Investigations in Meta-GAs: Panaceas or Pipe Dreams?

Jeff Clune[1]*, Sherri Goings[2]*, Erik D. Goodman[3], William Punch[2]

Michigan State University, East Lansing, MI, USA
[1] Dept. of Philosophy, jclune@msu.edu
[2] Dept. of Computer Science, {goingssh, punch}@cse.msu.edu
[2] Dept. of Electrical and Computer Engineering, goodman@egr.msu.edu
* Both authors contributed equally to this paper

Abstract. A meta-GA (GA within a GA) is used to investigate evolving the parameter settings of genetic operators for genetic and evolutionary algorithms (GEA) in the hope of creating a self-adaptive GEA. We report three findings. First, the meta-GA can adapt its genetic operators to different problems and thereby perform well on average across diverse problems. Second, the meta-GA can change its parameters during the course of a run—seemingly a good idea—but this behavior may actually decrease performance. Finally, the genetic operator configurations the meta-GA evolves are far from optimal. We conclude that, while meta-GAs show promise for automating some parameter configurations, they are not likely to replace manually configured genetic and evolutionary algorithms without innovative alteration.

1 Introduction

One of the great promises of genetic and evolutionary algorithms (GEAs) was that they would solve difficult problems with minimal human intervention. The reality has become that getting GEAs to solve challenging problems usually requires a lot of esoteric knowledge about choosing a correct configuration of GEA parameters from an enormous number of possible setups. This is because what constitutes a good GEA setup—which genetic operators to use and with what frequency—changes from problem to problem [7], [17]. It is challenging and time consuming to come up with satisficing parameter configurations, and a failure to do so typically leads to unsatisfactory results [3], [4]. Furthermore, in some problems the optimal parameter settings may be different for various phases of the search process [4], [15], [8], [12]. This issue of problem 'temporality' only complicates the challenge of finding appropriate GEA settings for a given problem by adding this further dimension to the search space. Given that GEAs are good at finding suitable solutions amongst large multi-dimensional search spaces, it makes sense to try to find good GEA parameter setups using a GEA.

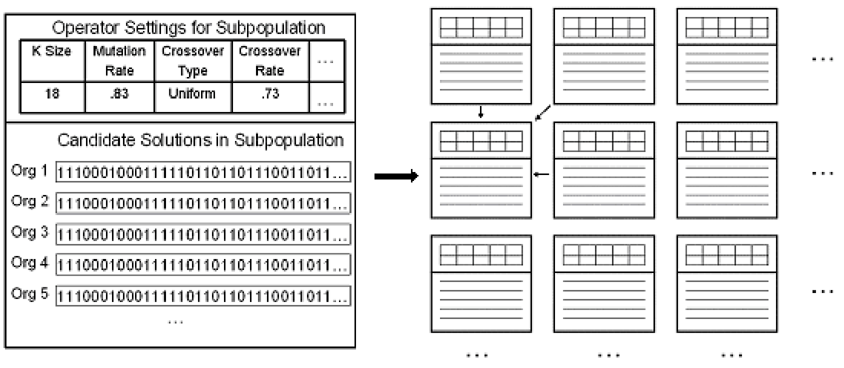
The research in this paper focuses on using a self-adaptive 'meta-GA' to evolve GA parameter settings. A meta-GA consists of a group of subpopulations that are each running under a unique configuration of genetic operators. The fitness of each subpopulation relative to other subpopulations is evaluated according to a fitness function different from the one being used within each subpopulation (i.e. a 'higher level' fitness function). Those configurations of genetic operators that are deemed more fit increase in frequency in subsequent generations. This is in contrast to the numerous attempts to put control parameters for genetic operators (e.g. mutation rate) directly on the chromosomes of individuals (reviewed in [8] and [4]). It is also in contrast to a predetermined rule specifying changes in operators over time, as used in simulated annealing [8]. Many variations on the theme of meta-GAs have been explored. Some evolve the operators with which to start a run, but do not allow for mid-course corrections via changing parameters during a run (e.g. [5]). Others initialize each subpopulation to have one unchanging configuration, and vary the size of each subpopulation according to its fitness, increasing the amount of searching done with good parameter sets [12]. This method limits the search space to those parameter sets initially present. It also prevents genetic operators from being individually tuned or coordinated in parallel (as in co-evolution)[4]. A further approach is to evolve one or a few genetic operators but not to allow a large set of operator types and frequencies to co-evolve (e.g. [14], [5], [10], [6], [1], [2]). Such approaches, however, fail to take advantage of the complex interactions between the types and frequencies of genetic operators, and may therefore overlook fruitful combinations of them [4]. Of all the work we know of, Lis and Lis utilize the most comprehensive set of evolvable operators [11]. They evolve the mutation probability, crossover rate, and population size, but do not include things such as crossover, mutation, and selection types, which can be important elements of the mix [4].

