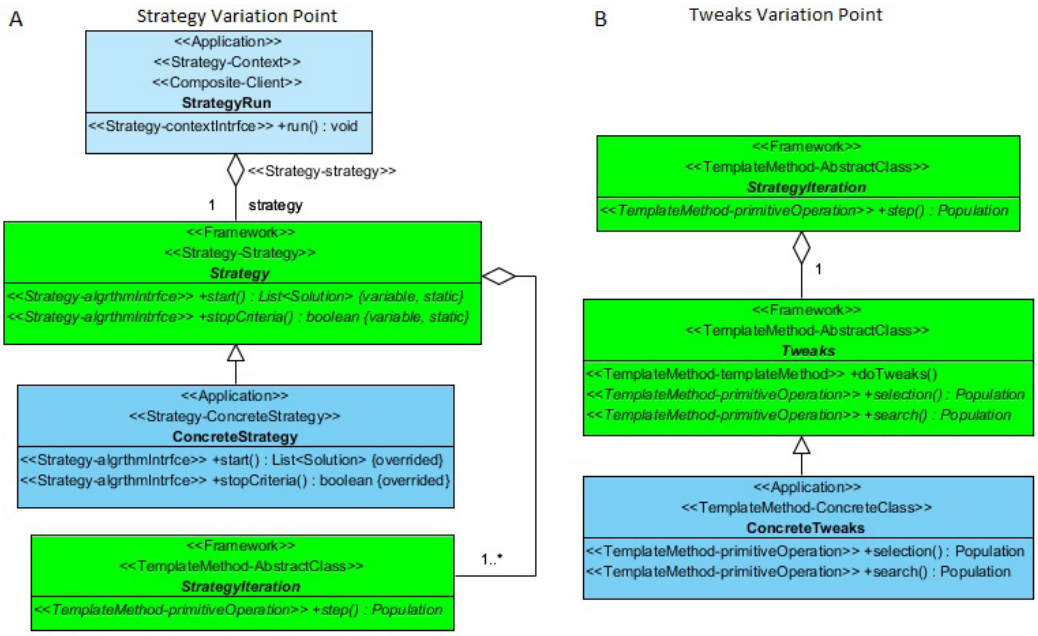

Figure 2. Specific variation points for the new algorithm. **A.** Strategy variation point. **B.** Tweaks variation point.

Solution

The solution construction process was defined at this variation point, i.e., the representation to be used for the CA at each solution and its respective evaluation scores were determined, taking into account the responsibilities of the selected pattern design (Figure 3A). The classes at this variation point were as follows:

Solution: The role director of the pattern; the construct method calls the methods that build the elements that compose the solution in the class with builder role.

Solution Builder: The builder role of the design pattern; its build Score and build Representation methods must be overridden in a concrete class of the algorithm under development.

CA Model: Part of the solution; the goal of the algorithm was to find a CA model.

Evaluator

The decorator design pattern allows the creation of single or composed evaluators. In

some algorithms, a score based on the number of hits or mathematical expressions was based on the results of the previous evaluations. Figure 3B illustrates the structure of this variation point. The classes at this variation point were as follows:

Evaluator: The base class for building of the evaluators; the evaluate method performs the mathematics behind the evaluation; it was overridden in the derived classes.

Concrete Evaluator: A class defined in the algorithm under development; the evaluate method performs evaluation of the simulation trajectory versus the evidences used for the data mining process of the CA rules.

Component Evaluator: The class structure to be utilized when the evaluation is dependent on the evaluation implemented in an existing evaluator, possible through class composition.

