Credit Assignment in Adaptive Memetic Algorithms

J.E. Smith
School of Computer Science
University of the West of England
Bristol, BS16 1QY, UK
james.smith@uwe.ac.uk

ABSTRACT

Adaptive Memetic Algorithms couple an evolutionary algorithm with a number of local search heuristics for improving the evolving solutions. They are part of a broad family of meta-heuristics which maintain a set of local search operators applying them at different stages of the search. This creates a need to make decisions about which operator to use when. Several different schemes have been proposed, but most of them assume there is a fixed set of predefined operators. This makes them unsuitable for use within the broader context of adaptive learning systems where the set of available operators can change over time. Here we investigate a range of different schemes, and propose a novel method for estimating an operator's current utility, which is shown to avoid some of the problems of noise inherent in simpler schemes. Results on a range of combinatorial optimisation problems show that algorithms embodying this mechanism locate the global optimum more reliably, without a significant computational overhead compared to the simpler schemes.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and SearchHeuristic Methods

General Terms

Algorithms,Performance,Reliability

Keywords

Memetic algorithms, co-evolution, local search, adaptivity

1. INTRODUCTION

One of the more successful recent trends in heuristic optimisation has been the increasing focus on what Ong et. al. have termed Adaptive Memetic Algorithms [29]. These couple an evolutionary algorithm with a number of local search (LS) heuristics for improving the quality of the evolved solutions. They are part of a broad family of meta-heuristics which maintain a number of LS operators and use different ones at different stages of the search. Other well-known members of this class include Hyper-Heuristics [11], and Variable Neighbourhood Search [14].

Common to all of these meta-heuristics is the need to make decisions about which LS operator to use at any given time. A number of different schemes have been proposed, and the recent survey by Ong et. al. provides a taxonomy for their classification along with some initial comparisons in the field of continuous optimisation [29]. However many of the proposed schemes assume the use of a fixed set of LS operators - for example one method used in Hyper-Heuristics increases the likelihood of using a LS method not only according to its recent perceived utility, but also according to the length of time since it was last used. This, and similar methods are clearly not directly suitable for use within the broader context of adaptive learning systems as exemplified by a number of recent algorithms where the set of LS operators, and their definitions, can change over time e.g. [36, 40, 22].

This paper investigates a range of different strategies for choosing and rewarding adaptive LS operators. It does so in the co-evolutionary context of the Coevolution Of Memetic Algorithms (COMA) framework [36, 37, 40], and so is able to leverage a body of related research in collaborative co-evolution. A novel method is proposed for obtaining more reliable estimates of a LS operator's utility at a given stage of the search process, and it is demonstrated that this avoids some of the problems of noise inherent in simpler schemes. The resultant algorithms are shown to demonstrate improved reliability of locating the global optimum, while not incurring significant computational overheads compared to the simpler schemes, or to "static" memetic algorithms, on a range of test problems. The rest of this paper proceeds as follows. Section 2 provides a rationale for this research within the context of evolutionary algorithms. It then describes various meta-heuristics proposed within this framework and places this work within the more general context of studies into adaptation, development and learning. Section 3 describes the specific model used to investigate different credit assignment strategies, and briefly reviews relevant results from previous papers. Section 4 describes the experimental set-up and the choice of performance metrics for comparison. Section 5 details and discusses the main experimental results. Finally, Section 6 discusses the implications of these results, draws conclusions and suggests future work.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

2. BACKGROUND

2.1 What is a Memetic Algorithm?

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 [26] for a review,and [15] 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 [26].

In recent years it has been increasingly recognised that the particular 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. 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 and propose 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.

2.2 MAs with Multiple LS Operators

There are several recent examples of the use of multiple LS operators within evolutionary systems. Ong et al.[29] present an excellent recent review of work in the field of what they term "Adaptive Memetic Algorithms". This encompasses Krasnogor's "Multi-Memetic Algorithms" [20, 24, 25, 21], Smith's COMA framework [36, 38, 39, 40], Ong and Keane's "Meta-Lamarkian MAs [28], and Hyper-Heuristics [11, 9, 19, 8]. In another interesting related algorithm, Krasnogor and Gustafson's "Self-Generating MAs" use a grammar to specify for instance when local search takes place [22, 23]. 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 [14], 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.

