A Critical View of the Evolutionary Design of Self-Assembling Systems

N. Krasnogor, G. Terrazas [1], D.A. Pelta [2], and G. Ochoa[3]

[1] Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science and Information Technology
University of Nottingham
{nxk,gzt}@cs.nott.ac.uk

[2] Departamento de Ciencias de la Computacion
ETSI Informatica, Universidad de Granada
dpelta@decsai.ugr.es

[3] Departamento de Ciencias de la Computacion
Universidad Simon Bolivar
gabro@ldc.usb.ve

Abstract. The automated design of systems which self-assemble is a fundamental cornerstone of nanotechnology. In this paper we review some work in which we have applied Evolutionary Algorithms (EAs) for the *automated* design of systems self-assembly. We will focus in three important minimalist self-assembly problems and we discuss the difficulties encountered while applying EAs to these test cases. We also suggest some promising lines of work that could possibly help overcome current limitations in the evolutionary design of self-assembling systems.

1 Introduction

Self-assembly is a process that creates complex hierarchical structures through the statistical exploration of alternative configurations. These processes occur without external intervention. The specific system that is self-assembled (from a given set of components) is determined by the way the statistical exploration of conformations is performed. In turn, the exploration mechanisms are constrained by the individual components that undergo self-assembly and the conditions imposed upon them by their local environment. In general, components are autonomous, have no pre-programmed *master* assembly plan, and can only interact with their local environment and other components. Self-Assembly is a powerful autopoietic mechanism whose power, as a reusable engineering concept, lays in the fact that it is a distributed, not-necessarily synchronous, control mechanism for the bottom-up manufacture of complex systems. This control mechanism is distributed across a myriad of elemental components, none of which has either the storage or the computation capabilities to know and follow a master plan for the assembly of the intended system. Instead each component has a very limited behavioral repertoire which tells it what to do under a reduced set of well defined conditions. Self-Assembly processes are ubiquitous in nature. Understanding how nature produces self-assembled systems will represent an enormous leap forward in our technological capabilities. Although major advances in the design of systems that exhibit self-assembly properties have been reported in the literature (e.g. [17, 16]), much less has been said about the *automated* design of self-assembly. In [8] the author tackles the problem of automated design of self-assembly for a very specific class of problems which are amenable to analytical solution. However, it is unrealistic to expect that each and every self-assembly system will have properties that make it agreeable to a hand-made design. Instead, as in many other industrial settings, we will need to resort to computer aided automated design of components, interaction matrices and assembly skeletons.

The complexity of self-assemble squares under a generalized model of tile assembly[13] was investigated in [1]. Several interesting results on the intractability of certain self-assembly processes were described. Although these papers point to promises and limitations of specific self-assembly processes it is important to remark that NP-hardness results have not, in the past, deterred the advance of other branches of science and engineering. On the contrary, NP-hard problems are regularly tackled (and solved to industrial standard satisfaction) with an arsenal of modern algorithmic techniques ranging from integer and linear programming, lagrangian relaxations to sophisticated metaheuristics like tabu search[6], simulated annealing[7] and memetic evolutionary algorithms[15].

A principled methodological approach for automated self-assembly design would be to systematically investigate automated design methods on (tunable) conceptual, highly idealized problems as it has been done in other domains like protein folding[4], traveling salesman problem[12], etc. To this end, in [9] we introduced a family of tunnable problems for self assembly. In this paper we complement that paper by reviewing some work in which we have applied Evolutionary Algorithms (EAs) for the automated design of systems self-assembly. We focus in three important minimalist self-assembly problems and we discuss the difficulties encountered while applying EAs into these problems. In this paper we also suggest some promising lines of work that could possibly help overcome current technological limitations.

2 Protein Structure Prediction and Wang Tiles as Paradigmatic Self-Assembly Design Problems

In this section we introduce two problems which are paradigmatic self-assembly design problems, namely, the design of folding rules in protein structure prediction and the design of Wang tile families for the self-assembly of two-dimensional shapes.

