
Natural Computing **3:** 53–76, 2004.

A Study on the use of "self-generation" in memetic algorithms

NATALIO KRASNOGOR and STEVEN GUSTAFSON
Automated Scheduling, Optimization and Planning Group, School of Computer Science and IT, University of Nottingham, Nottingham NG8 1BB, U.K. (E-mail: {nxk,smg}@cs.nott.ac.uk)

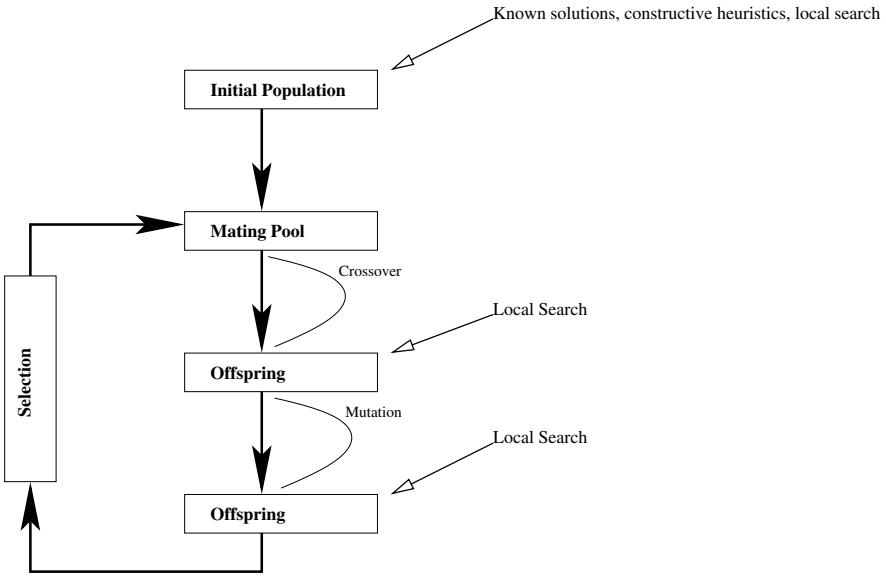
Abstract. A vast number of very successful applications of Global-Local Search Hybrids have been reported in the literature in the last years for a wide range of problem domains. The majority of these papers report the combination of highly specialized **pre-existing** local searchers and usually purpose-specific global operators (e.g. genetic operators in an Evolutionary Algorithm). In this paper we concentrate on one particular class of Global-Local Search Hybrids, Memetic Algorithms (MAs), and we describe the implementation of "self-generating" mechanisms to produce the local searches the MA uses. This implementation is tested in two problems, *NK-Landscape Problems* and the *Maximum Contact Map Overlap Problem (MAX-CMO)*.

Key words: contact map overlap, memetic algorithms, NK-Landscapes, self-assembling, self-generation

1. Introduction

Memetic algorithms are global-local search hybrids. In these algorithms the hybridisation is realized by integrating an evolutionary process (e.g. a genetic algorithm) with some kind of local search heuristic. The local search process is usually (but not always) represented by a well-tested heuristic. As an example of this consider the use of Lin-Kernighan and K-Opt local searchers for Traveling Salesman related problems (e.g. Merz and Freisleben, 1997) or other Monte-Carlo based approaches (e.g. Krasnogor and Smith, 2000) in conjunction with a genetic algorithm. In Figure 1, a schematic representation of a memetic algorithm is shown. The evolutionary process is meant to produce a robust global search on the space of potential solutions while the local search phase is generally aimed at exploiting, i.e. fine tuning, the search around specific regions of the search space.

A vast number of very successful applications of Memetic algorithms have been reported in the literature in the last years for a wide range of problem domains. These papers tackle problems as diverse as graph bipartition (Merz and Freisleben, 2000), protein structure prediction (Krasnogor et al., 2002),



(a)

Figure 1. Diagramatic representation of a Memetic algorithm.[1]

exam timetabling (Burke et al., 1996), shape matching (Ozcan and Mohan, 1998), etc.

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 like mutation and crossover. Moreover, those algorithms require a considerable effort devoted to the tuning of the local search and evolutionary parts of the algorithm.

In Krasnogor (2002) and Krasnogor and Gustafson (2002) we propose the so called "Self-Generating Metaheuristics". Self-Generating MAs are able to **create** 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 that individuals (also called agents) will use to alter the survival value of their chromosomes. As the memes (i.e. local search strategies) are propagated, mutated and are selected in a Darwinian sense,

[1] This figure is inspired by Eiben and Smith (2003), chapter 10.

the Self-Generating MAs we propose are closer to R. Dawkins concept of memes than previous works on memetic algorithms (e.g. França et al., 1999; Moscato, 1999, 2001; Burke and Smith, 1997).

In Krasnogor (2002), Krasnogor and Gustafson (2002), Smith (2002b, 2002a) 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.

L.M. Gabora (1993) mentions three phenomena that are unique to cultural (i.e. memetic) evolution, those are, *knowledge-based*, *imitation* and *mental simulation*. It is these three phenomena that our Self-Generating Memetic Algorithm implements and by virtue of which it can produce its own local searchers. The representation of the low level operators (in this paper 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 worth or not to be applied on a particular solution or on a particular region of a solution and, more importantly, the move operator itself in which the low level heuristic will be based (Krasnogor, 2002).

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 algorithms 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).

The goal of this paper is to demonstrate that local searchers can be evolved in the realm of NK-Landscape problems and a graph theory combinatorial problem. More importantly, to 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*. For an overview of selecto-recombinative evolutionary algorithms from a building-blocks perspective please refer to Goldberg (2002).

