Co-evolving Memetic Algorithms: Initial Investigations

Jim Smith

Intelligent Computer Systems Centre,
University of the West of England,
Bristol BS16 12QY, U.K.
james.smith@uwe.ac.uk,

Abstract. This paper presents and examines the behaviour of a system whereby the rules governing local search within a Memetic Algorithm are co-evolved alongside the problem representation. We describe the rationale for such a system, and the implementation of a simple version in which the evolving rules are encoded as (condition:action) patterns applied to the problem representation, and are effectively self-adapted. We investigate the behaviour of the algorithm on a test suite of problems, and show significant performance improvements over a simple Genetic Algorithm, a Memetic Algorithm using a fixed neighbourhood function, and a similar Memetic Algorithm which uses random rules, i.e. with the learning mechanism disabled.
Analysis of these results enables us to draw some conclusions about the way that even the simplified system is able to discover and exploit different forms of structure and regularities within the problems. We suggest that this "meta-learning" of problem features provides a means both of creating highly scaleable algorithms, and of capturing features of the solution space in an understandable form.

1 Introduction

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 combinatorial optimisation [1], optimisation of non-stationary functions [2], and multi-objective optimisation [3]. See [4] for a comprehensive bibliography. Commonly in these algorithms, the Local Search 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 three principal components which affect the workings of the LS algorithm. The first is the choice of pivot rule, which is usually either *Steepest Ascent* or *Greedy Ascent*. The second component is the "depth" of local search, which can vary from one iteration, to the search continuing to local optimality. Considerable attention has been paid to studying the effect of changing these parameters within MAs e.g. [5].

The third factor is the choice of neighbourhood function, which can be thought of as defining a set of points that can be reached by the application of some move operator to a point. We can consider the graphs defined by different move operators as “fitness landscapes” [6]. Merz and Freisleben [7] discuss 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.

In some cases, domain specific information may be used to guide this choice. However, it has recently been shown that the optimal choice of operators can be not only instance specific within a class of problems [7, pp254–258], but also dependant on the state of the evolutionary search [8]. The idea that periodically changing the move operator may provide a means of escape from local optima by redefining the neighbourhood structure, has been demonstrated in the *Variable Neighbourhood Search* algorithm [9].

The aim of this work is to provide a means whereby the definition of the local search operator used within a MA can be varied over time, and then to examine whether evolutionary processes can be used to control that variation, so that a beneficial adaptation takes place. In order to accomplish this aim, we must address four major issues. The first of these is the representation of LS operators in a form that can be processed by an evolutionary algorithm, and the choice of of initialisation and variation operators. The second is the credit assignment mechanism for assigning fitness to the LS population members. The third is the choice of population structures and sizes, along with selection and replacement methods for managing the LS population. Finally, we require a set of experiments, problems and measurements designed to permit evaluation and analysis of the behaviour of the system. This paper represents the first stage of this analysis.

2 Rule-Based Adaptation of Move Operators

The representation chosen for the LS operators is a tuple <*Search_Depth, Pivot_Rule, Pairing, Move*>. The first two elements are self-explanatory. Values for *Pairing* are taken from the set {*Linked, Random,Fitness_Based*}. When the Local Search phase is applied to a member of the solution population, the value of the *Pairing* in the corresponding member of the LS population is examined. If the value is *Linked* then that LS member is used to act on the solution. If the *Pairing* takes one of the values *Random* or *Fitness_Based* then a selection is made from the available (i.e. unlinked)members of the LS population according. By making this variable part of the rule and subject to evolution, the system can be varied between the extremes of a fully unlinked system,(in which although still interacting the two populations evolve separately), and a fully linked system. In the latter the LS operators can be considered to be extra genetic material which is inherited and varied along with the problem representation, in an exactly analogous way to the inheritance of strategy parameters in Self Adapting EAs. We note that "Self-Adaptation" can be considered as co-evolution with hereditary symbiosis, i.e. where the two populations share a common updating mechanism.

The representation chosen for the move operators was as *condition:action* pairs, which specify a pattern to be looked for in the problem representation, and a different pattern it should be changed to. 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 wildcard characters but also the specifications of repetitions and iterations. Further, permitting the use of different length patterns in the two parts of the rule gives scope for *cut* and *splice* operators working on variable length solutions.