2.1 Protein Structure Prediction

Proteins are hetero-polymers composed of amino acids. Under physiological conditions proteins fold into a three dimensional native state where they adopt their

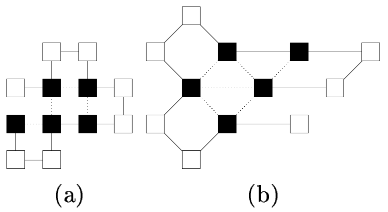

Fig. 1. *HP protein embedded in the square lattice (a) and triangular lattice(b). Black boxes represent hydrophobic residues, while white boxes represent hydrophilic ones.*

biological function. The protein structure prediction problem is concerned with the determination of the native state from the identity of the amino acids that constitutes a given protein. That is, protein folding might be regarded as the self-assembly problem par excellence. The particular simplified model we are concerned with in this paper is the Hydrophobic-Polar model introduced by K. Dill[4]. The HP model (and its variants) abstracts the hydrophobic interaction process in protein folding by reducing the 20 naturally occurring amino acids into a binary alphabet, thus a protein becomes an hetero-polymer of non-polar or hydrophobic (H) and polar (P) or hydrophilic amino acids. An n amino acids protein is represented by sequence $s \in \{H, P\}^+$ with $|s| = n$. The sequence s is to be mapped to a lattice, where each residue in s occupies a different lattice cell and the mapping is required to be self-avoiding. Although simple to state, this problem remains NP-Hard[3].

The energy potential in the HP model reflects the fact that hydrophobic amino acids have a propensity to form a hydrophobic core. To capture this feature of protein structures, the HP model adds a value ϵ for every pair of hydrophobes that form a topological contact; a topological contact is formed by a pair of amino acids that are adjacent on the lattice but not consecutive in s. After normalization, the interaction energy between two non-polar amino acids is $\epsilon_{H,H} = -1$ while all other interactions (i.e. HP and PP) are 0. In this model optimally self-assembled native structures minimize an energy function that is a simple count of the number of HH contacts in the self-assembled conformation. Figure 1(a) and (b) shows sequences embedded in the square and the triangular lattices, with hydrophobic-hydrophobic contacts (HH contacts) highlighted with dotted lines. The conformation in Figures 1(a) and 1(b) show the embedding of the same protein instance into two different lattices, which result in energies of -4 and -6 respectively.

Automated Design of Protein Self-Assembly In this paper we will address the problem of automatically designing, by means of an evolutionary algorithm, the rules that are necesary to drive the dynamical process of folding towards the native state of specific proteins. We will employ two different computational abstractions to represent these folding rules. The first abstraction we use is that of a one dimensional uniform, contiguous neighborhood, cellular automata to

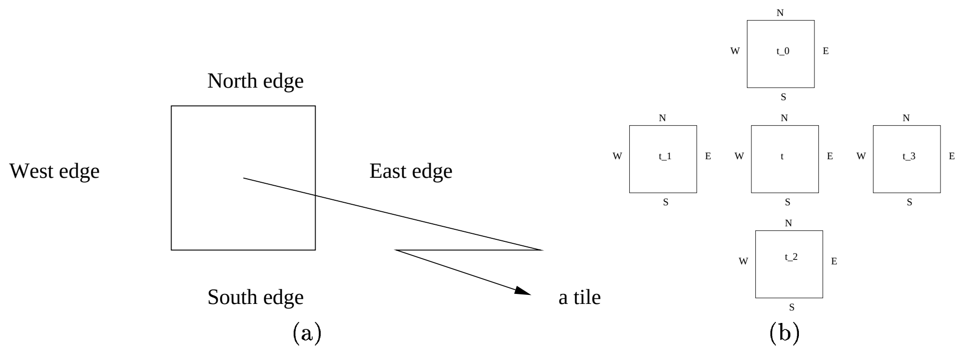

Fig. 2. (a) Schematic representation of a four edged tile. Each edge is distinguished by the labels *North, West, South, East.* (b) An example of a five tiles self-assembly

simulate the folding process. In this case, the evolutionary algorithm is required to design the rules that define the cellular automaton, with the intention that by executing those rules the protein sequence embedded in the automaton will self-assemble into its native state. In the second computational abstraction we represent the folding rules by an L-system grammar rather than by the rules of a cellular automaton. In this case the parallel interpretation of the L-system grammar drives the self-assembly of the protein structure into its target conformation.