The general consensus of the above work seems to be that using GEAs to tune GEA parameter settings shows promise [4]. However, most of the approaches still incorporate a large amount of knowledge from the scientist and do not take advantage of all the opportunities possible with self-adaptive GEAs [13], [4], [8]. In this paper, we investigate what happens when we move towards a GA that does not require fixed parameter settings because it evolves many of them during all phases of the run. Wang et al. created such a meta-GA [15], [16]. They evolved both the type and frequency ofthe application of genetic operators in an island model. Their preliminary results indicated that this relatively 'parameterless' meta-GA converged to an appropriate setup. They also suggest, anecdotally, that the meta-GA is discovering that different settings are preferable at different times during a run. In this paper we extend their work by doing enough runs to allow statistical analysis and adding experiments that help investigate to what extent this meta-GA may enable GEAs to deliver on their original promise of solving challenging problems with minimal human involvement.

We test the efficacy of the meta-GA by running it on two different toy problems: the 'counting ones' problem and a 4-bit 'deceptive trap.' If the meta-GA is superior to traditional GAs then it should be able to do better on both problems on average than any specific set of genetic parameters does on both problems on average. The concepts of 'generalists' and 'specialists' help frame the issue. There are GA settings that are specialized to individual problems. These will do well on the problem they are tuned to and perform poorly on most other problems. Conversely, there should exist general GA parameter settings whose performance is mediocre, but satisficing, for a large range of problems. We are interested in seeing whether the meta-GA can combine the best of both worlds by being a generalist, in that it will work well on a diverse set of problems, yet show the benefits of specialization, by becoming a specialist on the problem it faces. To satisfy this criterion, the meta-GA should prove its worth as a generalist by doing better than specialist configurations across different problems, as well as show near-specialist competency by performing nearly as well as specialists on the problems the specialists are tuned to. Additionally, we are interested in seeing whether the meta-GA can take advantage of the potential for changing the setup of the genetic operators over time during different phases of evolving its solution. Finally, we compare the configurations the meta-GA evolves versus those more commonly used by GA scientists.

2 Methods

We perform all of our experiments using the software package DAGA2 (available on the MSU GARAGE website at http://garage.cps.msu.edu/) [15] , [16]. DAGA2 is a two-level GA that evolves simultaneously on both levels. At the first level of DAGA2 are traditional GAs, each of which consists of a subpopulation of candidate solutions to some problem and an associated set of operators which govern the evolution of that subpopulation. The second level consists of 36 of these level-1 subpopulations and evolves the parameters associated with them. A diagram of the DAGA2 topology is shown in figure 1.


Operator Settings for Subpopulation

K Size	Mutation Rate	Crossover Type	Crossover Rate	...
18	.83	Uniform	.73	...

Candidate Solutions in Subpopulation

Org 1	111000100011111011011011110011011...
Org 2	111000100011111011011011110011011...
Org 3	111000100011111011011011110011011...
Org 4	111000100011111011011011110011011...
Org 5	111000100011111011011011110011011...