2. The NK-Landscapes case

An NK-Landscape problem instance can be defined by two integers n, k such as $0 < n, 0 \leq k \leq n-1$ and an $n \times 2^{k+1}$ matrix E with elements sampled randomly from the uniform distribution $U(0, n)$. E represents the epistatic interactions of k bits. A solution to the problem is a binary string S, such that $|S| = n$. To measure the quality or fitness of a solution S a function $fitness(S) = \frac{1}{n} * \sum_{i=1}^{i=n} f_i(S_i, S_{i_1}, \ldots, S_{i_k})$ is used where $f_i(\cdot)$ is an entry into E, S_i the value of string S at position i and S_{i_j} is the value of string S at the $j-th$ neighbor of bit i. The neighbors, not necessarily adjacent, j of bit i are part of the input.

NK-Landscapes are particularly useful to understand the dynamics of evolutionary search (Kauffman, 1993) (particularly MAs) as they can be tunned to represent low or high epistasis regimes (i.e. low or high k values respectively) with the extreme of an uncorrelated random fitness landscape for the case of $k = n-1$. Moreover, the optimization version of this problem can be solved in polynomial time by dynamic programming if the neighborhood structure used is that of *adjacent neighbors* or can be NP-Hard if the structure used is that of *random neighbors* (Weinberger and Fassberg, 1996).

NK-Landscapes have been the subject of intensive and varied studies. In Macready et al. (1996), Kaufmann et al. 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. (2000) showed performance profiles for evolutionary search based algorithms where phase changes were also present. Krasnogor and Smith (2001) and Krasnogor (2002) 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 (2000) 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 (2000) performs some analysis on the parameter space of evolutionary search strategies for NK landscapes. As the *NK-Landscapes* represent a rich problem domain 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 Krasnogor (2002) and in previous sections we argued briefly about the need to creatively adapt every aspect of the local searchers. In this part of the

```
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
    Variate individuals in M(t);
    Improve_by_local_search( M(t));
    Compute the fitness f(p) ∀p ∈ M(t) ;
    Run Memetic Processes of imitation, innovation and mental;
    simulation;
    Generate P(t + 1) selecting some individuals from P(t) and M(t);
    t = t + 1;
  Od
  Return best p ∈ P(t − 1);
End.
```

Figure 2. The memetic algorithm employed.

paper we will focus only on the self-generation of the move operator itself as a proof of concept. The other aspects are actively being investigated. Following the terminology of Krasnogor (2002) the MA is a $D = 4$ Memetic Algorithm which implies that local search occurs as an independent process of Mutation and Crossover.[2] The pseudocode in Figure 2 depicts the algorithm we use in this paper.

Individuals in the MA population will be composed of genetic and memetic material. The genetic material will basically represent a solution to either *NK-Landscapes* or *MAX-CMO* problems (i.e. a bit string) while the memetic part will represent "mental constructs" to optimize solutions for these two domains. As such we will be evolving individuals whose goal is to self-optimize by (first process) genetic evolution and (second process) memetic evolution.

2.1. *The Self-Generating Memetic Algorithm*

The initial population in P is created at random. As mentioned before, each individual is composed of genetic material in the form of a bit string (B). The bit string represent the solution to the NK instance being solved. The memetic material is of the form $* \rightarrow S$ where the $*$ symbol matches

[2] See the Memetic Algorithms taxonomy in Krasnogor (2002).

any bit in the solution string and S is another bit string. The meaning of memes will be explained later on. The only variation mechanism is bit-wise mutation (applied with probability 0.05) to the chromosomes. The replacement strategy was a $(20, 50)$. There is no genetic crossover but the SIM mechanism, as described in Krasnogor and Smith (2001), was used to transfer memes between individuals. As mentioned before memes represent particular local search rules in each of the two problem domains. In the case of *NK-Landscapes* a rule is encoded as $* \rightarrow S$. Memetic mutation occurs with an innovation rate (Krasnogor, 2001) of 0.2. A meme can be mutated (with equal probability) in three ways: either a random bit is inserted in a random position, or 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.

2.1.1. *The local search procedure*

During the local search stage a 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 that 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 zeros memes can reflect the adjacent neighbor or the random neighbor 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, in this case as S is built out of all 0 a fully-adjacent neighborhood is considered. Suppose $B = 101010101010111110$ and the bit to be optimized is the fourth bit. $B_4 = 0$ in the example and its four adjacent neighbors 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 $B' = 101111111010111110$ provided B' has better fitness than B. The process is repeated in every bit of B once for every sample in the sample set.

In our implementation, and because our experiments are meant only as a proof of concept, we did not use all the code optimization described in Merz (2000) nor we use an exhaustive *k-opt* or *Lin-kernighan* heuristic as Merz employed. Certainly his recommendations on how to improve the efficiency of the code (in particular those related to the fitness updates) will be needed if larger problems are going to be studied. The evolved memes induce a variable-sampled *k-opt* local searcher. 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.

