Memetic Crossover for Genetic Programming: Evolution Through Imitation

Brent E. Eskridge and Dean F. Hougen

University of Oklahoma, Norman OK 73019, USA
{eskridge,hougen}@ou.edu,
http://air.cs.ou.edu/

Abstract. For problems where the evaluation of an individual is the dominant factor in the total computation time of the evolutionary process, minimizing the number of evaluations becomes critical. This paper introduces a new crossover operator for genetic programming, *memetic crossover*, that reduces the number of evaluations required to find an ideal solution. Memetic crossover selects individuals and crossover points by evaluating the observed strengths and weaknesses within areas of the problem. An individual that has done poorly in some parts of the problem may then imitate an individual that did well on those same parts. This results in an intelligent search of the feature-space and, therefore, fewer evaluations.

1 Introduction

As genetic programming is being applied to problems where the evaluation times dominate the total compuation time, the feasibility of finding solutions in a reasonable time frame becomes increasingly important. This requirement becomes even more evident when genetic programming is applied to domains such as robot control (e.g., [1]) where the time to evaluate an individual's fitness many times is prohibitively expensive, as the evaluation time is often orders of magnitude larger than the remaining computation time required by genetic programming. In such cases it is crucial that the maximum amount of information be extracted from each evaluation, even at the cost of greater computation time.

For these reasons we introduce *memetic crossover*. Memetic crossover is based on imitation—people learn, not only from their own direct experiences with the world, but also by following patterns, models, or examples they have observed. This imitation is an active process by which people acquire cultural knowledge.

This new approach extracts more information from an individual than just its overall fitness. It allows the system to evaluate how portions of the individual perform in specific areas of the problem (referred to as *sub-problems*). This information allows the system to intelligently combine individuals by identifying the most promising portions of each one. As a result, fewer evaluations are required and the feasibility of finding a solution in a reasonable time is increased.

2 Related Work

Dawkins [2] proposed the evolution of cultural knowledge through the use of cultural replicators he termed *memes*. The concept of *memetic algorithms* was proposed by Moscato [3] in an effort to model evolutionary computation on cultural evolution instead of biological evolution. Moscato recognized that there are at least two great differences between biological evolution and cultural evolution: (1) individuals cannot choose their own genes, whereas memes can be acquired intentionally, and (2) individuals may modify and improve upon the memes that they acquire, whereas they cannot do so with their genes.

Hougen et al. [4] noted that Moscato chose to implement only the second difference and, therefore, presented a new approach called memetic *learning* algorithms that implements the first difference. In this approach, individuals are given the opportunity to learn through the imitation of other individuals. An individual will imitate when it is able to find another that was more successful than itself in some situation. In the present paper and, concurrently, elsewhere [5] we extend the concept of memetics from genetic algorithms to genetic programming. (In that other work we consider the artificial ant problem using the quite difficult Los Altos trail and the royal tree problem. Here we consider the artificial ant problem using the better-known Santa Fe trail and the bumblebee problem.)

The use of Koza-style Automatically Defined Functions (ADFs) [6,7] restricts the crossover operator to operate within the branches of the individual, namely the main execution branch and the defined functions. However, the selection of the branches or nodes on which to perform crossover is done randomly.

Work has been done on restricting crossover to nodes that reside in similar contexts within the individual [8]. Here again, emphasis is placed on the genotype of the individual, namely the locations of the nodes, not the phenotype.

Langdon [9] tracks the progress of an individual while it is being evaluated in an effort to only perform crossover on trees that have performed poorly in a particular portion of a problem. However, once a tree has been selected for crossover, node selection is performed randomly.

The automatic discovery of subroutines during the evolutionary process [10] has been investigated as a way to detect useful building blocks within an individual. This approach requires two methods of analysis. First, viable individuals are selected when they demonstrate marked improvement in fitness with respect to the fitness of their worst parent. Second, possibly useful subroutines are chosen by their relatively high execution rate within the individual. This approach differs from our approach in that only potentially successful areas are identified, not successful and failed nodes, and the fitness of the various potential subroutines is inferred from the overall fitness of the individual, not observed directly.

