

Coevolution and Tartarus

Dan Ashlock
Mathematics Department
Iowa State University
Ames, Iowa, 50011
danwell@iastate.edu

Stephen Willson
Mathematics Department
Iowa State University
Ames, Iowa, 50011
swillson@iastate.edu

Nicole Leahy
Genetics Department
University of Georgia
Athens, GA, 30606
nleahy@uga.edu

Abstract—Coevolution is the process of mutual adaptation of two populations. When a difficult optimization is performed with evolutionary computation, a population of adaptive test cases can strongly affect the progress of evolution. This study applies coevolution to the Tartarus task, a grid robot test problem. If the coevolving test cases are viewed as a form of parasite, then the question of virulence becomes an important feature of the algorithm. This study compares different types of parasites for the Tartarus problem. The impact of coevolution in this study is at odds with intuition and statistically significant. Analysis of the different types of coevolution suggests that disruptive crossover has a key effect. In the presence of disruptive crossover, coevolution may need to be modified to be effective. Examples of these modification are presented. The key method of dealing with disruptive crossover is tracking the age of the Tartarus agents. The age of an agent is defined to be the number of selection steps the agent has survived. Using only older agents to drive coevolution of test cases substantially enhances the performance of one of the two type of coevolution studied.

I. INTRODUCTION

Evolutionary computation borrows ideas from biology to design algorithms. Evolutionary algorithms operate by iterating selection, reproduction, and variation on a population of data structures. The choice of selection method, of variation operators, and of data structure all can have substantial impact on performance. Within evolutionary computation there are two broad classes of algorithm. The first has a static or fixed method of evaluating the fitness of a data structure as part of the selection process. Function optimizers [14], [9], [11] are a good example of this type of evolutionary algorithm. The second class of evolutionary algorithm makes the quality of a structure depend on the other members of the current population. Examples of this class of evolutionary algorithms include [5], [16], [15].

The second class of evolutionary algorithms have a coevolutionary flavor. The agents are not solving a fixed problem, but are instead adapting to one another while generating an example of a state of the system (robot soccer, prisoner's dilemma). An optimizer can be improved by giving it a coevolutionary character. The seminal work in this area [10] is an attempt to coevolve smaller sorting networks with a population of lists which viewed the coevolving lists to be sorted as parasites.

In this study we will compare two different types of parasites. The first operate in a situation where fitness is strictly divided between parasite and host. The second operate in a situation where parasite fitness is an abstraction of its effect

on the host population. We will refer to these parasites as *hard* and *soft* parasites respectively.

The hypothesis tested in this study is that soft parasites enhance evolution of the host population more than hard parasites do. This idea is raised in [6] in the context of Hillis's original work on sorting networks. The results of this study point in a different direction from the results obtained there on sorters. An explanation of the discrepancy is advanced and supported. The key difference is that the sorting networks use a crossover operator that is substantially less disruptive than the one used in the genetic programming system of this study. Systems with disruptive crossover require a different design of coevolution.

Parasites that strictly divide fitness with their hosts (hard parasites) have the potential to either supinely roll over and play dead or to utterly overwhelm their hosts. A supine parasite population is one in a local optimum of the space of parasite behaviors in which the hosts get almost all the fitness and the parasites evolve to divide up the residue. Parasites of this type will provide at best modest selection pressure for their hosts and, so, will not encourage evolutionary progress in the hosts. At the other extreme, deadly parasites represent a situation in which the test cases have become so hard that the host population can make no progress against them. Both these extremes are undesirable, and keeping a coevolving population of hosts and parasites balanced between these extremes seems something that might be difficult to maintain.

Soft parasites, rather than directly competing with their hosts, attempt to coexist to some degree. In [6] the reward for defeating hosts was low if few hosts were defeated, high if an intermediate number were defeated, and again low if a large number were defeated. The fitness evaluation for parasites was designed to move the system to a point between supine and deadly parasites. In theory this intermediate point, where parasites are selected to create a fitness gradient for the host population, should yield superior performance. We take a similar approach in this study: the fitness of a soft parasite is the standard deviation of the fitness foregone by the hosts encountering it. Soft parasites are rewarded for distinguishing between hosts, rather than for beating them.

II. OTHER WORK ON CO-EVOLUTION

In [18] several of the issues in this paper are investigated from a different perspective for the games tic-tac-toe and Nim. The paper investigates rewarding rarity in a hosts's ability to

defeat parasites by dividing the fitness available for a given parasite among those hosts that can defeat it. In addition the authors attempt to ensure the ability to defeat parasites that have died out is not lost by maintaining a parasite “Hall of Fame”. The approaches to coevolution presented in that paper are probably synergistic with the ones presented here.

No paper on coevolution would be complete without acknowledging the theoretical [1], [2], [7], [8], [20] and applied [13], [17] contributions of Jordan Pollack and his collaborators in the area.

The current paper not only continues to develop the thoughts from [6] on hard and soft coevolution but adds a potentially critical new thought that helps create effective parasites. If the representation used in an instance of evolutionary computation has disruptive variation operators then the age of hosts used for the selection of parasites has a substantial impact. As is documented subsequently, including young hosts in parasite selection yields bad performance. We conjecture that such young hosts are “too easy” in many cases because of variation-operator induced incompetence.

