

Soft Computing based Decision Support Systems: Two Prototypes for Combinatorial Optimization Problems

José I. Peláez **Alejandro Mesas**
Dept. of Languages and Computer Sciences
University of Malaga
Malaga 29071, Spain
jignacio@lcc.uma.es

David Pelta
Dept. of Computer Science and AI
University of Granada, 18071
Granada, Spain
dpelta@decsai.ugr.es

Abstract

Soft computing techniques are, nowadays, frequently and successfully used in optimization. In this context, two decision support systems's prototypes are developed for solving two hard combinatorial problems. A relevant point is the use of a fuzzy sets-based heuristic at the core of both systems that allowed to obtain very good results.

Keywords: soft computing, decision support, bioinformatics, cutting problems

1 Introduction

The benefits on the use of soft computing techniques in the context of optimization is nowadays, out of question. If we consider heuristics and metaheuristics (instead of just genetic algorithms), as one of the main components of soft computing, the range of successful applications increases exponentially. Just a look at two recent books [6, 7] may give the interested reader, an overview of the current trends.

The use of the basic ideas of fuzzy sets and systems within the context of heuristic optimization give raise to what is known as fuzzy sets-based heuristics. Relevant contributions in this topic, were recently compiled in [11].

In this work, we will show how a fuzzy sets-based heuristic is used in two prototypes of decision support systems (DSS) corresponding to two optimization problems: the guillotine cut problem and the protein structure prediction problem on lattice models. In these DSS's, the user can adjust the parameters of the algorithm and manipulate

the problem's modelization. Everything is done through a very simple user interface and graphical output of the results is also provided. The main screen of both prototypes is shown below:



Figure 1: Main screen of the DSS' Prototypes

The paper is organized as follows: in Section 2 we describe the fuzzy adaptive neighborhood search heuristic used. Then in Section 3 we introduce the protein structure prediction problem, we describe the implemented DSS and we comment its usage. The same scheme is used in Section 4 for the guillotine cut problem. Finally, Section 5 is devoted to the conclusions.

2 A Fuzzy Sets-based Heuristic for Optimization

The Fuzzy Adaptive Neighborhood Search Method (*FANS*) [3, 10] is a local search procedure which differs from other local search methods in two aspects. The first one is how solutions are evaluated; within *FANS* a fuzzy valuation representing some (maybe fuzzy) property P is used together with the objective function to obtain a

```

Procedure FANS:
Begin
  While (not-end) Do
    /* The neighborhood scheduler NS is called */
     $S_{new} = NS(\mathcal{O}, \mu(), S_{cur});$ 
    If ( $S_{new}$  is good enough in terms of  $\mu()$ ) Then
       $S_{cur} := S_{new};$ 
      adaptFuzzyValuation( $\mu(), S_{cur}$ );
    Else
      /* NS failed to return a good solution with  $\mathcal{O}$  */
      /* The operator scheduler will modify the operator */
       $\mathcal{O} := OpSchedul(\mathcal{O});$ 
    Fi
    If (TrappedSituation()) Then
      doEscape();
    Fi
  Od
End.

```

Figure 2: Scheme of *FANS*

“semantic evaluation” of the solution. In this way, we may talk about solutions satisfying P in certain degree. Under this view, the neighborhood of a solution effectively becomes a fuzzy set where the neighbor solutions are the elements and the fuzzy valuation is the membership function.

The fuzzy valuation enables the algorithm to achieve the qualitative behavior of other classical local search schemes [3]. *FANS* moves between solutions satisfying P with at least certain degree, until it became trapped in a local optimum. In this situation, the second novel aspect arise: the operator used to construct solutions is changed, so solutions coming from different neighborhoods are explored next. This process is repeated once for each of a set of available operators until some finalization criterion for the local search is met.