While the framework that we have describe above is intended to permit a full exploration of several research issues, we shall initially restrict ourselves to considering a simple system, and examining its behaviour on a well understood set of binary encoded test problems. For these initial investigations we therefore restricted the LS operators to a single improvement step, a greedy acceptance mechanism, and full linkage. This restriction to what are effectively self-adaptive systems provides a means of dealing with the credit assignment and population management isssues noted above.

We also initially restrict ourselves to considering only rules where the *condition* and *action* patterns are of equal length and are composed of values taken from the set {0,1,#}.The last of these is a "don't care" symbol which is only allowed to appear in the *condition* part of the rule.

The neighbourhood of a point i is defined by finding the (unordered) set of positions where the substring denoted by *condition* is matched in the representation of i. For each of these a new string is then made by replacing the matching substring with the *action* pattern. To give an example, if we have a solution represented by the binary string 1100111000 and a rule 1#0:111, then this matches the first, second, sixth and seventh positions, and the neighbourhood is the set {1110111000, 11111111000, 1100111100,1100111110}. Note that in this initial work we do not considered the string as toroidal.

3 Related Work

The COMA system can be related to a number of different branches of research, all of which offer different ways of perspectives and means of analysing it's behaviour. Space constraints preclude a full discussion of each of these, so we will briefly outline some of these perspectives.

Although we are not aware of other algorithms in which the LS operators used by an MA are adapted in this fashion, there are other examples of the use of multiple LS operators within evolutionary systems. Krasnogor and Smith [8] describe what they call a "MultiMemetic Algorithm", in they used a simple Self-Adaptive mechanism to evolve the choice of which a fixed set of static LS operators ("memes") should be applied to individual solutions. They report that their systems are able to adapt to use the best meme available for different instances of TSP.

As noted above, if the populations are of the same size, and are considered to be linked, then this instantiation of the COMA framework can be considered as a type of Self Adaptation. The use of the intrinsic evolutionary processes to adapt search strategies, and the conditions necessary, is well documented e.g. for mutation step sizes [10, 11], mutation probabilities [12], recombination operators[13, 14] and general variation operators [15], amongst many others.

If the two populations are not linked, then we have a co-operative coevolutionary system, where the fitness of the members of the LS population is assigned as some function of the relative improvement they cause in the "solution" population. Co-operative co-evolutionary (or "symbiotic") systems have been used with good reported results for function optimisation [16–18] and Bull conducted a series of more general studies on the conditions necessary for co-operative co-evolution to occur [19–21]. These issues will be explored in future work.

If we were to simply apply the rule selected from in the LS population (possibly iteratively) to transform an individual solution, without considering a pivot rule, then we could also view the system as a type of "developmental learning" akin to the studies in the evolution of Genetic Code [22]

COMA differs from the last two paradigms because the LS population can potentially modify the genotypes within the solution population. This phase of improvement by LS can be viewed as a kind of lifetime learning, which leads naturally to the question of whether a Baldwinian approach might be preferable to the Lamarkian Learning currently implemented. However, even if a Baldwinian approach was used, the principal difference between the COMA approach and the co-evolutionary systems above lies in the use of a pivot rule within the LS, such that detrimental changes are rejected.

Finally, and perhaps most importantly, we should consider that if the same rule has an improving effect on different parts of a solution chromosome over as number of generations, then the evolution of rules can be seen as a process of learning generalisations about patterns within the problem representation, and hence the solution space. This point of view is akin to that of Learning Classifier Systems. For the case of unlinked fitness-based selection of LS operators, insight from this field can be used to guide the credit assignment process.

4 The Test Suite and Experimental set-up

In order to examine the behaviour of the system it was used with a set of variants of a test function whose properties are well known. This was a sixty four bit problem composed of 16 subproblems of Deb's 4-bit fully deceptive function given in [23]. The fitness of each subproblem i is given by its unitation $u(i)$ (i.e. the number of bits set to 1):

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

In addition to a "concatenated" version (which we will refer to as 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 ensures that even the longest rules allowed in these experiments would be unable to alter more than one element in any of the subfunctions.