While multi-objective fitness [11] attempts to utilize more information learned from the evaluation of an individual than a simple fitness value, it does not delve into the individual itself in an effort to determine the fitness of its components.

3 Proposed Approach

Memetic crossover is directly inspired by Dawkins' idea of the acquisition of memes through imitation [2]. For imitation to be effective in reducing the amount of time spent learning, it must be able to discriminate between parts that lead to success and parts that lead to failure.

Memetic crossover achieves this requirement through a three-step preparatory process. First, the problem must be broken down into *sub-problems*. These sub-problems can be pre-defined, determined upon initialization, or observed during execution. Second, the evaluation order of nodes within each sub-problem is tracked during the fitness evaluation of the individual. Note that this tracking in no way affects the process of evaluating fitness of the individual. Third, for each node, the sub-problems for which it contributed to success and failure are analyzed and used to rank the nodes for each result category. In standard crossover, a bias towards selecting non-terminal nodes 90% of the time is made. We favor this option so a multiplier is applied to non-terminal nodes during the ranking process. This allows us to bias the ranking without falling back to random selection (as in standard crossover).

When a *recipient* individual is selected for memetic crossover, an effort is made to find a compatible *donor* individual. The sub-problem performance of the worst nodes of the recipient and the best nodes of the donor are compared. If the match between the sub-problems for a pair of nodes exceeds the threshold value for that generation, the individuals are considered compatible and crossover occurs with the recipient receiving the donor's "good" node. If a compatible donor is not found after a predetermined number of selections, the donor containing the best found match is used. When appropriate, the required match threshold between nodes starts low and increases with each generation. This is done to counter the relatively high failure rates in sub-problems early on that result from the fact that the initial population is generated randomly. As individuals become more successful in sub-problems, the probability of finding a compatible match between nodes increases.

It is important to note that the memetic crossover operator does not require any problem-specific information. Sub-problems are only referred to by an identifier for the matching process. Thus, no passing of problem information to the operator is required. While the addition of problem-specific information is not prohibited, this was not done in any of the experiments presented so as not to give our approach an unfair advantage.

4 Experiments

To evaluate memetic crossover, two standard problems were chosen. The artificial ant problem was chosen because it can easily be decomposed into unique sub-problems. The bumblebee problem was chosen because the sub-problems are not unique.

4.1 The Artificial Ant Problem

Background. For our first test of memetic crossover, we use the standard genetic programming benchmark problem of the artificial ant using the Santa Fe trail [6]. The goal is to evolve a program that can successfully traverse a food trail with gaps. Fitness is calculated as the amount of uneaten food that remains after traversal. It is important to note that the program may be evaluated as many times as is necessary to either eat all the food or reach the movement limit.

Configuration. First we must find the sub-problems involved in order to adequately evaluate the fitness of the individual's components. For this experiment, we define a sub-problem as navigating a gap in the trail (i.e., eating the food items on either side of the gap in succession). The trail is analyzed upon initialization and the gaps are labeled. Note that there is no preferred order of traversal for the gap. To have a preferred order, the trail would have to be analyzed by traversing it, thus negating the reason for evolving a solution.

As the individual is evaluated, the order of execution of the nodes in the evaluation is recorded. If a node is involved when the ant eats the food on either side of the gap in succession, it is labeled as contributing to the successful completion of the sub-problem. If, however, the node is involved when the ant wanders off the path at a gap and runs into the trail at another point, it is labeled as contributing to the failed completion of the sub-problem. Once evaluation of the individual is completed, nodes are ranked in two categories: best and worst. The top N (in our case, 5, see section 5) are saved for use in crossover.

The parameters used for the system are the same as those used by Langdon and Poli [12]. The baseline implementation uses 90% standard, one child crossover and 10% elite cloning. It is also important to note that instead of the 400 maximum moves used by Koza [6], a value of 600 was used. To adequately explore the effects of varying the amount of memetic crossover and the non-terminal multiplier, several values for each variable were evaluated. Memetic crossover contributions of 10% to 90% were used in conjunction with 10% cloning—the remaining percentage being standard crossover. Non-terminal multiplier values of 1 to 9 were used for each memetic crossover percentage evaluated. For each combination, including the standard implementation, 500 runs were made.

