Chapter 11
Self-adaptative and Coevolving Memetic Algorithms

James E. Smith

11.1 Introduction

Results from applications of meta-heuristics, and Evolutionary Computation in particular, have led to the widespread acknowledgement of two facts. The first is that evolutionary optimisation can be improved by the use of local search methods, creating so-called Memetic Algorithms. The second is that there is no single "best" choice of memetic operators and parameters- rather the situation changes according to both the problem and the particular stage of search. This has created a growing interest in "Adaptive" Memetic Algorithms which combine a portfolio of local search operators with some method to choose between them. Here we describe techniques which extend these ideas to allow the behaviours of the local search operators to adapt during the search process. In the first case these maybe thought of as Self-Adaptive, so that each member of the evolving population encodes for both an initial solution to a problem, and a learning mechanism which acts on that solution to improve it. More generally, we show that these can be treated as separate co-evolving populations of "genes" and "memes" . Following a review of related work, we next describe a framework for meme-gene self-adaptation and co-evolution. This is followed by a summary of the "proof-of-concept" and of findings concerning representation and scalability with self-adaptive memes. Next the paper considers in more depth issues relevant to co-evolution such as credit assignment, and the ratio of population sizes - which can be thought of as the memetic "load" that an evolving population can support.

James E. Smith
Department of Computer Science, University of the West of England,
Bristol, BS16 1QY, UK Name
e-mail: james.smith@uwe.ac.uk

F. Neri et al. (Eds.): Handbook of Memetic Algorithms, SCI 379, pp. 167–188.
springerlink.com

11.2 Background

The performance benefits which can be achieved by hybridising Evolutionary Algorithms (EAs) with Local Search (LS) operators, so-called *Memetic Algorithms* (MAs), have now been well documented across a wide range of problem domains such as optimisation of combinatorial, non-stationary and multi-objective problems (see [493] for a review, and [376] for a collection of recent algorithmic and theoretical work). Typically in these algorithms, a LS improvement step is performed on each of the products of the generating (recombination and mutation) operators, prior to selection for the next population. There are of course many variants on this theme, but these can easily be fitted within a general syntactic framework [493].

In recent years it has been increasingly recognised that the choice of LS operator will have a major impact on the efficacy of the hybridisation. Of particular importance is the choice of move operator, which defines the neighbourhood function, and so governs the way in which new solutions are generated and tested. For example Krasnogor and Smith used Polynomial Local search (PLS) theory to show that the worst-case runtime of an MA is not improved over the underlying EA if the LS neighbourhood function does not differ from those of the EAs variation operators [494]. However, points which are locally optimal with respect to one neighbourhood structure will not in general be so with respect to another, unless of course they are globally optimal. It therefore follows that even if a population only contains local optima, then changing the LS move operator (neighbourhood) may provide a means of progression in addition to recombination and mutation. This observation has led a number of authors to investigate mechanisms for choosing between a set of predefined LS operators which may be used during a particular run of a meta-heuristic such as an EA.

11.2.1 MAs with Multiple LS Operators

There are several recent examples of the use of multiple LS operators within evolutionary systems. Ong *et al.* [683] present an excellent recent review of work in the field of what they term "Adaptive Memetic Algorithms". This encompasses Krasnogor's "Multi-Memetic Algorithms" [486, 487, 491, 492, 496], Smith's COMA framework [821, 824, 825, 830], Ong and Keane's "Meta-Lamarkian MAs [680], and Hyper-Heuristics [96, 97, 169, 456]. In another interesting related algorithm, Krasnogor and Gustafson's "Self-Generating MAs" use a grammar to specify for instance when local search takes place [488, 490]. Essentially all of these approaches maintain a pool of LS operators available to be used by the algorithm, and at each decision point make a choice of which to apply. There is a clear analogy between these algorithms and Variable Neighbourhood Search [363], which uses a heuristic to control the order of application of a set of predefined LS operators to a single improving solution. The difference here lies in the population based nature of MAs, so that not only do we have multiple LS operators but also multiple candidate solutions, which makes the task of deciding which LS operator to apply to any given one more complex.

Ong's classification uses terminology developed elsewhere to describe adaptation of operators and parameters in Evolutionary Algorithms [237, 238, 386, 829]. This categorises algorithms according to the way that these decisions are made. One way (termed "static") is to use a fixed strategy . Another is to use feedback of which operators have provided the best improvement recently. This is termed "Adaptive", and is further subdivided into "external", "local" (to a deme or region of search space), and "global" (to the population) according to the nature of the knowledge considered. Finally they note that LS operators may be linked to candidate solutions (Self-Adaptive). We will adopt this terminology, and also make use of the general term "meme" to denote an object specifying a particular local search strategy.

11.2.2 Self-adaptation in EAs

We are concerned with meta-heuristics which maintain two sets of objects - one of genes and one of memes. If these sets are adaptive, and use evolutionary processes to manage what may now be termed populations, then we can draw some immediate parallels to other work. If the populations are of the same size and selection of the two is tightly coupled (to use the notation of [22]) then this can be considered as a form of Self Adaptation. The use of the intrinsic evolutionary processes to adapt mutation step sizes has long been used in Evolution Strategies [799], and Evolutionary Programming [268]. Similar approaches have been used to self-adapt mutation probabilities [31, 828] and recombination operators[793, 827] in genetic algorithms (GAs) as well as more complex generating operators combining both mutation and recombination [826]. More recently Smith and Serpell have showed that self-adaptation can very effectively govern both the choice and parameterisation of different mutation operators for GAs with permutation representations [807].

11.2.3 Co-evolutionary Systems

If selection is performed separately for the two populations, with memes' fitness assigned as some function of the relative improvement they cause in the "solution" population, then we have a co-operative co-evolutionary system. Following initial work by Husbands and Mill [399] this metaphor has gained increasing interest. Paredis has examined the co-evolution of solutions and their representations [709]. Potter and DeJong have also used co-operative co-evolution of partial solutions in situations where an obvious problem decomposition was available [727]. Both reported good results. Bull [90] conducted a series of more general studies on co-operative co-evolution using Kauffman's static NKC model. In [92] he examined the evolution of linkage flags in co-evolving "symbiotic" systems and showed that the strategies which emerge depend heavily on the extent to which the two populations affect each others fitness landscape. In highly interdependent situations linkage of the two species' chromosomes was preferred –which in our context is equivalent to memes self-adapting as part of the solutions' genotypes. Bull also examined the effect of various strategies for pairing members of different populations for

evaluation [91]. This showed mixed results, although the NKC systems he investigated used fixed interaction patterns. This work has recently been revisited and extended by Wiegand *et al.* with very similar findings [933]. Wiegand's work also focused attention on the number of partners with which a member of either population should be evaluated, which draws attention to the trade-off between accurately estimating the value of an object (solution or meme), and using up evaluations doing so. Parker and Blumenthal's "Punctuated Anytime Learning with samples" [714] is another recent approach to the pairing problem by using periodic sampling to estimate fitness, but this is more suited to approaches where the two populations evolve at different rates. Closely related to this, Bull, Holland and Blackmore have examined the effect of changing the relative speed of evolution of populations which they termed "genes" and "memes" [93]. Their results showed that as the relative speed of meme evolution increased a point was reached beyond which gene evolution effectively ceases. However, the NKC systems they use severely limit the types of interaction permitted to an abstraction rather different from most MA applications.