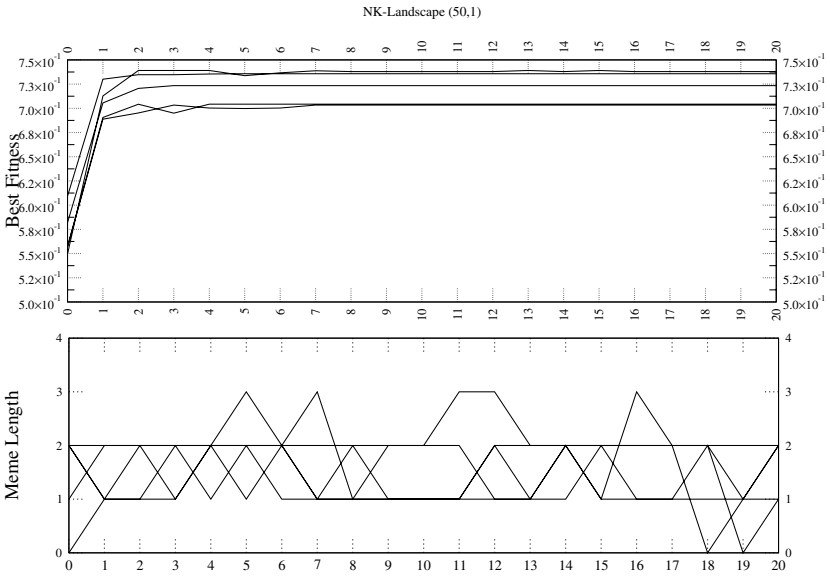
2.2. *Experimental setting*

In previous sections we described our self-generating MAs. What sort of behaviors 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 negligible 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 is: low epistasis, poly-time solvable landscapes with adjacent neighbors ((50, 1), (50, 4)), high epistasis, poly-time solvable landscapes with adjacent neighbors ((50, 8), (50, 10), (50, 12), (50, 14)), low epistasis, NP-hard landscapes with random neighbors ((50, 1), (50, 4)) and high epistasis, NP-hard landscapes with random neighbors ((50, 8), (50, 10), (50, 12), (50, 14)).

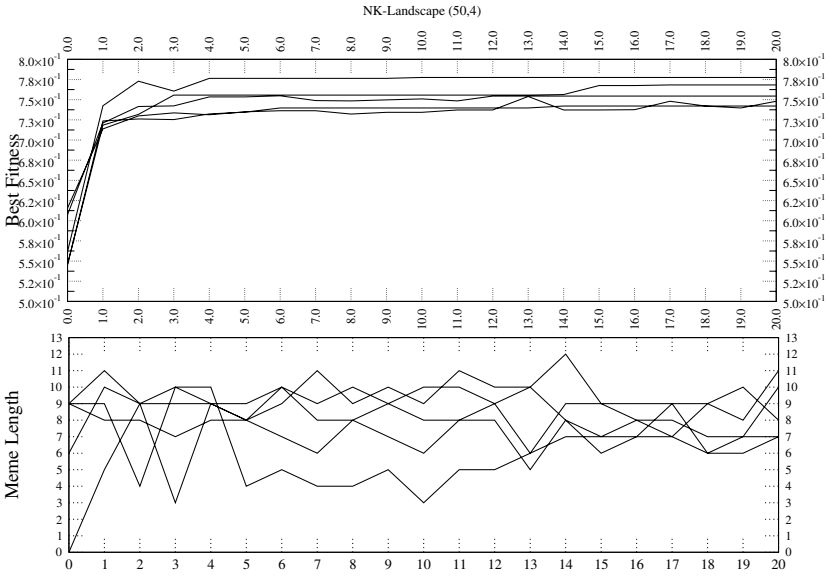
2.3. *Results*

In the following graphs 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.[3]

[3] Several sets of 10 runs each were carried out with slightly different parameters. The overal results are similar to the ones reported here.



(a)



(b)

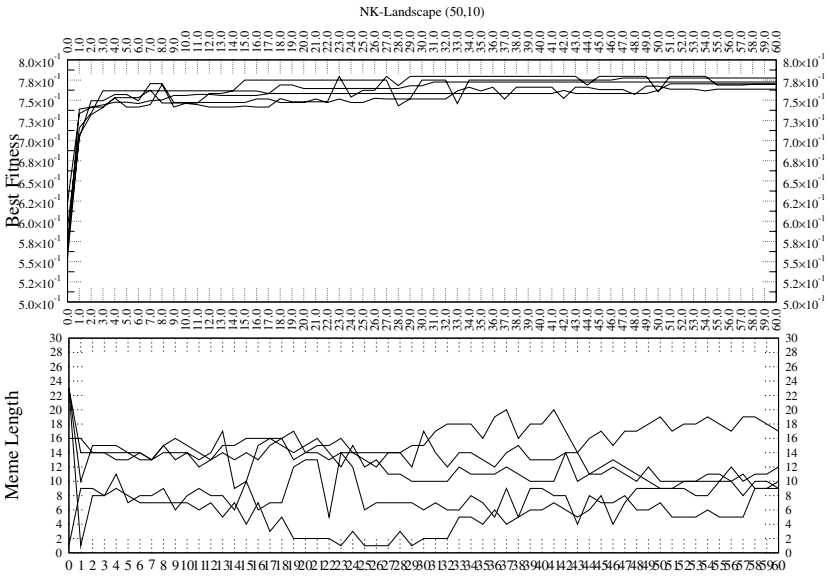
Figure 3. $NK(50, 1)$ in (a), $NK(50, 4)$ in (b). Adjacent neighbors.

2.3.1. *Low and high epistasis, poly-time solvable*
In Figures 3(a) and 3(b) we can observe the behavior 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, the simplest possible *NK-Landscape* regimes, it does not seem to be of benefit to learn any specific meme, but rather, a random rule seems to suffice.

