Multimeme Algorithms Using Fuzzy Logic Based Memes For Protein Structure Prediction

David A. Pelta[1] and Natalio Krasnogor[2]

[1] Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada, Spain
http:\\www.ugr.es\~dpelta
dpelta@ugr.es
[2] Automatic Scheduling, Optimisation and Planning Group
School of Computer Science and IT
University of Nottingham, U.K.
http:\\www.cs.nott.ac.uk\~nxk
natalio.krasnogor@nottingham.ac.uk

Summary. In this chapter we extend our previous studies on the self-adaptation of local searchers within a Memetic Algorithm.Self-adaptation allows the MA to learn which local searcher to use during search. In particular, we extend our results in [12], where memes were instantiated as Fuzzy-Logic based local searchers, and we show that our Multimeme algorithms are capable of producing new optimum solutions to instances of the *Protein Structure Prediction Problem in the HP-model.*

1 Introduction

Fuzzy Adaptive Neighborhood Search (*FANS*) was introduced in [4, 23]. Building upon local search, a classical method often used in optimization and operational research, and some basic elements of Fuzzy Sets theory, *FANS* was shown to be a robust optimization tool. This was noted for a variety of domains like knapsack problems [4], continuous function minimization [23] and more recently [23, 24, 26] in the protein structure prediction problem.

In our previous work [4], *FANS* was compared against a genetic algorithm. It was verified that both algorithms have similar performance for the range of problems studied. However, one of the advantages of using *FANS* is the easier implementation and parameter tuning. On the other hand, *FANS* performs its search by sampling one solution at a time which in some cases compromises its global search capabilities; as the Genetic Algorithm keeps a population of solutions it (more) consistently avoid local optima and performs a more global search.

In [12] we hybridized a Multimeme Algorithm [23, 28] with a simplified version of *FANS* in order to implement the pool of local searchers that the Mul-

timeme algorithm used. We demonstrated how *FANS*, and in turn fuzzy sets and systems ideas, could be successfully used to design a wide range of memes' behaviors. Moreover, we showed some benefits of using a Fuzzy-Evolutionary hybrid to tackle the Protein Structure Prediction problem (PSP).

The problem of predicting the three-dimensional structure of a protein is, perhaps, the single most important problem that biochemistry and bioinformatics face today. Even after almost five decades of intensive research it has not been "cracked". All-atom models of the folding process are extremely expensive. Moreover, there is no unique and ideal model for folding simulations, therefore, researchers use simplified descriptions of the phenomenon and tackle the slightly simpler (yet still intractable) problem of predicting the final structure of the folding process rather than the process itself. In this research we use such model, known as the HP-model [8]. The later has been widely used to benchmark folding and structure prediction algorithms and it was the source of important theoretical insights on the Protein Folding process [10].

This paper is organized as follows: in section 2 the protein structure prediction problem is introduced. Then in section 3 a brief descriptions of Memetic and Multimeme algorithms are presented. The hybrid approach we propose, i.e. a Multimeme Algorithm that includes *FANS* as local searchers, is described in Section 4. In order to assess the usefulness of the approach, several computational experiments were performed. These are described in Section 6 and the results discussed there. A section with conclusions ends the chapter.

2 The Protein Structure Prediction Problem

A protein is a chain of amino acid residues that folds into a specific *native* tertiary structure under certain physiological conditions. Proteins unfold when folding conditions provided by the environment are disrupted, and many proteins spontaneously re-fold to their native structures when physiological conditions are restored. This observation is the basis for the belief that prediction of the native structure of a protein can be done *computationally* from the information contained in the amino acid sequence alone.

In practice, solving the structure prediction problem means finding an adequate energy formulation (that correctly identifies native states) and being able to (by means of an adequate algorithm) search for candidate native states under that energy formulation. Exhaustive search of a protein's conformational space is clearly not a feasible algorithmic strategy for *PSP*. The number of possible conformations is exponential in the length of the protein sequence, and even powerful computational hardware is not capable of enumerating this space for even moderately large proteins. As an example consider the case where a protein structure is confined to a three dimensional cubic lattice. In this case, for a protein of length n there are potentially 4.7^n accessible conformations. Furthermore, recent computational analysis of *PSP* have

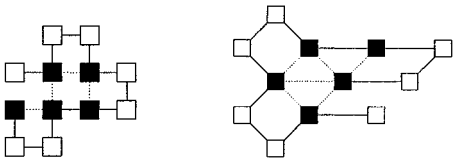


Fig. 1. HP sequence embedded in the square lattice and triangular lattice.

shown that this problem is intractable even on simple lattice models [1, 2, 7] such as the three dimensional case mentioned above.