A third variant of this problem (Shifted-Trap) was designed to be more difficult than the first for the COMA algorithm to learn a single generalisation, by making patterns which were optimal in one sub-problem, sub-optimal in all others. This was achieved by noting that each sub-problem as defined above is a function of unitation, and therefore can be arbitrarily translated by defining a 4-bit string and using the Hamming distance from this string in place of the unitation. Since we have 16 sub-problems, we simply used the binary coding of the sub-problem's index as basis for its fitness calculation.

We used a generational genetic algorithm, with deterministic binary tournament selection for parents and no elitism. 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 values were taken as "standard" from the literature, bearing in mind the nature of the 4-Trap function. 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).

For each problem, 20 runs were made for each population size {100,250,500}. Each run was continued until the global optimum was reached, subject to a maximum of 1 million evaluations. Note that since one iteration of LS may involve several evaluations, this allows more generations to the GA, i.e. we compare algorithms strictly on the basis of the number of calls to the evaluation function.

The algorithms used are: a "vanilla" GA i.e. with no use of Local Search (GA), a simple MA using one iteration of greedy ascent over the neighbourhood at Hamming distance 1 (MA), a version of COMA using a randomly created rule in each application, (i.e. with the learning disabled) (RandComa), variants of COMA using rules of fixed lengths in the ranges $\{1, \ldots, 9\}$ (1-Coma,...,9-Coma), and finally an adaptive version of COMA (A-Coma). For A-Coma the rule lengths are randomly initialised in the range [1,16]. During mutation, a value of $+/-1$ is randomly chosen and added with probability 0.0625, subject to staying in range.

5 Comparison Results

Figure 1 shows the results of these experiments as a plot of mean time to optimum for 4-Trap with three different population sizes. When an algorithm failed to reach the optimum in all twenty runs, the mean is taken over the successful runs, and this number is shown. The error bars represent one standard deviation. It should be noted that the scale on the y-axis is logarithmic. We can see that the GA and MA,and 1-Coma algorithms fail to find the optimum as frequently, or when they do as fast, for the smaller population sizes. For all population sizes there is greater variance in the performance of these three algorithms than for the other variants.

Because the variances are unequal, we applied the conservative Tamhane's T2 test to the solution times for the successful runs to detect statistically significant differences in performance. The GA, MA and 1-coma algorithms are significantly slower than the rest at the 5% level for a population of 100. For a population size of 250 the GA and MA algorithms are significantly slower. When the population size is increased to 500, the worse performance seen with the GA and MA is no longer significant, according to this conservative test. However, the Rand-Coma is now significantly slower than all but the GA, MA and 2-Coma. 1-Coma is significantly slower than all but the GA, and 2-Coma is slower than all but GA, MA and Rand-Coma.

In short, what we can observe is that for fixed rule lengths of between 3 and 9, and for the adaptive version, the COMA system derives performance benefits from evolving LS rules. Significantly, and unlike the GA and MA, the COMA algorithm does not depend on a certain size population before it is able to solve the problem reliably. This is indicative of a far more scaleable algorithm.

Figure 2 shows the results of the experiments on the variants of the trap functions. For the "Shifted" trap function, the performances of the GA and MA are not significantly different from those on the unshifted version. this refelects the fact that these algorithms solve the sub-problems independently and are "blind" to whether the optimal string for each is different. When we examine the results for the COMA algorithms, we see slower solution times than on the previous problem, resulting from the fact that no one rule can be found which will given good performance in every subproblem. However we see a similar pattern of reliable problem solving for all but 1-Coma and 2-Coma. Analysis reveals that even these last two are statistically significantly better than GA or MA for all but the largest population size. Interestingly, the RandComa algorithm performs well here, probably as a natural consequence of using a random rule every time, so promoting diversity in the rule base.

Considering Dist-Trap, we first note that the GA, MA and Rand-COMA failed to solve the function to optimality in any run, regardless of population size. The poor results of the GA can be attributed to the mismatch between the distributed nature of the representation and the high positional bias of the 1-point Crossover used. When we consider the action of the bitflipping LS operator on a subproblem, this will lead towards the sub-optimal solution, whenever the unitation is 0,1 or 2, and the greedy search of the neighbourhood will also lead towards the deceptive optimum 75% of the time when the unitation is 3. This observation helps us to understand the poor results of the simple MA, and the 1-Coma algorithm.