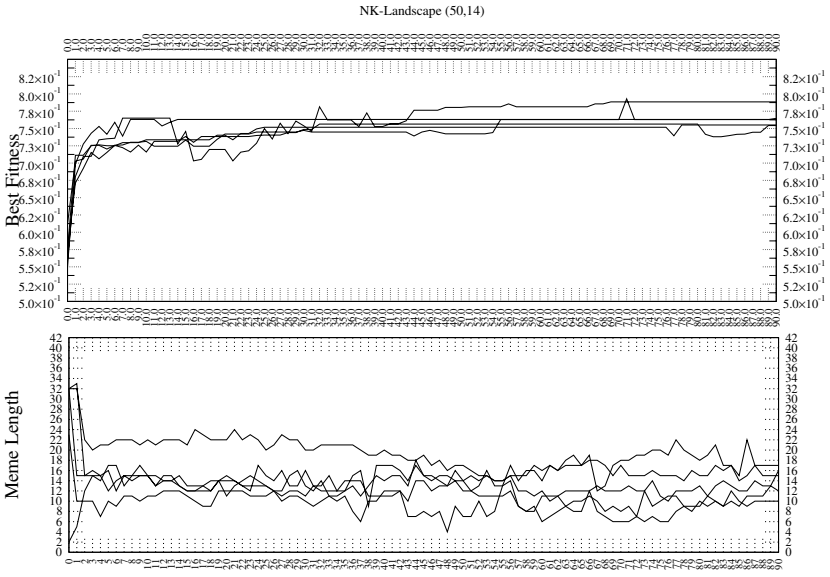
In Figure 4 we can see the system's behavior for a value of k after the phase transition mentioned in Macready et al. (1996) and Krasnogor (2002). In this case there is effective evolutionary activity during the whole period depicted and we can see clearly that the length of the meme employed by the most successful individual converges towards the value of k. In Figure 4(a) 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 Figure 4(b) where meme lengths converge to values around to $k = 14$ (similar results are obtained for the $NK(50, 12)$ landscapes). 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 Merz (2000) (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.

2.3.2. *Low and high epistasis, NP-hard*
The graphs associated with the low epistasis, NP-hard regime (not shown here for the sake of space) present features that are very similar to those of the adjacent neighborhoods landscapes in Figure of 3(a) and (b). We concentrate instead in the high epistasis, np-hard, regime. The experiments with ($n = 50, k = 8$) under the random neighbors model reveal marked differences with



(a)



(b)

Figure 4. $NK(50, 10)$ in (a) and $NK(50, 14)$ in (b). Adjacent neighbors.

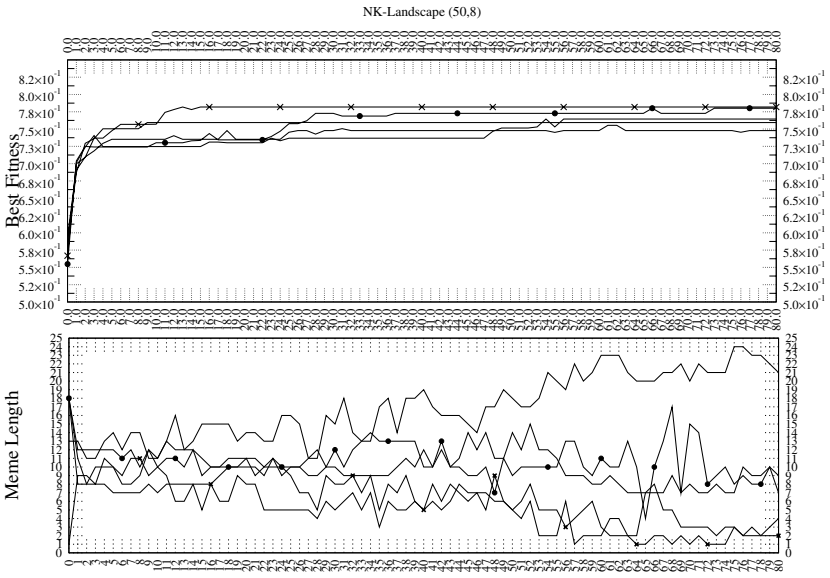
the consecutive neighbor model. While in the later all the runs converged toward a meme length very close to k, the random model shows a richer dynamics. Meme length were divided into 3 groups (see Figure 5(a)). In one group, the emerged meme length were very close to the value of k, 8 in this case. 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 enough, the run that converges first to the local optimum is the one that uses very short memes, in contrast, the one that uses memes length equivalent to a value of k shows a continued improvement. It is important to note that none of the evolved memes converged towards the expected length of 12.

The runs that correspond to instances of $(n = 50, k = 12)$ differ notably from previous ones (Figure 5(b)). The meme length seems to be converging towards a value 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, 6, presents similar features as that of 5(b) but two clusters appear, one that suggest length around the value of k and another with length values of 6. Similar results are obtained in the $NK(50, 10)$ landscapes. From the analysis of the previous graphs we can see that our expectations, namely, that memes of length proportional to k will arise were confirmed.

