Meme Fitness and Memepool Sizes in Coevolutionary Memetic Algorithms

Jim Smith

***Abstract*— This paper investigates the search efficiency of a class of adaptive memetic algorithms where the pivot function, depth, and definition of the local search operators are co-evolved alongside a population of potential solutions to the problem in hand. Such co-evolutionary mechanism requires a means for assigning meme fitness based in some way on the improvement they cause in solutions at a particular stage in the search process. We examine schemes based on both the extremal and mean improvement caused, and compare these to the implicit self-adaptive scheme. Simultaneously we examine the effect of using different fixed or adaptive pivot functions and depths of search. Results show that provided the fitness is correctly assigned the system successfully adapts the global/local search trade-off via evolution of the memes' search depth. The system is also able to adapt the optimal choice of greedy or steepest ascent. Unlike recent work on adaptive operator choice, results suggest that a fitness based on a meme's mean, rather than extremal affect provides more reliably effective optimisation results. Despite the close coupling between the two population, the self-adaptive schemes which use implicit fitness assignment are less successful than a well designed co-evolutionary scheme. Finally we examine the effect of changing the size of the meme pool and show that a surprisingly large number can be processed and benefit evolution.**

I. 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 - so-called Memetic Algorithms (MAs). 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 [1]. Taking inspiration from Dawkins' original concept of memes as evolving entities which influence the behaviour of individuals coded for by a population of genes, the COevolutionary Memetic Algorithms framework (COMA) was designed as a testbed for investigating a range of behaviours and effects. Starting with simple fixed length pattern-matching memes, and successively building in more complexity, experimental results have shown significant performance benefits over "fixed" MAs on a range of problems [2], [3], [4], [5], [6].

In this paper we address three outstanding issues:

- Increasing flexibility so that memes can evolve the pattern matching, pivot rule (greedy or steepest ascent) and depth of local search.
- Revisiting the issue of credit assignment in the light of recent results from the field of Adaptive Operator Selection, which have suggested that it may be better to assign fitness rewards based on extreme, rather than mean benefit caused.
- Evaluating different meme population sizes. Previous results showed that rewarding memes according to their effect on just one solution is too noisy which suggests there is a trade-off in terms of the number of memes, or learning strategies, that a population of solutions can effectively support and exploit. On one hand a small population is less diverse, but each meme can be evaluated at many different points in space. In contrast a larger population may be more diverse, but each meme is evaluated in the context of fewer solutions.

Jim Smith is with the Department of Computer Science, University of the West of England, Bristol, BS16 1QY, UK; (email james.smith@uwe.ac.uk)

II. Background

A. MAs with Multiple LS Operators

There are several recent examples of the use of multiple LS operators within evolutionary systems. Ong et al.[1] present an excellent recent review of work in the field of what they term "Adaptive Memetic Algorithms". This encompasses Krasnogor's "Multi-Memetic Algorithms" [7], [8], [9], [10], [11], Smith's COMA framework [2], [3], [4], [5], [6], Ong and Keane's "Meta-Lamarkian MAs [12], and Hyper-Heuristics [13], [14], [15], [16]. In another interesting related algorithm, Krasnogor and Gustafson's "Self-Generating MAs" use a grammar to specify for instance when local search takes place [17], [18]. 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. Ong's classification uses terminology developed elsewhere to describe adaptation of operators and parameters in Evolutionary Algorithms [19], [20], [21], [22]. This categorises algorithms according to the way that these decisions are made. One way ("static") is to use a fixed strategy. Another ("Adaptive") uses use feedback of which operators have provided the best improvement recently, 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).

B. Credit Assignment in 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. Bull [23] conducted a series of more general studies on co-operative co-evolution using Kauffman's static NKC model. In [24] 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 [25], with inconclusive results. This work has recently been revisited and extended by Wiegand et al. with very similar findings [26]. Wiegand's work also considered 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. "Punctuated Anytime Learning with samples" [27] is another recent approach to the pairing problem which uses periodic sampling to estimate fitness, but is more suited to cases where the populations evolve at different rates.

Our previous results using simple fitness improvement, or memory based schemes using variants of Paredis' "LifeTime Fitness Evaluation" [28], [29] were inconclusive [3], [6]. Results showed that simple co-evolutionary schemes suffered from too much noise depending on the solution they were partnered with, (especially with a greedy pivot), whereas the memory based systems did not adapt quickly enough. Despite inherent inefficiencies, the best results came from a scheme which used each meme with two different solutions, and vice versa, accepting only the best.