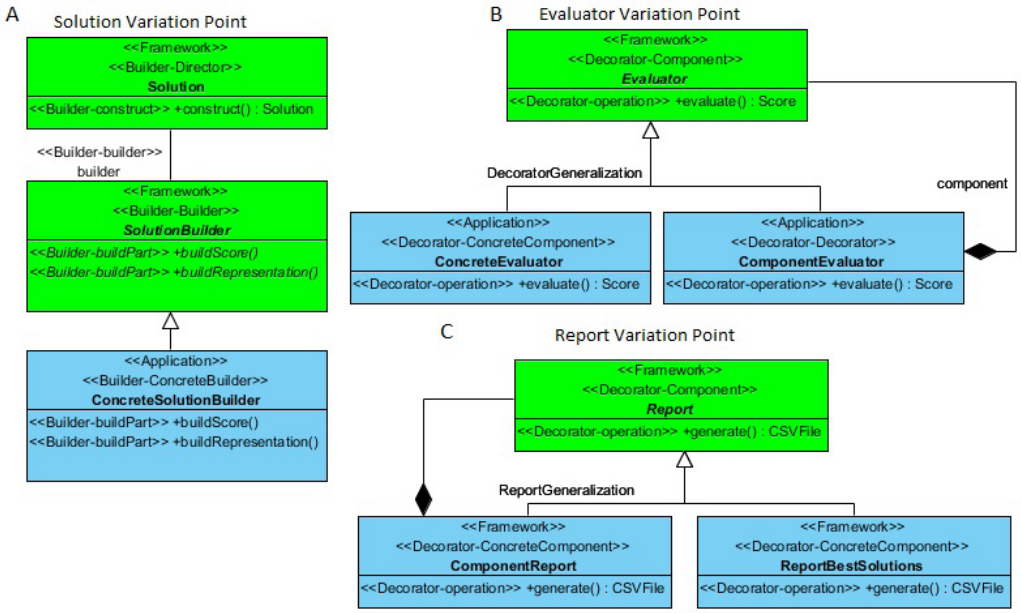

Figure 3. Solution-related variation points in the new algorithm. **A.** Solution variation point. **B.** Evaluator variation point. **C.** Report variation point.

Report

Based on the design pattern decorator, the report variation point allows recording of the statistics collected during running of the algorithm. The structure was intended to allow the utilization and design of reports that utilize previous or new reports. Figure 3C illustrates the structure of report in the framework. The classes that comprise this variation point were as follows:

Report: Definition of the method to be overridden in the derived classes; the generate method allows flexible implementation of reports according to the needs of the algorithm.

Component Report: Allows the composition of reports in order to develop more complex reports based on existing reports.

Report Best Solutions: The basic report, which records the scores for the best solutions of each iteration, fulfilling the basic needs of many algorithms.

Simulator

The simulator variation point had a broad implementation in the framework; a full CA simulator was implemented, which can be used with a graphical interface (visual simulation) or simply for the generation of a simulation trajectory for evaluation and analysis. Figure 4 demonstrates the structure based on the design pattern bridge and adapter. The classes at this variation point were as follows:

Bridge Simulator: Defines the method and increases adaptability of the different simulators to be used (start method). Every simulator must determine the bridge operation method to be either a bridge simulator implementation or an adapter implementation.

Simulator: Provides the interface to be implemented in the simulators (bridged or adapted).

CA Simulator: implements a bridged simulator that simulates a CA and records the trajectory.

CA Graphic Simulator: Implements a bridged simulator that performs the simulation in graphical view and records the trajectory.

External Simulator: Allows the utilization of external simulation tools through an adapter implementation.

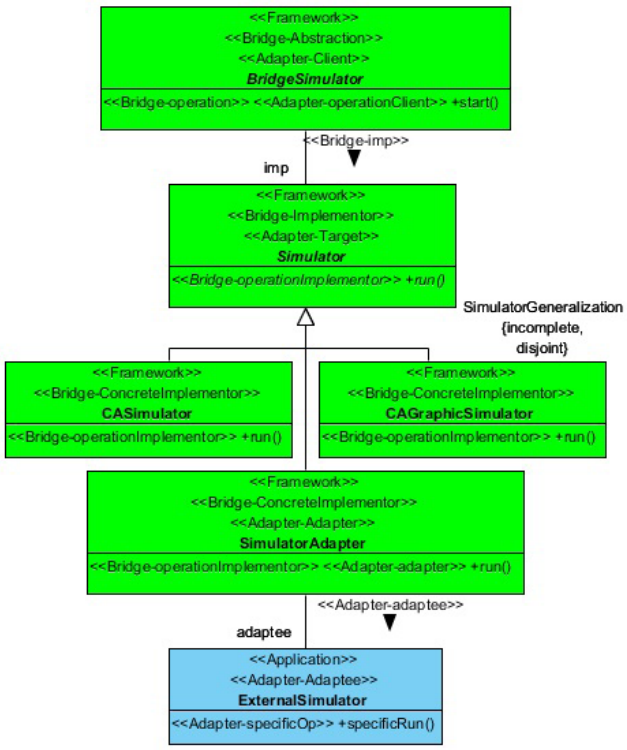

Figure 4. Simulator variation point.

Evidences

The evidences variation point allows data input after preprocessing. This variation point was designed with the design pattern composite, taking into account the capability of the complex structure definitions to manipulate data. The behavior of this framework component was dependent on the parser functionality. Figure 5A demonstrates the architecture of the structure. The classes at this variation point were as follows:

Evidences: Defines the methods that are mandatory in the composition; the get Evidences method processes the input trajectory simulation and formats the data in a lattice (matrix) representation. The load Evidences method obtains the previously processed and formatted evidence files.

Leaf Evidences: The minimal structure at the composite pattern, it implements a list of files wherein each file is a consecutive global configuration in a lattice format (e.g., a 2D matrix for a 2D CA).