Fig. 1. On the left is an example subpopulation containing a number of candidate solutions and one chromosome of operator settings that governs the evolution of those solutions. On the right is the level-2 DAGA population containing many of these subpopulations. The small arrows show the flow of data being sent to the sample subpopulation from its three neighbors.

The parameters that may be modified at the first level are crossover type, crossover rate, genomic mutation rate, selection type, and the parameters associated with each selection type (tournament size for tournament selection and the scaling factor for roulette selection). The parameter ranges (shown in Table 2) were chosen to be wide enough for flexible search while minimizing effects of mutational balance. Each subpopulation's initial parameters are randomly initialized within their respective ranges.

Table 1. Ranges for parameter settings in DAGA2

Crossover Type	{1-pt, 2-pt, Uniform}
Crossover Rate	{0-100%}

Genomic Mutation Rate	On Chromosomes Length 1000: {0-5}
	On Chromosomes Length 3000: {0-15}

Selection Type	{Tournament, Roulette, Stochastic Remainder Sampling?]
Tournament Size	{0-50}
Roulette Scaling Factor	{0-5}

After finding in hundreds of preliminary runs that tournament selection was selected for overwhelmingly in every single run, we locked this selection type in to speed up experiments. Mutation type was fixed as bit mutation, since this was the type relevant to our test problems. While this is not a completely parameterless meta-GA, since other entries could be added to each type category (e.g. additional selection types such as rank selection) and other things, like representations, could be evolved, it is a move toward a system in which the various parameters interact and we can thus see whether they successfully co-evolve.

The second level of DAGA2 evaluates the "fitness" of each subpopulation as the fitness of its best individual and acts as a traditional GA would in setting up a competitive environment in which the best populations propagate more often than others. The 36 randomly initialized subpopulations are placed on a toroidal grid and every five generations each receives information from three neighbors: the subpopulations directly above, to the right, and to the upper-right diagonal of it. The most fit neighbor is then calculated and is used in a subpopulation's second-level evolution process, consisting of migration, mutation, and crossover, in that order. Migration involves a subpopulation receiving copies of the top 7% of organisms from its best neighbor and placing them in its own population. The organisms to be replaced are chosen randomly with the caveat that the best individual in the population is never overwritten. Second-level mutation causes each parameter value of a subpopulation to mutate with a 20% probability. In 90% of these parameter mutations, a delta is taken randomly from a normal distribution (with a mean of 0 and a standard deviation of 1/20th of the range for that parameter setting) and added to the current parameter value. The other 10% of the time the parameter is simply changed to a random number in the given parameter range. Second-level crossover (uniform) consists of a subpopulation copying its best neighbor's level-2 parameters over its own with an 80% probability of occurrence for each individual parameter. In this way the operators and other GA parameters that lead to more effective evolution of individuals in subpopulations will be discovered and propagate, leading to more effective evolution of the level-1 individuals overall. Note that, unlike the method of Grefenstette [5], this evolution is occurring while the level-1 individuals are working on the solution of the problem, and the experience they accumulate about the fitness landscape is not discarded.

In our experiments, the second level was made up of 36 first-level subpopulations, each of which contained 100 randomly initialized chromosomes representing candidate solutions. All experimental data reported here is averaged over 25 replicate runs differing only in random number seed. P-values for comparisons between experiments are calculated using an Independent Group t-test. Two level-1 fitness functions (problems) were tested: counting ones (or one-max) and a repeating, deceptive 4-bit trap. The fitness of a solution to the counting ones problem is simply the number of ones on its binary chromosome, which had a length of 3,000 bits. The fitness of a solution to the trap problem is the total of the fitnesses of each of its non-overlapping 4 bit sections. A sample chromosome of length 1000 would have 250 traps: bits 0-3 would be the first trap, bits 4-7 the second, and so on to bits 996-999 as the last trap. The fitness of each section is a function of how many of the 4 bits are 1's, as shown in figure 2. The landscape of this trap problem has many local optima. The chromosome length for the trap problem was 1,000 or 3,000. Fitnesses are reported in tables as a percentage of the max possible fitness for that problem.