Results. For the standard crossover alone, 465,500 of individuals need to be processed to yield a solution 99% of the time (Fig. 1), which is consistent with Koza's result [6] of 450,000 individuals. When 30% of a generation is created with memetic crossover and a non-terminal multiplier of 7, only 322,000 individuals need to be processed, a 30% reduction (Fig. 2). On average for all non-terminal multipliers evaluated for a 30% memetic crossover contribution, 394,500 individuals need to be processed, a 15% reduction.

As the memetic crossover contribution is increased, however, a different picture arises. At the extreme of 90% memetic crossover and 10% cloning, only

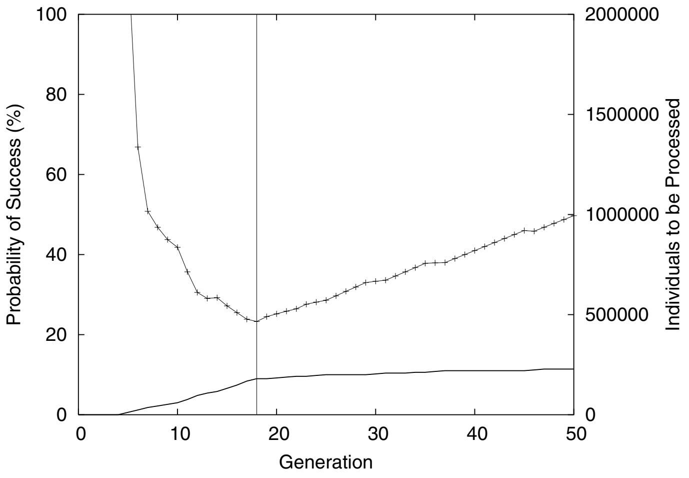

Fig. 1. Performance curves for the ant problem with no memetic crossover. The vertical line indicates E = 465,500, the minimal number of individuals to be processed to yield a solution 99% of the time. The solid plot represents the cumulative probability of success $P(M, i)$ and the marked plot represents the number of individuals that need to be processed.

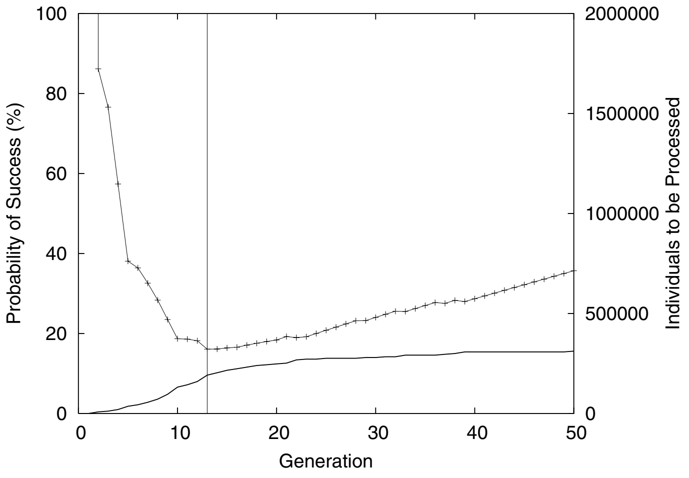

Fig. 2. Performance curves for the ant problem with 30% memetic crossover and a non-terminal multiplier of 7. (E = 322,000 individuals)

two solutions were found in 4,500 runs and those were found in the initial generation. While initially surprising, the result is logical. Without the standard crossover, there is no exploration of the search space. All the individuals are trying to imitate one another and none are attempting to learn anything new.

Of the percentages examined, a 30% memetic crossover contribution showed the best performance (Fig. 3). However, even with memetic crossover, the problem remains quite difficult as less than 20% of runs end in success.