C. Credit Assignment in Adaptive Operator Selection

Since the beginnings of the field of Evolutionary Computation, the question of how to assign the probabilities of applying different operators, and the choice of associate parameters has been a subject of intense and ongoing interest. A wide range of different strategies have been proposed for adapting the operator probabilities in response to their perceived utility (the interested reader can find a recent review in [22]). There are two principal categories: self-adaptive schemes (where utility is implicitly assumed via association with fitter solutions that survive selection) and adaptive schemes that track the qualities of offspring produced by different operators and then recalculate probabilities periodically. The use of the intrinsic evolutionary processes to adapt mutation step sizes has long been used in Evolution Strategies [30], and Evolutionary Programming [31]. Similar approaches have been used to self-adapt mutation probabilities [32], [33] and recombination operators[34], [35] as well as more complex generating operators [36]. 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 [37].

Recent work in the area of adaptive operator selection by Schoenauer et al.[38], and Thierens [39], has divided the problem into two areas - first how to assign a "quality" metric to an operator that changes responsively over time, and second how to allocate probabilities to operators on that evolving basis.A major result emerging from this stream of work is that it appears beneficial to use extreme values - i.e. the maximum positive difference between offspring and parent fitness, rather than the mean value of the effect of an operator. This is in the spirit of rewarding operators that produce occasional large jumps in fitness rather than those which produce steady, but small, fitness improvements.

In COMA the "probability allocation" is dealt with by the action of selection in the meme population.

Clearly it is beneficial to evaluate memes in the context of more than one solution, and equally clearly this mechanism needs to be responsive to the current (rather than historical) state of the population of candidate solutions. Initial experiments (not shown for reasons of space) show that the former can be achieved by increasing the selection pressure in the meme population - by using tournaments of size 5. Based on the review above three possibilities can be identified for assigning meme fitness. The first is implicit i.e use the fitness of the attached solution. This does not necessarily imply self-adaptation, since the selection processes could be decoupled, but does imply the same-sized populations. The second is to record the effect of every time meme is applied, and use the mean improvement caused. This could be normalised by either the number of solutions to which it is applied ("usage") or by the total number of calls to the fitness evaluations. The third method is to use the maximum difference in fitness observed when a meme is applied to a candidate solution.

III. A Framework for Self-Adaption and Co-evolution of Memes and Genes

The pseudo-code in Figure 1 illustrates the algorithmic framework 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*$>$. The representation of the tuple elements leads naturally to the choice of evolutionary variation operators. The *Pivot* element is naturally binary. The *Depth* element is mapped as an integer, which permits shows the maximum number of iterations allowed. An arbitrarily large number is used to signify that search should always progress until a local optima is reached. The *Pairing* elements is one of $\{$*Self-Adaptive, Random, Fitness_Based*$\}$ and determines how memes are created and applied to solutions. As is illustrated in the pseudo-code, a range of behaviours from self-adaptive, through collaborative co-evolution to random meme drift can be obtained by fixing the elements, and selectively allowing mutation to operate on them creates various different adaptive schemes.

Note that for clarity we have omitted some of the parameters - for example $Recombine(parent1, parent2)$ is assumed to return a copy of the first parent with probability $1 - Px$ (where Px is the probability of applying crossover).

COevolving Memetic Algorithm for Binary Coded Problems :

```
Begin
  /* Given populations P of μ_s solutions and M of μ_m memes */
  initialise P and M randomly ;
  set generations = 0;
  set evaluations = 0;
  Repeat Until ( run_termination condition is satisfied )
  Do
    /* Create μ_s solution offspring and store parent ids */
    For i := 1 To i = μ_s Do
      set FirstParent[i] = Select_One_Parent(P);
      set SecondParent[i] = Select_One_Parent(P);
      set Offspring[i] = Recombine(FirstParent[i],SecondParent[i]);
      Mutate(Offspring[i]);
      set i = i +1;
    Od

    /* Create mu_m meme offspring according to pairing */
    For i := 1 To i = μ_m Do
      set Pairing = Get_Pairing(M,i);
      If (Pairing = SelfAdaptive) Then
        set MemeParent1[i] = FirstParent[i];
        set MemeParent2[i] = SecondParent[i];
        /* note this requires μ_m = μ_s. */
      Fi
      Else If (Pairing = Fitness_Based) Then
        set MemeParent1[i] = Select_One_Parent(M);
        set MemeParent2[i] = Select_One_Parent(M);
      Fi
      Else
        set MemeParent1[i] = RandInt(1,μ_m);
        set MemeParent2[i] = RandInt(1,μ_m);
      Esle
      set NewMemes[i] = Recombine(MemeParent1[i],MemeParent2[i]);
      Mutate(NewMemes[i]);
      set i = i+1;
    Od

    /* Apply local search to Offspring Using Memes */
    For i := 1 To i = μ_s Do
      set original_fitness = Get_Fitness(Offspring[i]);
      If (Pairing = SelfAdaptive) Then
        set meme = i;
      Fi
      Else
        set meme = Select_Random(NewMemes);
      Esle
      set Neighbours = Apply_Rule_To_Offspring(Offspring[i],NewMemes[meme]);
      Evaluate_Fitness(Neighbours);
      set Offspring[i] = Apply_Pivot_Rule(Neighbours);
      set Δfitness = Get_Fitness(Offspring[i]) - original_fitness;
      Update_Meme_Fitness(NewMemes[meme], Δfitness);
      set evaluations = evaluations +1 + |Neighbours|;
      set i = i +1;
    Od
    set P = Offspring;
    set M = NewMemes;
  Od
End.
```