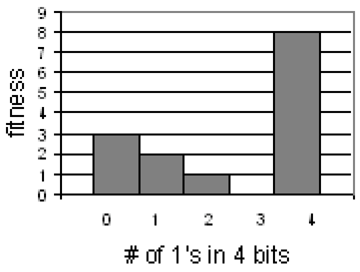


Fig. 2. Histogram showing the fitness function for the 4-bit trap problem. The highest fitness is gained by setting all 4 bits to 1, but having any 3 bits set to 1 gives the lowest fitness, creating a trap for a simple hill-climber

3 Experiments, Results and Discussion

Our first task was to demonstrate that the meta-GA was indeed finding different parameter settings for the two different problems. We did 25 runs with the broad parameter ranges described in methods that the meta-GA could search through. The average parameter setting values after 100 generations are reported in Table 2. The most notable difference between the two sets of parameters is that uniform crossover is selected for when the fitness function is the counting ones problem whereas two-point crossover dominates when it is the trap problem. This makes sense when considering the two different problems. Uniform crossover is ideal for combining the best alleles of different genomes but does not maintain groups of genes tightly linked on the chromosome. This is a bad tactic in the trap problem, where groups of all zeros in the right place should be maintained. One- and two-point crossover are less likely to break up solutions to individual traps and thus are much better on this problem, at least in the first hundred generations.

Table 2. The averages across 25 runs of the parameter settings that evolved on two different problems. Tournament selection was locked in since doing so sped up experiments and it was universally chosen in all the preliminary runs performed with the meta-GA. The rest of the parameters are drawn from the system after 100 generations of evolution. Note that the meta-GA overwhelmingly uses uniform crossover for the counting ones problem, which does not have building blocks, whereas the trap problem, which does have building blocks, yields the evolution of crossover types that keep sections of genomes together. The rest of the parameters are not significantly different ($p > .1$)

	Counting Ones	4-bit Trap
Genomic Mutation Rate	2.15	1.54
1pt Crossover	1%	11%
2pt Crossover	1%	87%
Uniform Crossover	97%	1%
Crossover Rate	89%	86%
Tournament Size	24	22
Fitness	98%	82%

While the mutation rates appear different, they are not significantly so ($p > .1$). The convergence across problems to similar values for crossover rates and for tournament sizes is counter-intuitive. One can rule out mutation-selection balance as the explanation for the similar crossover rates, given that the range is from 0 to 100%. We looked into whether the mutational force is dominating selection in the tournament sizes (which were in the middle of the 1-50 range) by testing a series of ranges: (e.g. 0-10, 0-15, 0-20, 0-25, … 0-45, data not shown). If mutation is the dominant force, we would expect the resulting setting after a period of evolution to be in the middle of its possible range. We found this to be the case in ranges with an upper limit above approximately 15. In ranges with a lower ceiling, selection was strong and pushed the tournament size up to nearly the ceiling. Selection thus prevents tournament size ranges from persisting below 15 or so, but it appears that mutation-selection balance is the reason the tournament size averages on the separate problems are so similar within the wider 0-50 range. Looking at all the evolved parameter settings, we conclude that some of the parameter settings have been adapted to the problem at hand while others work for both problems.

We next addressed the issue of whether it would have been better to use either of the parameter settings on both problems. In other words, does either of the specialist parameter settings that we evolved do better than the meta-GA in general? To test this we used the parameter settings found by the meta-GA in the 100th generation on each problem as the immutable parameter settings for our specialist on that problem (for the crossover type, we choose the obvious leader). We then did 25 runs on each problem for each specialist parameter setting by locking those parameter values into the meta-GA. This is done by restricting the search range of the meta-GA to the desired outcome. The parameter settings for each specialist are shown in Table 3.