A way of partially overcoming both the problem of the energy formulation and the enormous amount of candidate structures to analyze, is to use reduced protein models and knowledge-based potentials. Such simplified protein models are continuously playing an important role in improving our understanding of the fundamental physical properties of real-life proteins while paving the way for the development of algorithms to predict their native conformations using just the information of the amino acid sequence.

HP models abstract the hydrophobic interaction process in protein folding by reducing a protein to a heteropolymer that represents a predetermined pattern of hydrophobicity in the protein; non-polar amino acids are classified as hydrophobics and polar amino acids are classified as hydrophilics. A sequence is $s \in \{H, P\}^+$, where H represents a hydrophobic amino acid and P represents a hydrophilic amino acid.

The HP model restricts the space of conformations to self-avoiding paths on a lattice in which vertices are labelled by the amino acids. 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 and not consecutive in the sequence. The value of ϵ is typically taken to be -1.

Figure 1 shows a sequence embedded in the square and the triangular lattice, with hydrophobic-hydrophobic contacts (HH contacts) highlighted with dotted lines. The conformation in Fig. 1(a) embedded in a square lattice, has an energy of -4, while the embedding in the triangular lattice (b) has an energy of -6 (there are 4 and 6 dotted lines, i.e. contacts, in the figure).

The particular version of the problem that we are going to tackle in this chapter is given by:

Maximum Protein Structure Prediction
Instance: A protein, i.e. a string over the alphabet $\{H, P\}$ ($s \in \{H, P\}^*$).
Solution: A self avoiding embedding of s into a 2D square lattice.
Measure: The number of Hs that are topological neighbors in the embedding (neighbors in the lattice but not consecutive in s)

Protein structure prediction has been shown to be NP-complete for a variety of simple lattice models (see Atkins and Hart [1] for a recent review), including the HP-model version on the square [7] and cubic lattices [2]. A wide variety of global optimization techniques have been applied to various models of the PSP problem, e.g. see the papers in Biegler et al. [3], Pardalos, Shalloway and Xue [21] and Pelta et al. [26]. Evolutionary algorithms (in their various forms) were shown to be particularly robust and effective global optimization techniques for molecular conformation problems. In particular, evolutionary methods have been used by several researchers engaged in proteomics related activities [9, 10, 11, 23, 15, 16, 22, 23, 27, 28, 29, 30, 31].

3 Memetic Algorithms

Memetic Algorithms are metaheuristics designed to find solutions to complex and difficult optimization problems. They are evolutionary algorithms that include a stage of individual optimization or learning as part of their search strategy. Memetic Algorithms are also called hybrid genetic algorithms, genetic local search, etc. A simple Memetic Algorithm scheme is shown in Fig. 2.

The inclusion of a local search stage into the traditional evolutionary cycle of crossover-mutation-selection is not a minor change of the evolutionary algorithm architecture. On the contrary, it is a crucial deviation that affects how local and global search is performed. The reader should also note that the pseudocode shown in Fig. 2 is just one possible way to hybridize a genetic algorithm with local search. In fact, a great number of distinct memetic algorithms' architectures have been presented in the literature and even integrated into formal models [23, 13].

An interesting variant of memetic algorithms are the Multimeme Algorithms (*MMA* in what follows) as introduced in [23, 28]. *MMA* are memetic algorithms where several types of local searchers, called memes, are available to the evolutive process during the local optimization phase. An individual in a *MMA* is composed of a genetic part, representing the solution to the problem being solved, and a memetic part, encoding a meme or local searcher, that is employed during the individual optimization stage.

The set of memes available to the algorithm is called the *memepool* and its design is a critical aspect for the success of the metaheuristic. Several design criteria for the memepool are described in [23]. Multimeme algorithms for the Protein Structure Prediction problem and Protein Structure Comparison Problem are reported in [11] and [5] respectively.

```
Memetic_Algorithm():
Begin
  t = 0;
  /* We put the evolutionary clock (generations), to null */
  Randomly generate an initial population P(t);
  Repeat Until ( Termination Criterion Fulfilled ) Do
    Compute the fitness f(p) ∀p ∈ P(t) ;
    Accordingly to f(p) choose a subset of P(T), store them in M(t);
    Recombine and variate individuals in M(t), store result in M'(t);
    Improve_by_local_search( M'(t));
    Compute the fitness f(p) ∀p ∈ M'(t) ;
    Generate P(t+1) selecting some individuals from P(t) and M'(t);
    t = t + 1;
  endDo
  Return best p ∈ P(t − 1);
End.
```

Fig. 2. A basic version of a memetic algorithm.