Composite Evidences: Allows the implementation of complex structures by composition of previously defined composites or leafs.

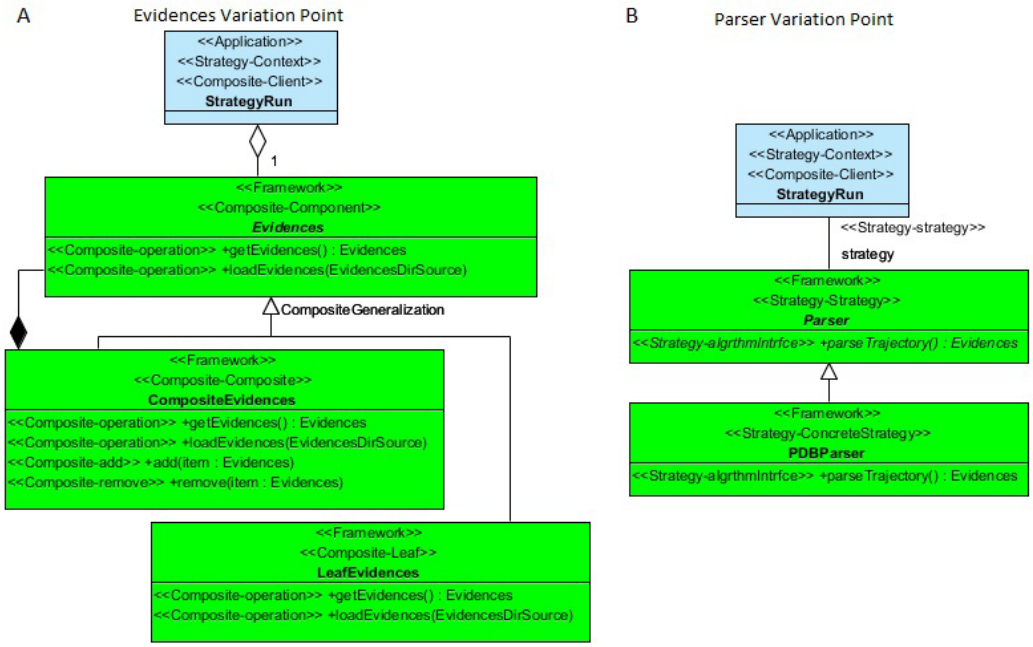

Figure 5. Evidences - Parser variation points. **A.** Evidences variation point. **B.** Parser variation point.

Parser

The parser variation point was designed with the pattern strategy and algorithm interface; the parse Trajectory method was implemented in each derived class. The main script of the application must provide the context of the strategy; the structure is illustrated in Figure 5B. The classes at this variation point were as follows:

Parser: Defines the interface for the design pattern strategy; the parse Trajectory method processes the input trajectory and must be implemented in derived classes, creating a set of formatted evidence files.

PDB Parser: An implementation provided by the framework, it processes an input trajectory simulation in PDB format (Berman et al., 2007), a common format for protein-folding simulation tools.

Examples of framework instantiation

The proposed framework was tested during the development of several algorithms. The application selected involved the identification of CA models in the CM space. A CM (Vendruscolo and Domany, 1998) is a suitable representation of a 3D protein structure, because it eliminates the effects of rotations and alignments of the structure, making it possible to reconstruct the 3D structure from the CM (Vassura et al., 2008). The CM of a protein structure is a binary 2D matrix, in which 0 represents non-adjacency between a pair of amino acids (AA) and 1 represents adjacency. The adjacency is defined in a threshold (e.g., [4Å, 7 Å] or <8Å). If the distance between an AA pair is out of range, it is a non-contact, otherwise, a contact.

The goal of the developed instances of the framework (algorithms) was to explore the search space of 2D binary CAs for a model that represents a dynamic behavior similar to that of the input protein-folding trajectory. The algorithms developed were two evolutionary metaheuristics, a GA (Goldberg, 1999) and a co-evolutive algorithm (Yoo et al., 2006), a tabu search algorithm (Glover and Laguna, 1999) (metaheuristic non-population-based), and a hybrid metaheuristic (GA plus tabu search). Evolutionary metaheuristics are commonly used in CA identification, GAs being used for this task, and therefore, we selected it in our study. Single model metaheuristics (e.g., TS) are not popular in CA identification, although we chose it as a test case for framework generality.

Below, we describe the design of each algorithm as well as the classes that were required in their implementation.

Genetic algorithm

The GA fit the structure proposed by the framework. Roughly, a GA had a population that evolved in each successive iteration of the search strategy; evolution was conducted by the tweaks selection, mutation, and crossover (in the search for new CA models). The input was a molecular-dynamics simulation trajectory in PDB format, which was transformed to CMs (evidences). A solution in the GA is a binary string of length 49, each bit denoting a cell in a matrix of 7 x 7. The matrix codifies the topological configuration of the neighborhood of the CA. The rules for each possible CA solution were extracted from the evidences as part of the evaluation process. The high-level diagram of classes is shown in Figure 6.

