Self-Assembling of Local Searchers in Memetic Algorithms

Natalio Krasnogor and Steven Gustafson

Automatic Scheduling, Optimisation and Planning Group
School of Computer Science and IT
University of Nottingham, U.K.
http:\\www.cs.nott.ac.uk/~{nxk,smg}
{Natalio.Krasnogor,Steven.Gustafson}@nottingham.ac.uk

Summary. In this chapter we concentrate on one particular class of Global-Local Search Hybrids, Memetic Algorithms (MAs), and we describe the implementation of "self-assembling" mechanisms to produce the local searches the MA uses. To understand the context in which self-assembling is applied we discuss some important aspects of Memetic theory and how these concepts could be harnessed to implement more competitive MAs. Our implementation is tested in two problems, Maximum Contact Map Overlap Problem (MAX-CMO) and the NK-Landscape Problems.

Three lessons can be drawn from this paper:

- Memetic theory provides a rich set of metaphors and insights that can be harnessed within optimisation algorithms as to provide better search methods.
- The optimization of solutions can be done **simultaneously** with the self-assembling of local search strategies which can then be exploited by the Memetic Algorithm (or other metaheuristic)
- Local search strategies that are evolved to **supply building blocks** can greatly improve the quality of the search obtained by the Memetic Algorithm and do not seem to suffer from premature convergence (an ubiquitous problem for global-local hybrids).

1 Introduction

A vast number of very successful applications of Memetic algorithms (MAs) have been reported in the literature in the last years for a wide range of problem domains. The majority of the papers dealing with MAs are the result of the combination of highly specialized **pre-existing** local searchers and usually purpose-specific genetic operators. Moreover, those algorithms require a considerable effort devoted to the tuning of the local search and evolutionary parts of the algorithm.

In [23] and [25] we propose the so called "Self-Generating Metaheuristics". Self-Generating Metaheuristics can create on-the-fly the type of operators needed to successfully perform certain task. The self-generation concept

can be applied to any existing metaheuristic like simulated annealing, tabu search, etc. In the case of Memetic Algorithms, self-Generation implies that the MAs are able to **self-assemble** their own local searchers and to co-evolve the behaviors it needs to successfully solve a given problem. In Self-Generating Memetic Algorithms two evolutionary processes occur. On one hand evolution takes place at the chromosome level as in any other Evolutionary Algorithm; chromosomes and genes represent solutions and features of the problem one is trying to solve. On the other hand, evolution also happens at the memetic level. That is, the behaviors and strategies that individuals (also called agents) use to alter the survival value of their chromosomes are self-assembled from a set of components by means of, for example, an evolutionary process. As the self-assembeld memes (i.e. local search strategies) are propagated, mutated and are selected in a Darwinian sense, the Self-Generating MAs we propose are closer to Dawkins concept of memes than the previous works on memetic algorithms (e.g. [14],[33],[34],[4]). Additionally, they seem to be more robust and scalable than their single local searchers counterpart.

In this chapter we will review some important ideas arising from Memetic theory and we will describe the implementation we have choosen for the proposed algorithms. Results on the use of the Self-Assembling of local searchers for MAs are reported and future lines of research discussed.

2 The Memetic Metaphor

Memetic algorithms are not the first kind of algorithms to draw inspiration from natural phenomena. In this case the inspiration came from memetic theory. However, unlike Simulate annealing, Ant Colony optimization, GAs, etc., scholars working on MAs, as will be argued later, departed considerably from the metaphor and ignored its main features.

The common use of the term "memetic algorithm" refers to an evolutionary algorithm that employs as a distinctive part of its main evolutionary cycle (mutation, crossover and selection), a local search stage.

The name "memetic algorithm" is a very contested label that stirs critics and controversies among researchers and practitioners who usually adopt names such as lamarckian GAs, genetic local search, hybrid GAs, etc. Although very justificable in the large mayority of cases, these names obscure the fact that there is a large body of literature on memetic theory that is being neglected. We would like to argue in this section that if we were to put back the "memetic" into memetic algorithms then progress could be made with a new breed of algorithms that are more atune to the name "memetic algorithms".

Memetic theory started as such with the definition given by R. Dawkins of a meme in [11][1]:

[1] The definition was later refined in [12]

> I think that a new kind of replicator has recently emerged on this very planet. It is staring us in the face. It is still in its infancy, still drifting clumsily about in its primeval soup, but already it is achieving evolutionary change at a rate that leaves the old gene panting far behind. The new soup is the soup of human culture. We need a name for the new replicator, a noun that conveys the idea of a unit of cultural transmission, or a unit of imitation. "Mimeme" comes from a suitable Greek root, but I want a monosyllable that sounds a bit like "gene". I hope my classicist friends will forgive me if I abbreviate mimeme to meme.(2) If it is any consolation, it could alternatively be thought of as being related to "memory", or to the French word "même". It should be pronounced to rhyme with "cream". Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation.

Many other researchers and philosophers "flirted" with the idea that cultural phenomena can somehow be explained in evolutionary terms even before Dawkins' introduction of a meme. Other symbols were introduced to refer to the elementary unit of cultural change and/or transmission (e.g., *m-culture* and *i-culture* [7], *culture-type* [42], etc.). See [13] for a comprehensive analysis. The merit of Dawkins contribution can be attributed to his insight into correctly assigning a new signifier, i.e., a label or symbol, to the thing being signified, i.e., the unit of cultural transmission. The term meme was a new word hence it was not loaded with preconceptions and misconceptions. From the computer sciences perspective it was appealing because it defined that concept as a discrete structure which can be easily harnessed in a computer program.

The fundamental innovation of memetic theory is the recognition that a dual system of inheritance, by means of the existence of two distinct replicators, mould human culture. Moreover, these two replicators interact and co-evolve shaping each other's environment. As a consequence evolutionary changes at the gene level are expected to influence the second replicator, the memes. Symmetrically, evolutionary changes in the meme pool can have consequences for the genes.

2.1 Memetic Theory in Evolutionary Computation

In any of the major evolutionary computation paradigms, e.g., GAs, Evolution Programs, Evolutionary Strategies, GPs, etc, the computation cycle shown in graph 1 takes place.

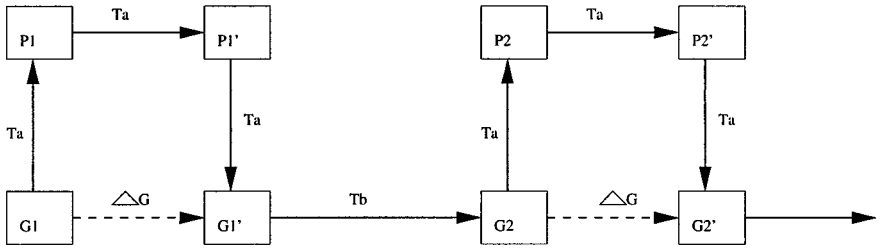


Fig. 1. *Evolutionary genetic cycle.*

In graph [2] 1 a hypothetical population of individuals is represented at two different points in time, generation 1 (G1) and at a later generation (G2). In the lower line, G_i for $i = 1, 2$ represents the distribution of genotypes in the population. In the upper line, P_i represents the distribution of phenotypes at the given time. Transformations T_A account for epigenetic phenomena, e.g., interactions with the environment, in-migration and out-migrations, individual development, etc., all of them affecting the distribution of phenotypes and producing a change in the distribution of genotypes during this generation. On the other hand transformations T_B account for the Mendelian principles that govern genetic inheritance and transforms a distribution of genotypes G_1' into another one G_2. Evolutionary computation endeavors concentrate on the study and assessment of many different ways the cycle depicted in 1 can be implemented. This evolutionary cycle implicitly assumes the existence of only one replicator: genes.

On the other hand what memetic algorithmicists should somehow investigate, if they were more faithful to the natural phenomena that inspired the methodology, is the implementation of a more general and complex dual evolutionary cycle where two replicators co-exist. This is shown in 2[3].

In the context of memetic algorithms, memes represent instructions to self-improve. That is, memes specify sets of rules, programs, heuristics, strategies, behaviors, etc, individuals can use in order to improve their own fitnesses under certain metric.

As we mentioned earlier, the fundamental difference between the later graph and the former resides in the fact that graph 2 reflects a coevolutionary system where two replicators of a different nature interact. Moreover the interactions between genes and memes are indirect and mediated by the common carrier of both: individuals. A truly memetic system should not be confused with other coevolutionary approaches where different "species", subpopulations or just different individuals interact by ways of a combination of cooperation, competition, parasitism, symbiosis, etc. In coevolutionary approaches like those described by [19],[36],[37],[38],[39],[40] and others, only

[2] This graph is adapted from [13] page 114.
[3] This graph is adapted from [13] page 186.