2.2 Wang Tiles Self-Assembly

Computation and self-assembly are connected by the theory of tiling, of which *Wang Tiles*[14] are a prime example. A Wang tile system is defined by a family of two dimensional square tiles embedded in the plane. Each side of a tile might have a specific glue type attached to it. When tiles move around in the plane, and two of them colide, they will either stay attached or they will separate and continue their brownian motion as independent entities. Whether they self-assemble or stay separated depends on the strength and compatibility of the glue types in their coliding sides. This process is initialized with a specific kinetic energy associated to the tile set (i.e. temperature). When tiles attach to each other they form complex shapes and the specific shapes which emerge are said to be self-assembled. This process can be mathematically described:

Let Σ be the set of symbols used to label the edges associated to each tile. This set of symbols encodes the "glue" types associated to each edge and includes the special case λ representing and edge with no glue. The set of tiles is $\mathcal{T} = \{t | t = (x_0, x_1, x_2, x_3)\}$ such that for any $k \leq 3$ $\exists a, a \in \Sigma, p >= 0$ *and* $x_k = a^p$. If $p = 0$ then a^0 is taken to be equivalent to λ, i.e., the no glue state for a given edge of the tile. A label a^p on an edge x_k encodes an "a" glue type with strength p.

We can associate x_0, x_1, x_2 and x_3 with the north, west, south and east edges respectively as shown in figure 2(a). Let also τ be the "temperature" parameter as in [1]. After coliding, two tiles t_i, t_j will self-assemble by their edges e_i, e_j if the glue types and strengths in those edges are equivalent and the glue strength larger than the temperature.

Given tiles t, t_0, t_1, t_2, t_3 they will self-assemble with t in the center (as shown in Fig 2(b)) if the glue strength of each attaching edges is bigger than 0 and the sum of all glue strengths bigger than τ. More precisely, t and t_i for $0 \leq i \leq 3$ will self-assemble if the following conditions hold:
$t = (x_0, x_1, x_2, x_3)$ and $t_i = (x_{i_0}, x_{i_1}, x_{i_2}, x_{i_3}, 0 <= i <= 3$ with $x_{0_2} = x_0, x_{1_3} = x_1, x_{2_0} = x_2, x_{3_1} = x_3$ and $|x_0| + |x_1| + |x_2| + |x_3| >= \tau$
Please note that the conditions on x_k above can be succintedly written as $x_{k_g} = x_k$ with $g = (k+2)\%3$ where % stands for the module operation. The reader must note that the labeling of edges as "north,west,south,east" is only a useful convention to simplify the exposition.

Automated Design of Wang Tile Families The third and last automated design problem we will address is that of the automated design of $\mathcal{T}$ (i.e. a families of Wang tiles), which can self-assemble into a specific two dimensional shape, which in this paper is a square.

3 Evolutionary Algorithms for the Automated Design of Protein Self-Assembly by Cellular Automata (CA)

Cellular automata have been used as models of physical and biological phenomena such as fluid flow, galaxy formation, earthquakes, biological pattern formation, etc. and as models of computation (see for example [18]). Briefly a CA consists of two components. The first one is *a lattice of N identical cells*, each of which have a state. Each cell is updated based on its current state and the state of its neighbors in the lattice. The neighborhood considered depends on the particular CA. The second component is the *transition rules* that give the updated state for each cell as a function of the neighborhood.

We used a CA to model the rules and dynamics which would drive a self-assembly process towards the native state of a given protein sequence. We had previously addressed this problem using a circular one-dimensional CA with only four states (1, 2, 3, 4), each one corresponding with the absolute moves **U**p, **D**own, **L**eft or **R**ight (relative to the position of the previous amino acid in the sequence) [10]. An example is shown in Figure 3 (a). Allowed rule radii were 1, 2 and 3. The evaluation of an individual involved: running the CA with the individual's value (set of rules), getting the final configuration of the automaton (the folded structure), applying this fold to the protein to obtain the energy value.

We also performed experiments with an extended set of rules which took into consideration the specific amino acids the rule was being applied to. This is shown in Fig 3 (b). To evolve the rule set that defined the CA we used a Genetic Algorithm. Implementation and parameter details are described in [10].

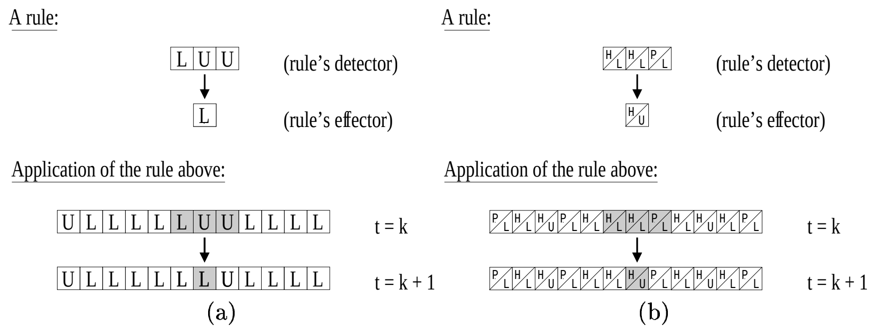

Fig. 3. (a) First approach to CA rule scheme. (b) Second approachto CA rule scheme.