4 Fuzzy Memes for Multimeme Algorithms

The Fuzzy Adaptive Neighborhood Search Method (*FANS*) [4, 25] is a local search procedure which differs from other local searchers in two aspects. The first aspect is how the solutions are evaluated. Within *FANS* a fuzzy valuation representing some (maybe fuzzy) property P is used together with the objective function to obtain a "semantic evaluation" of the solution. In this way, we may talk about solutions satisfying P to a certain degree. Thus, the neighborhood of a solution effectively becomes a fuzzy set with the neighbor solutions as elements and the fuzzy valuation as the membership function.

The fuzzy valuation enables the algorithm to achieve the qualitative behavior of other classical local search schemes [4]. *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 arises: 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 simplified scheme of *FANS* used here is shown in Fig. 3. The execution of the algorithm finishes when some external condition holds. In this research this happens when the number of cost function evaluations reached a prespecified limit. Each iteration begins with a call to the *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 $\mathcal{O}^k$ (a parameterized operator which is used to construct solutions). The neighborhood scheduler can return two

```
Procedure FANS:
Begin
  InitVariables();
  k:=maxK;;
  While ( not-end ) Do
    /* The neighborhood scheduler NS is called */
    S_new = NS(O^k, μ, S_cur);
    If (S_new is good enough in terms of μ()) Then
      S_cur := S_new;
      adaptFuzzyValuation(μ(), S_cur);
      Else
      /* NS could not obtain a good enough solution */
      /* The operator will be changed modifying the parameter k */
      If ((k=1)) Then
        k:= maxK;;
        Else
        k := k-1;;
      endIf
    endIf
  endDo
End.
```

Fig. 3. Scheme of *FANS*

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 state of the search. This mechanism allows the local search stages to adapt during the search, hence accordingly to [23] the *FANS* based memes are adaptive helpers. 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. In the full version of *FANS*, the strategy for this adaptation is encapsulated in the so called *operator scheduler* OS. Here we simply decrease the value of the parameter k of the operator $\mathcal{O}$. Effectively this induces, for each fixed operator, a variable radius search. At the beginning, the radius of the search is wide and it will be reduced as the search progresses. The next time NS is executed, it will have a modified operator (i.e., a different radius) to search for solutions.

The reader should note that what varies at each iteration are the parameters used in the $\mathcal{NS}$ call. The algorithm starts with $\mathcal{NS}$ $(s_0, \mathcal{O}^{t_0}, \mu_0)$. If $\mathcal{NS}$ could retrieve an acceptable neighborhood solution, the next iteration the call will be $\mathcal{NS}$ $(s_1, \mathcal{O}^{t_0}, \mu_1)$, the current solution is changed and the fuzzy valuation is adapted. If $\mathcal{NS}$ 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 $\mathcal{NS}\ (s_l, \mathcal{O}^{t_1}, \mu_l)$.

5 Description of the Memepool

Multimeme algorithms (the overall strategy guiding the search behind our approach) have been described in detail elsewhere [5, 11, 23, 28], so we only describe here the memepool our *MMA* employs.

The memes of our *MMA* are implemented as simplified versions of *FANS* as a way to obtain a wide range of behaviors in a simple and unified fashion [12]. For the neighborhood scheduler, a *First* strategy was implemented: given the current solution s, the scheduler samples the search space with the operator $\mathcal{O}$ and returns the first solution satisfying $\mu(f(s), f(O(s))) \geq \lambda$ using at most certain number of trials (length of the local search), defined here as $n/2$ where n is the size of the instance. The value λ represents the minimum level of acceptability required for a solution to be considered as a "good enough" solution.

Each meme is identified by a 3-tuple:

$$(< basic\,operator >< fuzzy\,valuation >< value\,of\,\lambda >) \qquad (1)$$

where each element will be described below.

The $<$ *basic operator* $>$ can be instantiated to anyone of the following basic moves:

0. $Reflex(i,k)$: This operator reflects the protein structure across one of its symmetry axes. The change takes place between residues i and $i+k$.
1. $Shuffle(i,k)$: This operator performs a random re-positioning of the residues i^{th} to $(i+k)^{th}$.
2. $Stretch(i,k)$: The stretch operator unfolds a substructure of length k starting from residue i.
3. $Pivot(k)$: The pivot operator represents a rigid rotation. In this case, k random residues are selected and rigid rotations are performed sequentially on each one of them.