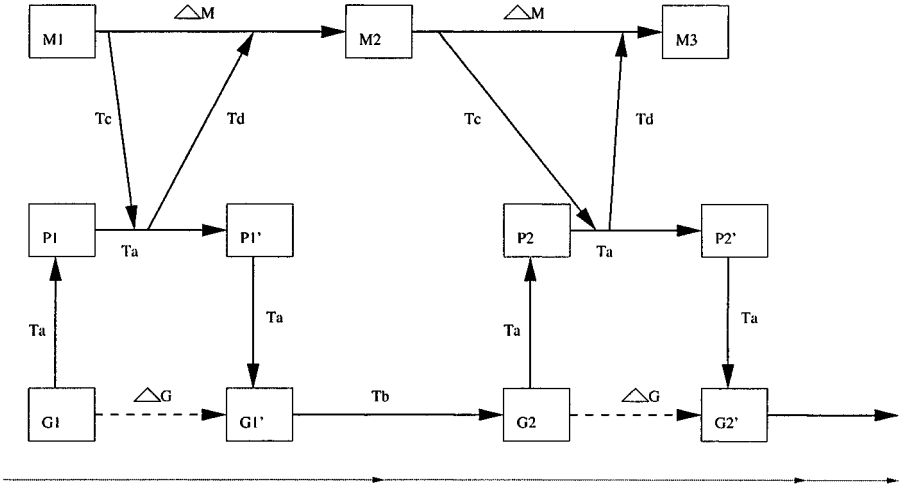


Fig. 2. *Coevolutionary memetic-genetic cycle.*

Mendelian transformations are allowed and sometimes in-migration and out-migration operators are also included. In a memetic system, memes can potentially change and evolve using rules and time scales other than the traditional genetic ones. In the words of Feldman and Cavalli-Sforza[6] memetic evolution is driven by:

> ...the balance of several evolutionary forces: (1) *mutation*, which is both purposive (innovation) and random (copy error); (2) *transmission*, which is not as inert as in biology [i.e., conveyance may also be horizontal and oblique]; (3) *cultural drift* (sampling fluctuations); (4) *cultural selection)* (decisions by individuals); and (5) *natural selection* (the consequences at the level of Darwinian fitness) ...

In graph 2 we have the same set of transformations as before between genes and phenotypes, but also meme-phenotypes and memes-memes relations are shown. There are mainly two transformations for memes that are depicted, T_C and T_D. Transformations T_C represents the various ways in which "cultural" instructions can re-shape phenotypes distributions, e.g., individuals learn, adopt or imitate certain memes or modify other memes. T_D, on the other hand, reflects the changes in memetic distribution that can be expected from changes in phenotypic distributions, e.g., those attributed to teaching, preaching, etc.

Memetic Algorithms as they were used so far failed completely, or almost completely, to implement this dual inheritance system to any degree, except for the works initiated with [22],[23] and continued with [25],[45],[24],[26]. Consequently, it is not surprising that researchers hesitate to call a GA (or other evolutionary approach) that uses local search a memetic algorithm.

2.2 Memes Self-Assembling

In [23],[25],[45],[24] it was proposed and demonstrated that the concept of Self-Generating Memetic algorithms can be implemented and, at least for the domains considered in those papers, beneficial. In the context of SGMAs, memes specify sets of rules, programs, heuristics, strategies, behaviors, or move operators the individuals in the population can use in order to improve their own fitnesses (under a given metric). Moreover the interactions between genes and memes are indirect and mediated by the common carrier of both: individuals (sometimes also called agents).

Gabora[15] mentions three phenomena that are unique to cultural (i.e. memetic) evolution. Those phenomena are *Knowledge-based, imitation* and *mental simulation.*

It is these three phenomena that our Self-Generating Memetic Algorithm implements and use to self-assemble its own local search strategies. The representation of the low level operators (in this chapter the local searchers) includes features such as the acceptance strategy (e.g. next ascent, steepest ascent, random walk, etc), the maximum number of neighborhood members to be sampled, the number of iterations for which the heuristic should be run, a decision function that will tell the heuristic whether it is beneficial for a particular solution or a particular region of a solution and, more importantly, the move operator itself in which the low level heuristic will be based[23].

Previous technologies for Evolutionary Algorithms, GRASP, Simulated Annealing, Tabu Search, etc have concentrated so far in, for example, self-adapting the probabilities with which different move operators[46] are used during the search for a problem's solution, the size of the tabu lists[49] or the size of populations[48], the adaptation of the aspiration criteria [1], the crossover points[43], mutations frequencies and intensities[47], the weights in the choice functions of Hyperheuristics [9], the appropriate local searcher that must be used by a Memetic Algorithms [28], the intensity of search[30], the degree of exploitation and exploration of local search [27], etc. However, only recentely some exploration on the on-line self-assembling of local searchers has been developed.

The role played by local search in both Memetic and Multimeme algorithms has traditionally been associated to that of a "fine tuner". The evolutionary aspect of the algorithm is expected to provide for a global exploration of space while the local searchers are assumed to exploit current solutions and to fine tune the search in the vicinity of those solutions (i.e. exploitation)

We will show next that local search strategies can be self-assembled in the realm of NK-Landscape problems and a graph theory combinatorial problem. Equally important, we will suggest a new role for local search in evolutionary computation in general and memetic algorithms in particular: *the local searcher not as a fine-tuner but rather as a supplier of building-blocks*[4]. Ini-

[4] For an overview of selectorecombinative evolutionary algorithms from a building-blocks perspective please refer to [17]

tial explorations of the many concepts described here appeared before in [25], [26] and [24] and we invite the reader to also consult those papers for further details.

3 The NK-Landscapes Experiments

The NK model of rugged fitness landscapes are particularly useful to understand the dynamics of evolutionary search[20] as they can be tunned to represent low or high epistasis regimes. The amount of epistasis is related to the level of interdependency of genes within a genome. That is, the fitness contribution of a particular gene's allele depends not only on the identity of that allele by also on which are the specific alleles in the remaining genes. To model this situation an NK-landscape instance consists of two integer n and k representing the total number of genes n and the number of other genes a gene i is epistatically realted to. The values k can take are $0 \leq k \leq n-1$. Besides n and k, a $n \times 2^{k+1}$ matrix E with elements sampled randomly from the (usually) uniform distribution $U(0,1)$ is also required to completely define an instans. A solution to an NK-landscape problem instance is represented as a binary string S with length n. The fitness of S is given by $fitness(S) = \frac{1}{n} * \sum_{i=1}^{i=n} f_i(S_i, S_{i_1}, \ldots, S_{i_k})$ where $f_i(\cdot)$ is an entry in E, S_i the value of string S at position i and S_{i_j} is the value of string S at the $j-th$ neighbour of bit i. The neighbours, not necessarily adjacent, j of bit i are part of the input as well.

In figure 3 we show a $(10,3)$ NK-landscape. The genome is formed by 10 genes ($N = 10$) and each of the genes is epistatically linked to 3 other genes ($K = 3$). In the example this is depicted by the curved arrows going out from gene i towards adyacent genes.

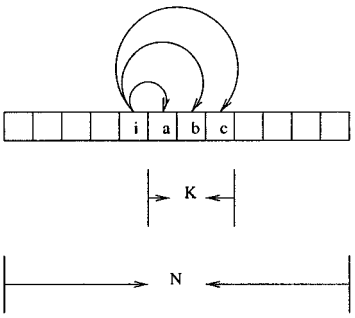


Fig. 3. *An example of NK-Landscapes*

Low values for k represent low epistatic problems while large k value make up highly epistatic landscapes. The extreme case of an uncorrelated random fitness landscape is when $k = n-1$. The optimization version of this problem

can be solved in polynomial time by dynamic programming if the neighborhood structure used is that of *adjacent neighbours*. The problem becomes NP-Hard if the structure used is that of *random neighbours*[50].

NK-Landscapes have been the subject of intensive and varied studies. Kaufmann et al. [31] explore a phase change in search when a parameter τ of a local search algorithm reaches a certain critical value on some NK-Landscape problems .

In their paper the authors show experimentally that the quality of the search follows an $s-$shape curve when plotted against τ making evident a change in phase. M. Oates et al. [35] showed performance profiles for evolutionary search based algorithms where phase changes were also present. Krasnogor and Smith [28] and Krasnogor [23] showed the existence of the "solvability" phase transition for GAs (instead than LS) and demonstrated that a self-adapting MA can learn the adequate set of parameters to use. Merz [32] devotes at least one whole chapter of his Ph.D. dissertation to the development of efficient Memetic Algorithms for this problem (we will return to his MAs later on). With a different target as the object of research O.Sharpe in [44] performs some analysis on the parameter space of evolutionary search strategies for NK landscapes.