Fig. 1. Pseudo-Code Definition of COMA algorithm

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 [40]. However while such richness tends to lead to complexity of expression suitable for practical applications, it can make it 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, 1111111000, 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.

IV. 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 (#) symbol which is intepreted as "don't care" when it appears in the *condition* part of the rule and as *invert* in the *action*. Each meme also contain an integer *rule_length* specifying the number of positions in the pattern string to consider, as well as the Pairing, Depth and Pivot elements. In [2] it was shown that mutation acting on *rule_length* permits successful evolution of rules with the appropriate lengths to capture structural dependencies in various different types of problems.

A. The Test Suite

The first set of problems used are composed of 16 subproblems of Deb's 4-bit fully deceptive function [41]. The fitness of each subproblem i is given by its unitation $u(i)$, that is the number of bits set to "one":

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

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 Royal Road function used is a simple R1 type with fitness rewards for groups of contiguous eight genes all set to 1. Watson's highly epistatic H-IFF function rewards matching pairs of adjacent bits in a solution s, i.e.

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

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 $l \in \{32, \ldots, 512, 1024\}$ 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 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 [42]. For lengths of 50 and 100 variables 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.

B. Experimental set-up and terminology

For the population of candidate solutions a generational genetic algorithm, with deterministic binary tournament selection for parents and no elitism was used. Population size μ_s was 400. One Point Crossover was applied with probability 0.7 followed by self-adaptive mutation using the scheme outlined in [43], [44], [45] . These choices were taken as "standard", and no attempt was made to tune them to the particular problems at hand.

Initially, and always for the self-adaptive variants, the size of the meme population was set to $\mu_m = 400$. As suggested in Figure /reffig:COMA the self-adaptive variants used as parent the meme that was previously associated with the relevant solution. The fitness-based variants used

binary tournaments based on meme fitness to implement $Select\_One\_Parent()$. No crossover was used in the meme population, so memes were produced by copying selected parents and then applying mutation to the rules with a allele-wise probability of 0.0625 - the inverse of the maximum rule length allowed to the adaptive version. If subject to mutation, the depth was flipped with probability 0.01. Rule lengths were randomly initialised in the range [1,16], and during mutation, with probability 0.01 a $N(0, 2)$ Gaussian deviate was added subject to staying in range. The depth of search was mutated in the same way if adaptive. The various variants of self- and co-adaptive algorithms that can be instantiated within this framework are denoted as *CAB-D-E* where *A* denotes the pairing and is one of *S* (Self-adaptive), or *T* (Tournament - variants of fitness based coevolution). *B* denotes the pivot function and is one of *G*reedy, *S*teepest or *A*daptive. *D* denotes the depth of search and is '1' , *L* (to local optima) or -*A*daptive. *E* denotes the reward function and is one of *M* (mean improvement per evaluation used), *U* (mean improvement per "raw" solution), or *X* (best improvement).

For each problem, 50 runs were made, each continuing until the global optimum was reached, subject to a maximum of 500,000 evaluations. For this paper we have focussed on the effectiveness of the search algorithm as measured by the Success Rate (SR) which is the number of runs finding the global optimum. 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.

V. Results

A. Calculation of meme fitness

Tables I and II shows the results of various fixed and adaptive co-evolutionary schemes with 400 memes, compared with the self-adaptive scheme, a simple GA, and four simple MAs. As can be seen all methods outperform the simple GA and MAs. Despite the large computational budget, on most of problems the steepest ascent is unsuccessful on the longer problems, since the neighbourhoods are potentially huge, upsetting the global/local search balance. Elsewhere [4] we have noted the opposite effect, but in each case the adaptive scheme is as good as, or nearly as good as, the better of the two pivot rules.