Table 3. These parameter settings, chosen based on the data in Table 2, serve as the 'specialist' parameter configurations

	Genomic Mutation Rate	Crossover Type	Crossover Rate	Selection Type	Trnmnt Size
Counting Ones Specialist	2.2	Uniform	89%	Trnmnt	24
4-bit Trap Specialist	1.5	2pt	86%	Trnmnt	22

We display the average fitness of the specialist parameter settings across 25 runs in Figure 3.

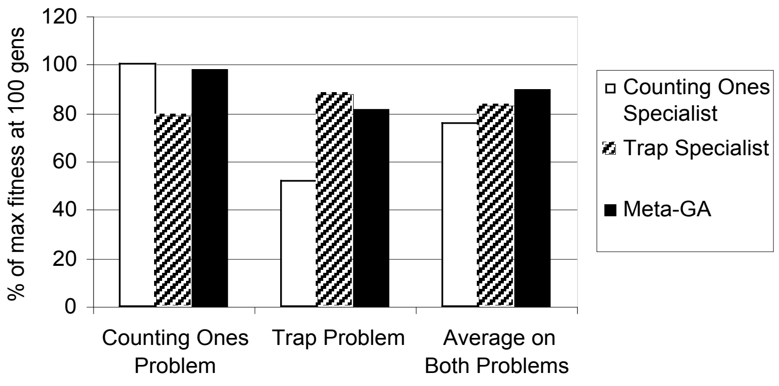


Fig. 3. Fitnesses are averaged over 25 runs for each parameter configuration (counting ones specialist, trap specialist and meta-GA) on two different problems (counting ones and trap). The specialist parameter configurations are taken from Table 3. For both configurations a 'specialist average' is computed, which is the mean fitness of that specialist configuration across both problems, and compared to the performance of the meta-GA. For each problem the parameter configuration specialized to that problem performed significantly better than the meta-GA ($p<.001$) which in turn performed significantly better than the parameter configuration specialized to the other problem ($p<.001$). The average score, however, of the meta-GA across both problems is much better than that of either specialist parameter configuration ($p < .001$)

If the problem was solved before the hundredth generation, a fitness of 100% was reported (this only happened with the counting ones specialist parameter configuration on the counting ones problem, and the earliest a problem was solved was in the 90th generation). These data fall perfectly in line with our expectations. Each specialist does better than the meta-GA on the problem it is tuned to, for the meta-GA has to pay the cost of learning a good parameter set and is at a disadvantage up to that point. The meta-GA also remains at a disadvantage as long as there is no temporality in the problem, for it is exploring other parameter sets instead of exploiting the one it has found. Note, though, that the difference between the specialist and the meta-GA is not tremendous. Each specialist, however, performs poorly relative to the meta-GA on a problem different from the one it is specialized for. Of particular interest is the finding that the meta-GA does better averaged across both problems than using either of the specialist parameter settings ($p < .001$). This evidence supports the general conclusion that meta-GAs may be appropriate as generalist problem solvers since they can function on different problems and achieve performance close to parameter settings specifically tuned to that problem.

The next question we investigated is whether problems exist that require different parameter settings at different phases in the run and, if so, whether the meta-GA exploits this property. Intuitively, this makes sense. It is not hard to imagine that one type of parameter setup is preferable at the beginning of a run, where a wide net should be cast, and a different setup is desirable at the end of a run, where a local search around a near-optimal solution should be performed. Evidence for this would be two or more shifts in parameter settings within the course of a run that are repeated across runs (that most runs move to a given parameter setup once is to be expected, since they should move from the initially random settings to a good set, but if all runs switch from this first setup to a second setup, this is evidence that what constitutes a ‘good’ setup has changed). We found this to be the case if we make the trap problem harder by increasing the number of traps three-fold. We call this the ‘long trap’ problem because it involves increasing the chromosome length from 1000 to 3000. The results show clear differences between parameter settings gleaned from earlier and later stages in the runs (Table 4).