We have conducted extensive experiments but due to space limitations we only show the results for 3 instances in Table 1. We recorded the number of times (out of 10) that the optimum, optimum-1 and optimum-2 conformations were found.

Sequence	Length	Optimum	Opt	Opt + 1	Opt + 2
PHPPHHPPHPPHPPHHPPHP	20	-8	5		5
HHPPHPPHPPHPPHPPHPPHPPHH	24	-9	1	2	5
PPHPPHHPPPPHHPPPPHHPPPPHH	25	-8			1

Table 1. Number of runs in which the GA achieved the stated energy value. The 'Optimum' column displays the native energy value, while 'Opt', 'Opt + 1' and 'Opt +2' display the number of runs in which either the optimum energy was achieved or conformations with energies with a gap of one or two above that value was found.

The results may be analized from two points of view. One is oriented to answer the question: *is it possible to find a set of rules to reach the native state from this particular unfolded state?*. The answer is yes on two of the three cases, although it is clear that more experimentation should be done.

The other point of view focus on the quality of the search process, and here the results seems to deteriorate with the size of the instance. For the smaller one, 50% of the runs, lead to set of rules that allowed to achieve the optimal configuration. This percentage goes down to 10% in the second instance and in the third one, just one run allowed to obtain a configuration with energy 6.

One may conclude that: a) in principle, it is indeed possible to find set of rules for a cellular automaton which instigates the self-assembly of the native structure; and b) the search procedure should be enhanced.

4 Evolutionary Algorithms for the Automated Design of Protein Self-Assembly by L-Systems

In [5] we introduced an L-systems' based evolutionary algorithm as the inference procedure for folded structures under the HP model in 2D lattices. The evolutionary algorithm attempts to find a set of rewriting rules (an L-system) that captures a target folded structure (which represents the native state for a given protein) on the selected lattice model.

The simplest class of L-systems, the D0L-systems, is deterministic and context free. We use D0L-systems to drive the self-assembly of the protein sequence.

Given a target structure (input), let say the one shown in Fig. 1(a), the evolutionary algorithm will evolve and L-system L (output) that, once evaluated, would produce a string (in internal coordinates) which matches the target structure (in the example, the end-product of the EA would be and L-system whose termination word is LRRLRRFLRR).

A Genetic algorithm was used to evolve the L-systems which would drive the self-assembly procees. Full details of the algorithm and experiments can be found in [5]. Table 2 shows some of the results we obtained evolving L-systems for the self-assembly of protein structures.

Instance	**Length**	**Success/Num. Runs**
$\frac{HHHPPHPHPHPPHPHPHPPH}{RRFRFRLFRRFLRLRFRR}$	18	$\frac{3}{40}$
$\frac{HHPPHPPHPPHPPHPPHPPHPPHH}{RLLFLFFRRFLLFRRLRFFRRF}$	22	$\frac{1}{50}$
$\frac{PPHPPHHPPPPHHPPPPHHPPPPHH}{FFRRFFFLLFFFFRRFFFFLLFF}$	23	$\frac{1}{50}$

Table 2. *Partial Results of the Automated Evolutionary Design of L-Systems for Protein Folding. The first column format $\frac{I}{S}$ denotes the protein sequence I with target self-assembled structure S, the second column shows the length of the protein sequence and the third column -following the same format as the first- shows the total number of runs of the EA and the number of successful runs.*

Similarly to the evolutionary design of CA rules for self-assembling, the automated design of L-systems met with partial success. On the one hand it is possible to show that the algorithm is capable of finding L-system which will induce the correct self-assembly behaviour. On the other hand however, the process is painfully slow and requires very many executions of the algorithm to obtain a successful L-system.

5 Evolutionary Algorithms for the Automated Design of Wang Tiles Self-Assembly

We have applied a Genetic Algorithm to the automated design of the tile sets $\mathcal{T}$ which can self-assemble into a 2D square of 10x10 tiles. The GA used various parameters for crossover, mutation, population sizes, etc., which will be reported elsewhere. In order to evaluate an individual (i.e. assess its fitness) we placed it in a Wang tile self-assembly simulator. As the individual specifies various tile families, several instances of each family were placed in the simulator. Each tile was initialy placed on a randomly selected empty lattice position. Then, tiles move randomly for the duration of the simulation. Once the simulation finished the fitness function tried to identify (within the lattice) the shape with the most similarity to the target structure. This was done by a *Hamming distance* function defined as $H(L, S) = a_i$, where L was the simulation's final 2D lattice configuration and a_i is the maximum amount of tiles appearing within a square region S. The region was slided accross the lattice in order to find the better match ensuring that the fitness of an individual is equivalent to the minimal Hamming distance. Figure 4(a) shows a scanning example.