The self-adaptive schemes using implicit fitness assignment are less successful than the schemes which take explicit fitness gains into account, except on the Shifted Trap and SAT problems. One likely reason is that the original solution is considered part of the neighbourhood induced by a meme. Thus a meme can survive via association with a fit candidate solution, even if it no longer matches any positions in the candidate solution, or it does but its effect is always rejected as creating less fit solutions.

Interestingly the "Usage" based fitness evaluation is notably more successful than taking the number of fitness evaluations into account on the SAT problems - e.g for SAT-100, CTG-L-U succeeds 692 times vs. 186 for CTG-L-M and 180 for CTG-L-X. Since there is no structure to be exploited here, this suggests that methods that ignore the cost of unused evaluations - so being more prone to reward occasional "lucky" changes may be more successful at preserving meme diversity on these problems.

The Onemax and Royal Road (and to a lesser extent HIFF) functions were specifically chosen to require local and global search respectively. This is reflected in the better results for depth *L* vs. *1* on Onemax, and vice versa on the other problems. In both cases the results suggests that the adaptive depth mechanism, despite its simplicity, has been able to successfully balance the trade-off between local and global search, provided that information regarding the size of the neighbourhood searched is considered (i.e. *CTZ-A-M* rather than *CTZ-A-U*). The results also suggest that the mean improvement *CTZ-A-M* is slightly more effective than the extreme value *CTZ-A-X*.

Notably the Steepest ascent *CTS*- methods typically perform less well than the Greedy or Adaptive versions (*CTG-*, *CTA-*) as the size of the problem, and hence the potential neighbourhoods for local search increase. Despite this trend, the 10 thousand bit Onemax problems are still solved to optimality every time for adaptive depth *CTS-A-* - showing the strength of the robustness of meme adaptation.

Finally we note different between mean and extreme reward strategies is more evident for steepest ascent (see. e.g. the 1024 bit Royal Road and HIFF functions). This suggests that the steepest ascent may be more prone to noise from one particularly "lucky" combination of meme and solution, which will be more distinct with *X* than *M*.

B. Size of Memepool

Tables III and IV shows the effect of changing the mempool size for the *CTA–* coevolutionary algorithm with adaptive pivot and either mean or extreme fitness reward. The greedy and steepest ascent variants are omitted for reasons of space, but show similar patterns of results.

It might be expected that with small populations, where each meme is evaluated in the context of multiple solutions, and extreme value might be needed to provide sufficient information for selection. In practice, this difference is nor observed - again where there are differences the *M* strategy slightly outperforms *X*.

However two distinct trends can be observed. On the SAT, Trap and OneMax functions the success rate increases as the number of memes is increased. However on the Shifted-trap, HIFF and Royal Road problems the opposite trend is observed - performance on these is worse with 400 memes than with fewer.

VI. Conclusions

This paper set out to answer three questions: (i) do simple co-evolutionary models provide enough information to adapt