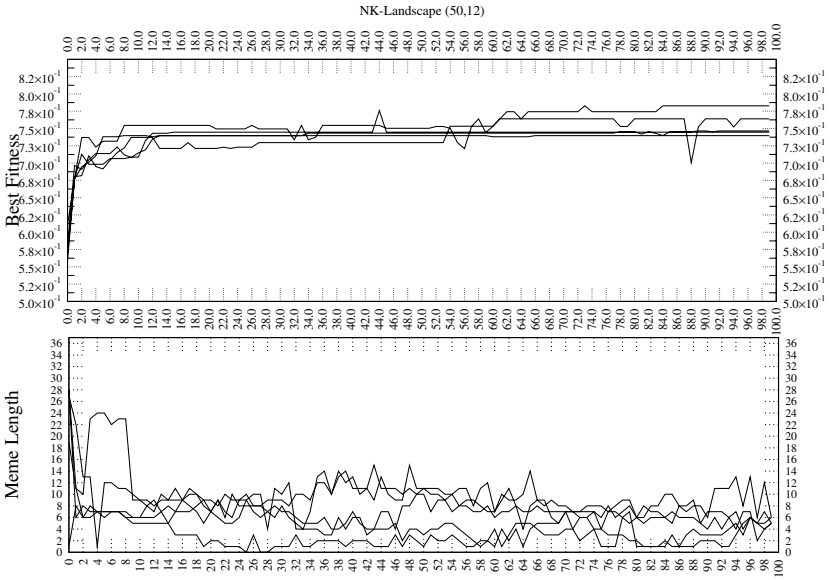
However, other interesting features are evident. There are clear differences between memes that are evolved to solve the poly-time solvable cases and the NP-hard cases. In the first case all memes length and for $k > 4$ converged toward values in the proximity of k. However, for the random neighborhood model and for high epistasis ($k > 4$) problems, the runs were clustered mainly around either meme lengths of values close to k or to lengths around 6 (regardless the value of k). This is indeed a very interesting behavior that deserves further studies as values of k in the range [4, 5, 6] are *on the edge* of the phase transitions described in Macready et al. (1996), Krasnogor (2002) and Krasnogor and Smith (2001), 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 neighborhood with $k = 8$ three clusters are noted; we speculate that problems in this range are on the so called "edge of chaos" where emergent behaviors are more likely to occur (Coveney and Highfield, 1995; Kauffman, 1993).

3. The maximum contact map overlap case

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*



(a)



(b)

Figure 5. $NK(50, 8)$ in (a) and $NK(50, 12)$ in (b). Random neighbors.

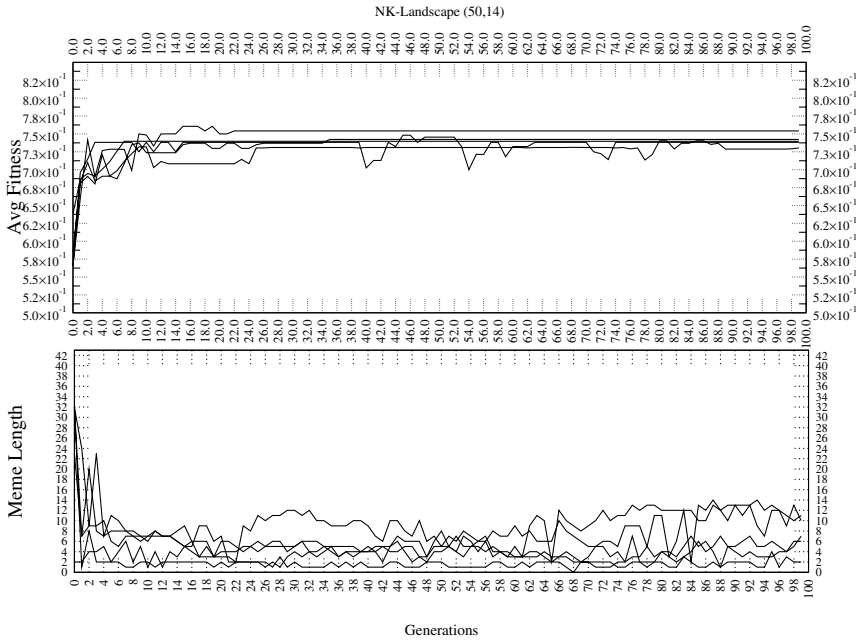


Figure 6. NK (50, 14). Random neighbors.

is represented as an undirected graph that gives a concise representation of a protein's 3D fold. In this graph, each residue[4] is a node and there exists an edge between two nodes if they are neighbors. Two residues are deemed neighbors if their 3D location places them closer than certain threshold. 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 the Max CMO problem was studied in Goldman et al. (1999) and later in Krasnogor (2002) and shown to be NP-hard.

[4] A residue is a constituting element of a protein.

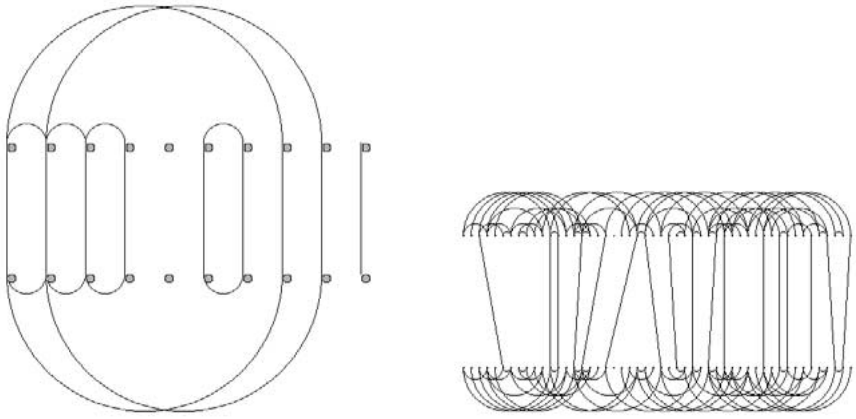
3.1. *Self-Generating Memetic Algorithms for MAX-CMO*

In a genetic algorithm for *Max CMO* (Lancia et al., 2001), a chromosome is represented by a vector $c \in [0, \ldots, m]^n$ where 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 jth 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 $c[i] > c[j]$ or $i > j$ and $c[i] < c[j]$ (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 Lancia et al. (2001) to mate individuals and create one offspring. Although both parents are 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 Lancia et al. (2001) a few variations on the sliding mutation were described and used. In our previous work (Carr et al., 2002) we employed a multimeme algorithm that, besides using the same mutation and crossover as the mentioned GA, had a set of 6 Human-designed local search operators. Four of the local searchers implemented were parameterized variations of the sliding operator. The direction of movement, left or right sliding, and the tilting factor, i.e., the number added or subtracted, were chosen at random in each local search stage. The size of the window was taken from the set $\{2, 4, 8, 16\}$. Two new operators were defined: a "wiper" move and a "split" move. Details of the operators described here can be found in Lancia et al. (2001), Krasnogor (2002), and Carr et al. (2002).