The NK-Landscapes represent a rich problem and they are an ideal test case for our purposes. We will describe the behavior of our Self-Generating Memetic Algorithms in 4 different regimes: *low epistasis and poly-time solvable*, *high epistasis and poly-time solvable*, *low epistasis and NP-hard* and *high epistasis and NP-hard.*

In [23] and in previous sections we argued briefly about the need to creatively adapt every aspect of the local searchers, that is, the acceptance strategy, the maximum number of neighborhood members to be sampled, the number of iterations for which the meme should be run, a decision function that will tell the meme whether it is worth or not to be applied on a particular individual and, more importantly, the move operator itself. In this part of the chapter we will focus only on the self-generation of the move operator itself as a proof of concept[5].

The MA will be composed of two simultaneous processes. Individuals in the MA population will be composed of genetic and memetic material. The genetic material will represent a solution to NK-Landscapes problems (i.e. a bit string) while the memetic part will represent "mental constructs" to optimize the NK-Landscape string. As such we will be evolving individuals whose goal is to self-optimize by genetic evolution (first process) and memetic evolution (second process) as sugested by figure 2.

3.1 The Self-Generating Memetic Algorithm

The pseudocode in Figure 2 depicts the algorithm used to solve the NK-Landscape Problem.

[5] The other aspects are actively being investigated.

```
Memetic_Algorithm():
Begin
  t = 0;
  /* Initialize the evolutionary clock(generations) to 0 */
  Randomly generate an initial population P(t);
  /* The individuals in the population */
  /* are composed by genes & memes */
  /* both randomly initialized */
  Repeat Until ( Termination Criterion Fulfilled ) Do
    Variate individuals in M(t);
    /* The variation of an individual includes */
    /* both genetic and memetic variation */
    Improve_by_local_search( M(t));
    /* The local search is performed accordingly */
    /* to the individual's meme */
    Compute the fitness f(p) ∀p ∈ M(t) ;
    Generate P(t+1) by selecting from P(t) and M(t);
    t = t+1;
  endDo
  Return best p ∈ P(t-1);
End.
```

Fig. 4. The memetic algorithm employed.

The initial population in P is created at random. Each individual is composed of genetic material in the form of a bit string (B). The bit string represent the solution to the NK landscape instance being solved. The memetic material is of the form $* \rightarrow S$ where the $*$ symbol matches any bit in the solution string and S is another bit string. The only variation mechanism is bitwise mutation (applied with probability 0.05) to the chromosomes. The replacement strategy is a $(20, 50)$. There is no genetic crossover but the SIM mechanism, as described in [28], is used to transfer memes between individuals. Memetic mutation occurs with an innovation rate[23] of 0.2. A meme can be mutated (with equal probability) in three ways: either a random bit is inserted in a random position, a bit is deleted from a random position, or a bit is flipped at a random position. The length of memes cannot decrease below 0 nor increase beyond $3 * k$ for an $(n, k)-$problem.

The Local Search Procedure: Memes description for NK-Landscapes

A meme is represented as a rule of the form $* \rightarrow S$. During the local search stage this meme is interpreted as follows:

Every bit in the chromosome B has the opportunity to be improved by steepest hill-climbing. In general NK-Landscapes are epistatic problems so flipping only one bit at a time cannot produce reasonable improvements except of course in problems with very low k. To accommodate for this fact, for each bit, one wants to optimize the value of that bit and that of $|S|$ other bits. A sample of size n is taken from all the $(|S|+1)!$ possible binary strings. Based on the content of S, these sample strings serve as bits template with which the original chromosome B will be modified. If $|S| = 0$ then only B_i (the i^{th} bit of B) will be subjected to hill-climbing. On the other hand, if $|S| > 0$ then the local searchers scans the bits of S one after the other. If the first bit of S is a 0, then the bit $B_{(i+1)}$ will be set accordingly to what one of the n samples template mandates. On the other hand, if B_i is a 1 then bit $B_{(i+r)\%n}$ will be set as what one of the n samples template mandates. Here r is a random number between 0 and $n-1$. By distinguishing in S between ones and zeroes memes can reflect the adjacent neighbour or the random neighbour version of the NK-landscapes. The larger the size of S the more bits will be affected by the local search process. As an example consider the case where the rule is $* \rightarrow 0000$. This rules implies $S = 0000$. In this case, for every bit i in B we will produce a sample of size n out of the possible 2^5 binary strings. Each one of these samples will be used as a template to modify B. As S is built out of all 0's, a fully-adjacent neighbourhood is considered. Suppose $B = 1010101010101111 0$ and the bit to be optimized is the fourth bit. B_4 equals 0 in the example and its four adjacent neighbours are $B_5 = 1, B_6 = 0, B_7 = 1, B_8 = 0$. If one of the n samples is 11111 then B will be set to $B' = 1011111101010111 0$ provided B' has better fitness than B. The process is repeated in every bit of B once for every sample in the sample set.

Several complex local search strategies have been applied to the NK-landscapes domain. For example Merz in [32] uses various optimisation features with the aim of acelerating his MAs. Furthermore, in chapter 6 in this book he describes varios $K-opt$, $Lin-Kernighan$ and other sophisticated heuristics for NK problems. In the following case studies, initially explored in [25] and [57], we use simpler local searchers to serve only as a proof of concept. The evolved memes induce a variable-sampled $k-opt$ local search strategy. We say variable as k varies with the size of S and it can be as small as 0 or as large as $3 * k$. It is sampled as we do not exhaustively explore all the 2^{k+1} possible ways of settings the bits in a chromosome but rather take a reduced sample of size n.

3.2 Results

In previous sections we described our self-generating MAs. What sort of behaviors can we expect to see emerging? Four different scenarios needs to be analysed: low epistasis-poly-time solvable, high epistasis-poly-time solvable, low epistasis-NP-hard and high epistasis-NP-hard landscapes. The level of epistasis is controlled by the n and k. The closer k is to 0 the more negli-

gible the epistatic interactions among loci. If k grows up to $n-1$ then the induced problems is a random field. The transition between polynomial time solvability to NP-hardness depends on the type of neighborhood used as it was explained before. We should expect the emergence of short strings (i.e. $|S|$ not too big) for the low epistasis regimes while longer strings will be favored in high epistasis cases. We should be able to compare the length of the evolved local searcher with the k of the problem that is being solved. That is, we expect to see memes emerging with lengths close to k. We should probably also see distinct patterns of activity for the different problem regimes. The range of problems we experimented with are:

- low epistasis, poly-time solvable: $(50,1),(50,4)$ with adjacent neighbours.
- high epistasis, poly-time solvable: $(50,8),(50,10),(50,12),(50,14)$ with adjacent neighbours
- low epistasis, NP-hard: $(50,1),(50,4)$ with random neighbours.
- high epistasis, NP-hard: $(50,8),(50,10),(50,12),(50,14)$ with random neighbours.

3.3 Discussion

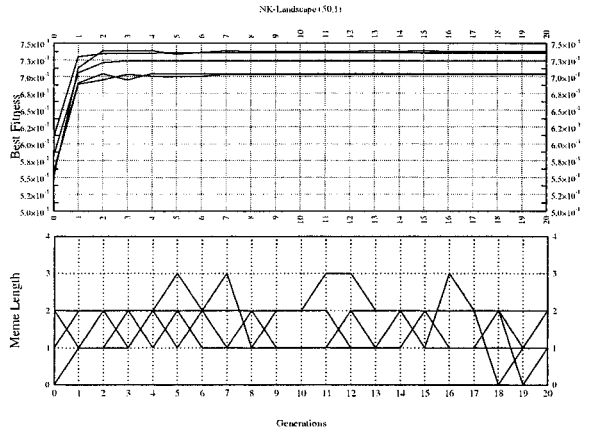
In the following figures we plot the evolution of the length of the meme associated with the fittest individual as a function of time and the evolution of fitness. For clarity, just 5 runs are depicted.