TABLE II

NUMBER OF SUCCESSFUL RUNS (OUT OF 50, 1250 FOR SAT) WITH DIFFERENT CREDIT MECHANISMS ON DIFFERENT FUNCTIONS, 400 MEMES

Algorithm	Royal Road				HIFF						OneMax					SAT	
	64	256	512	1024	32	64	128	256	512	1024	500	1000	2500	5k	10k	50	100
CSA-1	49	6	0	0	50	50	33	4	0	0	0	0	0	0	0	768	106
CSA-A	50	4	1	0	50	50	50	36	3	0	50	50	50	50	0	1121	376
CSA-L	49	5	0	0	50	50	50	28	1	0	50	50	50	50	0	1145	380
CSG-1	50	4	0	0	50	46	35	15	1	0	22	0	0	0	0	669	94
CSG-A	50	14	2	0	50	50	50	49	21	4	50	50	50	50	0	1134	573
CSG-L	50	13	0	0	50	50	50	49	20	1	50	50	50	50	0	1162	589
CSS-1	50	0	0	0	50	47	34	4	0	0	0	0	0	0	0	782	103
CSS-A	49	13	0	0	50	50	50	10	0	0	50	50	50	50	0	1120	214
CSS-L	50	3	0	0	50	50	50	11	0	0	50	50	50	50	0	1128	194
CTA-1-M	50	50	50	25	50	49	47	46	39	34	50	50	2	0	0	747	120
CTA-1-U	50	29	0	0	50	49	48	29	0	0	7	0	0	0	0	866	122
CTA-1-X	50	49	46	18	50	47	35	36	27	22	50	46	0	0	0	694	93
CTA-A-M	50	49	48	13	50	50	50	49	46	1	50	50	50	50	50	774	105
CTA-A-U	50	34	1	0	50	50	50	14	4	0	50	50	50	50	50	1188	385
CTA-A-X	50	49	47	8	50	50	48	43	42	0	50	50	50	50	50	692	78
CTA-L-M	50	50	49	6	50	50	50	50	6	2	50	50	50	50	50	905	183
CTA-L-U	50	38	0	0	50	50	50	13	0	2	50	50	50	50	50	1196	355
CTA-L-X	50	50	48	1	50	50	50	50	8	2	50	50	50	50	50	928	202
CTG-1-M	50	50	50	43	50	50	46	46	45	41	50	50	4	0	0	773	110
CTG-1-U	50	42	0	0	50	50	47	45	42	38	50	47	0	0	0	784	117
CTG-1-X	50	48	48	30	50	49	39	38	28	32	50	47	0	0	0	689	98
CTG-A-M	50	50	49	23	50	50	50	49	45	39	50	50	50	50	50	747	106
CTG-A-U	50	40	8	0	50	50	50	50	29	5	50	50	50	50	50	1182	668
CTG-A-X	49	50	46	18	50	50	49	46	47	37	50	50	50	50	50	679	75
CTG-L-M	50	50	50	11	50	50	50	50	50	39	50	50	50	50	50	907	186
CTG-L-U	50	45	9	0	50	50	50	50	30	2	50	50	50	50	50	1184	692
CTG-L-X	50	50	48	3	50	50	50	50	50	44	50	50	50	50	50	931	180
CTS-1-M	50	50	50	40	50	50	43	43	37	28	50	50	0	0	0	646	93
CTS-1-U	50	16	0	0	50	49	48	33	1	0	1	0	0	0	0	882	132
CTS-1-X	50	48	41	9	50	47	39	24	11	9	49	29	0	0	0	618	93
CTS-A-M	50	50	50	15	50	50	48	46	6	2	50	50	50	50	50	642	71
CTS-A-U	50	26	1	0	50	50	50	2	0	2	50	50	50	50	50	1174	162
CTS-A-X	50	50	43	2	50	50	46	39	2	2	50	50	50	50	50	667	90
CTS-L-M	50	50	49	1	50	50	50	50	1	1	50	50	50	50	50	819	137
CTS-L-U	50	28	2	0	50	50	50	4	1	1	50	50	50	50	50	1198	158
CTS-L-X	50	50	46	0	50	50	50	46	1	1	50	50	50	50	50	797	125
GA	15	0	0	0	33	2	0	0	0	0	50	34	0	0	0	221	20
SMA-G-1	50	0	0	0	49	24	1	0	0	0	50	50	0	0	0	784	87
SMA-G-L	50	0	0	0	50	32	0	0	0	0	50	50	50	50	50	1219	737
SMA-S-1	50	0	0	0	50	39	0	0	0	0	0	0	0	0	0	869	45
SMA-S-L	50	0	0	0	50	47	0	0	0	0	50	50	50	50	50	1225	52

the local-global search trade-off via the the encoded depth of local search; (ii) is extreme-valued reward preferable to mean improvement, and (iii) is a small population of memes needed to provide sufficiently accurate evaluation of a meme's value at multiple points in the search space.

While the results strongly support a positive answer to the first question, and to a lesser extend a negative answer to the second, the third question is less conclusively answered. On some problems the use of the mean improvement-based fitness clearly supports a larger and more diverse population of memes, and this is reflected in improved success rates. On others the pattern is different, and this does not appear to be directly related to the number of local optima (viz. OneMax in the first group and Trap in the second) or the "richness" of the search space- for example neither the 50-variable SAT problems nor the Shifted-Trap possess repeating exploitable structures but e.g. the *CTA-A-* algorithms perform better on the former and worse on the latter when the meme pool is increased from 200 to 400 memes. Clearly this requires further investigation.

REFERENCES

[1] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Transactions on Systems Man and Cybernetics Part B*, vol. 36, no. 1, 2006.

[2] J. Smith, "Co-evolution of memetic algorithms: Initial investigations," in *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, J. M. Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, Eds., no. 2439. Springer, Berlin, Heidelberg, New York, 2002, pp. 537–548.

[3] ——, "Protein structure prediction with co-evolving memetic algorithms," in *2003 Congress on Evolutionary Computation (CEC'2003)*. IEEE Press, Piscataway, NJ, 2003, pp. 2346–2353.

[4] ——, "The co-evolution of memetic algorithms for protein structure prediction," in *Recent Advances in Memetic Algorithms*, W. Hart, N. Krasnogor, and J. Smith, Eds. Springer, Berlin, Heidelberg, New York, 2004, pp. 105–128.