The operator has a parameter k indicating the number of positions to change. This value of k will be modified when the neighborhood scheduler fails to return an acceptable solution. In this case, the value of k is decremented by 1. When the failure occurs with $k = 1$, the value is set again to $k = maxK$.

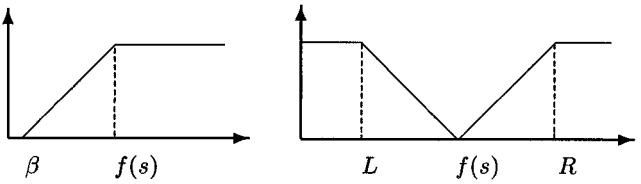


Fig. 4. Fuzzy Valuations μ_1 (left) and μ_2 (right).

There are two options available for the item $< fuzzy\ valuation >$:

1. The first fuzzy valuation proposed, μ_1, has the following definition:

$$\mu_1(s,q) = \begin{cases} 0.0 & \text{if } f(q) < \beta \\ (f(q)-\beta)/(f(s)-\beta) & \text{if } \beta \leq f(q) \leq f(s) \\ 1.0 & \text{if } f(q) > f(s) \end{cases} \quad (2)$$

where β is a threshold specifying what is, and what is not, considered an acceptable deterioration in solution quality. Given that the energy of a structure can take negative values (e.g. when the structure is not self-avoiding), the parameter β has two definitions[3]: when $f > 0$ then $\beta = f * 0.5$ (a deterioration in cost of 50% is allowed); when $f < 0$ then $\beta = f * 1.2$ (a deterioration in cost of 20% is allowed). This fuzzy valuation promotes acceptability to solutions improving the current cost. When used with $\lambda = 1$, it induces in *FANS* a hillclimber like behavior, allowing transitions only to improving solutions. The graphical representation of μ_1 is shown in Fig. 4 (left).

2. The second fuzzy valuation proposed, μ_2, has the following definition:

$$\mu_2(s,q) = \begin{cases} 1.0 & \text{if } R \leq f(q) \leq L \\ (f(s)-f(q))/(f(s)-L) & \text{if } L < f(q) \leq f(s) \\ (f(q)-f(s))/(R-f(s)) & \text{if } f(s) < f(q) \leq R \end{cases} \quad (3)$$

here, the parameters L and R are defined as follows: when $f(s) > 0$ then $L = f(s) * 0.5$ and $R = f(s) * 1.5$; when $f(s) > 0$ then $L = f(s) * 1.5$ and $R = f(s) * 0.5$. This fuzzy valuation promotes diversity, in the sense of cost. Solutions similar in cost to the current one, get very low degrees of acceptability and those differing in more than 50% with respect to the cost of the current solution gets the highest degree of acceptability. The graphical representation of μ_2 is shown in Fig. 4 (right).

[3] Although in Protein Structure Prediction one tries to minimize the energy of the conformation, in this chapter we recast the problem to a maximization problem by simply multiplying the energies by -1.

Table 1. HP model test Instances for the 2D Square Lattice.

Instance	Sequence	Opt	Size
I 1	PPHPPHHPPHHPPPPPHHHHHHHH HHPPPPPPHHPPHHPPHPPHHHH	-22	48
I 2	HHPHPHPHPHHHHPHPPPHPPPHPP PPHPPPHPPPHPHHHHPHPHPHPH	-21	50
I 3	PPHHHPHHHHHHHHPPPHHHHHHHH HHPHPPPHHHHHHHHHHHHPPPPHH HHHHPHHPHP	-34	60
I 4	HHHHHHHHHHHHPHPHPPHHPPHHP PHPPHHPPHHPPHPPHHPPHHPPHP HPHHHHHHHHHHHH	-42	64

The last element to define a meme is the $< value\,of\ \lambda >$. This parameter defines the minimum level of acceptability that a solution needs to be considered as the next solution in the search. Each pair $(\mu(), \lambda)$ defines a particular behavior for the meme. For example, with the fuzzy valuation μ_1 and $\lambda = 1$, the meme will only accept transitions to improving solutions. As $\lambda \to 0$ the chance to move to cost deteriorating solutions is increased. We can say that as λ increases, the use of μ_1 leads to exploitative memes. In turn, the use of the fuzzy valuation μ_2 leads to explorative memes. The higher values of acceptability are assigned to those solution with quite different cost with respect to that of the current solution. The lower values correspond to solutions similar in cost. In this work we consider three values for λ, where $\lambda \in \{0.4, 0.8, 1.0\}$.