When we examine the other COMA results, noting that the success rate is less than for the other problems, we again see the same pattern of better performance for the adaptive version and fixed rulelengths in the range 3-5, tailing off at the extreme lengths. Note that although the mean solution time drops for long rule length, so too does the number of successful runs which we take as our primary criterion. We also note that the failure of the RandComa algorithm indicates that some learning is required here.

6 Discussion and Analysis

The results given above are promising from the point of view of improved optimisation performance, but require some analysis and explanation. The deceptive functions used were specifically chosen because GA theory suggests that they are solved by finding and mixing optimal solutions to sub-problems. When we consider the results, we can see that the performance is best on 4-Trap, with a rule length of 4, which would support the hypothesis that the system is "learning" the structure of the sub-problems. Although not immediately apparent from the logarithmic scales, the solutions times here are less than half those on the other problems.

However we should note that the algorithms are not aware of the sub-problem boundaries. On 4-Trap and Shifted-Trap, for lengths of 4 or less occasionally, and always for lengths greater than 4, the changes made will overlap several sub-problems. This must always happen for Dist-Trap. It is clear from the results with different rule lengths, and from the distributed problem, that there is a second form of improvement working on a longer timescale., which does not arise simply from the use of random rules.

In order to examine the behaviour of the algorithm we plotted the population means of the effective rule length (only relevant for A-Coma), the "specificity" (i.e. the proportion of values in the condition not set to #) and the "unitation" (the proportion of bits in the action set to 1), and also the highest fitness in the population (with 100 as the optimum) as a function of the number of elapsed generations. Figure 3 shows the A-Coma results averaged over 20 runs on each of the three problems, with a population of 250. We also manually inspected the evolving rule bases on a large number of runs for each problem.

For the 4-Trap function (left hand graph), the system rapidly evolves medium length $(3 - 4)$, general (specificity $< 50\%$) rules whose action is to set all the bits to 1 (mean unitation approaches 100%). Note that in the absence of selective pressure (i.e. the pivot rules meant that the solutions were left unchanged), all three of these values would be expected to remain at their initial values, so these changes result from beneficial adaptation. Closer inspection of the evolving rulebase confirms that the optimal subproblem string is being learned and applied.

For the Shifted-Trap function, where the optimal sub-blocks are all different (middle) the rule length decreases more slowly. The specificity also remains higher, and the unitation remains at 50%, indicating that different rules are being maintained. This is borne out by closer examination of the rule sets.

The behaviour on Dist-Trap is similar to that on 4-Trap, albeit over a longer timescale. Rather than learning specific rules about sub-problems, which cannot possibly be happening (since no rule is able to affect more than one locus of any subproblem), the system is apparently learning the general rule of setting all bits to 1.

The rules are generally shorter than for 4-Trap, (although this is slightly obscured by the averaging) which means that the number of potential neighbours is higher for any given rule. Equally, the use of wildcard characters, coupled with the fact that there may be matches in the two parts of the rules, means that length of the rules used defines a maximum radius in Hamming space for the neighbourhood, rather than a fixed distance from the original solution. Both of these observations, when taken in tandem with the longer times to solution, suggest that when the system is unable to find a single rule that matches the problems' structure, a more diverse search using a more complex neighbourhood is used, which slowly adapts itself to the state of the current population of solutions.

7 Conclusions

We have presented a framework in which rules governing LS operators to be used in memetic algorithms can be co-evolved with the population of solutions. A simple version was implemented which used Self-Adaptation of the patterns defining the move operators, and this was shown to give improved performance over both GAs and simple MAs on the test set. We showed that the system was able to learn generalisations about the problem when these were useful. We also noted that the COMA algorithms were far less dependant on a critical population size to locate the global optima, and suggested that this indicates a far more scaleable type of algorithm.