3.1.1. *Description of memes*

As mentioned in previous sections, we seek to produce a metaheuristic that creates from scratch the appropriate local searcher to use under different circumstances. 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 (i.e. local searchers). One can achieve this by using a formal grammar that describes memeplexes and by letting a genetic programming (Koza et al., 1999) based system to evolve sentences in the language generated by that grammar (Krasnogor, 2003). The sentences in the language generated by this grammar represent syntactically valid complex



(a) (b)

Figure 7. Two contact maps snapshots. In (a) the two randomly generated proteins have 10 residues, while in (b) the patterns of contacts are maintained but the protein is 50 residues long.

local searchers and they are the instructions used to implement specific search behaviors and strategies. For space limitations we do not describe here the grammar used to represent valid memes. For details on how to achieve that the reader is referred to Krasnogor (2002). As we did for *NK-Landscapes*, we evolved only the move operator itself.

In Figure 7 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 (with a given probability) a residue is connected to either its nearest neighbor residue, to a residue that is 7 residues away in the protein sequence, or to both. In 7(a) the contact map is 10 residues long, while in (b) it is 50 residues long.

This contact pattern can be represented by the string 1–7, 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 $(i+7)$th. That is, the pattern 1–7 is a succinct 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 7(a) and (b) 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. In general, two contact maps will present different connectivity patterns, hence the meme representation needs to allow the specification of different patterns. This is accomplished by an encoding of the form $P_{cm_1} \rightarrow P_{cm_2}$, where P_{cm_1}, P_{cm_2} represent the (potential) patterns in the first and second contact map respectively that are going to be search for and subsequently aligned. 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 that of the second 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 (Creighton, 1993) 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. 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 move operator thus defined 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. That is, 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, namely, imitation, innovation and mental simulation.[5] Upon reproduction, a newly created offsprings inherited the meme of one of its parents accordingly to the simple inheritance mechanism described in Krasnogor and Smith (2001). 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

[5] Mental simulation was not used for the NK-Landscape problems.

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.

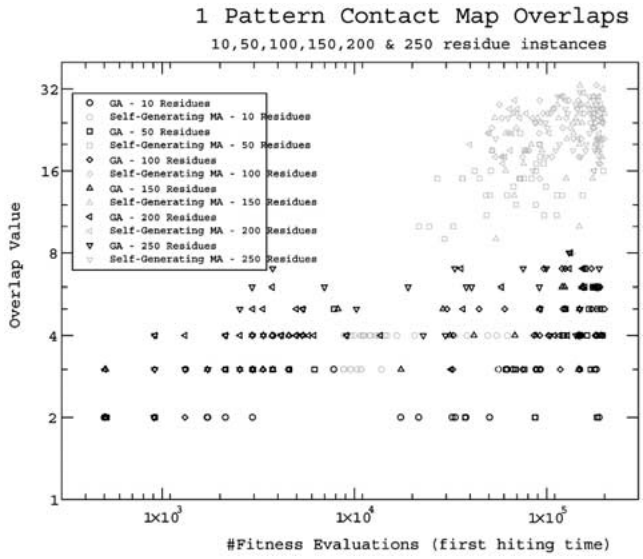
3.2. *Experimental setting*

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:

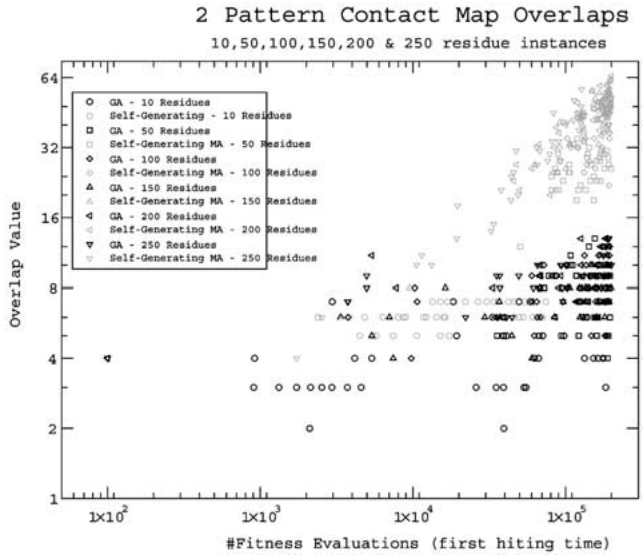
$r\ dn\ p_1\ pr_1\ p_2\ pr_2\ \ldots\ p_n\ pr_n$ where r is the number of residues in the randomly generated contact map, d is the density of random edges (i.e. noise) and n is the number of patterns in the contact map. For each of the n patterns two numbers are available, p_i and pr_i, where p_i specifies 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. 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.