Some parts of the algorithm are not shown, because the framework implementation had full functionality, i.e., the simulator of CA models and trajectory processing.

The best CAs obtained demonstrated rules that capture conserved structural characteristics of protein structure, such as alpha helix stability. Although CAs are known for their expressiveness, the rules of the identified CAs were not easy to understand, necessitating the use of a representative decision tree (Polaka and Tom, 2010) to aid comprehension. More extensive description of this work has been discussed in previous research (Diaz and Tischer, 2011a,b).

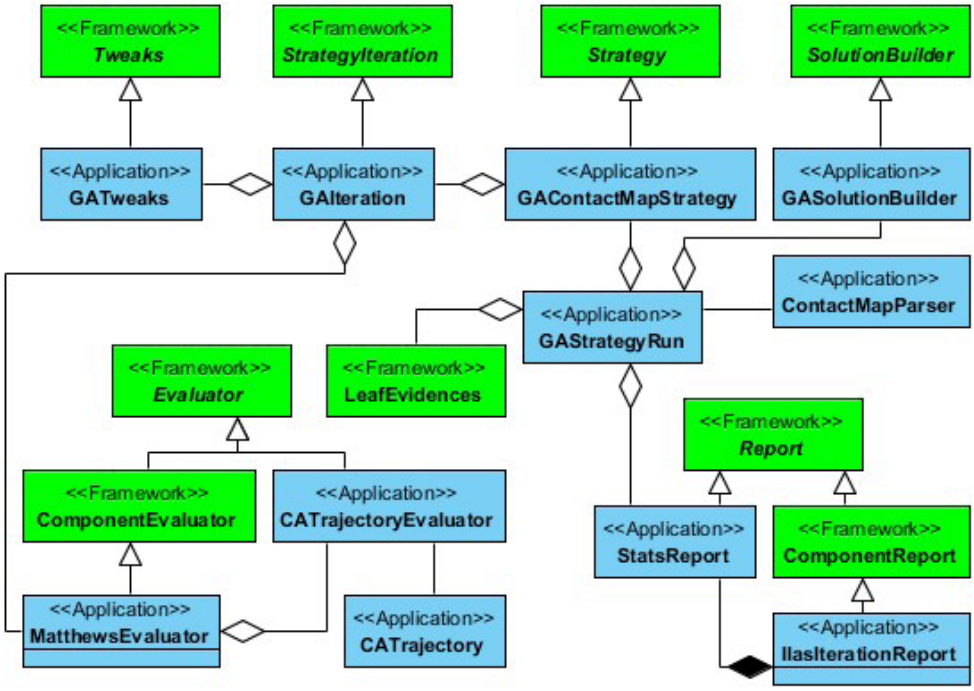

Figure 6. High-level class diagram for genetic algorithm application. <<Application>> stereotype identifies classes defined by the developer.

Co-evolutive strategy

A co-evolutive strategy algorithm (COES) is similar to a GA in that it shares the same parameters, adding extra complexity to the data management, solution evaluation, and population design. One version of COES is the n-populations competitive co-evolution space embedded (Luke et al., 2009). This approach consists of nine populations that are restricted to a lattice of 3 x 3; each cell of the lattice contains a population, and each one contests the nearest solutions of the other populations in a Von Neumann neighborhood (Ganguly et al., 2001). The advantage of COES versus GA is its more intensive exploration of the search space with multiple populations, converging in fewer iterations. The disadvantages are the computational cost and a more complex algorithm design. Implementation of the designed COES utilizes the same parameters as that of the GA.

Notwithstanding the complexity of this approach, the implementation from an architectural standpoint was nearly identical to the that of the GA. This structural similarity demonstrates the impact of the framework in their designs and reveals some additional components that could be included in the framework, such as reports implementation and the evaluators. The extra effort in the implementation of the COES is apparent in this architectural view, but it is clear in the code implementation. A more detailed description of the COES is discussed in previous research (Gómez et al., 2012). Figure 7 illustrates the general architecture of the algorithm; only five classes are needed to implement it.