The test set used here was designed to maximise different types of difficulty for COMA, namely deception, inappropriate representation ordering, and multiple different optima. Although space precludes their inclusion, we have repeated these experiments with a wide range of different problem types and found similar performance benefits. Clearly further experimentation and analysis is needed, and there are many issues to be explored, however we believe that this paper represents the first stage in a promising line of research.

References

1. Merz, P., Freisleben, B.: A comparion of Memetic Algorithms, Tabu Search, and ant colonies for the quadratic assignment problem. In: Proceedings of the 1999 Congress on Evolutionary Computation, IEEE Service Center (1999) 2063–2070
2. Vavak, F., Fogarty, T., Jukes, K.: A genetic algorithm with variable range of local search for tracking changing environments. In Voigt, H.M., Ebeling, W., I.Rechenberg, Schwefel, H.P., eds.: Proceedings of the Fourth Conference on Parallel Problem Solving from Nature, Springer Verlag (1996) 376–385
3. Knowles, J., Corne, D.: A comparative assessment of memetic, evolutionary and constructive algorithms for the multi-objective d-msat problem. In: Gecco-2001 Workshop Program. (2001) 162–167
4. Moscato, P.: Memetic algorithms' home page. Technical report, http://www.densis.fee.unicamp.br/˜moscato/memetic_home.html (2002)
5. Hart, W.E.: Adaptive Global Optimization with Local Search. PhD thesis, University of California, San Diego (1994)
6. Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, The University of New Mexico, Albuquerque, NM (1995)

7. Merz, P., Freisleben, B.: Fitness landscapes and memetic algorithm design. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization, McGraw Hill (1999) 245–260
8. Krasnogor, N., Smith, J.: Emergence of profitiable search strategies based on a simple inheritance mechanism. In Spector, L., Goodman, E., Wu, A., Langdon, W.B., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Morgan Kaufmann (2001) 432–439
9. Hansen, P., Mladenoviĉ, N.: An introduction to variable neighborhood search. In Voß, S., Martello, S., Osman, I.H., Roucairol, C., eds.: Meta-Heuristics: Advances and trends in local search paradigms for optimization. Proceedings of MIC 97 Conference. Kluwer Academic Publishers, Dordrecht, The Netherlands (1998)
10. Schwefel, H.P.: Numerical Optimisation of Computer Models. John Wiley and Sons, New York (1981)
11. Fogel, D.: Evolving Artificial Intelligence. PhD thesis, University of California (1992)
12. Back, T.: Self adaptation in genetic algorithms. In Varela, F., Bourgine, P., eds.: Towards a Practice on Autonomous Systems: Proceedings of the First European Conference on Artificial Life, MIT Press (1992) 263–271
13. Schaffer, J., Morishima, A.: An adaptive crossover distribution mechanism for genetic algorithms. In J.J.Grefenstette, ed.: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum (1987) 36–40
14. Smith, J., Fogarty, T.C.: An adaptive poly-parental recombination strategy. In Fogarty, T.C., ed.: Evolutionary Computing 2, Springer Verlag (1995) 48–61
15. Smith, J., Fogarty, T.: Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In Voigt, Ebeling, Rechenberg, Schwefel, eds.: Proceedings of the Fourth Conference on Parallel Problem Solving from Nature, Springer Verlag (1996) 441–450
16. Husbands, P.: Distributed co-evolutionary genetic algorithms for multi-criteria and multi-constraint optimisiation. In Fogarty, T., ed.: Evolutionary Computing: Proceedings of the AISB workshop. LNCS 865, Springer Verlag (1994) 150–165
17. Paredis, J.: The symbiotic evolution of solutions and their representations. In Eshelman, L.J., ed.: Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann (1995) 359–365
18. Potter, M.A., DeJong, K.: A cooperative coevolutionary approach to function optimisation. In Davidor, Y., ed.: Proceedings of the Third Conference on Parallel Problem Solving from Nature, Springer Verlag (1994) 248–257
19. Bull, L.: Artificial Symbiology. PhD thesis, University of the West of England (1995)
20. Bull, L.: Evolutionary computing in multi agent environments: Partners. In Back, T., ed.: Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann (1997) 370–377
21. Bull, L., Holland, O., Blackmore, S.: On meme-gene coevolution. Artificial Life **6** (2000) 227–235
22. Keller, R.E., Banzhaf, W.: The evolution of genetic code in genetic programming. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 1077–1082
23. Back, T., Fogel, D., Michalwicz, Z.: Handbook of Evolutionary Computation. Volume 1. Oxford University Press (1997)