Running Time. Because some researchers may be interested in applying memetic crossover in problem domains in which evaluation time per individual may not be the dominating factor, we also looked at the run-time cost of memetic crossover using the artificial ant problem. An additional 500 runs were performed using the standard crossover method and the memetic crossover configuration that yielded the best results. In the standard case, the total calculated run time to reach a 99% probability of finding an ideal individual was 178,966 ms, while the run time of the memetic case was 218,490 ms. While using memetic crossover for this problem increased the total run time by approximately 22%, it is important to note that this problem represents the extreme lower end of problems in which memetic crossover is applicable as the evaluation time for individuals is quite low (approximately 0.15 ms per individual). Furthermore, we should note that our code has not been optimized for run time and believe that the runtime for the memetic case would decrease should we attempt to optimized the code. In other words, in cases where the evaluation time for each individual is exceedingly small, using memetic crossover may be worse in terms of run time than standard genetic programming methods. However, we are most interested in cases in which evaluation time for each individual *does* dominate. In such a case, memetic crossover has the potential for tremendous speed-up. In the limit (as evaluation time per individual goes to infinity) a 30% reduction in evaluations, for example, would result in an overall 30% reduction in effort.

Table 1. The minimum number of individuals to be processed (E) observed in the ant problem for memetic crossover rates of 10% to 90% and non-terminal multiplier values of 1 to 9. The standard, non-memetic approach yields a result of E = 465,500. "MAX" indicates that there were no solutions found in any run and E is at the maximum value.

	1	2	3	4	5	6	7	8	9
0.1	427500	448500	351000	375000	360000	390000	389500	390000	382500
0.2	357000	348500	434000	392000	389500	429000	360000	480000	399500
0.3	384000	357000	392000	441000	470000	340000	322000	391000	450500
0.4	496000	405000	389500	528000	470000	416500	460000	448000	352000
0.5	527000	374000	563500	427500	413000	442500	498000	462000	512000
0.6	520000	773500	616000	589000	622500	800000	646000	892500	622500
0.7	978000	960500	772500	1059500	870000	700000	1120000	960500	833000
0.8	1287000	1798500	1026000	1137500	1573000	1704000	864000	1910000	1040000
0.9	MAX	1150500	MAX	MAX	MAX	MAX	MAX	1150500	MAX

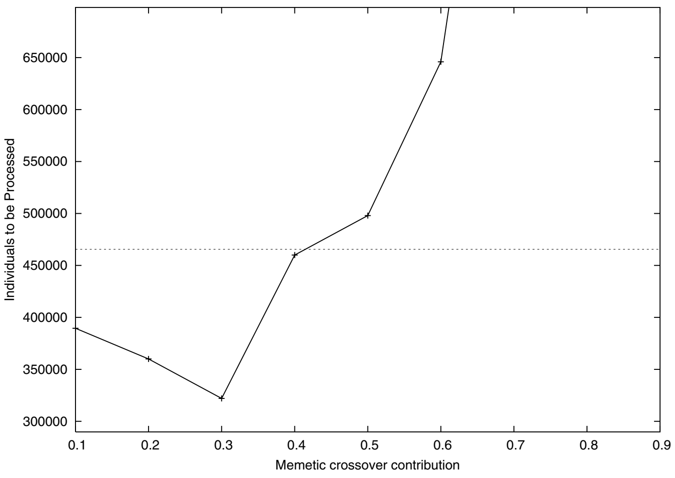

Fig. 3. The minimum number of individuals to be processed for varying percentages of memetic crossover. The dashed line indicates the number of individuals to be processed when no memetic crossover is used. (E = 465,500 individuals)

4.2 The Bumblebee Problem

Background. The second test uses the well-known bumblebee problem [7]. For this problem, the goal is to evolve a program that navigates a bee to all the flowers located in a plane. This problem was specifically designed to be scalable in difficulty. The more flowers that exist in the plane, the more difficult the problem becomes. Since the individual will only be executed once during evaluation, it will be large and full, with its size increasing as the difficulty of the problem increases. For the flower counts used in this experiment, the bumblebee problem is much easier to solve than the artificial ant problem.

Configuration. Since the locations of the flowers for each fitness case are determined randomly upon startup, individual sub-problems—navigating from one flower to another—will have to be observed during execution and not determined beforehand. Furthermore, there is no preferred order of traversal for the flowers as the next flower to visit is generated randomly each time. Since enumeration of all the possibilities would be prohibitive as the problem scales, we only consider the total number of successful and failed sub-problems for each node.