The scheme of *FANS* is shown in Fig. 2. The execution of the algorithm finishes when some external condition holds, here when the number of cost function evaluations reached a pre-specified limit. Each iteration begins with a call to the so called *neighborhood scheduler NS*, which is responsible for the generation and selection of the next solution in the optimization path. The call is done with parameters S_{cur} (the current solution), $\mu()$ (the fuzzy valuation), and \mathcal{O} (a parameterized operator which is used to construct solutions). The neighborhood scheduler can return two alternative results; either a good enough (in terms of

$\mu()$) solution (S_{new}) was found or not.

In the first case S_{new} is taken as the current solution and $\mu()$ parameters are adapted. In this way, the fuzzy valuation is changed as a function of the context or, in other terms, as a function of the state of the search. This mechanism allows the local search stages to adapt during the search. If *NS* failed to return an acceptable solution (no solution was good enough in the neighborhood induced by the operator), the parameters of the operator are changed. The adaptation strategy is encapsulated in the so called *operator scheduler OS*. The next time *NS* is executed, it will have a modified operator to search for solutions.

When the whole set of operators available was used and the search was still stagnated ($TrappedSituation = True$), a classical random restart procedure is applied, and *FANS* continues the search from the new solution.

At each iteration, the parameters used in the *NS* call change. *FANS* starts with *NS* ($s_0, \mathcal{O}^{t_0}, \mu_0$). If *NS* could retrieve an acceptable neighborhood solution, the next iteration the call will be *NS* ($s_1, \mathcal{O}^{t_0}, \mu_1$), the current solution is changed and the fuzzy valuation is adapted. If *NS* failed to retrieve an acceptable neighborhood solution (at certain iteration l), the operator scheduler will be executed returning a modified version of the operator, so the call will be *NS* ($s_l, \mathcal{O}^{t_1}, \mu_l$).

3 A DSS Prototype for the Protein Structure Prediction Problem

Protein Structure Prediction (PSP) is one of the most exciting problems that computational biology faces today. In simple terms, it can be formulated as follows: given a sequence of amino acids, which is the corresponding 3D structure of minimum energy?

One of the most studied simple protein models is the hydrophobic-hydrophilic model (HP model) proposed by K. Dill [5]. HP models abstract the hydrophobic interaction process in protein folding reducing a protein to a chain that represents a pattern of hydrophobicity in the protein; non-polar amino acids are classified as hydrophobic and polar amino acids are classified as hydrophilic. A sequence is $s \in \{H, P\}^+$, where H represents a hydrophobic amino acid and P represents a hydrophilic one.

The HP model restricts the space of conformations to self-avoiding paths on a lattice where vertices are labelled by the amino acids. The energy potential of the model reflects that hydrophobic amino acids tend to form a hydrophobic core. To capture this feature of protein structures, the HP model adds a value $\epsilon = -1$ for every pair of hydrophobics lying adjacent in the lattice but not consecutive in the sequence (a so-called topological contact). PSP under this model means to find the embedding of minimum energy (or maximum number of contacts). Figure 3 shows strings embedded in the square, triangular and cubic lattices, with HH contacts highlighted with dotted lines. The conformation in Figure 3(a) has a score of -4 (four contacts) and the conformation in Figure 3(b) has a score of -5 (six contacts).

PSP was shown NP-Hard on the square and cubic lattices. Genetic Algorithms, Simulated Annealing, GRASP and also Cellular Automata were applied to this problem. See for example, [8] and the references therein for more information.

The structures can be represented by Cartesian Coordinates, Internal Coordinates or Distance Geometry. In this article, we concentrate on both types of internal coordinates: absolute and relative. Under the absolute encoding, the struc-

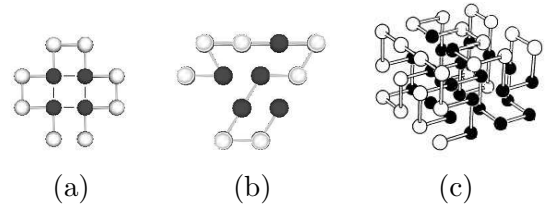


Figure 3: HP sequences embedded in a square (a), triangular (b) and cubic (c) lattices.