There has also been a large body of research into competitive co-evolution (see [710] for an overview). Here the fitnesses assigned to the two populations are directly related to how well individuals perform against the other population - what has been termed "predator-prey" interactions. Luke and Spector [541] have proposed a general framework within which populations can be co-evolved under different pressures of competition and co-operation. This uses speciation both to aid the preservation of diversity and as a way of tackling the credit assignment problem.

In all the co-evolutionary work cited above, the different populations only affect each other's perceived fitness, unlike the COMA framework where the meme population can directly affect the genotypes within the solution population. This raises the question of whether the modifications arising from Local Search should be written back into the genotype (Lamarckian Learning) or not (Baldwinian Learning). Although the pseudo-code and the discussion below, assumes Lamarckian learning, this is not a prerequisite of the COMA framework. However, even if a Baldwinian approach was used, COMA differs from the co-evolutionary systems above because there is a selection phase within the local search, so that if all of the neighbours of a point defined by the meme's rule are of inferior fitness, then the point is retained unchanged within the population. If one was to discard this criterion and simply apply the rule (possibly iteratively), the system could be viewed as a type of "developmental learning" akin to the studies in Genetic Code e.g. [443] and the "Developmental Genetic Programming" of Keller and Banzhaf [453, 454].

11.3 A Framework for Self-adaption and Co-evolution of Memes and Genes

In this section we describe a conceptual framework designed to support a wide range of algorithms for meme adaptation.

11.3.1 Specifying Local Search

The primary factor that affects the behaviour of the LS is the choice of neighbourhood generating function. This can be thought of as defining a set of points $n(i)$ that can be reached by the application of some move operator to the point i. One representation is as a graph $G = (v, e)$ where the set of vertices v are the points in the search space, and the edges relate to applications of the move operator i.e $e_{ij} \in G \iff j \in n(i)$. The provision of a scalar fitness value, f, defined over the search space means that we can consider the graphs defined by different move operators as "fitness landscapes" [433]. Merz and Freisleben [585] present a number of statistical measures which can be used to characterise fitness landscapes, and have been proposed as potential measures of problem difficulty. They show that the choice of move operator can have a dramatic effect on the efficiency and effectiveness of the Local Search, and hence of the resultant MA.

The second component of Local Search is the choice of pivot rule, which can be *Steepest Ascent* or *Greedy Ascent*. In the former the "termination condition" is that the entire neighbourhood $n(i)$ has been searched, whereas the latter stops as soon as an improvement is found. Note that one can consider only a randomly drawn sample of size $N << \mid n(i) \mid$ if the neighbourhood is too large to search.

The final component is the "depth" of the Local Search. This lies in the continuum between only one improving step being applied to the search continuing to local optimality. Studies with MAs e.g. [366] have shown it affects the performance both in terms of time taken and of quality of solution found.

11.3.2 Adapting the Specification of Local Search

The aim of this work is to provide a means whereby the definition of the LS operator used within a MA can be varied over time, and then to examine whether evolutionary processes can control that variation so that beneficial adaptation takes place. Accomplishing this aim requires the provision of four major components, namely:

- A means of representing a LS operator in an evolvable form i.e. as a meme.
- A means of assigning fitness to memes.
- A choice of population structures and sizes, selection and replacement methods for managing the meme population.
- A set of experiments to permit evaluation and analysis of the system.

The pseudo-code in Algorithm 21 illustrates the algorithmic framework of a COevolutionary Memetic Algorithm (COMA) developed to support this research. Note that although this pseudo-code assumes synchronous evolution, this need not in general be the case. The representation of the memes is a tuple *<Pivot, Depth, Pairing,Move>*, which can readily encompass all of the other requirements identified above. The representation of the tuple elements leads naturally to the choice of evolutionary variation operators. The *Pivot,Depth* and *Pairing* elements can be easily mapped onto integer or cardinal representations. The latter element, is one

Algorithm 21. Pseudo-Code Definition of COMA algorithm

```
   // Given populations P of μ_s solutions and M of μ_m memes
 1 initialise P and M randomly ;
 2 set generations ← 0;
 3 set evaluations ← 0;
 4 while run_termination condition is satisfied do
       // Create μ_s solution offspring and store parent ids
 5     for i ← 1 to μ_s do
 6         set FirstParent[i] ← Select_One_Parent(P);
 7         set SecondParent[i] ← Select_One_Parent(P);
 8         set Offspring[i] ← Recombine(FirstParent[i],SecondParent[i]);
 9         Mutate(Offspring[i]);
10         set i ← i +1;
11     endfor
       // Create mu_m meme offspring according to pairing
12     for i ← 1 to μ_m do
13         set Pairing ← Get_Pairing(M,i);
14         if Pairing = SelfAdaptive then
15             set MemeParent1[i] ← FirstParent[i];
16             set MemeParent2[i] ← SecondParent[i];
               // note this requires μ_m = μ_s .
17         endif
18         else if Pairing = Fitness_Based then
19             set MemeParent1[i] ← Select_One_Parent(M);
20             set MemeParent2[i] ← Select_One_Parent(M);
21         endif
22         else
23             set MemeParent1[i] ← RandInt(1,μ_m);
24             set MemeParent2[i] ← RandInt(1,μ_m);
25         endif
26         set NewMemes[i] ← Recombine(MemeParent1[i],MemeParent2[i]);
27         Mutate(NewMemes[i]);
28         set i ← i+1;
29     endfor
       // Apply local search to Offspring Using Memes
30     for i ← 1 to μ_s do
31         set original_fitness ← Get_Fitness(Offspring[i]);
32         if Pairing = SelfAdaptive then
33             set meme ← i;
34         endif
35         else
36             set meme ← Select_Random(NewMemes);
37         endif
38         set Neighbours ← Apply_Rule_To_Offspring(Offspring[i],NewMemes[meme]);
39         Evaluate_Fitness(Neighbours);
40         set Offspring[i] ← Apply_Pivot_Rule(Neighbours);
41         set Δfitness ← Get_Fitness(Offspring[i]) - original_fitness;
42         Update_Meme_Fitness(NewMemes[meme], Δfitness);
43         set evaluations ← evaluations +1 + |Neighbours|;
44         set i ← i +1;
45     endfor
46     set P ← Offspring;
47     set M ← NewMemes;
48 endw
```

of {*Self-Adaptive, Random, Fitness_Based*} and determines how memes are created and applied to solutions. As is illustrated in the *If..Else* section of the pseudo-code, a range of behaviours from self-adaptive, through collaborative co-evolution to random meme drift can be obtained.