The classification of Ong et al. uses the terminology developed elsewhere to describe adaptation of operators and parameters in Evolutionary Algorithms [44, 16, 12]. They categorise algorithms according to the way that these decisions are made. One way is to use a fixed strategy, this is termed "static". 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). In this terminology, the COMA algorithm may be local-Adaptive or Self-Adaptive. In this paper we will adopt this terminology, and also make use of the general term "meme" to denote an object specifying a particular local search strategy.

2.3 Self-Adaptation and Co-Evolution

As noted above, we are concerned with meta-heuristics which maintain two sets of objects - one of genes and one of memes. If we consider these to be 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 [2]) then this situation 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 [35], and Evolutionary Programming [13]. Similar approaches have been used to self-adapt mutation probabilities [3, 43] and recombination operators[34, 42] in genetic algorithms as well as more complex generating operators which combined both mutation and recombination [41].

If selection is performed separately for the two populations then we have a co-operative co-evolutionary system, where the fitness of the members of the meme population is assigned as some function of the relative improvement they cause in the "solution" population. Following initial work by Husbands and Mill [17] the metaphor of co-operative (or collaborative) co-evolution has gained increasing interest. Paredis has examined the co-evolution of solutions and their representations [30]. Potter and DeJong have also used co-operative co-evolution of partial solutions in situations where an obvious problem decomposition was available [33]. Both reported good results. Bull [5] conducted a series of more general studies on co-operative co-evolution using Kauffman's static NKC model. In [7] 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. It was shown that linkage of the two species chromosomes was preferred in highly interdependent situations. In the context we are using here, the idea of linked population 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 [6]. 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 [46]. 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" [32] 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.

There has also been a large body of research into competitive co-evolution (see [31] 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 [27] 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.

3. EVOLVING MEMES

This section describes the COMA framework chosen for these investigations, which provides a broad general model within which many different schemes can be implemented. The framework maintains two populations: one of genes encoding for candidate solutions, and one of memes encoding for LS operators to be used within the MA. Local Search is considered to be specified by three components: the *pivot* function (e.g. steepest or greedy ascent), the depth (number of iterations applied), and the choice of neighbourhood generating function. This latter defines a set of candidate solutions $n(i)$ that can be reached by the application of some move operator to the solution i. The memes are encoded as tuples of the form *<Depth, Pivot_Rule, Pairing, Move>*, where the first three elementrs take natural integer representations. The move operators are represented as *condition:action* pairs, specifying a pattern to be looked for in the problem representation, and a different pattern it should be changed to.

Most relevant is the element *Pairing*, which indicates how the choice of which members of the two populations to combine for evaluation is managed. This element allows the system to be specified to follow the extremes of a fully unlinked system, in which although still interacting the two populations co-evolve separately, and a fully linked, self-adapting system. Here the memes are effectively extra genetic material inherited and varied along with the problem representation. The full algorithm is described in pseudo-code in Figure 1, and as is illustrated in the *If..Else* section the full range of LS selection schemes can be achieved by encoding the pairing as a value taken from the set {*Linked, Random, Fitness_Based*}.

This framework, along with a road-map for exploring various issues and possibilities was initially proposed in [36] and these have been explored in a series of papers [37, 38, 39, 40]. The reported results were significantly better than the static MAs tested on a wide range of combinatorial optimisation problems both in terms of efficiency and effectiveness. However, to date investigation of the effects of different pairing strategies was less conclusive, with the reported results showing that the self-adaptive strategies usually, but not always, out-performed those using fitness-based linkage. This provided an extra incentive for the use of the COMA framework within this research.

4. EXPERIMENTAL SET-UP

4.1 Test Functions

A range of test functions were chosen to embody different problem characteristics. The first three were based around a 4-bit deceptive "Trap" function introduced by Deb [4]:

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

where $u(i)$ is the unitation of a substring i.

For the first test problem (4-Trap), instances of this were concatenated to form a problem with variable lengths in the range $\{20, 40, \ldots, 200\}$. Early experiments [37] showed that on this problem, the COMA system is able to first identify rules of the form $\#\#\#\# \rightarrow 1111$, and then exploit them through repeated application. The use of this problem is thus intended to test the ability of the different forms of credit assignment to identify and then maintain such rules in the meme population.

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" for the COMA algorithm in a different way 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. Analysis of previous results [37] suggests that to solve both of these problems it is necessary to maintain a diverse rule-set in the meme population - a different test for the credit assignment mechanism.

The fourth test function was Watson's highly epistatic H-IFF function. At the bottom level, fitness is awarded to matching pairs of adjacent bits in a solution s, i.e.