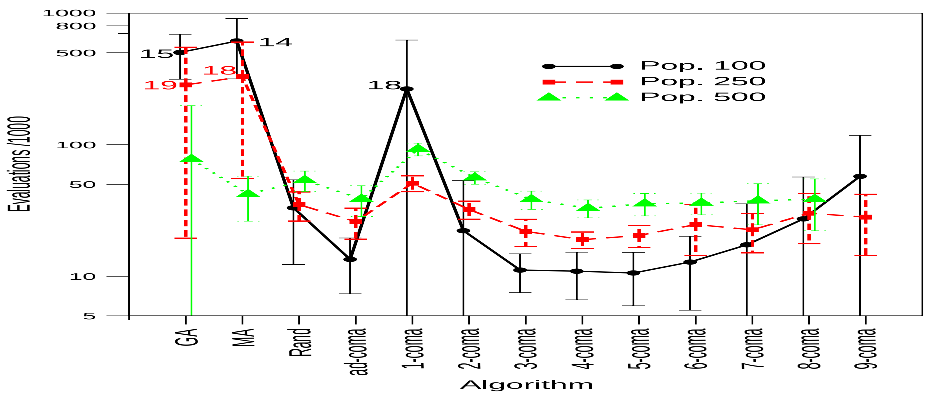

Fig. 1. Times to optimum for the 4-Trap function. Note logarithmic y-axis

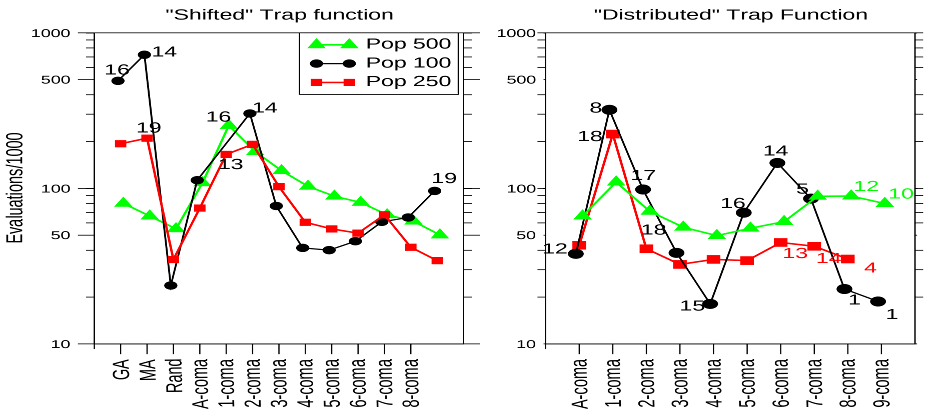

Fig. 2. Times to optimum for the Shifted-Trap and Dist-Trap. Note logarithmic y-axis. Error bars omitted for clarity

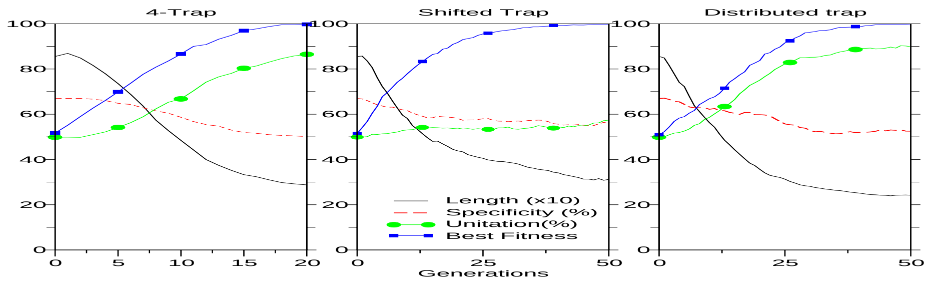

Fig. 3. Analysis of Evolving Rules by Function Type