3.3. *Results*

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 Lancia et al. (2001) and Carr et al. (2002) hence we considered our implementations as equivalent to the earlier ones. The difference between the GA and the SGMA are described below. In

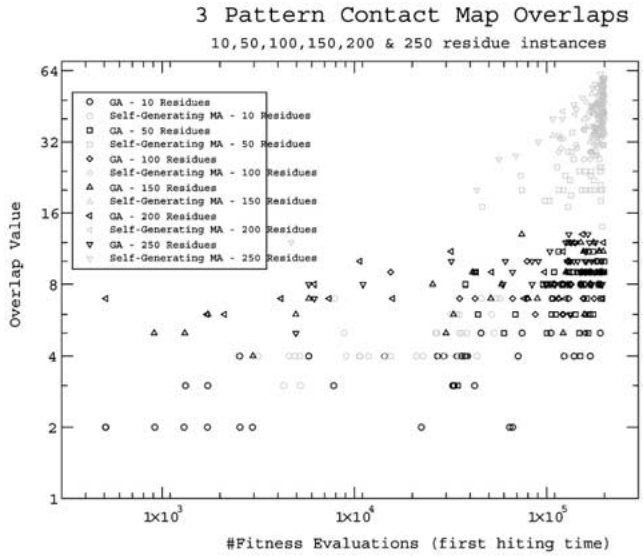


(a)

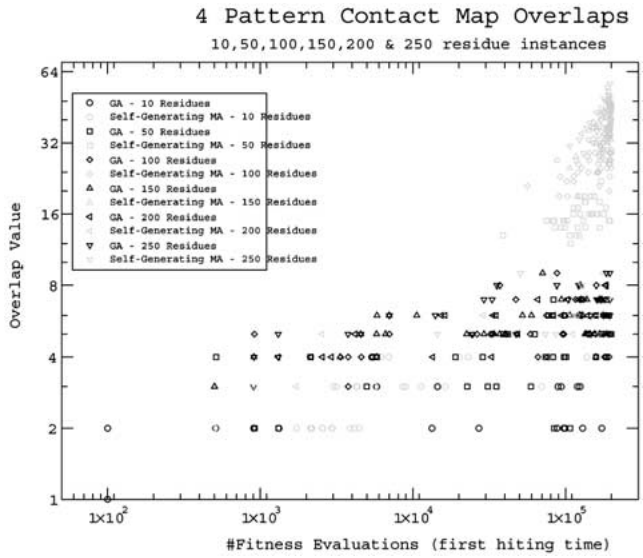


(b)

Figure 8. 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 (within each grap) and number of connectivity patterns (across graphs).



(c)



(d)

Figure 8. Continued.

graphs 8(a, b, c & d) we compare the overlap values[6] 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 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.

The graphs in 8(a, b, c & d) 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.

From Figures 8(a, b, c & d) 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 soon 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. On the contrary, as the SGMA continuously improves its solutions, 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 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,

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

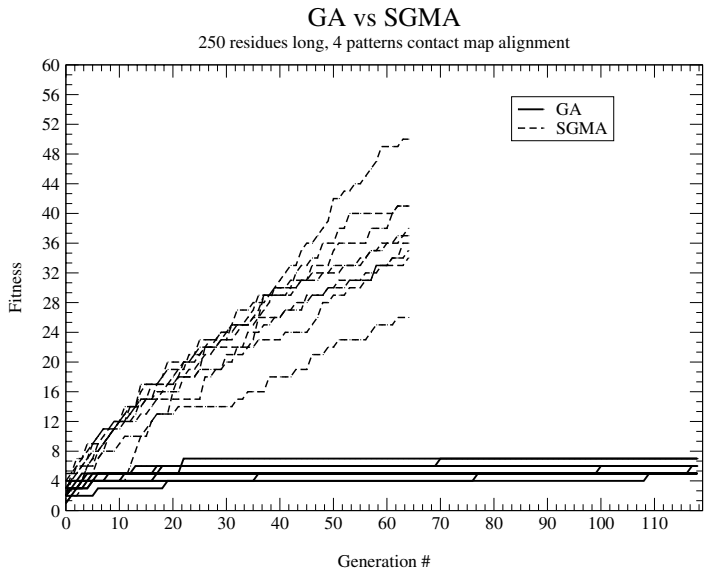


Figure 9. Representative example of GA and SGMA runs for a 250 residues and 4 patterns instance.

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 8(d)). 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 (Goldberg, 2002), 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 9 10 runs of the GA are compared against 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 during all the run. All the runs in Figure 9 use the same total number of fitness evaluations.

4. Conclusions

In this paper we investigated the concept of "Self-Generating Metaheuristics" and we exemplified its use in two hard combinatorial problems, *NK-Landscapes* and *MAX-CMO*. The particular implementation of Self-Generating Metaheuristics used in this paper was based on Memetic Algorithms. Unlike commonly held views on Memetic Algorithms and Hybrid Global-Local searchers, we do not resort here to human-designed local searchers but rather we allowed the SGMA to discover and assemble *on-the-fly* the local searcher that best suits the particular situation. In this paper we argued that from an optimization point of view there are obvious advantages in self-generating the local search behaviors for memetic algorithms. MAs that can self-generate the local searchers might 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 s earchers 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 global operators.

It is our hope that the ideas discussed on this paper would tempt researchers confronted with new problems for which there are not "silver bullet" local search heuristics[7] with which to hybridize a Memetic Algorithm to try the obvious: the Dawkins method of self-generation of local search behaviors, that is, the use of memes.

Acknowledgements

N. Krasnogor would like to thank J.E. Smith for many useful discussions over coffee, phone and e-mail on Self-Generating, alternatively called Co-evolving, Memetic Algorithms.

[7] Like it is the case for TSP and Graph Partitioning where *K-opt* and Lin-Kernighan are known to be extremely efficient.