tures are represented as a list of absolute moves in the corresponding space. For example, if a 2D square lattice is used, a structure s is codified as a string $s = \{\mathbf{Up}, \mathbf{Down}, \mathbf{Left}, \mathbf{Right}\}^+$. When using a relative encoding, each move must be interpreted in terms of the previous one, like those of the LOGO turtle: a structure s is encoded as a string in the alphabet $s = \{\mathbf{Forward}, \mathbf{TurnRight}, \mathbf{TurnLeft}\}^+$. The structure of Figure 3(a) is coded either as $s = RURDRDLLDLU$ (absolute encoding) or $s = FLRRLRRFLRR$ (relative one).

3.1 The System

The DSS prototype allow the user to apply *FANS* for several variants of the protein structure prediction problem on lattice models. The corresponding screen is shown in Fig. 4. In particular, the choices available for model setting are:

Lattice: 3 options are available: 1) two dimensional square lattice, 2) two dimensional triangular lattice and 3) three dimensional cubic lattice

Cost Function: 3 options are available: *standard*, *gecco* and *functional*. The standard function is the one described previously. The *gecco* function is described in [8], while the *functional* appeared in [2]. This last option is not available when the 2d triangular lattice is selected.

Representation: absolute and relative codification schemes are available for every lattice.

Instance Database: depending on the lattice selected, a set of test instances is shown. The user may input any other of his/her choice.

Some parameters of *FANS* can also be adjusted from the interface:

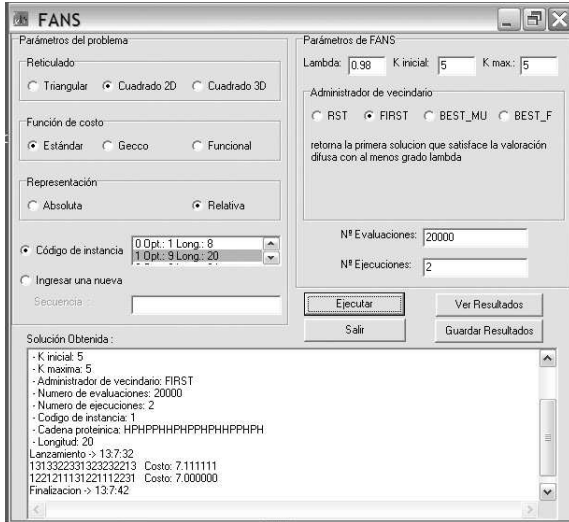


Figure 4: DSS prototype for Protein Structure Prediction: Main Screen

- Neighborhood Scheduler: four possible definitions are provided
- value of λ : the user should indicate which is the minimum level of acceptability required
- Operator Parameters: the modification operator changes k values from the current solution. The initial and final values for k should be provided
- Evaluations Available: this value determines the number of solutions generated.
- Number of Runs: repetitions to be performed. Each run starts from a different random seed.

The results can be saved and then graphically analyzed. Some snapshots are shown in Fig. 5.

3.2 Usage

This prototype has been used to perform several analyzes. The first one was oriented to understand how different encodings (relative or absolute) affected the behavior of FANS over different lattices and parameters' definitions.

From the point of view of FANS, it is being used to assessed the behavior of this fuzzy sets- based heuristic over a hard combinatorial optimization

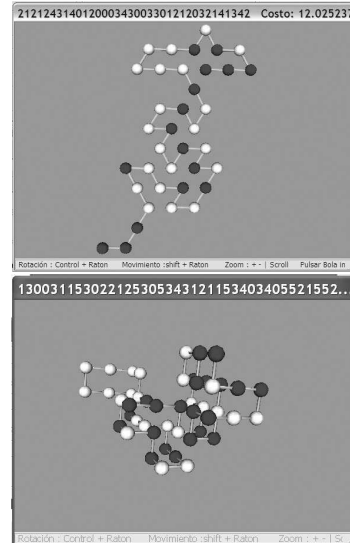


Figure 5: Results's Screen

problem. The results are extremely successful and encouraging.