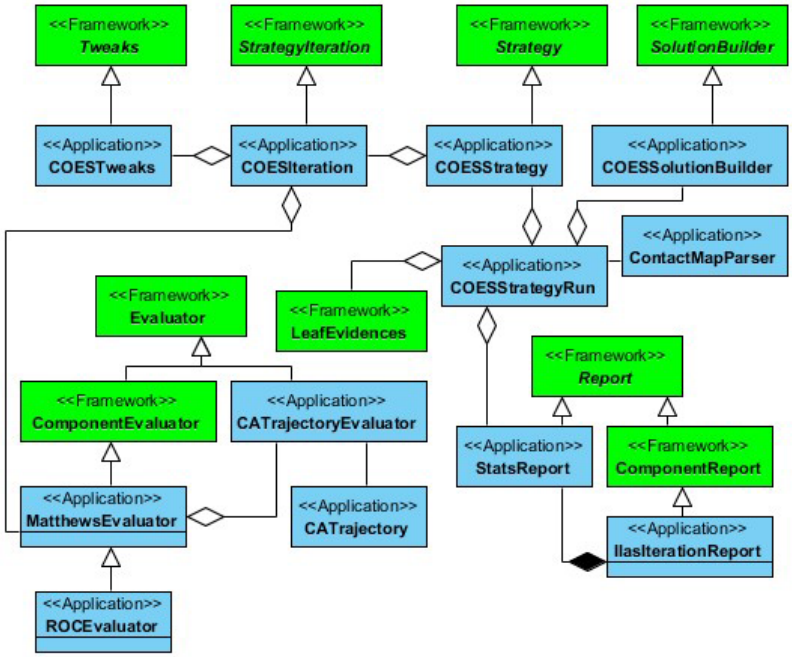

Figure 7. High-level class diagram for co-evolutive strategy algorithm instantiation.

Tabu search

Unlike the previous algorithms, the TS algorithm is based on a local search rather than a population-based exploration of the search space. The iterative process was carried out around one specific CA, and other possible solutions were defined through a concept of locality (e.g., CAs that differ in one cell of the neighborhood). A specific characteristic of TS is the use of a tabu list, which contains the CAs under consideration, in order to avoid the exploration of a recently evaluated CA.

Once again, the architecture was very similar to the previous applications (Figure 8), exhibiting the capability of the framework to aid the design of several metaheuristics. The extended description of this approach can be found in previous research (Gallego et al., 2012).

Hybrid algorithm

The proposed hybrid algorithm aimed to enhance the exploration of the GA search space using the advantages of TS searching to accommodate the best solution. The hybrid algorithm uses the previous GA and in the GA Iteration class, notes a lack of progress in the search. When this occurs, the GA searches in the vicinity of the best CA at that iteration, generating a populational shift and a consequential shift in the landscape exploration.

The GA architecture was changed only by the composition of the TSS trategy Run class in the GA Contact Map Strategy class, given that the concrete Strategy class contained all the data and configuration parameters necessary for the TS algorithm. Therefore, the architecture was almost identical to that shown in Figure 6.

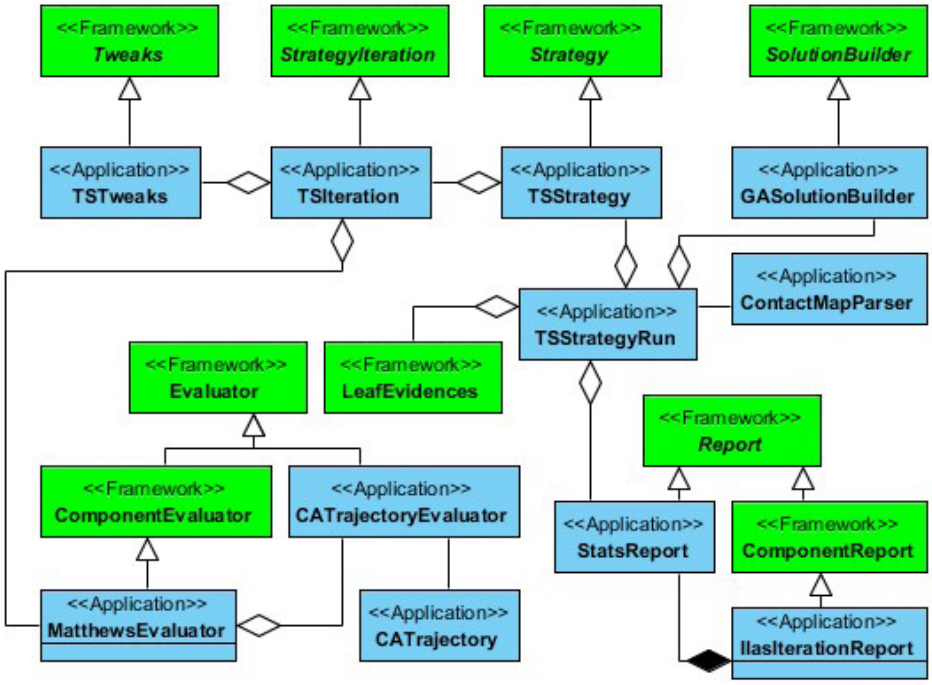

Figure 8. High-level class diagram for tabu search instantiation.