Table 4. The average parameters produced by the meta-GA at 200 generations (earlier in the run) and 400 generations (later in the run) on the long trap problem. Note the dramatic switch from 2pt crossover to uniform crossover. This change occurred in all 25 runs. The mutation rates are also different ($p < .001$). That the configurations are so divergent indicates a preference by the meta-GA for dissimilar parameter settings at different times during the run. Numbers are rounded and therefore do not necessarily total to 100%

	Early Parameters	Late Parameters
Genomic Mutation Rate	2.22	.96
1pt Crossover	7%	3%
2pt Crossover	85%	6%
Uniform Crossover	7%	90%
Crossover Rate	88%	89%
Tournament Size	25	23
Fitness	69%	76%

In all 25 runs, the preferred crossover type shifts from two-point crossover to uniform crossover. Figure 4 shows a representative run.

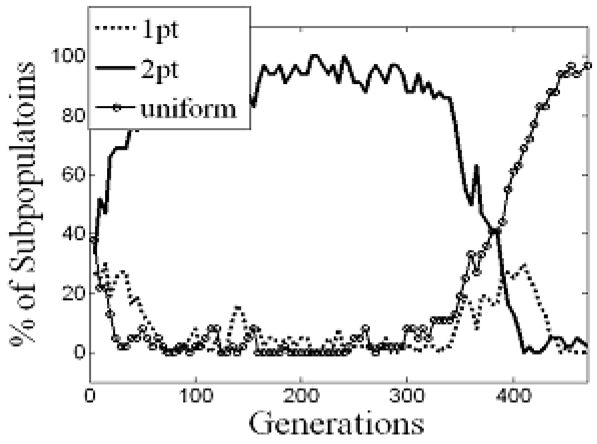


Fig. 4. The percent of 36 subpopulations employing a given crossover type over the course of one run. The switch seen here from 2pt to uniform crossover happened in 100% of the 25 runs

The mutation rates are also significantly different ($p < .001$). It seems clear, then, that the meta-GA prefers different parameter choices at various times during the run. If this mid-course correction benefits performance, then the meta-GA should outperform the parameter settings gleaned either before or after the change. To test this theory we used the average genetic operator settings taken at 200 generations and at 400 generations (Table 4), again choosing the obvious leader for crossover type, and did a series of 25 runs on the long trap problem with each of those parameter settings. We looked at the average fitness for each parameter set at generation 500, which is just after all runs stopped improving and thus represents the maximum fitness reached, and report the results in Figure 5.

This interesting result is counter to our expectations. Despite the fact that 25 out of 25 runs change their crossover type (amongst other things), performance would have been better had they retained the parameter configuration they had before this switch. While more investigation is called for to determine the reason for this phenomenon, we suspect that this is an example of the shortsightedness of evolution. Converting to uniform crossover must confer some short-term benefit, or it would not take off so reliably, but that must confine the populations to a local optimum that is worse in the long run than had they stayed with their originally dominant crossover type.

An alternate explanation is that it is the difference in other parameters that offsets the gain of moving over to uniform crossover. For example, as we will see in the next experiment, mutation rate settings higher than those the meta-GA chooses may lead to increased performance. Note that the mutation rate in the ‘early parameter’ setup is much higher than in the late parameter setup.

These results complicate our assessment of the usefulness of meta-GAs. On the one hand, there is evidence here for problem temporality and for the fact that the meta-GA adapts to that temporality. On the other hand, this ability to adapt can also lead the meta-GA to get stuck in local optima. It seems that this is a double-edged sword, like most things in evolutionary computation (and searching in general) [7], [17]. Adding a dimension increases both the potential for success and the potential for getting stuck.