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

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 the same factor. This problem has a number of Hamming sub-optima, and two global optima corresponding to the $u(i) \in \{0, 1\}$. Problem sizes $l \in \{8, 16, \ldots, 512\}$ were used, corresponding to 3 to 9 levels. In this case both the rules $\# \rightarrow 1$ and $\# \rightarrow 0$ will yield improvements at the lower levels, and in order to solve the problem the credit assignment mechanism must filter out these lower level effects to permit the population to "specialise" and approach one of the global optima.

The final test function was Max-SAT problem, a classic from the field of combinatorial optimisation, which consists 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 [1]. 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) with approximately 4.3 clauses per variable.

4.2 Performance Metrics

On the Shifted-Trap and Dist-Trap each algorithm was

```
COevolving Memetic Algorithm for Binary Coded Problems :
Begin

  /* given starting populations of solutions (P) and memes (M) both of size μ */
  initialise P with randomly selected binary genes;
  initialise M with randomly selected memes;
  set generations = 0;
  set evaluations = 0;

  Repeat Until ( run_termination condition is satisfied )
  Do

    /* Create μ offspring in each population */
    For child := 1 To child = μ Do
      /* Create offspring by selection, recombination and mutation, storing the parents id's */
      set Parent1[child] = Select_One_Parent(P);
      set Parent2[child] = Select_One_Parent(P);
      set Offspring[child] = Recombine(Parent1[child],Parent2[child]);
      Mutate(Offspring[child]);

      /* Select parents of the new meme */
      set Pairing = Get_Pairing(M,child);
      If (Pairing = Linked) Then
        set MemeParent1[child] = Parent1[child];
        set MemeParent2[child] = Parent2[child];
      Fi
      Else If (Pairing = Fitness_Based) Then
        set MemeParent1[child] = Select_One_Parent(M);
        set MemeParent2[child] = Select_One_Parent(M);
      Fi
      Else
        set MemeParent1[child] = RandInt(1,μ);
        set MemeParent2[child] = RandInt(1,μ);
      Esle

      /* Create new meme from these parents using recombination and mutation */
      set NewMemes[child] = Recombine(MemeParent1[child],MemeParent2[child]);
      Mutate(NewMemes[child]);

      /* Finally apply local search to Offspring Using Memes */
      set original_fitness = Get_Fitness(Offspring[child]);
      /* Applying the rule part of a meme to an offspring produces a set of neighbours */
      set Neighbours = Apply_Rule_To_Offspring(Offspring[child],NewMemes[child]);
      Evaluate_Fitness(Neighbours);
      /* Pivot rule of meme determines choice of neighbour */
      set Offspring[child] = Apply_Pivot_Rule_To_Neighbours(Neighbours,NewMemes[child]);
      /* Finally update meme fitness according to increase in solution fitness */
      set Δfitness = Get_Fitness(Offspring[child]) - original_fitness;
      Update_Meme_Fitness(NewMemes[child], Δfitness);
      set child = child +1;
    Od

    set evaluations = evaluations +1 + |Neighbours|;
    set P = Offspring;
    set M = NewMemes;
  Od
End.
```

Figure 1: Pseudo-Code Definition of COMA algorithm

run 10 times. For the 4Trap and H-IFF functions this was repeated for each length used. For the SAT problems, each algorithm was run ten times on each instance, giving 250 runs for each combination of algorithm and length. Each run was continued until the global optimum was reached, subject to a maximum of 500,00 evaluations. Two performance metrics were considered. To measure effectiveness we used the Success Rate (SR) which is the number of runs finding the global optimum. To measure efficiency we used 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 LS may involve several evaluations, this allows

more generations to the GA, i.e. algorithms are compared strictly on the basis of the number solutions evaluated.

4.3 COMA set-up

A generational genetic algorithm, with deterministic binary tournament selection for parents and no elitism was used. 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" from the literature, and no attempt was made to tune them to the particular problems at hand. Mutation was applied to the rules with a probability of 0.0625 of selecting a new allele value in each locus (the inverse of the maximum rule length allowed to the adaptive version). Each meme used a greedy ascent pivot rule for one step of hill-climbing.