For this problem, we define successful completion of a sub-problem as navigating to the next flower using the minimum number of moves. Since all movement is orthogonal in the plane, the minimum number of moves in the general case is two. Each node's execution is tracked as the individual solves a sub-problem. Once the individual has been evaluated, the top N nodes in each category are saved. The memetic crossover implementation used for this problem is similar to that of the ant problem with one exception—the specific sub-problems of the donor and recipient nodes are not compared with one another since sub-problems are not unique. We predict that this will hinder the effectiveness of the memetic approach as we are less able to gauge effectively the match between nodes.

The parameters used for the system are the same as described by Koza [7]. The baseline implementation uses 90% standard crossover and 10% elite cloning. Again, several values for the memetic crossover contribution and non-terminal multiplier were used. Values ranging from 1 to 5 were used for the non-terminal multiplier because they seemed to provide the most interesting insight. A value of 9 was also used since standard crossover is biased and selects non-terminals 90% of the time. A total of 60 runs were made for each combination of variables.

Results. As was predicted, the benefit of memetic crossover was hampered by the the operator's inability to to create an effective match between the crossover nodes. While the purely standard crossover approach on a 10-flower problem yielded a result of 80,000 individuals requiring processing, the best memetic crossover approach yielded 68,000. This is certainly an improvement over the standard method, but not nearly as dramatic as the previous results.

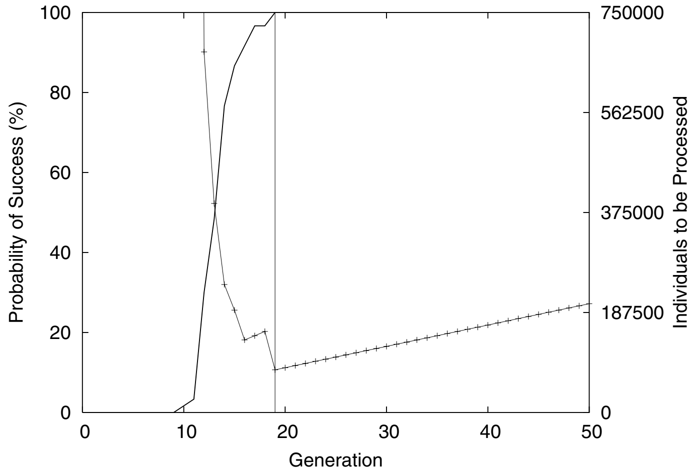

Fig. 4. Performance curves for the bumblebee problem with 10 flowers using standard crossover. (E = 80,000 individuals)

For the 15-flower version of the problem, the addition of memetic crossover yielded a much better result, 128,000 individuals to be processed as opposed to the 176,000 individuals of the purely standard method. However, the benefit only appeared in standard and memetic crossover combinations with relatively low contributions from memetic crossover. In both versions of this problem, the negative impact of a purely memetic approach was not as apparent as in the ant

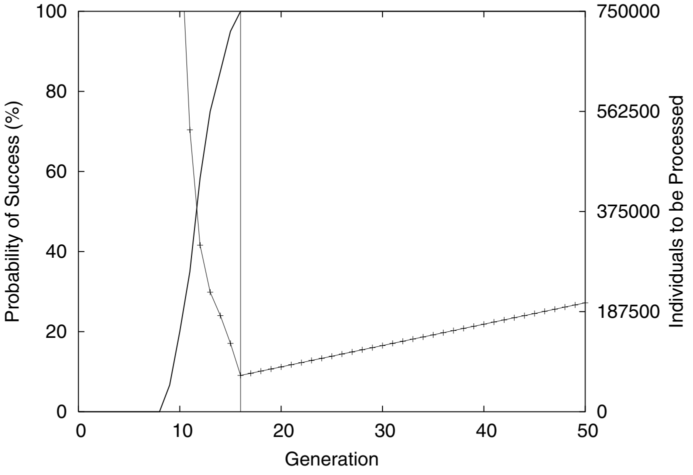

Fig. 5. Performance curves for the bumblebee problem with 10 flowers using 30% memetic crossover and a non-terminal multiplier of 3. (E = 68,000 individuals)