Finally, it was also used within the MODO group (<http://decsai.ugr.es/modo>) for instructional purposes, to introduce the protein structure prediction problem, and later on, to the closely related, protein folding problem.

4 A DSS Prototype for the Guillotine Cut Problem

The two-dimensional cutting problem using guillotine cuts (CGC2D) is defined as follows: Let $A_0 = (\alpha_0, \beta_0)$ be a rectangular stock sheet having length α_0 and width β_0 , and let R be a set of m smaller rectangular pieces R_1, R_2, \dots, R_m with dimensions $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots, (\alpha_m, \beta_m)$, and a corresponding value v_1, v_2, \dots, v_m and maximum cardinality b_1, b_2, \dots, b_m .

The objective of this problem is to construct a guillotine cutting pattern for A_0 with the highest possible total value using no more than b_i replicates of each rectangle R_i in the pattern with the following restrictions:

- All dimensions (α_i, β_i) for $i = 1, 2, \dots, m$ are assumed integers and therefore the cuts on the rectangles are to be made in integer steps along the x-axis or y-axis.

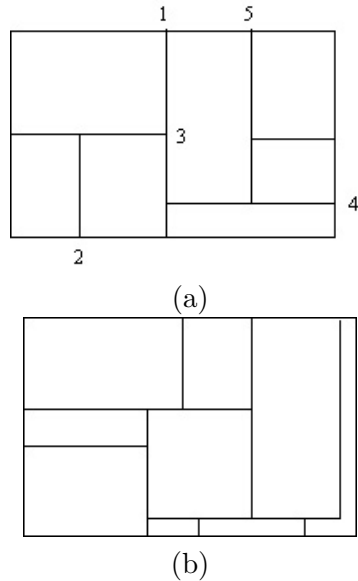


Figure 6: (a) A guillotine cutting pattern. (b) A cutting pattern infeasible with guillotine cuts.

b) The orientation of the pieces is considered fixed (a piece of length e and width f is not the same as a piece of length f and width e).

In the CGC2D problem let us assume that the cutting of A_0 into rectangles is performed in stages, where the number of stages is the number of cuts made in A_0 . In Fig. 6 two possible cutting patterns are shown, the first one (a) is a feasible four-stage cutting pattern. The number next to the cuts indicates the stage at which the cut is made. Thus, the cut direction at the first stage is parallel to the y -axis; at the second stage two cuts are made parallel to the x -axis; at the third stage two cuts are made parallel to the y -axis and at the fourth stage one cut is made parallel to the x -axis. In the second figure (b) a cutting pattern infeasible is shown. See [1, 4] for more information.

4.1 The System

The prototype developed for the guillotine cut problem shares several elements with the one presented before for the protein structure prediction problem and was developed having in mind the results presented in [9].

In this case, the information of the problem needs to be loaded from a text file whose format is extremely easy. Once the problem is loaded, the user has to decide what type of cut he/she will like: free cuts or only valid ones. In the last case, the potential positions of the cuts are constrained by the dimensions of the pieces. For example, if we have a piece with value $(30, 10)$, then the cut on the x -axis may be done at positions $0, 30, 60, 90, \dots$, etc.

Then, as before, the parameters of *FANS* need to be adjusted from the interface:

- Neighborhood Scheduler: three possible definitions are provided
- value of λ : the user should indicate which is the minimum level of acceptability required
- Operator Parameters: the modification operator changes k values from the current solution. The initial and final values for k should be provided
- Evaluations Available: this value determines the number of solutions generated.
- Number of Runs: repetitions to be performed. Each run starts from a different random seed.

The solutions obtained after running the heuristic can be saved for a later analysis.

4.2 Usage

This prototype has been used to tackle real problems arising at a company dedicated to the fabrication of chimneys, where metal plates are cut with different shapes and sizes as a function of the chimney models.