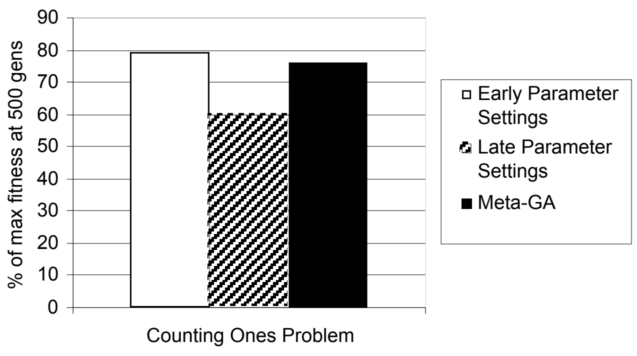


Fig. 5. Fitness averages over 25 runs are reported for parameter settings at 200 generations (early parameter settings) of the meta-GA on the trap problem and again at 400 generations (late parameter settings). While the meta-GA beats the average of these two scores, it performs worse than the early parameter settings.

That the meta-GA is choosing a strategy that is suboptimal raises the question of whether it is truly finding good parameter settings. Is it often the case that meta-GAs, like normal GEAs, get stuck on suboptimal peaks and cannot get off them? Our final experiment tests this hypothesis further by comparing evolved mutation rates against higher ones that scientists setting up GEAs often use. We took the parameters from the 100th generation of the meta-GA running on the original trap problem (chromosome 1000-long, Table 3) and doubled the mutation rate. The other parameter settings were left unmodified. We then compared their performance on the trap problem and report the findings in Table 5.

Table 5. Comparison of the average fitness for the parameters evolved by the meta-GA at 100 generations on the trap problem versus those same parameters with twice the mutation rate. This test shows that injected knowledge commonplace in the GEA community, such as the benefit of higher mutation rates, improves upon the effectiveness of the parameter configurations evolved by the meta-GA. This casts doubt on the ability of the meta-GA to discover good parameter configurations

	Mutation Rate	Average Fitness
Evolved Parameters	1.5	81%
Higher Mutation Rate	3	94%

Clearly the meta-GA has not found the optimal parameter settings. The slight injection of the general knowledge that higher genomic mutation rates than 1.5 are frequently good for GEA parameter settings improves the evolved parameter set. This is an indication that meta-GAs, at least in the form used here, cannot be relied on to discover optimal (or near optimal) parameter settings on their own.

4 Conclusions and Future Work

Our investigations support the claim that meta-GAs show some promise but have significant shortfalls. On the plus side, we have shown that they can successfully adapt themselves to two different problems. We have also shown that, while there are costs involved in the meta-GA learning the parameters appropriate for a problem, evidenced by the meta-GA performing slightly worse than specialist parameter settings on the problem the specialist is tuned to, these costs are not substantial. Furthermore, the costs can be seen as compensated for by the generality of the meta-GA: In our experiments, the meta-GA performed better average on both problems than either set of specialized parameters did on both problems. The meta-GA can thus be considered a 'Jack of all trades, but master of none.' In situations in which the human investment required to set up runs is more precious than performance, the meta-GA could be preferred.

On the negative side, while arguments have been put forward that a strength of meta-GAs is their ability to optimize parameter settings throughout the course of the run, we found indications that doing so may not benefit long-term performance. We also found that the meta-GA is not discovering great parameter configurations: a slight change to one of the parameter settings resulted in a significant increase in performance.

In future work, we will extend this research to a broader base of problems in the hopes of discovering whether these lessons hold true in general. Additionally, we plan to add more evolvable parameters into the mix and investigate different second-level fitness functions. We are also interested in learning more about what causes the meta-GA to be shortsighted and reliably converge upon suboptimal parameter configurations. If we can learn how to keep natural selection, especially at the meta level, focused on long-term goals without requiring human intervention, we will have gone a long way toward delivering on the promise of GEAs to solve challenging problems on their own.