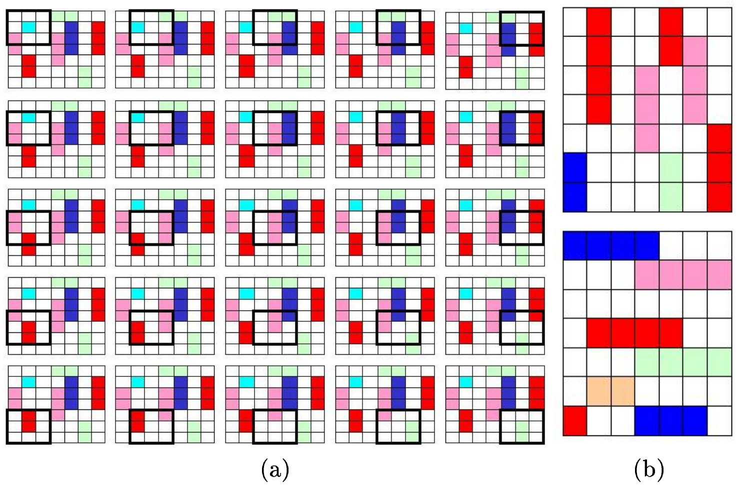

Fig. 4. (a) Scanning a lattice for a 3×3 square. (b) Self-Assembled rows and columns.

With the aim of determining which is the best set of parameters for both the GA and the Wang tile simulator we run an extensive set of experiments. After carefully selecting the best parameters the evolutionay algorithm was unable to evolve suitable tile sets that could self-assemble into the target structure. However, some intermediate structures were discovered by the algorithm. In this case, horizontal and vertical tiled strips (shown in Fig. 4(b)) were found.

6 Discussion and Conclusions

In previous sections we briefly sketched the application of evolutionary algorithms, more specifically genetic algorithms, to the automated design of components which could self-assemble into specific systems. Two of the showcases dealt with the design of rules, either for a cellular automaton or of an L-system, which could drive the process of protein folding (albeit in a very idealized model). In the third case we applied the GA to the design of tile sets and their glue types in order that they could self-assemble a target 2-dimensional shape. Although the application domain, the type of components and dynamic laws governing their use were different some common lessons could be drawn.

Firstly, in the three showcases large populations with short runs or small populations with long runs were required. That is, in the three cases studied the evolutionary design was computationally expensive. This requires a carefull consideration of the various parameters which define the GA behaviour as well as those parameters which are specific to the simulators. It may be possible that a co-evolutionary approach would be benefitial by simultaneously exploring the design space of system self-assembly and the parameter space of the GA.

Secondly, although in all three cases it was possible to achieve a moderate degree, yet not substantial, of success evolving the desired self-assembling system, the remarkable common fact is that intermediate self-assembled products -which are essential for the formation of the target system- were *always* discovered. That is, in L-systems and Cellular Automata we were able to find rules which allowed for the self-assembly of so called protein's "secondary structures". At the same time, the evolutionary design of Wang tiles was able to discover the equivalent of secondary structures in the form of self-assembled columns and rows. This common behaviour across three different domains and with differently customized evolutionary algorithms suggests an *evolutionary divide-and-conquer* methodology. That is, rather than trying to evolve from scratch the final design for a self-assembling system, we could instead evolve designs for *generalized secondary structures* and used those designs to bootstrap the final design. As an example consider the evolutionary design of Wang tiles to self-assembling a square. Instead of starting from completely random tile families we could seed the GA with those families known to form columns and rows as these features will certainly appear in any self-assembled square. Alternatively, in the case of L-systems we could evolve problem specific knowledge (e.g. specific rules for alpha-helices, beta sheets, etc) as to accelerate the design process of self-assembling rules for the whole protein structure.