This framework is designed it be generic in the way that move operators are described - for example they could be GP-like expressions as per [288]. However while such richness tends to lead to complexity of expression suitable for practical applications, it can make analysis of evolved behaviour more difficult. Therefore for the initial development work a simpler format was used together with well-understood test problems. In what follows, move operators are encoded as *condition:action* pairs, which specify one pattern to be looked for in the problem representation, and another to replace it. The neighbourhood of a point i then consists of i itself, plus all those points where the substring denoted by *condition* appears in the representation of i and is replaced by the *action*. To give an example, a rule $1\#0 \rightarrow 111$ matches the binary string 1100111000 in the first, second, sixth and seventh positions, and the neighbourhood is the set {1100111000, 1110111000, 11111111000, 1100111100, 1100111110}.

Note that the string is not treated as toroidal, and the neighbours are evaluated in a random order so as not to introduce positional bias into the local search when greedy ascent is used. Although this representation at first appears very simple, it has the potential to represent highly complex moves via the use of symbols to denote not only single/multiple wild-card characters (in a manner similar to that used for regular expressions in Unix) but also the specifications of repetitions and iterations. Further, permitting the use of different length patterns in the *condition* and *action* parts of the rule gives scope for *cut* and *splice* operators working on variable length solutions.

11.4 Test Suit and Methodology

A range of well understood test problems were used to examine the performance of various self-adaptive and coevolutionary MAs. Some of these are "standard" testbed functions for EAs, others were specifically designed to probe and evaluate certain behaviours. The initial systems only used rules where the *condition* and *action* patterns were of equal length and were composed of values taken from the set of permissible allele values of the problem representation, augmented by a "don't care" symbol (#) which is allowed to appear in the *condition* part of the rule and optionally in the *action* where it is treated as *invert*. In practise, each rule was augmented by a value *rule_length* specifying the number of positions in the pattern string to consider. This permitted not only the examination of the effects of different fixed rule sizes, but also the ability to adapt its value via mutation.

11.4.1 The Test Suite

The first set of problems used are composed of 16 subproblems of Deb's 4-bit fully deceptive function [35]. The fitness of each subproblem i is given by its unitation $u(i)$, that is the number of bits set to "one":

$$f(i) = \begin{cases} 0.6 - 0.2 \cdot u(i) & : \quad u(i) < 4 \\ 1 & : \quad u(i) = 4 \end{cases} \tag{11.1}$$

In addition to a "concatenated" version (4-Trap), a second "distributed"version (Dist-Trap) was used in which the subproblems were interleaved i.e. sub-problem i was composed of the genes $i, i+16, i+32, i+48$. This separation ensured that in a single application even the longest rules allowed in these experiments would be unable to alter more than one element in any of the sub-functions. A third variant of this problem (Shifted-Trap) was designed to be more "difficult" than the first for the COMA algorithm, by making patterns which were optimal in one sub-problem, sub-optimal in all others. Since unitation is simply the Hamming distance from the all-zeroes string, each sub-problem can be translated by replacing $u(i)$ with the Hamming distance from an arbitrary 4 bit string. There were 16 sub-problems so the binary coding of each ones' index was used as basis for its fitness calculation.

The second test function was Watson's Hierarchical-if-and-only-if (H-IFF) function, a highly epistatic problem designed to examine the virtues of recombination. At the bottom level, fitness is awarded to matching pairs of adjacent bits in a solution s, i.e.

$$f_1 s = \sum_{i=0}^{l/2-1} 1 - XOR(s_{2i}, s_{2i+1}) \tag{11.2}$$

and this process is applied recursively, so that a problem of size $l = 2^k$ has k levels. In each ascending level the number of blocks is reduced by a factor of two, and the fitness awarded for each matching pair is increased by a constant factor, in our case 2. This problem has a number of Hamming sub-optima, and two global optima corresponding to the $u(i) \in \{0,1\}$. Problem sizes $l \in \{16,\ldots,512,1024\}$ were used, corresponding to 3 to 10 levels. Note that for $l > 16$ the length of the blocks to be identified at the highest levels far exceeded the maximum rule length.

The Maximum satisfiability (Max-SAT) problem is a classical combinatorial optimisation problem, consisting of a number of Boolean variables and a set of clauses built from those variables. A full description and many examples can be found in [392]. For each length {50,100,250} the first 25 were taken from the sets of uniformly randomly created satisfiable instances around the phase transition (in terms of hardness) where there are approximately 4.3 clauses per variable.

11.4.2 Experimental Set-Up and Terminology

A generational genetic algorithm, with deterministic binary tournament selection for parents and no elitism was used. Population size μ_s was 250 unless otherwise

stated. One Point Crossover (with probability 0.7) and bit-flipping mutation (with a bitwise probability of 0.01) were used on the problem representation. These choices were taken as "standard", and no attempt was made to tune them to the particular problems at hand. Mutation was applied to the rules with a allele-wise probability of 0.0625 - the inverse of the maximum rule length allowed to the adaptive version. If the rule length was adaptive, they were randomly initialised in the range [1,16], and during mutation, a value of +/- 1 is randomly added with probability 0.0625, subject to staying in range.

For each problem, 20 runs were made, each continuing until the global optimum was reached, subject to a maximum of 500,000 evaluations. Two performance metrics were considered; the Success Rate (SR) which is the number of runs finding the global optimum, and the Average Evaluations to Success (AES) which is the mean time taken to locate the global optimum on successful runs. The reason for the large cut-off value was to try and avoid skewing results as can happen with an arbitrarily chosen lower cut-off, rather than to be indicative of the amount of time available for a "real world" problem. Note that since one iteration of a local search may involve several evaluations, this allows more generations to the GA, i.e. algorithms are compared strictly on the basis of the number of calls to the evaluation function. Any observed differences in performance were tested for statistical significance using ANOVA and pairwise post-hoc testing using the Tukey's Least Significant Difference (LSD) and Tamhane's T2 tests at the 95% confidence level.

The variants of self- and co-adaptive algorithms that can be instantiated within this framework are denoted as *CXY* where *X* denotes the pairing and is one of *L* (Linked, or self-adaptive), *R* (Random drift) or *T* (Tournament - variants of fitness based coevolution). *Y* denotes the pivot function and is one of *G*reedy, *S*teepest or *A*daptive. Rule lengths are adaptive unless denoted by a numeric prefix. Depth of search is one unless indicated by a suffix *-L* (to local optima) or *-A*daptive.