Low epistasis, poly-time solvable:

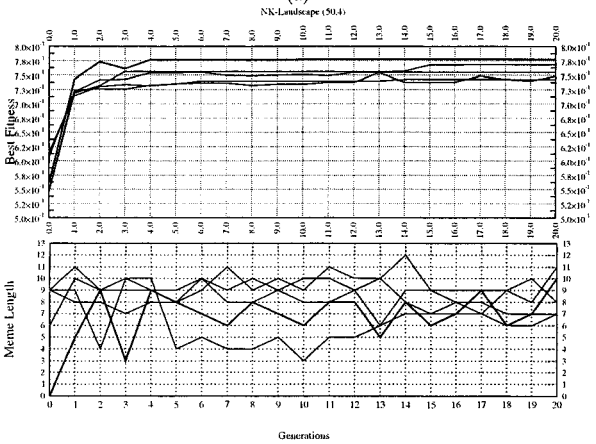
In Figures 5(a) and 5(b) we can observe the behaviour of the system. For the case $n = 50, k = 1$ the main activity occurs at the early generations (before generation 4). After that point the system becomes trapped in a local (possible global) optimum. The length of the memes evolved oscillates between 1 and 2. As the allowed length are restricted to be in the range $[0, 3 * k]$, the expected length of memes is 1.5. It is evident that the problem is solved before any creative learning can take place. When the Self-Generating MA is confronted with problem $n = 50, k = 4$ (a value of k just before the phase transitions mentioned in previous sections) the length of the meme in the best run oscillates between a minimum value of 3 (after generation 1) and a maximum of 10 for the run marked with a thick line (the best run). In this case the expected length (if a purely random rule was chosen) for a meme is 6 which is the most frequently visited value. For these simple NK-Landscape regimes, it does not seem to be of benefit to learn any specific meme, but rather, a random rule seems to suffice.

High epistasis, poly-time solvable:

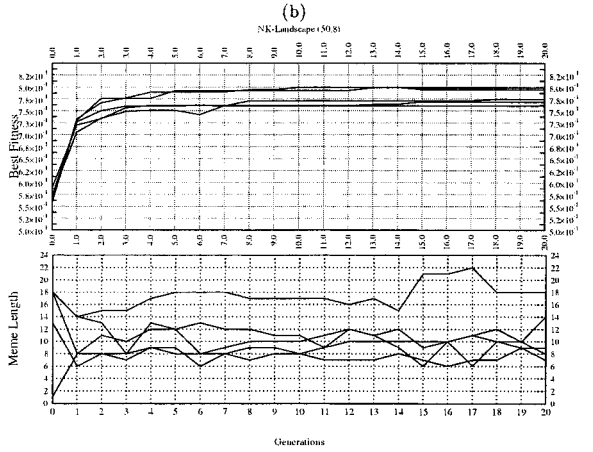
In Figure 5(c) we can see the system's behaviour for a value of k after the phase transition mentioned in [31] and [23]. In this case there is effective evolutionary



(a)



(b)



(c)

Fig. 5. *NK(50,1) in (a), NK(50,4) in (b) and NK(50,8) in (c). Adjacent neighbours.*

activity during the whole period depicted. Also, we can see clearly that the length of the meme employed by the most successful individual converges towards the value of k(in this case 8). If a purely random rule was used the expected length would have been 12. The case shown in Figure 6(a) is even clearer. All but one of the runs converge towards a meme length almost identical to $k = 10$, except for one that is very close to the expected length of 15.

The same trends can be seen in Figures 6(b) and 6(c) where meme lengths converge to values around to $k = 12$ and $k = 14$ respectively. It is interesting to note that although the values are very close to our predictions they do not remain at a fixed value but rather oscillates. This is a very intriguing behaviour as it resembles the variable-neighborhood nature of Lin-Kernighan, the most successful local search strategy for NK-Landscapes and other combinatorial problems. It will be interesting to investigate on the range of values that the Memetic Algorithms presented in [32] (which uses $K-opt$ and Lin-Kernighan) effectively employs; we speculate that the range of changes, i.e. the number of bits modified in each iteration of LS, will be close to the epistatic parameter of the problem instance.

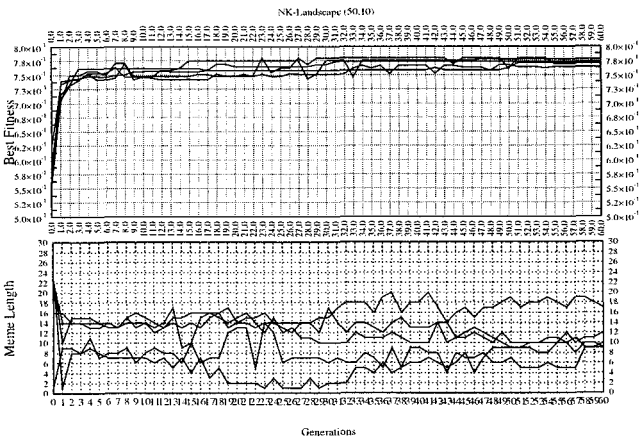
Low epistasis, NP-hard:

In Figure 7(a) we start to investigate the behaviour of the Self-Generating MA on the NP-Hard regime (i.e. the random neighborhood model). Figure 7(a) is similar to the adjacent neighborhoods version shown in Figure 5(a) except that oscillations are more frequent in the former. Comparisons between 7(b) and 5(b) reveal very similar trends.

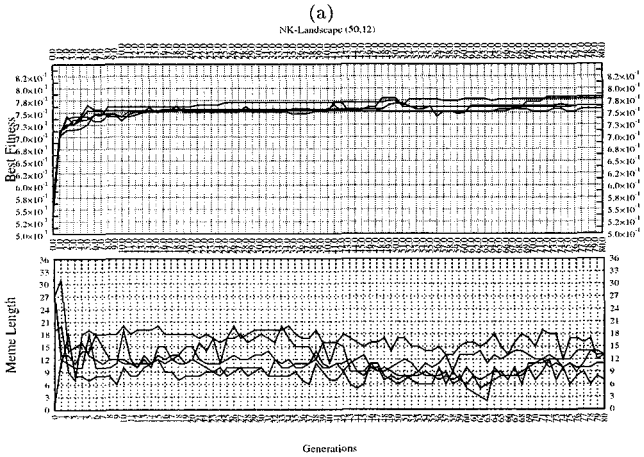
High epistasis, NP-hard:

The experiments with $(n = 50, k = 8)$ under the random neighbours model reveal marked differences with the consecutive neighbour model (see Figures 7(c) and 5(c) respectively). While in the later all the runs converged toward a meme length very close to k, the random model shows a richer dynamics. Meme length were divided into 3 groups. In one group, the emerged meme length were very close to the value of k, 8 in this case. The other two groups either continually increase the size of the memes or decreased it. Two of the most successful runs are identified with a cross or circle and each belong to a different group. Interestingly, the run that converges first to the local optimum is the one that uses very short memes. In contrast, the run that uses memes with length equivalent to a value of k show a continued improvement. It is important to note that none of the evolved memes converged towards the expected length of 12. Figure 8(a) seems to reveal a similar 3-grouped pattern.

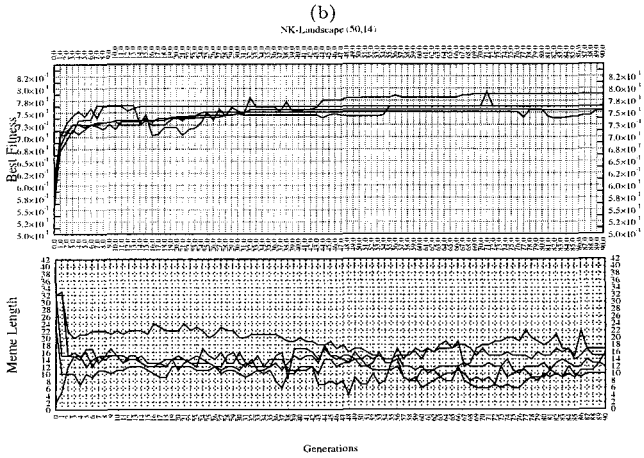
The runs that correspond to instances of $(n = 50, k = 12)$ differ notably from previous ones. The meme length seems to be converging towards a value



(a)

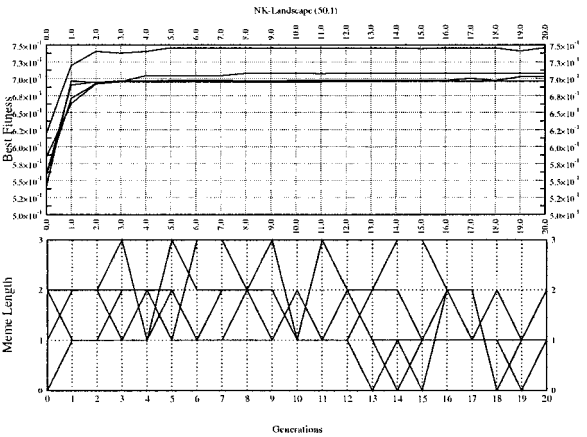


(b)