References

1. Davis, L.: Adapting Operator Probabilities In Genetic Algorithms. In: Grefenstette, J. J. (ed.): Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufman, New York (1989) 61-69
2. Davis, L.: Handbook of Genetic Algorithms. VanNostrand Reinhold, New York (1991)

3. DeJong, K.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Doctoral dissertation, University of Michigan (1975)
4. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Evolutionary Computation 2: Advanced Algorithms and Operators, Institute of Physics Publishing, Bristol (2000) 170-187
5. Grefenstette, J.J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Transactions of Systems, Man and Cybernetics. SMC 16(1) (1986) 122-128
6. Friesleben, B., Hartfelder, M.: Optimsation of Genetic Algorithms by Genetic Algorithms. In Albrecht, R., Reeves, C., Steele, N. (eds): Artificial Neural networks and Genetic Algorithms. Springer-Verlag (1993) 392-399
7. Hart, W. E., Belew, R. K.: Optimizing an Arbitrary Function is Hard for Genetic Algorithms. In: Belew, R. K., Booker, L. B. (eds): Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan-Kaufman, Los Altos (1990) 190-195
8. Hinterding, R., Michalewicz, Z., Eiben, A.: Adaptation in Evolutionary Computation: A Survey. In Back, T., Michalewicz, Z., Yao, X. (eds): Proceedings of the Fourth IEEE International Conference on Evolutionary Computation. IEEE Press, Piscataway (1997) 65-69
9. Kakuz, Y., Sakanashi, H., Suzuki, K.: Adaptive Search Strategy for Genetic Algorithms with Additional Genetic Algorithms. In: Manner, R., Manderick, B. (eds): Proceedings of the Second Conference on Parallel Problem Solving from Nature. Elsevier Science, Amsterdam (1992) 311-320
10. Lis, J.: Parallel Genetic Algorithm with Dynamic Control Parameter. In: Proceedings of the 1996 IEEE Conference on Evolutionary Computation. IEEE Press, Piscataway (1996) 324-329
11. Lis. J., Lis., M.: Self-adapting Parallel Genetic Algorithm with the Dynamic Mutation Probability, Crossover Rate, and Population Size. In: Arabas, J. (ed): Proceedings of the 1st Polish National Conference on Evolutionary Computation. Oficina Wydawnica Politechniki, Warszawskiej (1996) 324-329
12. Schlierkamp-Voosen, D., Muhlenbein, H.: Strategy Adaptation by Competing Subpopulations. In: Davidor, Y. (ed.): Proceeding of the Third Conference on Parallel Problem Solving from Nature. Springer-Verlag (1994) 199-209
13. Smith, J. E., Fogarty, T. C.: Operator and Parameter Adaptation in Genetic Algorithms. Soft Computing (1997) 1(2):81-87
14. Srinivas, M., Patnaik, L. M.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. IEEE Transactions on Systems, Man and Cybernetics, 24-4. (1994)
15. Wang, G. Goodman, E. D., Punch, W. F.: On the Optimization of a Class of Blackbox Optimization Algorithms. Proc. IEEE International Conference on Tools for Artificial Intelligence. (1997)
16. Wang, G., Dexter, T., Punch, W., Goodman, E. D.: Optimization of a GA Within a GA for a 2-Dimensional Layout Problem. Proceedings, First International Conference on Evolutionary Computation and its Applications. Presidium, Russian Academy of Sciences. (1996) 18-29
17. Wolpert, D. H., Macready, W. G.: No Free Lunch Theorems For Search. Technical Report. Santa Fe Institute, Santa Fe (1995)

Acknowledgements: Our work was supported in part by an NSF grant (DEB-9981397) to Richard Lenski, Charles Ofria, and colleagues. We would like to thank them and Robert T. Pennock for providing a stimulating environment in which do to research. Further support came from the College of Arts and Letters Dean's Recruitment Fellowship and a Graduate Research Award from the Quantitative Biology & Modeling Initiative at Michigan State University. We also thank all the members of the digital evolution group at Michigan State University.