11.5 Self-adaptation of Fixed and Varying Sized Rules

11.5.1 Self-adapting the Choice from a Fixed Set of Memes

The first experiments in this line of research explored the ability of evolutionary mechanisms to correctly select between a number of fixed memes. This can be achieved trivially within the COMA framework by the use of appropriate initialisation for the meme population, setting the meme recombination probabilities to zero and defining the mutation operator so that it chose between the set of fixed memes rather than operating "within" each meme. In [492] experiments were run on a range of TSP problems using MAs with one of set of ten memes which varied in both their move operators and depth. When the search progress was plotted together, it could clearly be seen that the optimal choice of meme was dependant on the state of the search as well as on the individual TSP instance. Next the population members of "multimeme" algorithm were allowed to self-adapt the choice of which meme to use. The results showed that the progress tracked that of the

currently best-performing meme from the ”static” MAs, ultimately outperforming each of them. The evolved patterns of meme usage closely matched what might have been ”designed” with hindsight, with periods of one meme dominating alternating with periods of broader usage as local optima were reached, then escaped from.

The concept of self-adapting the choice from a fixed set of memes was also successfully demonstrated by Krasnogor *et al.* for protein structure alignment [496].

11.5.2 Self-adaptation of Meme Definitions

Initial experiments were restricted to considering a simple system, and examining first whether the system was able to evolve useful rules for the ”trap” problems - first when the rule length naturally matched the structure of the problem, and then whether the system was able to adapt to an appropriate rule length for different problems. For this reason it was decided to avoid the various issues concerning population management, pairing strategies and credit assignment, and instead work with a single improvement step, a fully linked self-adaptive system and a greedy pivot rule. These choices were coded into the chromosomes at initialisation, and variation operators were not used on them. The algorithms used (and the abbreviations which will be used to refer to them hereafter) are as follows:

- A “vanilla” GA with no Local Search (GA).
- A simple bit-flipping MA (SMA-G).
- COMA using a random rules, i.e. with the learning disabled (CRG).
- COMA with self-adaptive memes, greedy pivot and adaptive rule lengths (CLG).
- COMA using fixed length memes (1-CLG,. . .,10-CLG),

Experiments were run with population sizes (μ_s, μ_m) of 100, 250 and 500.

11.5.3 Results on Trap Functions

The results on 4-Trap showed that the GA, SMA, and 1-CLG algorithms frequently failed to find the optimum but the other COMA variants, always did. On these problems there was a clear benefit to using adaptive neighbourhood local search, although since the CRG algorithm also found the optimum on every run, it cannot be concluded from the Success Rates that learning was taking place. Considering the AES, the GA, SMA and 1-CLG algorithms took significantly longer to locate the optimum. For a population of 500 2-CLG joined the significantly slower group.

In short, it could be observed that for fixed rule lengths of between 3 and 9, and for the adaptive version, the COMA system derived performance benefits from evolving LS rules according to both metrics on this function.

For the Shifted-Trap function, the performances of the GA and SMA were not significantly different from those on 4-Trap because these algorithms solved the subproblems independently and so were “blind” to whether the optimal string for each was different. The COMA results exhibited the same pattern of behaviour noted

above; fast, reliable problem solving for all but 1-CLG and 2-CLG, and even for these two the AES results were statistically significantly better than GA or SMA.

On Dist-Trap, GA, SMA and CRG never located the global optimum, regardless of population size. While the Success Rate for COMA was less than for the other problems (typically 10-15/20 for $\mu = 100$ and 15-20/20 for $\mu = 250$), the same pattern was observed of better performance (SR and AES) for the adaptive version and fixed rule lengths in the range 3-5, tailing off at the extremes of the length range.

11.5.4 Analysis of Results and Evolution of Rule Base

The deceptive functions used were specifically chosen because GA theory suggests they are best solved by finding and mixing optimal solutions to sub-problems. Thus the GA failed to solve the function when the crossover operator was not suited to the representation (Dist-Trap). Considering the action of a single bit-flipping LS operator on these "trap" subproblems, a search of the Hamming neighbourhood of a solution will always lead towards the sub-optimal solution when the unitation is 0,1 or 2, regardless of pivot rule. Additionally, the greedy search of the neighbourhood will lead towards the deceptive optimum 75% of the time when the unitation is 3. This explains the poor results of the SMA, and 1-CLG algorithms.

The behaviour of the CLG algorithm was examined by plotting the population mean against time of the rule length, the specificity of the condition (the proportion of values set to #), and the unitation of the action. These results are shown in Figure 11.1.

For the 4-Trap function, the system rapidly evolved medium length $(3-4)$, general (specificity $< 50\%$) rules whose action was to set all the bits to 1 (mean unitation 100%). Closer inspection of the evolving rule-base confirmed that the optimal subproblem string was being learnt and applied.

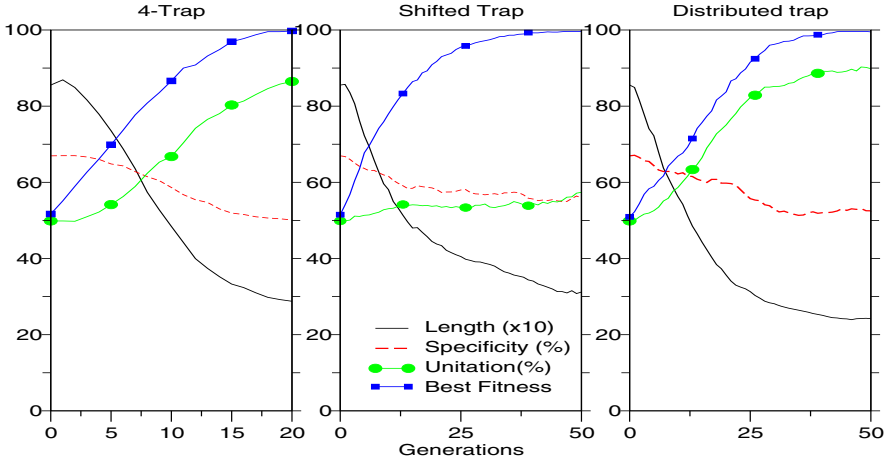

Fig. 11.1. Analysis of Evolved Rules on three problems with different properties

For the Shifted-Trap function, where the optimal sub-blocks are all different, the rule length decreased more slowly from its initial mean value of 8. The specificity also remained higher, and the mean unitation remained at 50%, indicating that different rules were being maintained. This was borne out by closer examination of the evolved rule sets.