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 MA using a neighbourhood of Hamming distance one, with one iteration of greedy ascent (SMA).
- COMA with "global-random" pairing (**CR**) achieved by making a random choice of parents for the meme created to use with each solution. Effectively this scheme equates to having a uniform credit assignment mechanism which removes selection pressure in the meme population.
- COMA with "local-adaptive" pairing achieved by the use of self-adaptation (**CS**). There is implicit credit assignment via the association of good memes with good solutions they help to create.
- COMA with "global-adaptive" pairing of memes and solutions. In this case the credit assignment mechanism sets the fitness of each meme to be the improvement it causes when applied to a solution. Memes are then selected to be parents using a simple binary tournament, hence this scheme is denoted **CT**. Note that in this case there is no "memory", so 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) [31] this uses a time-decaying fitness function of the form:

$$meme\_fitness' = meme\_fitness \cdot decay\_factor + improvement\_caused \quad (3)$$

  In the case that a meme is newly created, it takes the average of its parent's fitnesses, and a $decay\_factor$ of 0.5 was used.
- Early results [37] indicated that the uses of greedy ascent could be a source of noise that confused the credit assignment mechanism. This ties in with finding that there is no single best way of choosing a partner for evaluation in general co-evolutionary schemes (see e.g. [5, 6, 46]). However there is also a computational overhead every time a meme is applied to a solution. The compromise scheme tested involves a minor 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.

5. RESULTS

5.1 Reliability

Table 1 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) and a simple Genetic Algorithm (GA). The global-random scheme (CR) shows lower Success Rates than the other COMA algorithms on most problems. The local-adaptive scheme (CS) also has lower success rates on the longer H-IFF problems and the SAT problems that the global-adaptive variants.

Comparing the four different global-adaptive schemes, no single clear pattern emerges:

- On the 4-trap functions, the simpler schemes based on a single pairing (CT, CTD) find the optimum slightly more often for the longer instances.
- On the Shifted-Trap and Dist-Trap functions the performances are the same (100% Success), except that the CT algorithm was only successful 9 times on Dist-Trap.
- In contrast, on the H-IFF and SAT problems the schemes that use credit-assignment based on a collaboration poolsize of 2 are more successful, the averaging version (CT2M) especially so. Notably the CTD scheme with memory and a collaboration poolsize of 1 is markedly less successful than the others on the SAT functions.
- Taking the results for all functions into account, the CT2M algorithm has the highest success rate.

5.2 Efficiency

Figure 2 illustrates 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, SMA and CR are omitted from the first two for the sake of clarity as they are so poor. As can be seen, on the Trap functions the results with collaboration poolsize 1 (CT, CTD, CS) are obtained faster than with the poolsize of 2 (CT2M, CT2B). 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. However on the H-IFF function the CT approach is not only less successful than the CTD 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

Table 1: Success Rates of Algorithms on Different Functions

Function	Len	CR	CS	CT2B	CT2M	CTD	CT	GA	SMA
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
	Total	81	100	89	93	100	100	62	19
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
	Total	22	32	40	42	40	38	19	28
SAT	50	131	134	146	152	136	145	114	153
	100	28	21	24	26	16	25	38	27
	Total	159	155	170	178	152	170	152	180
Total		272	307	319	333	312	327	243	230

Table 2: Ranking of time to find solutions on different functions. $X < \{Y,Z\}$ indicates that X is faster than Y which is in turn faster than Z, but that only the first difference is significant with 95% confidence according to Tamhane's test

4Trap { GA, CTD, CT, SMA, CS, CR, CT2B, CT2M }
Shifted-Trap { GA, CTD, CS, CT, CR, CT2M, CT2B } < SMA
Dist-Trap { CTD, CS, CT, CT2B, CT2M }
H-IFF { GA,CR, CS} < { SMA, CTD, CT,CT2B, CT2M }
SAT { GA, SMA } < {CR,CS,CTD,CT} < {CT2MG,CT2BG}

between generations. On the SAT problems, where there is no regular problem structure to be learnt and exploited, the CT2M/B schemes again take longer.

For each function an ANOVA was performed followed by post-hoc testing using Tamhane's T2 test. This yielded the rankings in Table 2. This table shows that of the four more successful credit assignment mechanisms, the one with memory - CTD is always the fastest, followed by CT as expected from the discussion above. Hidden in the tables is the fact that the difference between the results for CTD and CT2M/CT2B is usually significant with 95% confidence, but that the difference between the CT and those two approaches is not significant.

5.3 Discussion

These results clearly demonstrate that the use of a credit assignment mechanism that does not rely solely on the improvement caused when a meme is applied to a single solution has its advantages. 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, and this does not seem to be at the expense of significantly increased runtimes. The memory-based approach (CTD) is faster, but can be mislead as shown by the lower Success Rates 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". In contrast the approach which exploits the creation of two offspring during recombination (CT2M/B) displays reliable optimum finding on all problems without a significantly increased overhead compared to the CT strategy.