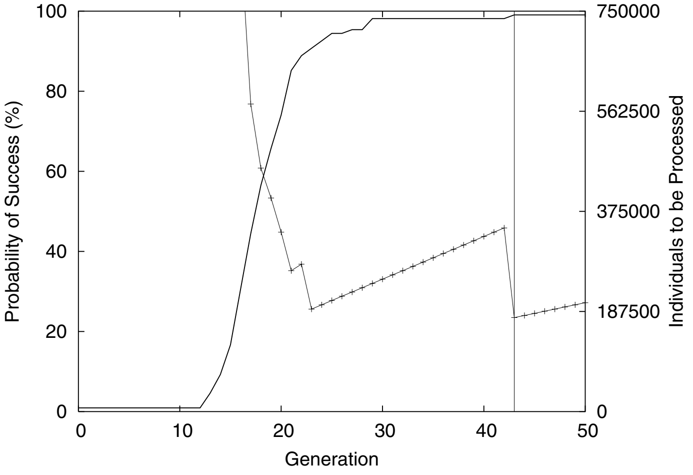

Fig. 6. Performance curves for the bumblebee problem with 15 flowers using standard crossover. (E = 176,000 individuals)

problem. This is most likely a result of the selection of crossover nodes being far more lenient and choosing a broader range and context of nodes.

5 Discussion

For both problems, the addition of the memetic crossover reduced the the number of individuals that are required for processing. The nature of the ant problem,

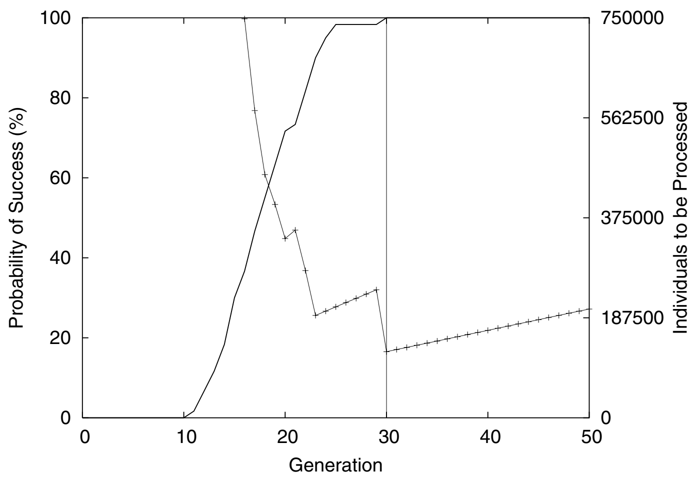

Fig. 7. Performance curves for the bumblebee problem with 15 flowers using 10% memetic crossover and a non-terminal multiplier of 5. (E = 128,000 individuals)

namely its easy decomposition into reasonable sub-problems, allowed it to exploit memetic crossover to a greater degree than the bumblebee problem. In the ant problem, memetic crossover was able to effectively match a failure-prone node to a success-prone node based on the related sub-problems. Thus, individuals were able to imitate the actions of another within a particular set of sub-problems. Since the sub-problems for the bumblebee problem were far less specific than for the ant problem, memetic crossover's impact was limited. Since there was no particular context associated the success or failure of a sub-problem, the imitation exhibited by individuals was far less intelligent. This disparity is not readily evident in the results as the improvement in the bumblebee problem is almost as dramatic as that in the ant problem. We believe that this is because each sub-problem requires effectively the same solution, minimizing the lack of context for sub-problems. Furthermore, for the complexities used, the bumblebee problem is much easier to solve with genetic programming than the ant problem.

The choice of retaining N = 5 nodes in each category for use in crossover was made to allow for multiple opportunities to find a effective match while not requiring the system to retain the full list of nodes. For the bumblebee problem, a value of N = 1 could have been used without affecting the results since no context was associated with the node's fitness.