[5] ——, "Co-evolving memetic algorithms: A review and progress report," *IEEE Transactions in Systems, Man and Cybernetics, part B*, vol. 37, no. 1, pp. 6–17, 2007.

[6] ——, "Credit assignment in adaptive memetic algorithms," in *Proceedings of Gecco, the ACM-SIGEVO conference on Evolutionary Computation*, 2007, pp. 1412–1419.

TABLE IV

NUMBER OF SUCCESSFUL RUNS (OUT OF 50, 1250 FOR SAT) AS A FUNCTION OF MEMEPOOL SIZE, DEPTH AND CREDIT FUNCTIONS

Algorithm	Royal Road			
	64	256	512	1024
CTA-1-M-50	50	40	39	30
CTA-1-M-100	50	48	46	39
CTA-1-M-200	50	49	49	41
CTA-1-M-400	50	50	50	25
CTA-1-X-50	50	37	24	16
CTA-1-X-100	50	47	44	20
CTA-1-X-200	50	49	45	34
CTA-1-X-400	50	49	46	18
CTA-A-M-50	50	46	33	31
CTA-A-M-100	50	44	44	40
CTA-A-M-200	50	48	48	35
CTA-A-M-400	50	49	48	13
CTA-A-X-50	50	37	18	6
CTA-A-X-100	49	44	33	23
CTA-A-X-200	50	49	47	20
CTA-A-X-400	50	49	47	8
CTA-L-M-50	50	45	34	29
CTA-L-M-100	50	48	48	36
CTA-L-M-200	50	50	49	33
CTA-L-M-400	50	50	49	6
CTA-L-X-50	50	44	30	10
CTA-L-X-100	50	47	44	24
CTA-L-X-200	50	49	47	23
CTA-L-X-400	50	50	48	1

HIFF				
64	128	256	512	1024
46	45	40	30	30
50	48	47	38	40
50	48	41	37	36
49	47	46	39	34
44	43	41	26	19
49	42	39	35	26
49	46	38	38	39
47	35	36	27	22
47	37	39	23	8
50	45	41	33	6
50	47	46	34	4
50	50	49	46	1
49	44	40	29	6
49	45	35	23	6
50	48	45	32	5
50	48	43	42	0
49	50	38	16	4
50	50	45	19	1
50	50	50	13	2
50	50	50	6	2
50	47	43	11	3
50	49	46	14	1
50	50	50	10	2
50	50	50	8	2

OneMax				
500	1000	2500	5000	10000
50	48	1	0	0
50	50	1	0	0
50	50	3	0	0
50	50	2	0	0
38	15	0	0	0
45	37	0	0	0
49	47	0	0	0
50	46	0	0	0
50	46	37	33	33
50	49	43	48	41
50	50	50	50	49
50	50	50	50	50
48	40	38	34	34
50	48	43	48	41
50	50	50	50	49
50	50	50	50	50
50	47	47	45	47
50	50	48	49	49
50	50	50	50	50
50	50	50	50	50
49	46	47	45	47
50	50	48	49	49
50	50	50	50	50
50	50	50	50	50

MAXSAT	
50	100
494	52
610	82
715	107
747	120
579	61
623	100
685	103
694	93
514	32
633	76
726	92
774	105
559	66
642	86
694	85
692	78
476	45
568	68
764	119
905	183
549	65
567	78
742	109
928	202

[7] N. Krasnogor, "Coevolution of genes and memes in memetic algorithms," in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, A. Wu, Ed., 1999.

[8] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Morgan Kaufmann, San Francisco, 2000, pp. 987–994.

[9] ——, "Emergence of profitable search strategies based on a simple inheritance mechanism," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. Morgan Kaufmann, San Francisco, 2001, pp. 432–439.

[10] N. Krasnogor, "Studies in the theory and design space of memetic algorithms," Ph.D. dissertation, University of the West of England, 2002.

[11] N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst, "Multimeme algorithms for protein structure prediction," in *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, J. M. Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, Eds., no. 2439. Springer, Berlin, Heidelberg, New York, 2002, pp. 769–778.

[12] Y. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[13] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," *Lecture Notes in Computer Science*, vol. 2079, pp. 176–95, 2001.

[14] E. Burke and A. Smith, "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 122–128, 2000.

[15] G. Kendall, P. Cowling, and E. Soubeiga, "Choice function and random hyperheuristics," in *Proceedings of Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL)*, 2002, pp. 667–671.

[16] E. Burke, G. Kendall, and E. Soubeiga, "A tabu search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, 2003.