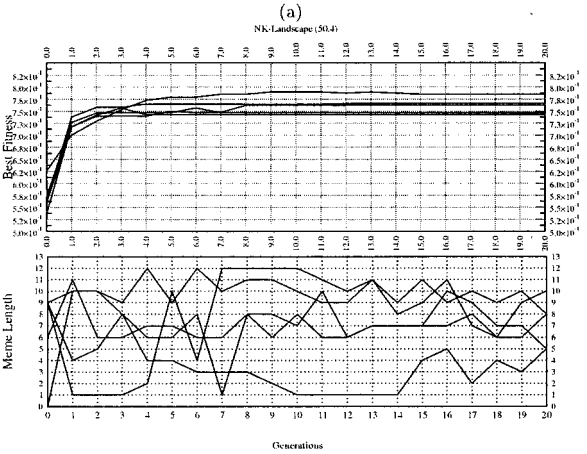


(c)

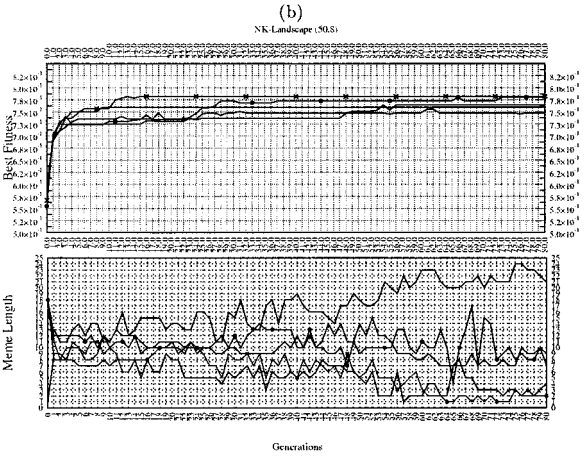
Fig. 6. *NK(50,10) in (a), NK(50,12) in (b) and NK(50,14) in (c). Adjacent neighbours.*



(a)

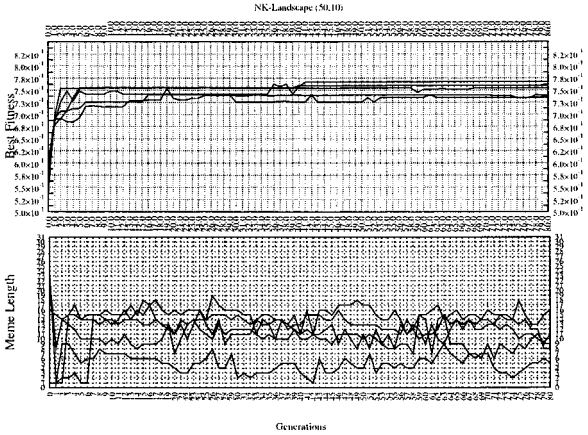


(b)

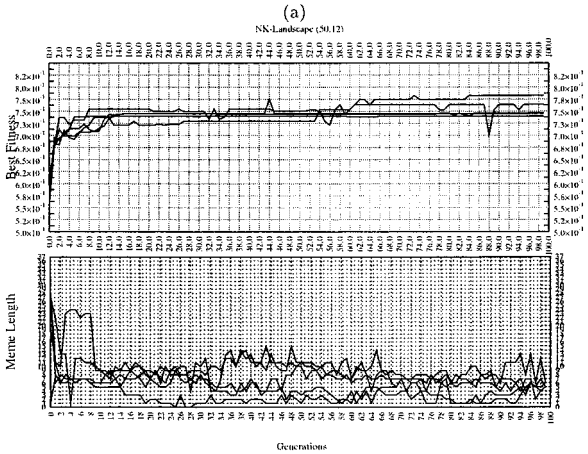


(c)

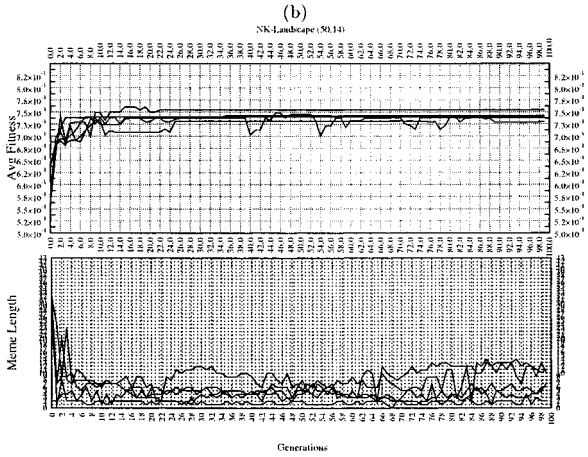
Fig. 7. *NK(50,1) in (a), NK(50,4) in (b) and NK(50,8) in (c). Random neighbours.*



(a)



(b)



(c)

Fig. 8. *NK(50,10) in (a), NK(50,12) in (b) and NK(50,14) in (c). Random neighbours.*

well below the expected length of 18 and even the epistatic value $k = 12$ for these problems. However, between generation 34 and 68 the meme lengths oscillates very close to $k = 12$ values. The next figure, 8(c), presents similar features as that of 8(b). However, now two clusters appear, one that suggest length around the value of k and another with length values of 6.

From the analysis of the previous figures we can see that our expectation that memes of length proportional to k will arise confirmed. However, other interesting features are evident. There are clear differences between memes that are evolved to solve the poly-time solvable cases and the NP-hard cases. In the first case, all the memes length for $k > 4$ converged toward values in the proximity of k. However, for the random neighborhood model and for high epistasis ($k > 4$) problems, the runs were clustered mainly around memes lengths close to k or close to around 6 (regardless the value of k). This is indeed a very interesting behaviour that deserves further studies as values of k in the range $[4, 5, 6]$ are *on the edge* of the phase transitions described in [31],[23] and [28]. That is, between 4,5 or 6 bits were the optimum number of bits that need to be considered to boost the efficiency of the search. Moreover, in the case of the NP-hard random neighbourhood with $k = 8$ three clusters are noted; we speculate that problems in this range are on the so called "edge of chaos" where emergent behaviours are more likely to occur[8],[20].

4 The Maximum Contact Map Overlap Experiments

We explore next the evolved local searcher as a supplier of building block in the context of a problem drawn from computational biology. A *contact map* is represented as an undirected graph that gives a concise representation of a protein's 3D fold. In this graph, each residue[6] is a node and there exists an edge between two nodes if they are neighbors. Two residues are deemed neighboors if their 3D location places them closer than certain threshold. Figures 9 & 10 show two contact maps. An *alignment* between two contact maps is an assignment of residues in the first contact map to residues on the second contact map. Residues that are thus aligned are considered equivalents. The value of an alignment between two contact maps is the number of contacts in the first map whose end-points are aligned with residues in the second map that, in turn, are in contact (i.e. the number of size 4 undirected cycles that are made between the two contact maps and the alignment edges). This number is called the *overlap* of the contact maps and the goal is to maximize this value. The complexity of Max CMO problem was studied in [18] and later in [23].

[6] A residue is a constitutent element of a protein.

4.1 Self-Generating Memetic Algorithms for MAX-CMO

The overall architecture of the Memetic Algorithm is similar to that described by the pseudocode in Figure 4. The backbone of the MA is a genetic algorithm in which chromosomes are represented by a vector $c \in [0, \ldots, m]^n$. Here m is the size of the longer protein and n the size of the shorter. A position j in c, $c[j]$, specifies that the j^{th} residue in the longer protein is aligned to the $c[j]^{th}$ residue in the shorter. A value of -1 in that position will signify that residue j is not aligned to any of the residues in the other protein (i.e., a structural alignment gap). Unfeasible configurations are not allowed, that is, if $i < j$ and $v[i] > v[j]$ or $i > j$ and $v[i] < v[j]$ (e.g., a crossing alignment)then the chromosome is discarded. It is simple to define genetic operators that preserve feasibilities based on this representation. Two-point crossover with boundary checks was used in [29] to mate individuals and create one offspring. Although both parents were feasible valid alignments the newly created offspring can result in invalid (crossed) alignments. After constructing the offspring, feasibility is restored by deleting any alignment that crosses other alignments. The mutation move employed in the experiments is called a sliding mutation. It selects a consecutive region of the chromosome vector and adds, slides right, or subtracts, slides left, a small number. The phenotypic effect produced is the tilting of the alignments. In [29] a few variations on the sliding mutation were described and used. Further implementation details can be found also in [23] and [5]. We describe next the make-up of memes.