6. CONCLUSIONS

The survey of meme-pairing and credit assignment schemes in Adaptive Memetic Algorithms [29] concluded that for continuous-variable problems a global adaptive scheme was preferable. In this work we have extended their work to consider combinatorial optimisation problems, and in particular to consider the adaptations that are necessary to the global-adaptive schemes when the set of memes available is not held static, but evolves. We have also taken into account the studies by Wiegend et. al. [46] on the optimal size of the "collaboration pool" in co-evolution. Our results suggest that for combinatorial optimisation, where the concept of "locality" is less well defined, a global-adaptive scheme is indeed preferable, subject to the ability of the credit assignment scheme to accurately reflect the value of a meme *at that point in time*. This can be done either by the use of a "memory" where the results of past encounters have a time-decaying contribution to the fitness of a meme, or by evaluating each meme with two different solutions. This of

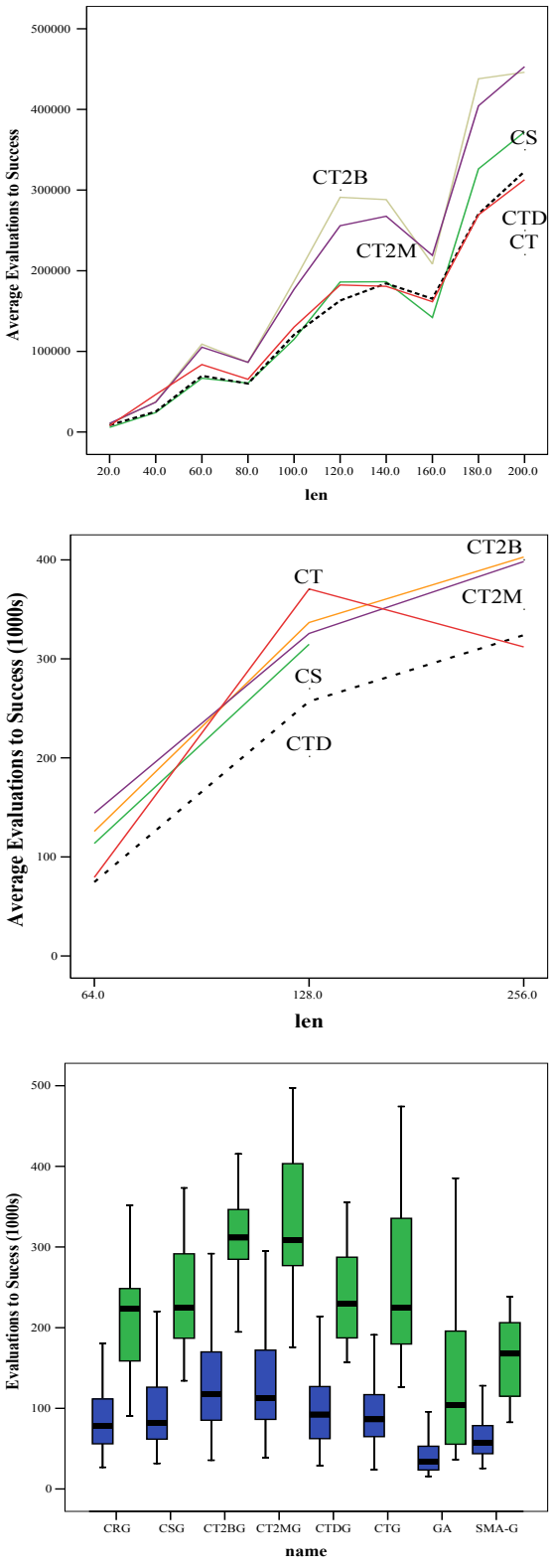

Figure 2: Average Evaluations to Success on Trap (top) and H-IFF (middle) functions. Box-plots of Evaluations to Success on SAT functions (bottom). Lighter boxes are for 50-variable instances, darker ones for 100 variables.

course has the benefit that each solution is subjected to LS using two potentially different neighbourhoods, which can enhance the escape from local optima. This effect can be seen on the SAT problems, and to a lesser extent on the H-IFF problems. Of the two schemes tested, the one which assigns to each meme the *average* of the improvements caused in the two solutions (CT2M) outperformed the use solely of the *better* of the two.