RESULTS AND DISCUSSION

The applications developed with the support of the framework were used in the identification of CA using molecular-dynamics simulation of the villin headpiece protein (HP35-NleNle) as a sample trajectory (Ensign et al., 2007). Statistics for an execution of the GA are shown in Figure 9. Typically, the central measure used to evaluate performance of evolutionary algorithms is the score of the best individual on each iteration. Our approach doesn't allow a typical assessment because of the changing data subset of the vast quantity of protein structures in the folding trajectory. Which is implemented by Iterative Learning with Alternating Strata management. In our sample trajectory, the average score/moving average overcomes this issue. The statistics in Figure 9 correspond to a convergent behavior of the GA in the example execution.

The best CAs achieve approximately 97% accuracy in replicating the folding simulation based solely on the identified rules. For experimental analysis, we restricted one of the established rules. The selected rule covers a wide range of native contacts, involving the contacts at positions i,j and i+2,j+2 in the time step T, which determines the contact state of AAs at i,j in T+1. Figure 10 illustrates the native contact map for the villin headpiece protein used in the experiments; the coloring of contacts is consistent with the diagonals. Each diagonal of the CM represents a specific separation between the AAs; therefore, the rule represents a dissimilar behavior. The left-bottom inset in Figure 10 illustrates a scheme for the 3D structure of the protein, displaying three recognizable alpha helices on the CM (helix 1, AA 2 a 10; helix 2, AA 13 a 20; and helix 3, AA 21 a 33).

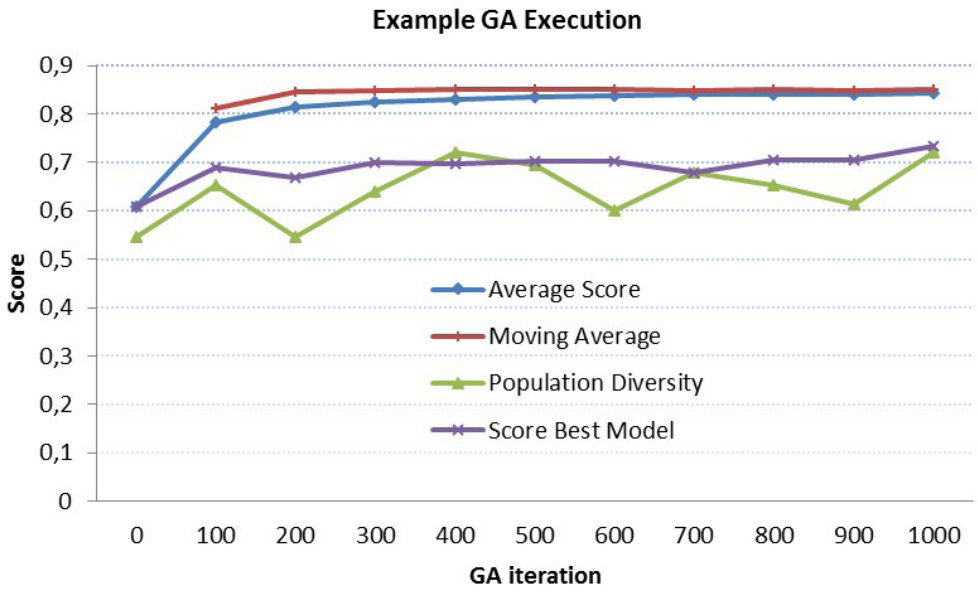

Figure 9. Statistics for genetic algorithm execution on the sample data set.