The behaviour on Dist-Trap was similar to that on 4Trap, albeit over a longer time-scale. The algorithm could not possibly be learning specific rules about sub-problems, since no rule was able to affect more than one locus of any subproblem. Rather, the system learnt the general rule of setting all bits to 1. The rules were generally shorter than for 4Trap, which means that the number of potential neighbours was higher for any given rule. The high incidence of #s meant that the rule length defined a maximum radius in Hamming space for the neighbourhood, rather than a fixed distance from the original solution. These two observations, together with the longer times to solution, suggest that when the system was unable to find a single rule that matched the problems' structure, a more diverse search took place using a more complex neighbourhood which slowly adapted itself to the current solution population. Full details of these experiments and analysis may be found in [821].

11.5.5 Benchmarking the Self-adaptive Systems

In order to test these hypotheses about how the memes self-adapt in different ways a further set of experiments was run using a wider range of problems, with 50 runs per problem-length to tease out statistically significant differences. For the first two sets of results, both steepest and greedy ascent pivot rules were tried, for the final, MAX-SAT problem, the pivot rule was also allowed to adapt under mutation.

11.5.5.1 Exploiting Search Space Regularities Gives Scalability

The hypothesis memes adapt to identify and exploit any regularities in the problem space was tested by varying the lengths of two problems. The first of these comprised multiple concatenated copies of (11.1) with lengths in the range {40, 60, 80,..., 200}. As expected from above, the results for SMA-G were extremely poor. The next worse algorithm was CRS. The SR steadily decreased 50 (100%) at length 40 to 5 at length 100 and zero above that. All the other algorithms showed SR of 49 or 50 up to length 160, but only the CLS (39) and CLG (50) solved the 200-bit problem. This provides evidence that learning is taking place in the meme populations. The AES results were revealing. The GA was faster than CLG and CLS but the increase in AES with length was worse than linear. The AES results of the successful COMA variants, and analysis of the evolving rule bases, supported the hypothesis of discovering and exploiting regularities. In this case it meant identifying a rule giving the optimal solution to the sub-problems, and then applying it to each sub-problem in the string in successive generations. as shown in Figure 11.2 CLG was the fastest algorithm, followed by CLS, and all three were near-linear. For example, a linear regression of AES to length for CLG fitted the data with a correlation co-efficient of 0.97.

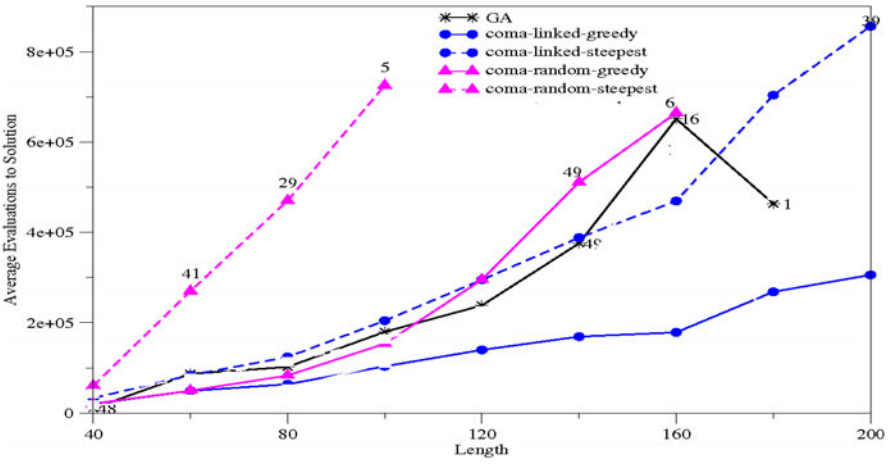

Fig. 11.2. Efficiency of different algorithms on 4 Trap functions with varying length. Annotations beside points show where Success Rates were less than 50/50.

On the H-IFF problems all of the MAs had higher Success Rates than the GA, and again the CLG and CLS were significantly the best. For example, out of 50 runs with $l = 128$ the SR values were 0 (GA, CRG, CRS), 4 (MA), 38 (CLG), 43 (CLS). Only the CLG (10) variant solved the 256 bit problem. As on other problems: the greedy ascent versions found the optimum faster (lower AES) than the equivalent steepest ascent versions but not as reliably (lower SR). ANOVA on the MBF results confirmed that the performance was statistically significantly different with 95% confidence. Post-hoc analysis showed that the CLG and CLS variants had a higher mean best fitness than the others but did not significantly differ.

11.5.5.2 Escaping Local Optima by Changing Neighbourhoods

Shifted-Trap, Dist-Trap and MAX-3SAT were used to examine the behaviour when there were no regularities that could be exploited. On the Dist-Trap function, only the CLS and CLG algorithms ever located the global optimum, and both always did, CLG significantly faster than CLS. On the Shifted-Trap function, the success rates were 39/50 (CRS) 45/50 (SMA-G) and 50/50 (all others). There was no significant difference in the mean times to solution.

On MAX-SAT the GA, steepest/greedy simple MAs (SMA-S, SMA-G), and self-adaptive COMA algorithms with greedy, steepest and adaptive pivot strategies (CLG, CLS, CLA) were run ten times on each instance. Table 11.1 shows the number of success from 250 runs. Full experimental details , and some results omitted here for brevity, may be found in [830].

As can be seen, for the 50 variable instances the simple MAs have the highest success rates, and the GA the worst. For the longer instances all methods are much less successful, and many instances are not solved by any algorithms. SMA-G and

Table 11.1. Success Rates (out of 250) for different length MAX-3SAT problems.

Algorithm	Length 50	Length 100
GA	125	21
SMA-S	154	0
SMA-G	153	25
CLS	141	0
CLG	135	25
CLA	144	8

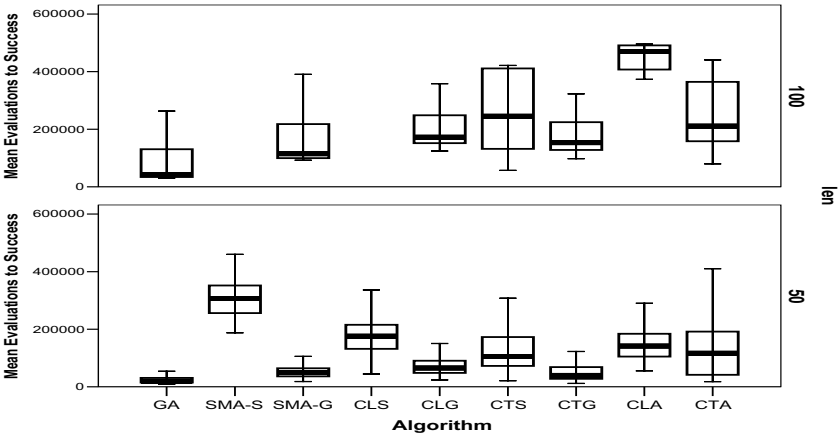

Fig. 11.3. Box plots of AES for 100 (top) and 50 (bottom) variable MAX-3SAT instances.