Here, we want to stress the overall intended dynamics of our metaheuristic:

- At the local level (i.e. the process of individual local search) FANS memes perform a fuzzy-based variable-operator **and** variable-radius local search.
- At the global level (i.e the process of population evolutionary search) the Multimeme Algorithm is co-adapting solutions to the Protein Structure Prediction and the best local searcher (i.e. meme) to use in each individual at different stages of the search.

The metaheuristic searches concurrently on both solution and searcher spaces.

6 Description of Experiments and Results

The experiments were done with the four instances of the HP model in the square lattice shown in Table 1. For each one, the length of the sequence and the optimum value of the corresponding structure are described.

We perform two experiments which differ in the size of the mating pool. In the first one, the memepool has 12 memes which arise from the combination of the four basic moves, the fuzzy valuation μ_1 and the three values of λ. In the

second experiment, the memepool size is 24 after adding 12 more memes which arise from the use of the fuzzy valuation μ_2. In this way, we are incorporating memes promoting diversification. We use also the following parameters that were used in previous experiments by the authors [12]:

1. Replacement Strategy: $(\mu = 350, \lambda = 350)$
2. Depth of the local search, i.e. number of iterations performed by each meme application: 3
3. Length of the local search, i.e. max. number of trials allocated in the neighborhood scheduler of *FANS*: $n/2$, with n the length of the sequence.

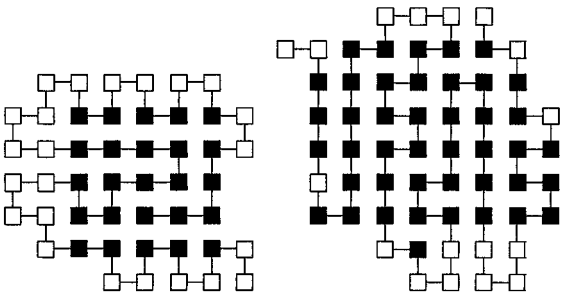
For each memepool size and instance we performed 30 runs of the *MMA*. Each run was allocated 200 generations. The initial population was generated randomly and consisted of 350 individuals. Two(consecutive)-point mutations and two-point crossover were employed with probabilities 0.2 and 0.8, respectively. In the case of mutation, the probability was per individual. The innovation rate was set to $IR = 0.2$. Tournament selection was used to select the mating parents and a tournament size of 2 individuals was used.

Three values were recorded at the end of every run: *bestF*, the fitness of the best solution found; *e2b*, the number of fitness evaluations used to reach the best solution; and *eDone*, the total amount of evaluations done in the whole run.

Tables 2, 3, 4 show the average, standard deviation, minimum and maximum values obtained for each variable over 30 runs. Each row is named $I < x > m < y >$ where $x \in \{1, 2, 3, 4\}$ stands for the instance used and $y \in \{12, 24\}$ represents the memepool size used within the *MMA*. The results obtained using the implementation of *FANS* presented in [12] are also included. *FANS* was executed 30 times, where each instance ended after 2^6 evaluations.

The first thing to notice is that for instances 1 and 3, structures with higher bonds than the known optima were obtained. For instance 1, a structure with 23 bonds was found while for instance 3, one with 35 bonds was obtained. Fig. 5 shows both structures. To the best of our knowledge, such optimal values were only achieved before in [20]. However, in that paper the authors do not measure the cost of their algorithm in number of total energy evaluations so it is impossible to provide comparisons. Moreover, they enforce a strong bias in the search to regions of the search space that contain secondary structure information derived from the native structure they are searching for. In other words, they used specific domain knowledge that is not present in our algorithm. Having this in mind, we deem our algorithms as the first blind search method (to the best of the authors knowledge) able to obtain these novel native structures.

The results in terms of *bestF* were quite similar using 12 or 24 memes. The algorithms using 24 memes had a slightly higher standard deviation and lower minimum values. The 12 memes version, achieved a higher maximum



(a) (b)

Fig. 5. New best structures obtained for instances 1, in (a), and instance 3, in (b).