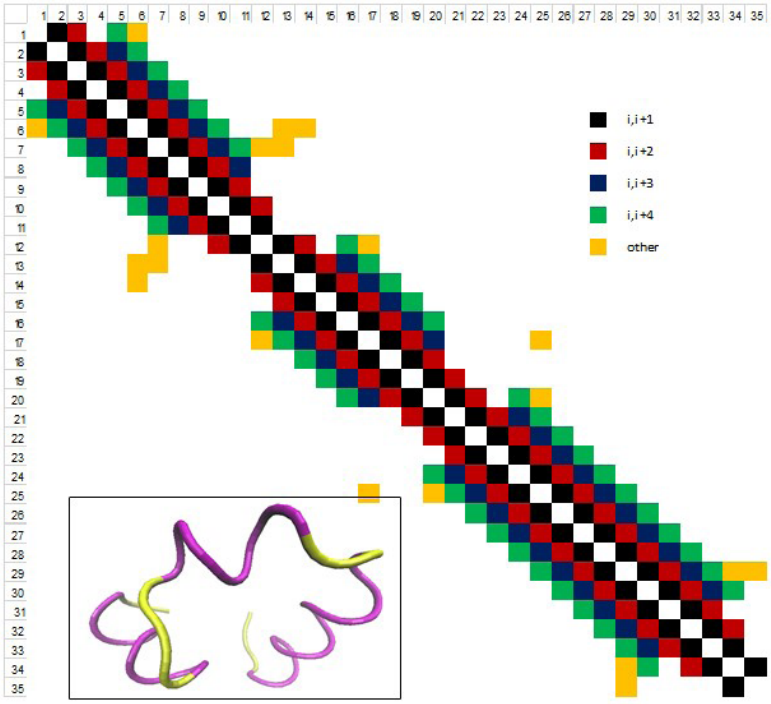

Figure 10. Native contact map of villin headpiece protein at 7Å. Inset: Villin headpiece native structure.

For brevity of analysis, Figure 11 considers only helix 2, displaying the pairs of cells in the CM to which the rule applies. Figure 11A displays the pairs of cells with a separation of two AAs in the protein sequence; for this section of the CM, the rule application leads to conservation of contacts in the next simulation, which is concordant with known stability characteristics of the alpha helix (i.e., AA i and i+4 are in contact). Figure 11B displays the pairs of cells with a separation of three AAs in the protein sequence; the contacts of this zone involve more distant AAs, thus these diagonals denote stable interactions between AAs. Figure 11C displays another set of more distant contacts; the AAs involved display 4 AAs of sequence separation.

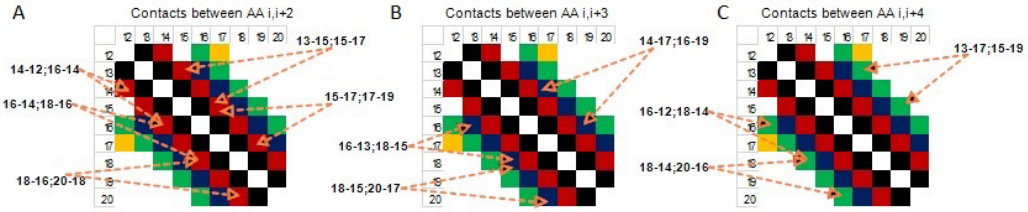

Figure 11. Rule application for contacts in helix 2 of villin headpiece protein (HP35-NleNle). **A.** Contacts between AA i-i+2. **B.** Contacts between AA i-i+3. **C.** Contacts between AA i-i+4.

An interesting behavior that can be observed in Figure 11A is the transitivity of the native contacts inside the helix, i.e., if AAs 14 and 16 are in contact and equivalent to the contact distance between AAs 16 and 18, then the contact of AAs 14 and 16 will be conserved from its occurrence, in the same way that the contact between AAs 16 and 18, 13 and 15, 12 and 14, 15 and 17, 17 and 19, and 18 and 20 will be conserved.

The overall process of the framework design was guided by UML-F. UML-F is a UML profile accompanied by a full guide with recommendations for the framework architect (Fontoura et al., 2002), lessening the difficulty of the complex task of designing a framework. The documentation that complements the design artifact is very useful for the application developer, because it is oriented to serve as a step-by-step guide to indicate the instantiation of each aspect of the framework.

The framework prototype was implemented in Python 2.7, and is publicly available at http://caif.googlecode.com. The use of a high-level language offers additional characteristics such as higher-order programming (e.g., the use of parameters that are definitions of objects or classes). Some aspects of the application set were repeated, indicating that the reports, parsers, and evaluators would be present in the next version of the framework.

For an extended description of the overall CA model results of the running of the algorithms, refer to previous research (Diaz and Tischer, 2011b). In spite of the expected ease of use of CA models, the results were very difficult to analyze in conjunction with the complex rules. The complexity of the rules was due, in part, to the size of the neighborhood in the best models. Moreover, the kind of knowledge subjacent to the CM was highly dependent on the cell position (in AA pairs nearer to the protein sequence a rule represents a behavior, while for distant AA pairs the same rule represents something completely different). Furthermore, a rule that represents interactions of nearer AAs inside a secondary structure can simultaneously express interactions between distant AAs in different secondary structures. To address this,

we plan to conduct a study of the conserved structures in the CM space of known protein structures and their representations in the CA rules.

The successful utilization of the framework in the running example verifies an ongoing proposal based on the simplified molecular-dynamics rules of the obtained CAs in the field of CM prediction. This kind of approach has not been previously used, making it novel in the field of CM prediction.

The proposed framework has proven its usefulness in the development of algorithms to mine CA models from protein-folding trajectories. The diversity of the algorithms implemented allows consideration of a significant quantity of other metaheuristic algorithms such as genetic programming, ant colonies, cuckoo search, harmonic search, multiobjective optimization, learning classifier systems, local guided search, and particle swarm optimization.