CLG show the same performance For the shorter instances the steepest ascent strategy is on average better, but there are differences between individual instances. For the longer instances the cost of searching the entire neighbourhood every iteration becomes prohibitive, so that SMA-S and CLS solve no instances. Analysis shows that the adaptive variant CLA performs on a par with whichever of the S or G variants is better for each instance, suggesting successful adaptation.

The AES results show that the GA is the fastest algorithm followed by a close grouping of SMA-G, CLG then CLA, with the CLS algorithm taking more time and having a higher variance. The adaptive pivot variants both fall between their respective greedy and steepest counterparts, both in terms of mean and variance. GA was significantly faster, and the SMA-S significantly slower than the other algorithms, CLS/CLA/CLG did not significantly differ. Analysis of the mean best fitness showed that the CLA algorithm came between the two fixed strategies, but again the ordering of CLG/CLS, and the magnitude of the difference between them, was instance dependant.

11.5.6 Summary of Self-adaptive Results

The results above highlight the problem of choosing the appropriate local search operator which provided the original rationale for the development of COMA. For example, although the Memetic Algorithm with a simple bit-flipping hill climber had the highest Success Rates and Mean Best Fitness on the Max-3SAT problems, it's performance on the other problems was derisory, and frequently worse than the simple GA. In contrast the self-adaptive MAs exhibited better performance than the GA or SMA over a wide range of problems, according to different metrics. Fuller details of the experiments with binary-coded problems may be found in [821, 822, 830], and details of successful application to a protein structure prediction problem may be found in [822, 825]. Overall adapting the pivot rule (CLA) is outperformed by whichever is better of steepest or greedy ascent, but the difference is often marginal, and more importantly the choice (CLS or CLG) is problem dependant.

11.6 Extension to True Co-evolution: the Credit Assignment Problem

Having established the basic principle of evolving memes which coded for LS rules as a means of enhancing optimisation performance in MAs, the next series of experiments used a full co-evolutionary model. Experiments reported in [824, 825, 830] showed that a major factor determining successful adaption was the credit assignment mechanism used to award fitness to a solution. The results also showed that with meme fitness dependant simply on the improvement caused, the choice of pivot and pairing strategies are intertwined.

Unsurprisingly, the greedy variants almost always used less evaluations than the steepest ascent equivalents on successful runs. However, for some problems (but not all) the extra noise introduced by using a greedy ascent was sufficient to "fool" the simple credit assignment mechanism. Thus a good rule will only get a low fitness if the first match only leads to a small improvement, whereas larger improvement (and hence fitness) might be seen if it was applied elsewhere in the solution. Another related source of noise is the choice of partner.

In [831] a number of methods were examined to try and overcome the difficulties of the greedy strategy by reducing the amount of noise present. Meme parent selection used binary tournaments based on fitness defined in the following ways.

- Simple "global-adaptive" scheme where the fitness of a meme was the improvement it caused when applied to a solution (CT). Note that even if a meme perfectly encapsulates the problem structure it can achieve zero fitness if it happens not to match or change the solution it is paired with.
- COMA with a "memory" (CTD). Inspired by Paredis' "Life Time Fitness Evaluation" (LTFE) [710] this uses a time-decaying fitness function of the form:

$$meme\_fitness' = meme\_fitness \cdot \alpha + improvement\_caused \tag{11.3}$$

A newly created meme takes the average of its parent's fitnesses. After initial experiments a decay factor of $\alpha = 0.5$ was used.

- A modification to the COMA algorithm so that two solution parents contribute to create two offspring solutions via recombination, and similarly for the memes. Each meme is then tested against both of the solutions and the fitness assigned is either the mean (**CT2M**) or better (**CT2B**) of the two improvements noted. In Wiegand's terminology this is a *collaboration poolsize* of two. Each solution takes the better of the two neighbours found for it.

11.6.1 Results: Reliability

Table 11.2 shows the Success Rates achieved with the different algorithms on each function and problem length. The results not just for the 3 Trap variants but also the H-IFF show the clear advantage of Adaptive Memetic Algorithms over both the static counterpart (SMA-G) and a simple Genetic Algorithm (GA). The global-random scheme (CRG) shows lower Success Rates than the other COMA algorithms on most problems. The self-adaptive scheme (CLG) also has lower success rates on the longer H-IFF problems and the SAT problems that the co-evolutionary

Table 11.2. Success Rates of Algorithms on Different Functions

Function	Len	CRG	CLG	CT2BG	CT2MG	CTDG	CTG	GA	SMAG
4Trap	20	10	10	10	10	10	10	10	10
	40	10	10	10	10	10	10	10	6
	60	10	10	10	10	10	10	10	3
	80	10	10	10	10	10	10	10	0
	100	10	10	10	10	10	10	10	0
	120	10	10	10	10	10	10	8	0
	140	9	10	10	10	10	10	3	0
	160	10	10	10	10	10	10	1	0
	180	2	10	7	9	10	10	0	0
	200	0	10	2	4	10	10	0	0
Shifted Trap	64	10	10	10	10	10	10	10	3
Dist-Trap	64	0	10	10	10	10	9	0	0
H-IFF	16	10	10	10	10	10	10	10	10
	32	10	10	10	10	10	10	5	10
	64	2	9	9	10	10	8	4	8
	128	0	3	5	8	6	7	0	0
	256	0	0	6	4	4	3	0	0
SAT	50	131	134	146	152	136	145	114	153
	100	28	21	24	26	16	25	38	27
Total		272	307	319	333	312	327	243	230