The Local Search Procedure: Memes description for MAX-CMO

As mentioned in previous sections, we seek to produce a metaheuristic that creates from scratch the appropriate local searcher to use under different circumstances. A meme represents one particular way of doing local search. Memes can adapt through changes in their parameter set or through changes in the actions they perform. The local search involved can be very complex and composed of several phases and processes. In the most general case we want to be able to explore the space of all possible memes. One can achieve this by using a formal grammar that describes memes and by letting a genetic programming[21] based system to evolve sentences in the language generated by that grammar[23]. The sentences in the language generated by this grammar represent syntactically valid complex local searchers and they are the instructions used to implement specific search behaviors and strategies. To describe the particular representation employed to self-assemble the move operator used by a local search strategy we resort to a few examples.

In Figure 9 we can see two contact maps ready to be aligned by our algorithm. To simplify the exposition, both contact maps are identical (i.e. we are aligning a contact map with itself) and have a very specific pattern of contacts among their residues. In the present example a residue is connected to either its nearest neighbor residue, to a residue that is 4 residues away in

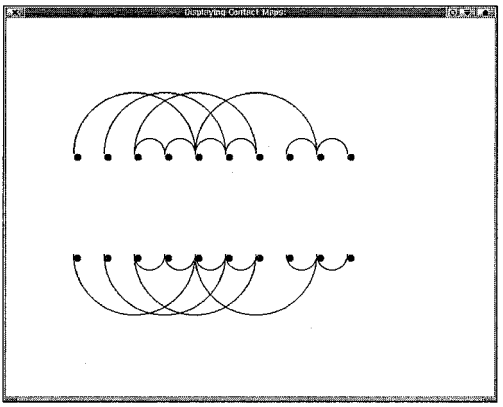


Fig. 9. *A contact map snapshot. The two randomly generated proteins have 10 residues.*

the protein sequence, or to both (with a given probability). In Figure 9 the contact map is 10 residues long, while in 10 it is 50 residues long (but with the same connectivity patterns). This contact pattern can be represented by the string $1-4$, meaning that the residue which occupies the ith position in the protein sequence is in contact in the native state with residues $(i+1)$th and $(i+4)$th. That is, the pattern $1-4$ is a succint representation of a possible building block which, if matched by the local searcher, could be propagated later on by crossover into other solutions.

An appropriate move operator for a local searcher acting in any of the contact maps on Figures 9 & 10 would be one that iterates through every residue in one of the contact maps, checking which residues on the lower contact map fulfills the pattern of connectivity and making a list of them. The same procedure would be applied to the top contact map producing a second list of residues. The local searcher then would pair residues of one list with residues of the second list thus producing a new and correct alignment which includes that building blocks.

The number of residues that verifies the pattern in each list puts an upper bound on how expensive the local search move operator can be. If the size of the first list is L_1 and the size of the second list is L_2, and without loss of generality we assume that $L_1 \leq L_2$ then there are at most $\sum_{i=1}^{i=L_1} \frac{L_2!}{(L_2-i)!}$. Clearly this number is too big to be searched exhaustively, this is why the previous grammar allows for the adaptation of the sample size. Moreover, although it is well known that real proteins present these contact patterns[10] it is impossible to know a priori *which* of these patterns will provide the best fitness improvement for a particular pair of protein structures. Hence, the Self-Generating MA needs to discover this itself.

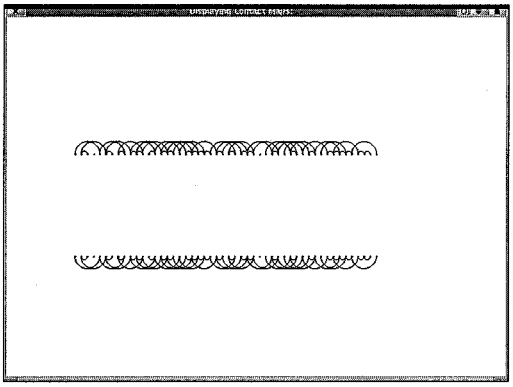


Fig. 10. *A contact map snapshot. The two randomly generated proteins have 50 residues and the patterns of contacts are similar to those in Fig. 9.*

If the graphs to be aligned were different (in the previous cases a graph was aligned with itself for the sake of clarity), then a move operator able to account for that variation in patterns must be evolved.

The defined move operator induces a neighborhood for every feasible alignment. If an alignment s is represented as explained above and L_1, L_2 are the list of vertices that matches the move operator, then every *feasible* solution that can be obtained by adding to s one or more alignments of vertices in L_1 with vertices on L_2 is a neighbor of s. The other components of a meme will then decide how to sample this neighborhood and which solutions to accept as the next one. As this paper is an account of the initial investigations we performed on the use of SGMA, we fixed several aspects of the memes that could otherwise be evolved. In this paper all memes employ first improvement ascent strategy and they are applied after crossover. The sample size was either 50 or 500 and the local search was iterated 2 times.

As described in the introduction, there were three memetic processes: imitation, innovation and mental simulation. Upon reproduction, a newly created offsprings inherits the meme of one of its parents accordingly to the simple inheritance mechanism described in [28]. In addition to this mechanism, and with a certain probability (called "imitation probability"), an agent could choose to override its parental meme by copying the meme of some successful agent in the population to which it was not (necessarily) genetically related. In order to select from which agent to imitate a search behavior, a tournament selection of size 4 was used among individuals in the population and the winner of the tournament was used as role model and its meme copied. Innovation was a random process of mutating a meme's specification by either extending, modifying or shortening the pattern in a meme (either before or after the $\rightarrow$). If during 10 consecutive generations no improvement was produced by either the local search or the evolutionary algorithm a stage of mental simulation was

started. During mental simulation, each individual (with certain probability) will intensively mutate its current meme, try it in the solution it currently holds, and if the mutant meme produces an improvement, both the newly created solution and the meme will be accepted as the next state for that agent. That is, mental simulation can be considered as a guided hill-climbing on memetic space. If ten mental simulation cycles finished without improvements, then metal simulation was terminated and the standard memetic cycle resumed.

4.2 Results

We designed a random instance generator with the purpose of parameterizing the complexity of the contact map overlap problems to be solved. The input to the random instance generator is a list of the form:

$r\ d\ n\ p_1\ pr_1\ p_2\ pr_2\ \ldots\ p_n\ pr_n$ where r is the number of residues in the randomly generated contact map, d is the density of random edges (i.e. noise) and n is the number of patterns in the contact map. For each of the n patterns two numbers are available, p_i and pr_i, where p_i specifies that a residue j is connected to residue $j+p_i$ with probability pr_i for all $i \in [1,n]$. That is, every pattern occurs with certain probability in each residue, thus an upper bound on the expected number of contacts is given by $r*d+r*\sum_{i=1}^{i=n} pr_i \leq r*(n+d)$. In our experiments $r \in \{10,50,100,150,200,250\}$, $d = 0.01$ and $n \in \{1,2,3,4\}$, that is, contact maps as short as 10 residues and as long as 250 residues were considered. For each contact map length, every possible number of patterns was used, this gives rise to 24 pairs of (r,n) values. For each pair, 5 random instances were generated spanning from low density contact maps to high density contact maps[7]. A total of 120 instances were generated. From all the possible parings of contact maps we randomly choose a total of 96 pairs to be aligned by means of 10 runs each.

We present next comparisons of the performance of a Genetic Algorithm versus that of the SGMA. In this experiment we would like to elucidate whether the overhead of learning suitable local searchers is amortized along the run and whether our proposed approach is ultimately useful. In order to run the experiments we implemented a GA as described previously. We were able to reproduce the results of [29] and [5] hence we considered our implementations as equivalent to the earlier ones. The difference between the GA and the SGMA are described below. In graphs 11,12,13 and 14 we compare the overlap values[8] against the *first hitting times*. First hitting time (FHT) is the time (in number of fitness evaluations) at which the best value of a run was encountered. Each graphs presents the results for 1,2,3 and 4 patterns respectively and for a range of contact maps sizes. The particular parameters used in the

[7] The program to generate random contact maps was written in java 1.1.8 as is available by request from the author.

[8] A higher overlap value means a better structural alignment.

GA are 0.15, 0.75 for mutation and crossover probabilities, and a (50, 75) replacement strategy. The Self-Generating MA uses 0.15,0.75,1.0,1.0,1.0,1.0 for the probabilities of mutation, crossover,local search, imitation, mental simulation and innovation respectively. The algorithms uses the same replacement strategy and for both local search and mental simulation a cpu budget of 50 samples is allocated.

4.3 Discussion

The graphs in 11,12,13 and 14 are good representatives of the results obtained with the two types of algorithms. That is, under a variety of changes to the parameter values mentioned above the results remain equivalent to those shown here.