ACKNOWLEDGMENTS

Research supported by the COLCIENCIAS National Call for Doctoral Studies #567, the Universidad del Valle and the Universidad del Cauca.

REFERENCES

Berman H, Henrick K, Nakamura H and Markley JL (2007). The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data. *Nucleic Acids Res.* 35: D301-D303. http://dx.doi.org/10.1093/nar/gkl971

Diaz N and Tischer I (2011a). Mining cellular automata models on protein folding trajectories. In: Computing Congress (CCC), 2011 6th Colombian, 1-6.

Díaz N and Tischer I (2011b). Metodología y framework computacional para el diseño inverso de modelos de autómata celular de secuencias cortas de aminoácidos soportado en un proceso de minería de datos. Universidad del Valle.

Ensign DL, Kasson PM and Pande VS (2007). Heterogeneity even at the speed limit of folding: large-scale molecular dynamics study of a fast-folding variant of the villin headpiece. *J. Mol. Biol.* 374: 806-816. http://dx.doi.org/10.1016/j.jmb.2007.09.069

Fontoura M, Pree W and Rumpe B (2000). UML-F: A modeling language for object-oriented frameworks. *ECOOP 2000 Object-Oriented Programming* 1: 63-82.

Fontoura M, Pree W and Rumpe B (2002). The UML profile for framework architectures. Addison-Wesley Professional.

Gallego R, Obando F and Díaz N (2012). Metaheurística híbrida para la identificación de modelos de autómata celular en trayectorias de simulación de plegamiento de proteína.

Gamma E, Helm R, Johnson R and Vlissides J (1995). Design patterns: elements of reusable object-oriented software. Vol. 206. Addison-Wesley Reading, MA.

Ganguly N, Sikdar BK, Deutsch A, Canright G, et al. (2001). A survey on cellular automata. *Project BISON* 1-30.

Glover F and Laguna M (1999). Tabu Search. Springer US.

Goldberg DE (1999). The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. *Evol. Design Comp.* 105-118.

Gómez A, Ocoró S and Díaz N (2012). Estrategia co-evolutiva para la identificación de modelos de autómata celular en trayectorias de simulación de plegamiento de proteína.

Gupta N, Mangal N and Biswas S (2005). Evolution and similarity evaluation of protein structures in contact map space. *Proteins* 59: 196-204. http://dx.doi.org/10.1002/prot.20415

Karplus M and Kuriyan J (2005). Molecular dynamics and protein function. *Proc. Natl. Acad. Sci. USA* 102: 6679-6685. http://dx.doi.org/10.1073/pnas.0408930102

Kolinski A (2004). Protein modeling and structure prediction with a reduced representation. *Acta Biochim. Pol.* 51: 349-371.

Luke S, Domeniconi C, De Jong K, Grefenstette J, et al. (2009). Essentials of Metaheuristics. Zeroth Edition.

Polaka I and Tom I (2010). Decision Tree Classifiers in Bioinformatics. *Sci. J. Riga Techn. Univ* 44: 119-124.

Rabino G and Laghi A (2002). Identification of cellular automata: theoretical remarks. In: Proceedings of the Annual Conference of The European Regional Science Association (ERSA) (Dortmund), 1-23.

Sharma S, Ding F and Dokholyan NV (2008). Probing protein aggregation using simplified models and discrete molecular dynamics. *Front. Biosci. A J. Virtual Library* 13: 4795. http://dx.doi.org/10.2741/3039

Vassura M, Margara L, Di Lena P, Medri F, et al. (2008). FT-COMAR: fault tolerant three-dimensional structure reconstruction from protein contact maps. *Bioinformatics* 24: 1313-1315. http://dx.doi.org/10.1093/bioinformatics/btn115

Vendruscolo M and Domany E (1998). Efficient dynamics in the space of contact maps. *Fold. Des.* 3: 329-336. http://dx.doi.org/10.1016/S1359-0278(98)00045-5

Wang Y, Wei Q, Pan F, Yang M, et al. (2014). Molecular dynamics simulations for the examination of mechanical properties of hydroxyapatite/ poly α-n-butyl cyanoacrylate under additive manufacturing. *Biomed. Mater. Eng.* 24: 825-833.

Wolfram S (2002). A New Kind of Science. Vol. 1. Champaign, IL: Wolfram Media Inc.

Xiao X, Wang P and Chou KC (2011). Cellular automata and its applications in protein bioinformatics. *Curr. Protein Pept. Sci.* 12: 508-519. http://dx.doi.org/10.2174/138920311796957720

Yoo JS, Shekhar S, Kim S and Celik M (2006). Discovery of co-evolving spatial event sets. *SIAM Intl. Conf. Data Mining* 1: 306-315.