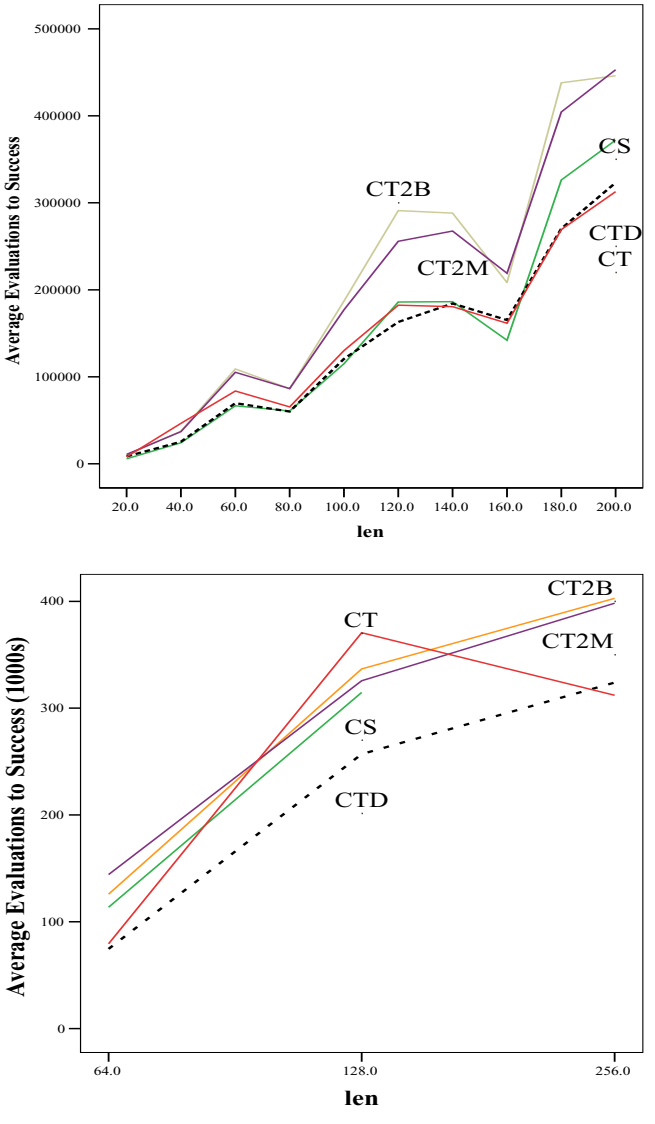

Fig. 11.4. Average Evaluations to Success on Trap (top) and H-IFF (bottom) functions.

variants. Comparing the four different fitness schemes for coevolution, no clear pattern emerges:

- On the 4-trap functions, the meme evolution task is to identify rules of the form #### → 1111, and then maintain and exploit them through repeated application. Here the simpler schemes based on a single pairing (CTG, CTDG) find the optimum slightly more often for the longer instances in the time allowed.

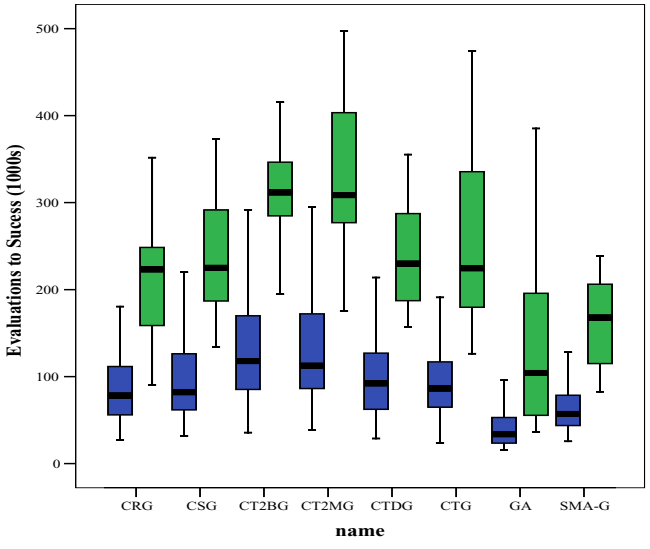

Fig. 11.5. Box-plots of Evaluations to Success on SAT functions. Lighter boxes are for 50-variable instances, darker ones for 100 variables.

- On the Shifted-Trap and Dist-Trap functions, where it is necessary to maintain a *diverse* rule-set in the meme population, algorithms perform the same (100% Success), except the simple CTG (9/10 on Dist-Trap).
- On the H-IFF and SAT problems the fitness schemes based on a collaboration poolsize of 2 are more successful, the averaging version (CT2MG) especially so. Notably the CTDG scheme with memory and a collaboration poolsize of 1 is markedly less successful than the others on the SAT functions.
- Overall the CT2MG algorithm has the highest success rate.

11.6.2 Results: Efficiency

Figures 11.4 and 11.5 illustrate the change in the mean time to locate the optimum for the Trap, H-IFF and SAT functions used with different length instances. The results for the GA, SMAG and CRG are omitted from the first two for the sake of clarity as they are so poor. On the Trap functions the results with collaboration poolsize 1 (CTG, CTDG, CLG) are obtained faster than with the poolsize of 2 (CT2MG, CT2BG), the difference being increasingly statistically significant for the longer instances. This is a natural result of the overhead of testing each meme against two solutions - since the solution just takes the better of the two improvements to be the result of its Lamarkian learning, the other evaluations are “wasted” from that point of view. This explains the lower SR for CT2MG/CT2BG on longer 4-Trap problems. However on the H-IFF function the CTG approach is not only less successful than

the CTDG approach, but takes more evaluations when it does find the optimum. This can be explained by the fact that the algorithm needs to make a decision between the all '1's solution and the all '0's solution, and the use of a memory helps make this decision consistent between generations. On the SAT problems, where there is no regular problem structure to be learnt and exploited, the CT2M/B G schemes again significantly take longer.

11.7 Varying the Population Sizes

The results in the previous section clearly demonstrate the advantages of a credit assignment mechanism that does not rely solely on the improvement caused when a meme is applied to a single solution. In general those schemes that make use of multiple collaborations (to use Wiegend's terminology) - either explicitly within the same generation, or via a memory - have higher success rates, but this is sometimes at the expense of significantly increased run-times. The memory-based approach (CTDG) is faster, but can be mislead as shown by the lower SR results for the SAT functions. We hypothesise that this is because the meme population is not converging in these runs, so the use of fitness inherited from both parents is more "noisy".

One obvious way to assess memes in the context of multiple solutions (points in the search space) without "wasting" evaluations is to reduce the size of the meme population μ_m relative to μ_s. To investigate this, a series of experiments were run using different size meme populations. After some brief initial experimentation, the following changes were made to the parameter settings, with the results shown in Table 11.3:

- The "#" character in a "action" string is taken to mean "invert the current value".
- The solution population size was increased to 500 and self-adaptive mutation was applied using the scheme outlined in [820, 823, 850].
- The tournament size in the meme population was increased from 2 to 5. This effectively reduces the size of the meme population since less fit memes have a smaller probability of being selected as parents.
- The fitness of each meme is assigned by summing improvement that meme caused in different solutions divided by the number of calls to the evaluation function used. However multiple copies of memes were allowed so this potentially provides a mix of what Schoenauer *et al.* have termed "extreme" and "average" value rewards in the context of adaptive operator selection in EAs [260].

The results of these experiments are presented in Table 11.3 and Figure 11.6, and can be summarised as follows:

- Overall the COMA algorithms are clearly more effective (higher SR) than the GA and SMA.
- Although not shown for reasons of clarity, the coevolutionary memetic algorithms are also overall more efficient (lower AES) than the GA or SMA.