The solutions obtained using the program were considered as highly satisfactory because they helped to improve the cutting process from two points of view: material saving and experts satisfaction. The last point is related with the use of adaptation in *FANS* to change the way the cuts are done as a function of the state of the search. Some expert knowledge is stored on a “knowledge database” and such information is available to perform the adaptation of the operators.

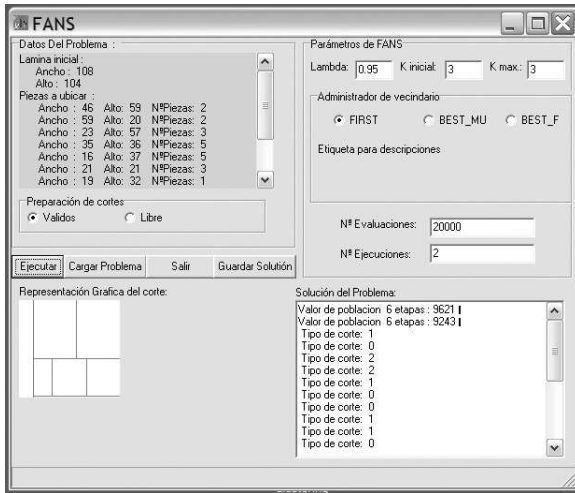


Figure 7: DSS prototype for Guillotine Cuts Problem: Main Screen

5 Conclusions

In this paper, we described two soft computing based DSS's for two hard combinatorial optimization problems: one from the bioinformatics area, and the other from the class of cutting/packing problems. Each DSS was briefly described and their current and past usage was presented.

At the core of both systems, a fuzzy sets-based heuristic is used to solve the combinatorial problems described. From an optimization point of view, the results obtained are successful, thus the door is open to perform the transition from prototypes to more complete decision support systems.

Acknowledgments

This work is supported by Project TIC2002-04242-C03-02 (Spanish Ministry of Science and Technology)

References

- [1] J. Beltrán, J. Calderón, R. Cabrera, J. M. Pérez, and J. Moreno-Vega. Fuzzy stopping rules for the strip packing problem. In *Procs of IPMU 2004*.
- [2] B. Blackburne and J. Hirst. Evolution of functional model proteins. *Journal of Chemical Physics*, 115(4):1935–1942, 2001.
- [3] A. Blanco, D. Pelta, and J. Verdegay. A fuzzy valuation-based local search framework for combinatorial problems. *Journal of Fuzzy Optimization and Decision Making*, 1(2):177–193, 2002.
- [4] N. Christofides and E. Hadjiconstantinou. An exact algorithm for orthogonal 2-d cutting problems using guillotine cuts. *European Journal of Operational Research*, 83(1):21–38, 1995.
- [5] K. A. Dill. Dominant forces in protein folding. *Biochemistry*, 24:1501, 1985.
- [6] F. Glover and G. A. Kochenberger, editors. *Handbook of Metaheuristics*, Vol. 57 of *Intern. Series in Operations Research & Management Science*. Kluwer, 2003.
- [7] W. Hart, N. Krasnogor, J. Smith (Eds). *Recent Advances in Memetic Algorithms*, Vol 166 of *Studies in Fuzziness and Soft Computing*. 2005,.
- [8] N. Krasnogor, W. Hart, J. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In W. Banzhaf et al. Eds, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pp 1596–1601. Morgan Kaufman, 1999.
- [9] J. Peláez, A. Mesas, and J. D. na. Un algoritmo genético para el problema del corte con guillotina en 2d. In *Actas del III Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, pages 30–36, 2004.
- [10] D. Pelta, A. Blanco, and J. L. Verdegay. *Fuzzy Sets based Heuristics for Optimization*, chapter Fuzzy Adaptive Neighborhood Search: Examples of Application. pp 1-20, 2003.
- [11] J. L. Verdegay. *Fuzzy Sets based Heuristics for Optimization*. Studies in Fuzziness and Soft Computing. Springer, 2003.