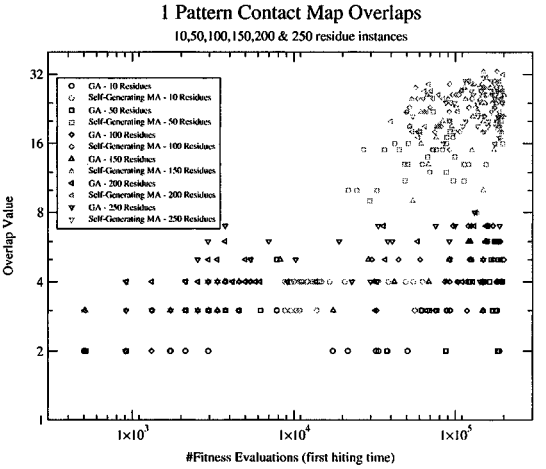


Fig. 11. *Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number. Contact maps present one pattern.*

From Figures 11,12,13 and 14 we can see that the Self-Generating Memetic Algorithm produces a much better amortized overlap value than the simple GA. That is, if enough time is given to the SGMA, it will sooner or later discover an appropriate local searcher move that will supply new building blocks. In turn, this will deliver an order of magnitude better overlaps than the Genetic Algorithm. Also, it seems that the GA is oblivious to the size (i.e. residues number) of the contact maps as it seems to produce mediocre local optima solutions even when given the maximum cpu time allocation (in these experiments $2 * 10^5$ fitness evaluations) for the whole range of 10 to 250 residues. The GA converges very quickly into local optima. This is seen in the graphs by bands parallel to the $x-axis$ over the range of energy evaluations for low overlap values. However, as the SGMA continuously improves its solutions,

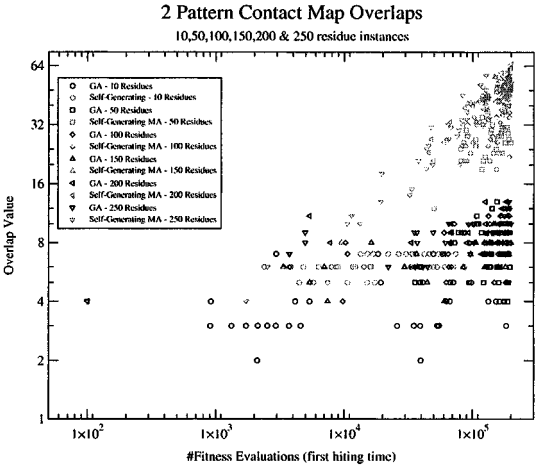


Fig. 12. *Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number.Contact maps present two patterns.*

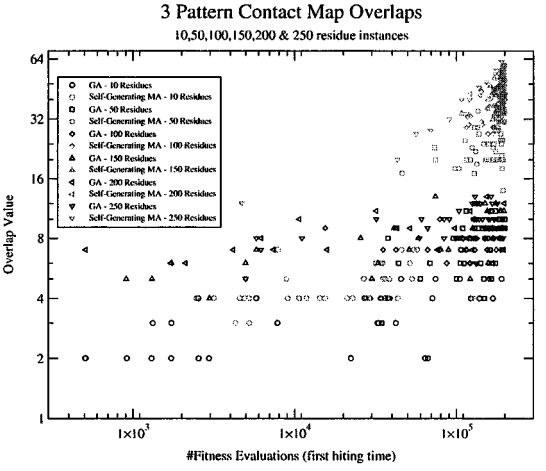


Fig. 13. *Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number.Contact maps present three patterns.*

it is not until very late in the execution (i.e. to the right of the $x - axis$) that the best solutions are found.

In contrast to the GA, the SGMA (as expected) is sensitive to the number of residues in the contact maps involved, that is, longer contact maps require larger cpu time to come up with the best value of the run (which is seen in the graph in the clustering patterns for the different residues number). Another

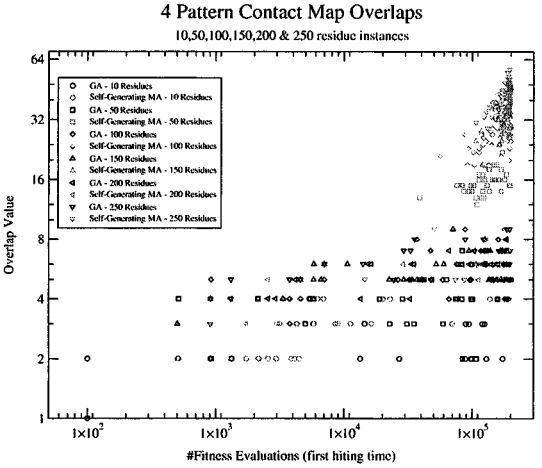


Fig. 14. *Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number.Contact maps present four patterns.*

important aspect to note is that both the $x - axis$ and the $y - axis$ are represented in logarithmic scales. Taking this into consideration it is evident that the quality of the overlaps produced by the SGMA are much better than those produce by the GA. As it is evident from the graphs, for sufficiently small instances (e.g all the 10 residues long and some of the 50 residues long) it is not worth using the SGMA as it requires more cpu effort to produce same quality of overlaps as the GA.

On the other hand, as the number of residues increases beyond 50, then instances are sufficiently complex to allow for the emergence of suitable local searchers in time to overtake and improve on the GA results. Also, as the number of patterns that are present in the instances increases both algorithms, as expected, require larger amounts of CPU to come up with the best solution of a run. However, it is still seen that the GA is insensitive to the number of residues, while the SGMA is clustered in the upper right corner (of Figure 14). This indicates that during all its execution the algorithm is making progress toward better and better solutions, the best of which is to be found near the end of the run. Moreover, this behavior indicates that the SGMA is not prematurely trapped in poor local optima as is the GA.

The ability of the SGMA to overcome local optima comes from the fact that the evolved local searchers will introduce good building-blocks that match the particular instance. This supply of building-blocks is essential for a synergistic operation of both the local searcher and the genetic operators. That is, using Goldberg's notation [17], we have that for the SGMA the *take over time* t^* is greater than the *innovation time* t_i, which allows the algorithms to continuously improve. In Figure 15 10 runs of the GA are compared agaisnt

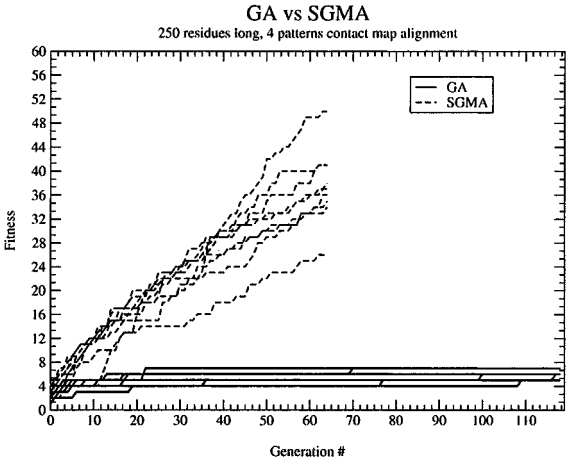


Fig. 15. *Representative example of GA and SGMA runs for a 250 residues and 4 patterns instance.*

10 runs of the SGMA. It can be seen that the GA runs get trapped very early (around the 20th generation) in poor local optima while the SGMA keeps improving durin all the run. All the runs in Figure 15 use the same total number of fitness evaluations.

5 Conclusions

In this chapter we discussed concepts arising from Memetic theory that could be use to produce a new breed of optimisation algorith,s We tied some of these memetic ideas with the concept of "Self-Generating Metaheuristics" and we exemplified the use of the resulting algorithms in two hard combinatorial problems.

The Memetic algorithms described in this paper do not resort to human-designed local searchers but rather they assemble *on-the-fly* the local search strategies that best suits each particular situation.

In this paper we argued that from an optimization point of view there are obvious advantages in self-assembling the local search behaviours for memetic algorithms. MAs that can self-generate the local searchers will be able to adapt to each problem, to every instance within a class of problem and to every stage of the search. A similar strategy could be use in other metaheuristics (e.g. Simulated Annealing, Tabu Search, Ant Colonies, GRASP, etc) where more sophisticated GP implementations might be needed to co-evolve the used operators.

One of the reasons for the success of the SGMA is that the evolved local searchers act as a (low and medium order) building block supplier. These

continuous supply of building blocks aids the evolutionary process to improve solutions continuously by producing a more synergistic operation of the local and gloabl operators.

It is our hope that researchers confronted with new problems for which there are not "silver bullet" local search heuristics (like is the case for TSP and Graph Partitioning where $K-$opt and Lin-Kernighan are known to be extremely efficient) with which to hybridize a Memetic Algorithm will try the obvious: the Dawkins method of self-assembling of local search behaviors. That is, use memes to evolutionary self-assemble appropriate local search strategies.