[17] N. Krasnogor, "Self-generating metaheuristics in bioinformatics: The protein structure comparison case," *Genetic Programming and Evolvable Machines. Kluwer academic Publishers*, vol. 5, no. 2, pp. 181–201, 2004.

[18] N. Krasnogor and S. Gustafson, "A study on the use of "self-generation" in memetic algorithms," *Natural Computing*, vol. 3, no. 1, pp. 53–76, 2004.

[19] J. Smith and T. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, vol. 1, no. 2, pp. 81–87, 1997.

[20] R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: A survey," in *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1997.

[21] A. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[22] A. Eiben, Z. Michalewicz, M. Schoenauer, and J. Smith, "Parameter Control in Evolutionary Algorithms," in *Parameter Setting in Evolutionary Algorithms*, ser. Studies in Computational Intelligence, Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz, Eds. Springer Verlag, vol. 54, pp. 19–46, 2007.

[23] L. Bull, "Artificial symbiology," Ph.D. dissertation, University of the West of England, 1995.

[24] L. Bull and T. Fogarty, "Horizontal gene transfer in endosymbiosis," in *Proceedings of the 5th International Workshop on Artificial Life : Synthesis and Simulation of Living Systems (ALIFE-96)*, C. Langton and K.Shimohara, Eds. MIT Press, Cambridge, MA, 1997, pp. 77–84.

[25] L. Bull, "Evolutionary computing in multi agent environments: Partners," in *Proceedings of the 7th International Conference on Genetic Algorithms*, T. Bäck, Ed. Morgan Kaufmann, San Francisco, 1997, pp. 370–377.

[26] R. Wiegand, W. Liles, and K. D. Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. Morgan Kaufmann, San Francisco, 2001, pp. 1235–1245.

[27] G. Parker and H. Blumenthal, "Varying sample sizes for the coevolution of heterogeneous agents," in *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*. IEEE Press, 2004, pp. 766–771.

[28] J. Paredis, "The symbiotic evolution of solutions and their representations," in *Proceedings of the 6th International Conference on Genetic Algorithms*, L. Eshelman, Ed. Morgan Kaufmann, San Francisco, 1995, pp. 359–365.

TABLE I

NUMBER OF SUCCESSFUL RUNS (OUT OF 50) WITH DIFFERENT CREDIT MECHANISMS ON TRAP FUNCTIONS, 400 MEMES

	4Trap - length						Dist	Shift
Algorithm	40	80	120	160	200	400	-Trap	-Trap
CSA-1	50	50	47	42	25	0	50	27
CSA-A	50	47	41	37	26	5	49	35
CSA-L	50	50	43	41	27	5	50	27
CSG-1	50	50	49	48	38	6	50	25
CSG-A	50	50	48	48	38	24	50	48
CSG-L	50	50	48	45	47	28	50	48
CSS-1	50	50	47	33	19	0	50	24
CSS-A	50	48	34	25	25	1	50	19
CSS-L	50	49	43	34	21	0	50	29
CTA-1	50	50	50	50	50	50	50	14
CTA-1-U	50	50	50	50	48	5	50	2
CTA-1-X	50	50	50	50	50	50	50	3
CTA-A-M	50	50	50	50	50	50	50	0
CTA-A-U	50	48	40	40	30	5	50	0
CTA-A-X	50	50	50	50	50	50	50	0
CTA-L-M	50	50	50	50	50	46	50	0
CTA-L-U	50	50	48	44	37	13	50	0
CTA-L-X	50	50	50	50	50	44	50	1
CTG-1-M	50	50	50	50	50	50	50	15
CTG-1-U	50	50	49	39	34	15	50	3
CTG-1-X	50	50	50	50	50	50	50	5
CTG-A-M	50	50	50	50	50	50	50	0
CTG-A-U	50	48	49	47	45	39	50	1
CTG-A-X	50	50	50	50	50	50	50	0
CTG-L-M	50	50	50	50	50	49	50	1
CTG-L-U	50	50	48	48	47	38	50	0
CTG-L-X	50	50	50	50	50	50	50	0
CTS-1-M	50	50	50	50	50	50	50	16
CTS-1-U	50	50	50	49	49	9	50	0
CTS-1-X	50	50	50	50	50	48	50	4
CTS-A-M	50	50	50	50	50	43	50	2
CTS-A-U	50	45	42	26	16	3	50	0
CTS-A-X	50	50	50	50	50	36	50	0
CTS-L-M	50	50	50	50	50	5	50	4
CTS-L-U	50	49	43	30	18	0	50	0
CTS-L-X	50	50	50	50	50	1	50	1
GA	34	0	0	0	0	0	9	0
SMA-G-1	40	7	0	0	0	0	11	0
SMA-G-L	39	0	0	0	0	0	1	0
SMA-S-1	50	44	0	0	0	0	49	0
SMA-S-L	50	0	0	0	0	0	4	0