Table 2. Statistics for *bestF*

Algorithm	Mean	SD	Min	Max
i1m12	20.55	0.97	19.02	23.02
i1m24	20.02	1.23	18.01	23.02
fans-i1	19.35	0.76	18.02	21.02
i2m12	19.98	1.00	18.01	21.01
i2m24	19.98	0.98	17.01	21.01
fans-i2	18.94	0.74	18.01	20.01
i3m12	32.82	0.96	31.02	35.02
i3m24	32.15	1.20	29.02	34.02
fans-i3	30.82	0.81	29.02	32.02
i4m12	33.45	1.65	30.02	38.02
i4m24	33.47	2.38	28.02	38.02
fans-i4	28.75	1.08	27.02	32.02

value on instance 3. *FANS* achieved the lowest values of standard deviation, but the higher ones in terms of the mean.

In terms of $e2b$, it is clear that the use of 24 memes allowed it to reach good results with less effort. This situation is reasonable if we consider in the number of trials that each meme has to perform to obtain an acceptable solution. Those memes using μ_2 can obtain acceptable solutions quite easily. For example, given a value $\lambda = 1$, the memes using μ_1 need to find solutions improving the cost, which may result in the use of a high number of trials. On the contrary, memes using μ_2 will accept any transition leading to a decrease in cost of more than 50%, and this is easy to achieve using a low number of trials. This aspect is confirmed looking at the statistics for *eDone*. Considering the mean, the *MMA* with 12 memes used approximately 2.35 million evaluations while the *MMA* with 24 never used more than 2 millions.

To finish the analysis, two additional aspects are considered. First, Fig. 6 shows the evolution of the *average cost of the best individual* through the generations. It can be seen that the use of 12 or 24 memes leads to very similar

Table 3. Statistics for *e2b*

Algorithm	Mean	SD	Min	Max
i1m12	1356104	481405	549691	2269640
i1m24	894661	331121	367357	1510390
fans-i1	933126	526296	160522	1960500
i2m12	1109461	545856	362400	2565580
i2m24	675173	198261	447300	1509880
fans-i2	1202764	589958	243614	2000050
i3m12	1358837	563218	605457	2790530
i3m24	990789	343991	356614	1599350
fans-i3	1066657	590136	65746	1940400
i4m12	1852648	399233	991046	2356310
i4m24	1520624	376036	469575	2005740
fans-i4	1074369	562966	76854	1965090

Table 4. Statistics for *eDone*

Algorithm	Mean	SD	Min	Max
i1m12	2310653	218389	1864000	2579590
i1m24	1581614	90237	1396960	1758450
i2m12	2384182	166804	2005530	2739440
i2m24	1630939	79154	1507610	1859810
i3m12	2511524	214088	1997610	2873460
i3m24	1678238	86323	1457230	1867790
i4m12	2468762	213454	2168150	3258060
i4m24	1964042	77697	1805970	2147600

patterns of evolution. Looking at the graph for instance 4, we can conclude that the *MMA* has not converged when the run finished. This fact may be considered an explanation about the quite low values of *bestF* obtained.

Second, Fig. 7 shows the evolution of the *average cost of the average fitness of the whole population* through the generations. It is clear that the use of diversification memes kept the overall fitness lower (i.e. better solutions)

7 Conclusions

A hybridization strategy between a fuzzy sets-based heuristic, and a Multimeme algorithm was proposed and tested.

The construction of the memepool using simplified versions of *FANS* enabled us to obtain a wide range of fuzzy memes, each one with its particular behavior. The advantage of using *FANS* as the memes for a *MMA* over using adaptive helpers as in [23, 27] is that it is much easier to tune the search of the memes. Moreover, human knowledge or instance specific knowledge (e.g. secondary structure information as that used in [20]) can be readily incorporated into *FANS* based memes.