References

1. R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.
2. S. Blackmore. *The Meme Machine.* Oxford University Press, 1999.
3. E.K. Burke, J.P. Newall, and R.F. Weare. A memetic algorithm for university exam timetabling. In E.K. Burke and P. Ross, editors, *The Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 241–250. Springer Verlag, 1996.
4. E.K. Burke and A.J. Smith. A memetic algorithm for the maintenance scheduling problem. In *Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference, Dunedin, New Zealand*, pages 469–472. Springer, 24-28 November 1997.
5. R.D. Carr, W.E. Hart, N. Krasnogor, E.K. Burke, J.D. Hirst, and J.E. Smith. Alignment of protein structures with a memetic evolutionary algorithm. In W.B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference*, 2002.
6. L.L. Cavalli-Sforza and M.W. Feldman. *Cultural Transmission and Evolution: A Quatitative Approach.* Princeton University Press, Princeton, NJ., 1981.
7. F.T. Cloak. Is a cultural ethology possible. *Human Ecology*, 3:161–182, 1975.
8. P. Coveney and R. Highfield. *Frontiers of Complexity, the search for order in a chaotic world.* faber and faber (ff), 1995.
9. P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In E. Burke and W. Erben editors, editors, *Theory of Automated Timetabling PATAT 2000, Springer Lecture Notes in Computer Science,*, pages 176–190. Springer, 2001.
10. T. E. Creighton, editor. *Protein Folding.* W. H. Freeman and Company, 1993.
11. R. Dawkins. *The Selfish Gene.* Oxford University Press, New York, 1976.
12. R. Dawkins. The extended phenotype. 1982.
13. W.H. Durham. *Coevolution: Genes, Culture and Human Diversity.* Stanford University Press, 1991.
14. P.M. França, A.S. Mendes, and P. Moscato. Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times. In *Proceedings of the 5th International Conference of the Decision Sciences Institute, Athens, Greece*, July 1999.

15. L.M. Gabora. Meme and variations: A computational model of cultural evolution. In L.Nadel and D.L. Stein, editors, *1993 Lectures in Complex Systems*, pages 471–494. Addison Wesley, 1993.
16. A. Godzik, J.Skolnick, and A.Kolinski. A topology fingerprint approach to inverse protein folding problem. *Journal of Molecular Biology*, 227:227–238, 1992.
17. D.E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms.* Kluwer Academic Publishers, 2002.
18. D. Goldman, S. Istrail, and C. Papadimitriou. Algorithmic aspects of protein structure similarity. *Proceedings of the 40th Annual Symposium on Foundations of Computer Sciences*, pages 512–522, 1999.
19. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Artificial Life II: Proc. of the 2nd Conf. on Artificial Life*, 1992.
20. S.A. Kauffman. *The Origins of Order, Self Organization and Selection in Evolution.* Oxford University Press, 1993.
21. J.R. Koza, F.H. Bennet, D. Andre, and M.A. Keane. *Genetic Programming III, Darwinian Invention and Problem Solving.* Morgan Kaufmann Publishers, 1999.
22. N. Krasnogor. Co-evolution of genes and memes in memetic algorithms. In A.S. Wu, editor, *Proceedings of the 1999 Genetic And Evolutionary Computation Conference Workshop Program*, 1999.
23. N. Krasnogor. http://www.cs.nott.ac.uk/ ~nxk/papers.html. In *Studies on the Theory and Design Space of Memetic Algorithms.* Ph.D. Thesis, University of the West of England, Bristol, United Kingdom., 2002.
24. N. Krasnogor. Self-generating metaheuristics in bioinformatics: The protein structure comparison case. *To appear in the Journal of Genetic Programming and Evolvable Machines. Kluwer academic Publishers*, 5(2), 2004.
25. N. Krasnogor and S. Gustafson. Toward truly "memetic" memetic algorithms: discussion and proof of concepts. In D.Corne, G.Fogel, W.Hart, J.Knowles, N.Krasnogor, R.Roy, J.E.Smith, and A.Tiwari, editors, *Advances in Nature-Inspired Computation: The PPSN VII Workshops.* PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9, 2002.
26. N. Krasnogor and S. Gustafson. The local searcher as a supplier of building blocks in self-generating memetic algorithms. In J.E. Smith W.E. Hart and N. Krasnogor, editors, *Fourth International Workshop on Memetic Algorithms (WOMA4).* In GECCO 2003 workshop proceedings, 2003.
27. N. Krasnogor and J.E. Smith. A memetic algorithm with self-adaptive local search: Tsp as a case study. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and Hans-Georg Beyer, editors, *GECCO 2000: Proceedings of the 2000 Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 2000.
28. N. Krasnogor and J.E. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In L. Spector, E.D. Goodman, A. Wu, W.B. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshj, M.H. Garzon, and E. Burke, editors, *GECCO 2001: Proceedings of the 2001 Genetic and Evolutionary Computation Conference.* Morgan Kaufmann, 2001.
29. G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap prob-

lem. *Proceedings of The Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001*, 2001.
30. M.W.S. Land. Evolutionary algorithms with local search for combinatorial optimization. *Ph.D. Thesis, University of California, San Diego*, 1998.
31. W.G. Macready, A.G. Siapas, and S.A. Kauffman. Criticality and parallelism in combinatorial optimization. *Science*, 261:56–58, 1996.
32. P. Merz. *Memetic Algorithms for Combinatorial Optimization Problems:Fitness Landscapes and Effecitve Search Strategies.* Ph.D. Thesis, Parallel Systems Research Group. Department of Electrical Engineering and Computer Science. University of Siegen., 2000.
33. P. Moscato. Memetic algorithms: A short introduction. In D. Corne, F. Glover, and M. Dorigo, editors, *New Ideas in Optimization.* McGraw-Hill, 1999.
34. P.A. Moscato. Problemas de otimizacão np, aproximabilidade e computacão evolutiva:da prática à teoria. *Ph.D Thesis, Universidade Estadual de Campinas,Brasil*, 2001.
35. M.J. Oates, D.W. Corne, and R.J. Loader. Tri-phase profile of evolutionary search on uni- and multi-modal search spaces. *Proceedings of the Congress on Evolutionary Computation (CEC2000)*, 1:357–364, 2000.
36. B. Olsson. A host-parasite genetic algorithm for asymmetric tasks. In C. Nédellec and C. Rouveirol, editors, *Machine Learning: ECML-98 (Proceedings of the 9th European Conference on Machine Learning)*, pages 346–351. Springer-Verlag, 1998.
37. B. Olsson. *Algorithms for Coevolution of Solutions and Fitness Cases in Asymmetric Problem Domains.* PhD thesis, University of Exeter, 1999.
38. B. Olsson. Handling asymmetric problems with host-parasite algorithms, 1999.
39. J. Paredis. Chapter c7.4: Coevolutionary algorithms. In T. Back, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, page C7.4. IOP publishing Ltd and Oxford Unviersity Press, 1997.
40. M.A. Potter and K.A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving From Nature*, 1994.
41. P.L. Privalov. Physical basis of the stability of the folded conformations of proteins. In T.E. Creighton, editor, *Protein Folding.* W.H. Freedman and Company, 1999.
42. P.J. Richerson and R. Boyd. A dual inheritance model of the human evolutionary process: I. basic postulates and a simple model. *Journal of Social and Biological Structures*, I:127–154, 1978.
43. J. Schaffer and A. Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J.J. Grefenstette, editor, *Proceedigns of Second International Conference on Genetic Algorithms.* Lawrence Erlbaum, 1987.
44. O. Sharpe. Introducing performance landscapes and a generic framework for evolutionary search algorithms. *Proceedings of the Congress on Evolutionary Computation (CEC2000)*, 1:341–348, 2000.
45. J.E. Smith. Co-evolution of memetic algorithms : Initial results. In Beyer Fgenandez-Villacans Merelo, Adamitis and Schwefel (eds), editors, *Parallel problem solving from Nature - PPSN VII, LNCS 2439.* Springer Verlag, 2002.
46. J.E. Smith and T.C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, pages 81–87, 1997.
47. Jim Smith and T.C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the Third IEEE International Conference on Evolutionary Computing*, pages 318–323. IEEE Press, 1996.

48. R.E. Smith and E. Smuda. Adaptively resizing populations: Algorithms, analysis and first results. *Complex Systems*, 1(9):47–72, 1995.
49. E. G. Talbi, Z. Hafidi, and J-M. Geib. A parallel adaptive tabu search approach. *Parallel Computing*, 24(14):2003–2019, 1998.
50. E.D. Weinberger and A. Fassberg. Np completness of kauffman's n-k model, a tuneably rugged fitness landscape. In *Santa Fe Institute Technical Reports*, 1996.