References

Burke E, Newall J and Weare R (1996) A memetic algorithm for university exam timetabling. In: Burke E and Ross P (eds) The Practice and Theory of Automated Timetabling, Vol. 1153 of Lecture Notes in Computer Science, pp. 241–250. Springer Verlag, Berlin

Burke E and Smith A (24–28 November 1997) A memetic algorithm for the maintenance scheduling problem. In: Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference, Dunedin, New Zealand, pp. 469–472. Springer, Berlin

Carr R, Hart W, Krasnogor N, Burke E, Hirst J and Smith J (2002) Alignment of protein structures with a memetic evolutionary algorithm. In: Langdon W, Cantu-Paz E, Mathias K, Roy R, Davis D, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter M, Schultz A, Miller J, Burke E and Jonoska N (eds) GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference

Coveney P and Highfield R (1995) Frontiers of Complexity, the Search for Order in a Chaotic World. Faber and faber (ff), London

Creighton TE (ed) (1993) Protein Folding. W. H. Freeman and Company, New York

Eiben A and Smith J (2003) Introduction to Evolutionary Computing, ISBN: 3-540-40184-9. Springer, Berlin

França P, Mendes A and Moscato P (July 1999) 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

Gabora L (1993) Meme and variations: A computational model of cultural evolution. In: Nadel L and Stein D (eds) 1993 Lectures in Complex Systems, pp. 471–494. Addison Wesley, Boston, MA, USA

Goldberg D (2002) The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Dordrecht

Goldman D, Istrail S and Papadimitriou C (1999) Algorithmic aspects of protein structure similarity. Proceedings of the 40th Annual Symposium on Foundations of Computer Sciences, pp. 512–522

Kauffman S (1993) The Origins of Order, Self Organization and Selection in Evolution. Oxford University Press, UK

Koza J, Bennet F, Andre D and Keane M (1999) Genetic Programming III, Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco, CA, USA

Krasnogor N (2002) Studies on the Theory and Design Space of Memetic Algorithms. Ph.D. Thesis, University of the West of England, Bristol, United Kingdom. http://www.cs.nott.ac.uk/~nxk/papers.html

Krasnogor N (2003) Self-generating metaheuristics in bioinformatics: The proteins structure comparison case. To appear in the Journal of Genetic Programming and Evolvable Machines

Krasnogor N, Blackburne B, Burke E and Hirst J (2002) Multimeme algorithms for protein structure prediction. In: Proceedings of the Parallel Problem Solving from Nature VII. Lecture notes in computer science

Krasnogor N and Gustafson S (2002) Toward truly "memetic" memetic algorithms: Discussion and proof of concepts. In: Corne D, Fogel G, Hart W, Knowles J, Krasnogor N, Roy R, Smith JE and Tiwari A (eds) Advances in Nature-Inspired Computation: The PPSN VII Workshops. PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading

Krasnogor N and Smith J (2000) A memetic algorithm with self-adaptive local search: TSP as a case study. In: Whitley D, Goldberg D, Cantu-Paz E, Spector L, Parmee I and Beyer

H-G (eds) GECCO 2000: Proceedings of the 2000 Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, San Francisco, CA, USA

Krasnogor N and Smith J (2001) Emergence of profitable search strategies based on a simple inheritance mechanism. In: Spector L, Goodman E, Wu A, Langdon W, Voigt H, Gen M, Sen S, Dorigo M, Pezeshj S, Garzon M and Burke E (eds) GECCO 2001: Proceedings of the 2001 Genetic and Evolutionary Computation Conference. Morgan Kaufmann Publishers, San Francisco, CA, USA

Lancia G, Carr R, Walenz B and Istrail S (2001) 101 optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. Proceedings of The Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001

Macready W, Siapas A and Kauffman S (1996) Criticality and parallelism in combinatorial optimization. Science 261: 56–58

Merz P (2000) 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

Merz P and Freisleben B (1997) Genetic local search for the TSP: New results. In: Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, pp. 159–164. IEEE Press, Piscataway, NJ, USA

Merz P and Freisleben B (2000) Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. Journal of Evolutionary Computation 8(1): 61–91

Moscato P (1999) Memetic algorithms: A short introduction. In: Corne D, Glover F and Dorigo M (eds) New Ideas in Optimization. McGraw-Hill, New York

Moscato P (2001) Problemas de Otimizacão NP, Aproximabilidade e Computacão Evolutiva: Da Prática à Teoria. Ph.D Thesis, Universidade Estadual de Campinas, Brazil

Oates M, Corne D and Loader R (2000) Tri-phase profile of evolutionary search on uni- and multi-modal search spaces. Proceedings of the Congress on Evolutionary Computation (CEC2000) 1: 357–364

Ozcan E and Mohan C (1998) Steady state memetic algorithm for partial shape matching. In: Porto V, Saravanan N and Waagen D (eds) Evolutionary Programming VII: 7th International Conference (EP98, San Diego, California, USA, March 25–27, 1998), Vol. 1447 of Lecture Notes in Computer Science, pp. 527–236. Springer, Berlin

Sharpe O (2000) Introducing performance landscapes and a generic framework for evolutionary search algorithms. Proceedings of the Congress on Evolutionary Computation (CEC2000) 1: 341–348

Smith J (2002a) Co-evolution of memetic algorithms: Initial results. In: Merelo BF-V and Adamitis S (eds) Parallel Problem Solving from Nature – PPSN VII, LNCS 2439. Springer Verlag, Berlin

Smith J (2002b) The co-evolution of memetic algorithms for protein structure prediction. In: Corne D, Fogel G, Hart W, Knowles J, Krasnogor N, Roy R, Smith J and Tiwari A (eds) Advances in Nature-Inspired Computation: The PPSN VII Workshops, PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading

Weinberger E and Fassberg A (1996) NP completness of Kauffman's N-K Model, a tuneably rugged fitness landscape. In: Santa Fe Institute Technical Reports