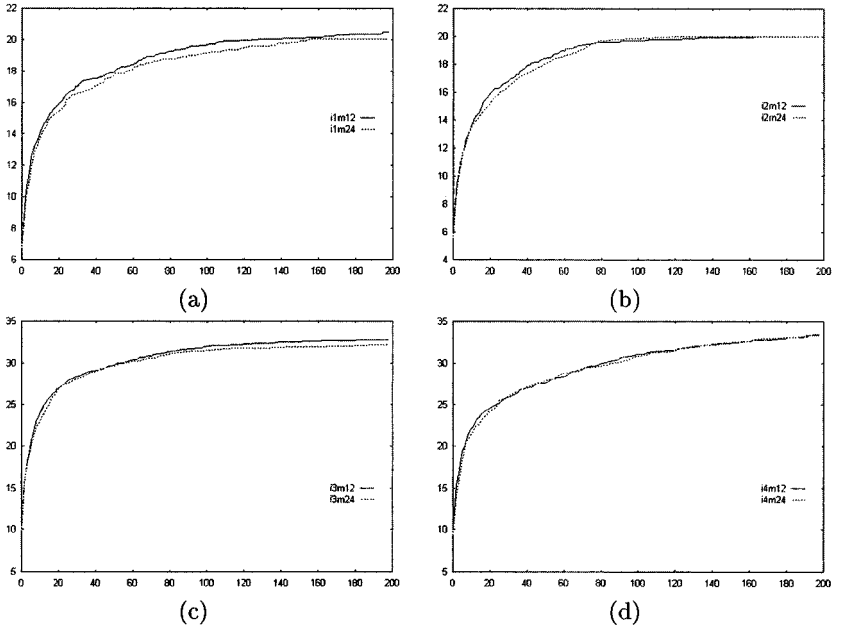


Fig. 6. Evolution of the cost of the best individual vs Generations for test instances 1 (a), 2 (b), 3 (c) and 4 (d) using a *MMA* with 12 and 24 memes.

The scheduling of memes by the simple inheritance mechanism was proven successful in the detection of the most suitable fuzzy meme for different stages of the search. This has been verified in other domains [23, 28], which deems Multimeme Algorithms a very robust metaheuristic.

The coupled effect of both elements lead to a robust and general purpose metaheuristic. In the test cases shown in this chapter it was able to improve previous results in the protein structure prediction problem. We suggest that this approach can be a powerfull metaheuristic for other combinatorial problems.

8 Acknowledgments

This research was partially founded by Fundacion Antorchas, Republica Argentina and supported in part by Project TIC2002-04242-CO3-02.

D.A. Pelta is a grant holder from Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Republica Argentina.

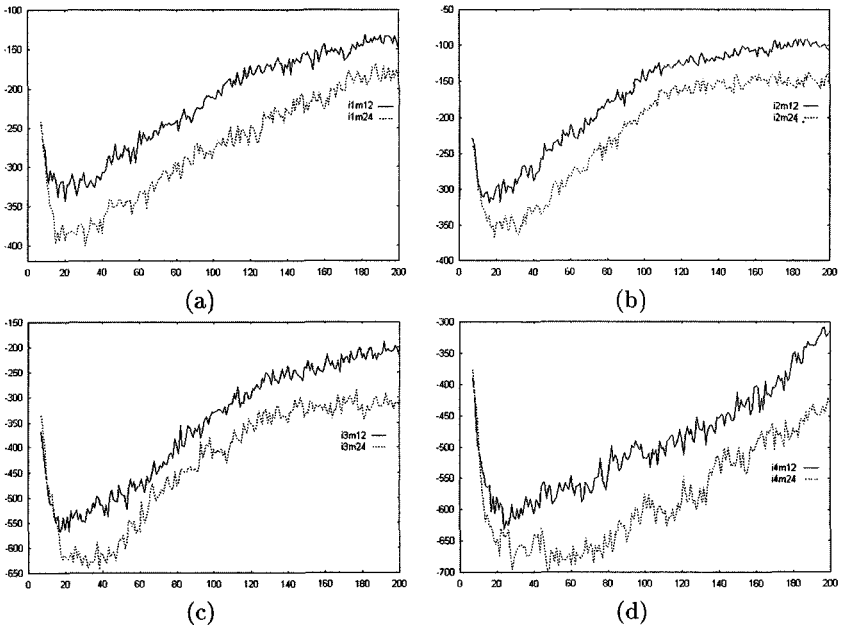


Fig. 7. Evolution of the cost of the average fitness of the population vs Generations for test instances 1 (a), 2 (b), 3 (c) and 4 (d) using a *MMA* with 12 and 24 memes.

References

1. J. Atkins and W. E. Hart. On the intractability of protein folding with a finite alphabet. *Algorithmica*, pages 279–294, 1999.
2. 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.
3. L. T. Biegler, T. F. Coleman, A. R. Conn, and F. N. Santosa, editors. *Large-Scale optimization with applications. Part III: Molecular structure and optimization*, volume 94 of *The IMA Volumes in Mathematics and its Applications*. Springer-Verlag, New York, 1997.
4. 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.
5. B. Carr, W.E. Hart, N. Krasnogor, E. Burke, J. Hirst, and J. Smith. Alignment of protein structures with a memetic evolutionary algorithm. In *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufman, 2002.
6. T. E. Creighton, editor. *Protein Folding*. W. H. Freeman and Company, 1993.
7. P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proceedings of The Second Annual International Conference on Computational Molecular Biology, RECOMB 98*, pages 51–62. ACM Press, 1998.