6 Conclusions

We have introduced a new genetic programming operator that allows for an intelligent search of feature-space. Memetic crossover's ability to exploit an individual's successful experiences in a particular sub-problem and allow others the opportunity to imitate the actions that led to that success has a dramatic impact on the time required to find a solution. Its purpose is not to replace the standard crossover operator but to enhance it. The standard crossover operator explores the feature-space in search of better solutions, but has the undesired effect of being destructive much of the time. Memetic crossover allows the population as a whole to better benefit from the knowledge gained through this exploration by propagating these partial solutions to others in context, which reduces the destructiveness associated with crossover. For the memetic approach to reach its potential, the problem being attempted must lend itself to the organized decomposition into sub-problems, for they provide the context in which the crossover operates. Without context, much of the intelligence of the approach is lost.

As was seen, memetic crossover does incur additional processing costs. However, when these costs are considered in the anticipated problem domain where the evaluation time for individuals is orders of magnitude larger than those encountered in the current work, these costs are negligible when compared to the time saved through the reduction in the number of evaluations.

7 Future Work

This paper should be viewed as an introduction to the memetic crossover approach. A number of areas are available as future work. The role of the non-terminal multiplier used in sorting successful and failed nodes is not well understood. Further investigation into exactly how this affects the overall learning in the system is warranted. Currently, selection of the donor uses tournament selection, which is random. If this selection could be made more intelligently, based on its successes with matching sub-problems, the resulting crossover may be more productive. It would also be interesting to apply this approach to problems with strongly typed variables [13]. This would allow node matching to be more accurate and allow for better replacement of poorly performing areas.

We wish to apply memetic crossover to problems, such as robot learning, in which evolutionary approaches are generally considered "off-line" methods due to their need for many expensive learning experiences [14].

Acknowledgments. This material is based upon work supported by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number DAAD19-03-1-0142. All work performed in this paper was done with the use of Sean Luke's ECJ package [15].

References

1. Nordin, P., Banzhaf, W.: Real time control of a Khepera robot using genetic programming. Cybernetics and Control **26** (1997) 533–561
2. Dawkins, R.: The Selfish Gene. Oxford University Press (1976)

3. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P 826, California Institute of Technology, Pasadena, CA (1989)
4. Hougen, D.F., Carmer, J., Woehrer, M.: Memetic learning: A novel learning method for multi-robot systems. In: International Workshop on Multi-Robot Systems. (2003) Avaliable at http://www.cs.ou.edu/∼hougen/mrs2003.pdf.
5. Eskridge, B.E., Hougen, D.F.: Imitating success: A memetic crossover operator for genetic programming. To appear in: Proceedings of the Congress on Evolutionary Computation, IEEE (2004)
6. Koza, J.R.: Genetic programming: On the programming of computers by natural selection. MIT Press, Cambridge, Mass. (1992)
7. Koza, J.R.: Genetic Programming II: Automatic Discovery of Reusable Programms. MIT Press, Cambridge, MA (1994)
8. D'haeseleer, P.: Context preserving crossover in genetic programming. In: Proceedings of the 1994 IEEE World Congress on Computational Intelligence. Volume 1., Orlando, Florida, USA, IEEE Press (1994) 256–261
9. Langdon, B.: Genetic Programming and Data Structures. PhD thesis, University College, London (1996)
10. Rosca, J.P., Ballard, D.H.: Discovery of subroutines in genetic programming. In Angeline, P.J., Kinnear, Jr., K.E., eds.: Advances in Genetic Programming 2. MIT Press, Cambridge, MA, USA (1996) 177–202
11. Coello, C.A.C.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowledge and Information Systems **1** (1999) 129–156
12. Langdon, W.B., Poli, R.: Fitness causes bloat. In Chawdhry, P.K., Roy, R., Pan, R.K., eds.: Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing, Springer-Verlag London (1997) 13–22
13. Montana, D.J.: Strongly typed genetic programming. Technical Report #7866, 10 Moulton Street, Cambridge, MA 02138, USA (1993)
14. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary algorithms for reinforcement learning. Journal of Artificial Intelligence Research **11** (1999) 241–276
15. Luke, S.: (ECJ 10: A java-based evolutionary compuation and genetic programming research system) Avaliable at http://cs.gmu.edu/∼eclab/projects/ecj/.