Table 11.3. Success Rates of different functions as number of memes is varied

	H-IFF							Trap					Max-Sat		
Algorithm	16	32	64	128	256	512	1024	Total	40	80	120	160	200	Total	50	100	Total
CTA-pop-10	50	48	45	24	23	15	5	210	48	46	34	34	32	194	196	40	236
CTA-pop-50	50	50	49	46	37	39	31	302	50	47	49	48	44	238	229	49	278
CTA-pop-100	50	50	50	44	42	37	29	302	50	50	50	49	49	248	232	54	286
CTA-pop-200	50	50	50	46	40	39	34	309	50	50	50	50	50	250	257	62	319
CTA-pop-400	50	50	50	48	41	36	29	304	50	50	50	50	50	250	260	56	316
GA	50	33	2	0	0	0	0	85	30	3	0	0	0	33	100	15	115
CTG-pop-10	50	49	42	34	30	17	9	231	50	39	39	36	32	196	212	45	267
CTG-pop-50	50	50	47	47	41	34	33	302	50	50	47	45	45	237	224	49	273
CTG-pop-100	50	50	49	49	41	37	31	307	50	50	50	50	48	248	247	49	296
CTG-pop-200	50	50	50	46	44	38	33	311	50	50	50	50	50	250	247	61	308
CTG-pop-400	50	50	50	49	41	36	38	314	50	50	50	50	50	250	257	57	314
SMA-G	50	49	24	1	0	0	0	124	38	6	0	0	0	44	246	48	294

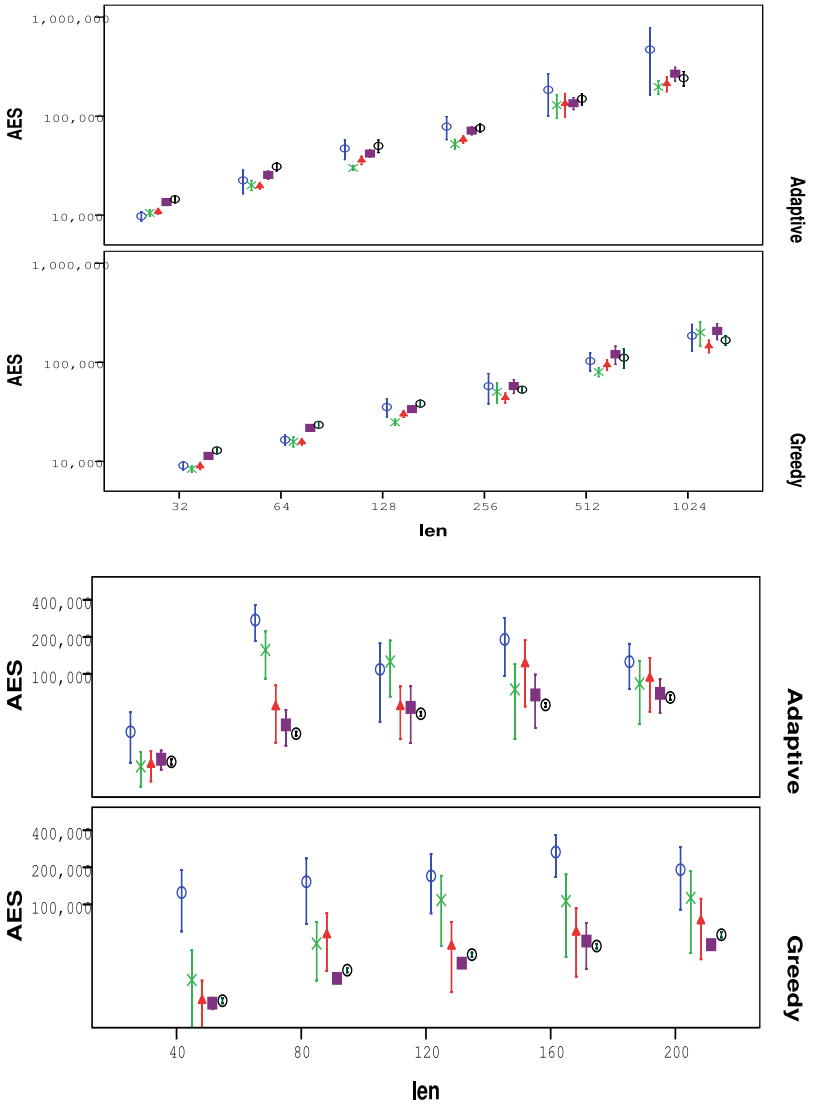

Fig. 11.6. Average Evaluations to Success on Trap (bottom) and H-IFF (top) functions as a function of length and number of memes. Error bars represent 95% conficence intervals for mean, grouping within each length is (l to r) 10,50,100,200,400 memes.

- On average there is little difference in effectiveness or efficiency between the fixed (CTG) and adaptive (CTA) pivot rules.
- Adapting the pivot rules creates more reliably efficient methods - the 95% confidence intervals for the AES are smaller for the CTA than for the corresponding CTG algorithms.

- The algorithms with low numbers of memes ($\mu_m \in \{10, 50\}$) are less effective. This may well arise from premature convergence or loss of diversity in the meme population, which could be ameliorated by reducing the selection pressure or increasing the mutation rate.
- The variation in efficiency reduces as the number of memes is increased - for similar reasons to the previous observation.
- The algorithms with 200 memes are the most effective (highest overall SR, especially on H-IFF and MAX-SAT) whilst not being significantly less effective than the algorithms with 400 memes (AES values not significantly different with 95& confidence).

Of particular interest is the relationship between the time taken to solve problems, and their length. As can be seen there appears to be a linear trend in Figure 11.6 - although the logarithmic scale should be noted. This is most evident for the H-IFF functions where a wider range of lengths is used. Using the SPSS tool to fit a curve to results for CTA, pooling the results for 200 and 400 memes reveals that a relationship of the form $AES = 233.3 \cdot len^{1}.018$ accounts for over 80% of the variation in solution times.

11.8 Conclusions

This chapter describes a conceptual framework within which self-adaptive and co-evolutionary memetic systems can be examined. Starting with systems which self-adapt the choice of which meme to use from a fixed set, and then moving through self-adaptation of the meme behaviours to a full co-evolutionary system, experimental results show progressively enhanced problem-solving behaviour using a variety of mechanisms.

The extension to co-evolution showed that the credit assignment mechanisms is critical, and selection within the meme population can be affected by noise arising from a number of sources. Mechanisms such as the use of multiple partners, or memory have been examined. The most promising appears to be a decoupling of the two populations with fewer memes than solutions.

Along the way the meme definitions have become progressively richer - permitting wildcards, inversion, and length adaptation in the pattern matching, and adapting the choice of pivot function. The stage is now well prepared for the use of richer definitions such as GP-like functions, which may be application specific as used elsewhere e.g. evolving MAX-SAT solvers using primitive elements derived from other heuristics [288, 461].