A third lessons, which we will also be tested in future experiments, is what we named "intelligent freezing". During the evolutionary design of self-assembling systems it was possible to observe that certain *critical generalized secondary structures (CGSS)* were formed. Some of the runs that discovered CGSS managed to maintain them long enough as to profit from their discovery. On the other hand, some runs tampered with the CGSS destroying their essential features. Intelligent freezing would implement a mechanism to detect CGSS (eg. by tracking evolutionary activity waves[2]) an will protect these CGSS from being disrupted by genetic or other mechanisms (i.e. they will be frozen). Another interesting avenue of research would be to use what has been termed the "Parisian Genetic Programming" approach [11] as it has been very successful in a not unrelated inverse design problem.

In conclusion, although the automated design of self-assembling systems is at its infancy it is possible to achieve a modest degree of success with current evolutionary metaheuristics. On the other hand, as the size and complexity of the target self-assembling system increases, its likely that more robust and efficient EA will be needed. We have described three showcases of the application of genetic algorithms for systems self-assembly and we have suggested some promising avenues for further research.

Acknowledgements:

N. Krasnogor would like to acknowledge many useful and enlightening conversations with G. Rozenberg, G. Paun, M. Gheorghe and P. Moriarty. He will also like to acknowledge the EPSRC (GR/T07534/01, EP/D021847/1) and BBSRC (BB/C511764/1) funding of his research on artificial intelligence as applied to bioinformatics and evolutionary chemistry. D. Pelta would like to acknowledge the funding for project TIC-2002-04242-C03-02.

References

1. L. Adleman, Q. Cheng, A. Goel, M. Huang, D. Kempe, P. Moisset de Espanes, and P.W.K. Rothemund. Combinatorial optimization problems in self-assembly. In *Proceedings of the Annual ACM Symposium on Theory of Computing(STOC)*. ACM Press, 2002.
2. M.A. Bedau and N.H.Packard. Measurement of evolutionary activity, teleology and life. In D. Farmer S. Rasmussen C.G. Langton, C. Taylor, editor, *Artificial Life II*, volume 98-03-023, pages 431–461. Addison-Wesley, 1992.
3. B. Berger and T. Leight. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. In *Proceedings of The Second Annual International Conference on Computational Molecular Biology, RECOMB 98*, pages 30–39. ACM Press, 1998.
4. K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24:1501, 1985.
5. G. Escuela, G. Ochoa, and N. Krasnogor. Evolving l-systems to capture protein structure native conformations. In *Proceedings of the 8th European Conference on Genetic Programming (EuroGP 2005), Lecture Notes in Computer Sciences 3447, pp 73-84*. Springer-Verlag, Berlin, 2005.
6. F. Glover, E. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41:3–28, 1993.
7. S. Kirkpatrick, C.D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 no 4598:671–680, 1983.
8. E. Klavins. Automatic synthesis of controllers for distributed assembly and formation forming. In *Proceedings of the IEEE Conference on Robotics and Automation*, 2002.
9. N. Krasnogor and S. Gustafson. A family of conceptual problems in the automated design of self-assembly. In *Proceedings of the 2nd International Conference on*

the Fundations of Nanoscience: Self-Assembled Architecture and Devices, Utah, Snowbird resort, April 24-29, 2005.

10. N. Krasnogor, D.A. Pelta, D.H. Marcos, and W.A. Risi. Protein structure prediction as a complex adaptive system. In *Proceedings of Frontiers in Evolutionary Algorithms 1998*, 1998.
11. P.Collet, E.Lutton, F.Raynal, and M.Schoenauer. Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. *Genetic Programming and Evolvable Machines*, 1:339–361, 2000.
12. G. Reinelt. Tsplib (http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/tsplib95/tsplib.html). In *mirror site: gopher://softlib.rice.edu/11/softlib/tsplib.*
13. P. Rothemund and E. Winfree. The program-size complexity of self-assembled squares. In *Proceedings of STOC*, 2000.
14. H. Wang. Probing theorems by pattern recognition. *Bell Systems Technical Journal*, 40:1–42, 1961.
15. N. Krasnogor W.E. Hart and J.E. Smith. *Recent Advances in Memetic Algorithms.* Studies in Fuzziness and Soft Computing Series - Springer, 2004.
16. G.M. Whiteside and M. Boncheva. Beyond molecules: Self-assembly of mesoscopic and macroscopic components. *Proceedings of the National Academy of Science (PNAS)*, 99(8):4769–4774, 2002.
17. G.M. Whiteside and B. Grzybowski. Self-assembly at all scales. *Science*, 295:2418–2421, 2002.
18. S. Wolfram. *A New Kind of Science*. Wolfram Media Inc., 2002.