TABLE III

NUMBER OF SUCCESSFUL RUNS (OUT OF 50) WITH DIFFERENT MEME POOL SIZES ON TRAP FUNCTIONS

	4Trap						Dist	Shift
Algorithm	40	80	120	160	200	400	-Trap	-Trap
CTA-1-M-50	49	44	47	42	46	33	49	3
CTA-1-M-100	50	50	48	49	49	47	49	9
CTA-1-M-200	50	50	50	50	50	49	50	8
CTA-1-M-400	50	50	50	50	50	50	50	14
CTA-1-X-50	50	49	44	46	46	33	49	17
CTA-1-X-100	50	50	50	48	47	46	50	17
CTA-1-X-200	50	50	50	49	50	49	50	23
CTA-1-X-400	50	50	50	50	50	50	50	3
CTA-A-M-50	49	47	45	46	45	34	48	4
CTA-A-M-100	50	49	49	47	47	38	49	2
CTA-A-M-200	50	48	50	49	49	47	50	0
CTA-A-M-400	50	50	50	50	50	50	50	0
CTA-A-X-50	49	48	37	43	43	29	49	7
CTA-A-X-100	50	50	47	46	46	38	50	7
CTA-A-X-200	50	50	50	49	47	46	50	4
CTA-A-X-400	50	50	50	50	50	50	50	0
CTA-L-M-50	50	45	46	44	37	26	50	1
CTA-L-M-100	50	48	46	43	47	35	49	3
CTA-L-M-200	50	50	48	48	48	43	50	0
CTA-L-M-400	50	50	50	50	50	46	50	0
CTA-L-X-50	49	47	48	47	41	24	49	12
CTA-L-X-100	50	50	49	45	41	33	50	5
CTA-L-X-200	50	50	50	48	49	37	50	1
CTA-L-X-400	50	50	50	50	50	44	50	1

[29] ——, "Coevolutionary algorithms," in *Handbook of Evolutionary Computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1998.

[30] H.-P. Schwefel, *Numerical Optimisation of Computer Models*. Wiley, New York, 1981.

[31] D. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, University of California, 1992.

[32] T. Bäck, "Self adaptation in genetic algorithms," in *Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, F. Varela and P. Bourgine, Eds. MIT Press, Cambridge, MA, 1992, pp. 263–271.

[33] J. Smith and T. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996, pp. 318–323.

[34] J. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*, J. Grefenstette, Ed. Lawrence Erlbaum, Hillsdale, New Jersey, 1987, pp. 36–40.

[35] J. Smith and T. Fogarty, "Recombination strategy adaptation via evolution of gene linkage," in *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996, pp. 826–831.

[36] ——, "Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm," in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., no. 1141. Springer, Berlin, Heidelberg, New York, 1996, pp. 441–450.

[37] M. Serpell and J. Smith, "Self-adaption of mutation operator and probability for permutation representations in genetic algorithms," *Evolutionary Computation*, vol. in press, 2010.

[38] J. Maturana, Á. Fialho, F. Saubion, M. Schoenauer, and M. Sebag, "Extreme Compass and Dynamic Multi-Armed Bandits for Adaptive Operator Selection," in *IEEE Congress on Evolutionary Computation*, Trondheim Norvège, 2009.

[39] D. Thierens, "Adaptive strategies for operator allocation," in *Parameter Setting in Evolutionary Algorithms*, F. Lobo, C. Lima, and Z. Michalewicz, Eds. Springer, Berlin, 2007, pp. 77–90.

[40] A. Fukunaga, "Automated discovery of local search heuristics for satisfiability testing," *Evolutionary Computation*, vol. 16, no. 1, pp. 31–61, 2008.

[41] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1997.

[42] "Satlib: http://www.satlib.org."

[43] C. Stone and J. Smith, "Strategy parameter variety in self-adaption," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, W. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, Eds. Morgan Kaufmann, San Francisco, 9-13 Jul. 2002, pp. 586–593.

[44] J. Smith, "Modelling GAs with self-adaptive mutation rates," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. Morgan Kaufmann, San Francisco, 2001, pp. 599–606.

[45] ——, "Parameter perturbation mechanisms in binary coded gas with self-adaptive mutation," in *Foundations of Genetic Algorithms 7*, Rowe, Poli, DeJong, and Cotta, Eds. Morgan Kaufmann, San Francisco, 2003, pp. 329–346.