The resulting algorithm, CT2M displays problem solving capability better than either the fixed (SMA) memetic algorithms or the other adaptive memetic algorithms, and appears to represent a computationally inexpensive solution to the problems noted in previous papers e.g. [37]. The next phase of research will concern testing these conclusions with other meta-heuristics.

7. REFERENCES

[1] Satlib: http://www.satlib.org.

[2] P. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence*, pages 152–161. IEEE Press, 1995.

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

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

[5] L. Bull. *Artificial Symbiology*. PhD thesis, University of the West of England, 1995.

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

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

[8] E. Burke, G. Kendall, and E. Soubeiga. A tabu search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6), 2003.

[9] E. Burke and A. Smith. Hybrid evolutionary techniques for the maintenance scheduling problem. *IEEE Transactions on Power Systems*, 15(1):122–128, 2000.

[10] *2003 Congress on Evolutionary Computation (CEC'2003)*. IEEE Press, Piscataway, NJ, 2003.

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

[12] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[13] D. Fogel. *Evolving Artificial Intelligence*. PhD thesis, University of California, 1992.

[14] P. Hansen and N. Mladenovìc. An introduction to variable neighborhood search. In S. Voß, S. Martello, I. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Proceedings of MIC 97 Conference*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[15] W. Hart, N. Krasnogor, and J. Smith, editors. *Recent Advances in Memetic Algorithms*. Springer, Berlin, Heidelberg, New York, 2004.

[16] 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.

[17] P. Husbands and F. Mill. Simulated coevolution as the mechanism for emergent planning and scheduling. In R. Belew and L. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, San Francisco, 1991.

[18] *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996.

[19] 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)*, pages 667–671, 2002.

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

[21] N. Krasnogor. *Studies in the Theory and Design Space of Memetic Algorithms*. PhD thesis, University of the West of England, 2002.

[22] N. Krasnogor. Self-generating metaheuristics in bioinformatics: The protein structure comparison case. *Genetic Programming and Evolvable Machines. Kluwer academic Publishers*, 5(2):181–201, 2004.

[23] N. Krasnogor and S. Gustafson. A study on the use of "self-generation" in memetic algorithms. *Natural Computing*, 3(1):53–76, 2004.

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

[25] N. Krasnogor and J. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. In Spector et al. [45], pages 432–439.

[26] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: Model, taxonomy and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[27] S. Luke and L. Spector. Evolving teamwork and coordination with genetic programming. In J. Koza, D. Goldberg, D. Fogel, and R. Riolo, editors, *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 141–149. MIT Press, Cambridge, MA, 1996.

[28] Y. Ong and A. Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, 2004.

[29] 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*, 36(1), 2006.

[30] J. Paredis. The symbiotic evolution of solutions and their representations. In L. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 359–365. Morgan Kaufmann, San Francisco, 1995.

[31] J. Paredis. Coevolutionary algorithms. In Bäck et al. [4].

[32] G. Parker and H. Blumenthal. Varying sample sizes for the co-evolution of heterogeneous agents. In *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*, pages 766–771. IEEE Press, 2004.

[33] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimisation. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science, pages 248–257. Springer, Berlin, Heidelberg, New York, 1994.

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

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

[36] J. Smith. Co-evolution of memetic algorithms: Initial investigations. In J. M. Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors, *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 537–548. Springer, Berlin, Heidelberg, New York, 2002.

[37] J. Smith. Co-evolving memetic algorithms: A learning approach to robust scalable optimisation. In CEC-2003 [10], pages 498–505.

[38] J. Smith. Protein structure prediction with co-evolving memetic algorithms. In CEC-2003 [10], pages 2346–2353.

[39] J. Smith. The co-evolution of memetic algorithms for protein structure prediction. In Hart et al. [15], pages 105–128.

[40] J. Smith. Co-evolving memetic algorithms: A review and progress report. *IEEE Transactions in Systems, Man and Cybernetics, part B*, 37(1):6–17, 2007.

[41] J. Smith and T. Fogarty. Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science, pages 441–450. Springer, Berlin, Heidelberg, New York, 1996.

[42] J. Smith and T. Fogarty. Recombination strategy adaptation via evolution of gene linkage. In ICEC-96 [18], pages 826–831.

[43] J. Smith and T. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In ICEC-96 [18], pages 318–323.

[44] J. Smith and T. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.

[45] L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. Morgan Kaufmann, San Francisco, 2001.

[46] R. Wiegand, W. Liles, and K. D. Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In Spector et al. [45], pages 1235–1245.