8. K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24:1501, 1985.
9. G. Greenwood, B. Lee, J. Shin, and G. Fogel. A survey of recent work on evolutionary approaches to the protein folding problem. In *Proceedings of the Congress of Evolutionary Computation (CEC)*, pages 488–495. IEEE, 1999.
10. M. Khimasia and P. Coveney. Protein structure prediction as a hard optimization problem: The genetic algorithm approach. In *Molecular Simulation*, volume 19, pages 205–226, 1997.
11. N. Krasnogor. *Studies on the Theory and Design Space of Memetic Algorithms.* Ph.D. Thesis, University of the West of England, Bristol, United Kingdom. (http://dirac.chem.nott.ac.uk/~natk/Public/papers.html), 2002.
12. N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst. Multimeme algorithms for protein structure prediction. In *Proceedings of the Parallel Problem Solving from Nature VII. Lecture notes in computer science*, 2002.
13. N. Krasnogor, W.E. Hart, J. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufman, 1999.
14. N. Krasnogor and D. Pelta. Fuzzy memes in multimeme algorithms: a fuzzy-evolutionary hybrid. In J. Verdegay, editor, *Fuzzy Sets based Heuristics for Optimization*, Studies in Fuzziness and Soft Computing, pages 49–66. Physica Verlag, 2003.
15. N. Krasnogor, D. Pelta, P. M. Lopez, P. Mocciola, and E. de la Canal. Genetic algorithms for the protein folding problem: A critical view. In C. F. E. Alpaydin, editor, *Proceedings of Engineering of Intelligent Systems.* ICSC Academic Press, 1998.
16. N. Krasnogor, D. 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.
17. N. Krasnogor and J. Smith. A memetic algorithm with self-adaptive local search: TSP as a case study. In *Proceedings of the 2000 Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 2000.
18. N. Krasnogor and J. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In *Proceedings of the 2001 Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 2001.
19. N. Krasnogor and J. Smith. Memetic algorithms: Syntactic model and taxonomy. 2001. submitted to The Journal of Heuristics. Available from the authors.
20. F. Liang and W. Wong. Evolutionary monte carlo for protein folding simulations. *Journal of Chemical Physics*, 115(7):3374–3380, 2001.
21. P. M. Pardalos, D. Shalloway, and G. L. Xue, editors. *Global minimization of nonconvex energy functions: Molecular conformation and protein folding*, volume 23 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, Providence, Rhode Island, 1996.
22. A. L. Patton. A standard ga approach to native protein conformation prediction. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 574–581. Morgan Kauffman, 1995.
23. D. Pelta, A. Blanco, and J. L. Verdegay. A fuzzy adaptive neighborhood search for function optimization. In *Fourth International Conference on Knowledge-Based Intelligent Engineering Systems & Allied Technologies, KES 2000*, volume 2, pages 594–597, 2000.

24. D. Pelta, A. Blanco, and J. L. Verdegay. Applying a fuzzy sets-based heuristic for the protein structure prediction problem. *Internation journal of Intelligent Systems*, 17(7):629–643, 2002.
25. D. Pelta, A. Blanco, and J. L. Verdegay. Fuzzy adaptive neighborhood search: Examples of application. In J. L. Verdegay, editor, *Fuzzy Sets based Heuristics for Optimization*, Studies in Fuzziness and Soft Computing, pages 1–20. Physica-Verlag, 2003.
26. D. Pelta, N. Krasnogor, A. Blanco, and J. L. Verdegay. F.a.n.s. for the protein folding problem: Comparing encodings and search modes. In *Fourth International Metaheuristics Conference, MIC 2001*, 2001.
27. A. Piccolboni and G. Mauri. Protein structure prediction as a hard optimization problem: The genetic algorithm approach. In N. e. a. Kasabov, editor, *Proceedings of ICONIP '97*. Springer, 1998.
28. A. A. Rabow and H. A. Scheraga. Improved genetic algorithm for the protein folding problem by use of a cartesian combination operator. *Protein Science*, 5:1800–1815, 1996.
29. S.-K. S. Genetic algorithms for protein tertiary structure prediction. In *Parallel Problem Solving from Nature - PPSN II*. North-Holland, 1992.
30. R. Unger and J. Moult. A genetic algorithm for three dimensional protein folding simulations. In *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA-93)*, pages 581–588. Morgan Kaufmann, 1993.
31. R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, 231(1):75–81, 1993.