III. TARTARUS

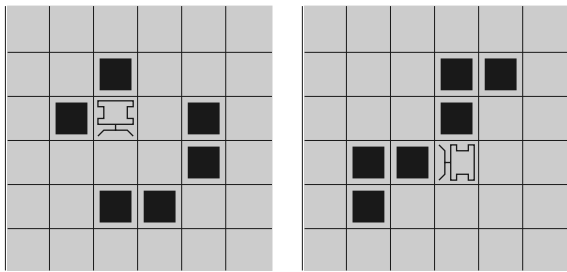


Fig. 1. Valid starting configurations for Tartarus

The *Tartarus* problem [19] is a standard evolutionary computation test problem. Tartarus takes place on a 6×6 grid surrounded by impenetrable walls. Six boxes are placed away from the walls with no boxes forming a close group of four boxes. An agent called the *bulldozer* is also placed on the grid, away from the walls. Examples of starting configurations for Tartarus are shown in Figure 1. During testing on a given board, the bulldozer is permitted 80 moves. On each of these moves, the bulldozer may turn left, turn right, or advance. The bulldozer may push one box ahead of it, but may not push two boxes or push a box through a wall. The bulldozer’s goal is to place box sides against the walls. A box in a corner scores two points while a box against the wall away from a corner scores one point. The maximum possible score on a board is thus 10 points. Figure 2 shows examples of final boards. The bulldozer is given information about the eight adjacent grids to plan its actions.

A good deal of past work on Tartarus exists, e.g. [19], [12], [4], [3]. The prohibition on having boxes in a close group of four deserves some discussion. Since the bulldozer cannot move any boxes in such a configuration, boards containing this configuration were judged to be of little worth in evaluating

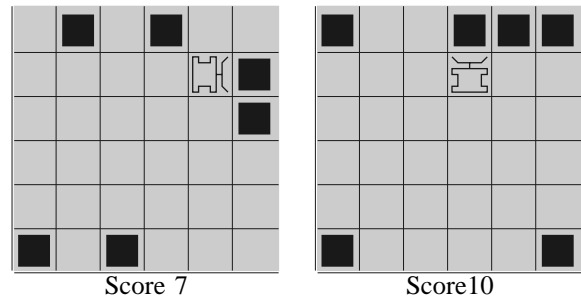


Fig. 2. Examples of final configuration with their scores

fitness. Stephen Willson discovered an additional configuration with equal worth to a close four in evaluating fitness. Examples of these configurations are shown in Figure 3. Out of a desire to use the standard Tartarus problem, these *Willson configurations* were not excluded in this study. The analysis of the outcome of the experiments in this study will require an enumeration of Tartarus configurations including Willson boards.

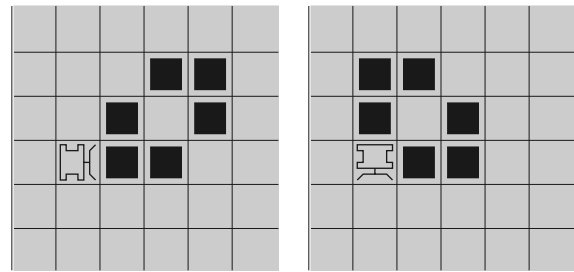


Fig. 3. Two Willson configurations

Theorem 1: (Enumeration of Configurations) There are 320,320 ways of laying out a Tartarus board at all. A total of 23,280 of these boards contain close fours. The number of valid starting configurations for standard Tartarus is thus the difference, 297,040. Of the valid Tartarus configurations, 288 are Willson configurations.

Proof:

There are 36 squares on the Tartarus board, 16 of which are a 4×4 square away from the wall. Placing six boxes can thus be done in $\binom{16}{6} = 8008$ ways. A close four can be placed with its upper left corner in 9 places and still fit in the 4×4 square where boxes may be placed. There are 12 remaining squares and two remaining boxes yielding $\binom{12}{2} = 66$ ways of placing the remaining two boxes. Note, however, that we can form additional close fours within a board if the two remaining boxes are placed to form a 3×2 block of six, and, so, these configurations will be double counted by the preceding method. There are 12 ways to place a 3×2 block within the 4×4 block, thus we must subtract 12 to compensate for over-counting and get $9 \cdot \binom{12}{2} - 12 = 582$ ways to place boxes to create a board containing at least one close four. This yields $8008 - 582 = 7426$ ways to place boxes in a valid board. Once the boxes are placed, the bulldozer is placed in one of the ten remaining squares away from the wall in one of four

orientations yielding 40 bulldozer placements per valid box placement. This yields $7426 \times 40 = 297,040$ valid boards. Placing the bulldozer on the excluded boards containing a close four yields $582 \times 40 = 23,280$ excluded boards. The total number of boards is $8008 \times 40 = 320,320$. It remains to count the Willson configurations. A Willson configuration fills a 3×3 area in one of two ways (an example of each is shown in Figure 3). There are four available ways to place a 3×3 area in the 4×4 box placement zone. The bulldozer may be placed in any of the 10 remaining grids *except* the one in the center of the six boxes, in any of four orientations. This yields $2 \times 4 \times 9 \times 4 = 288$ Willson configurations. \square

IV. GP-AUTOMATA

Start: 0→0			
	If Even	If Odd	Deciders
0	1→6	3→2	x_1
1	3→9	1→7	(ITE x_3 ($\sim x_8$) ($\sim x_1$))
2	2→9	1→9	(ITE x_5 x_5 (min (Odd x_3) ($\langle \rangle x_1 -2$)))
3	1→13	3→15	(ITE (Com 1) x_8 (Odd x_4))
4	3→0	0→9	(+ (Odd x_1) (max x_6 x_6))
5	1→15	3→9	(Odd (ITE -1 x_6 (min (Odd x_3) ($\langle \rangle x_1$ x_4))))
6	3→8	0→5	(Odd (ITE -1 x_6 (min (Odd x_3) ($\langle \rangle x_1$ x_4))))
7	1→13	0→4	(ITE x_5 x_5 (min -1 (Odd x_3)))
8	1→17	2→2	(> (ITE ($\langle \rangle x_1$ ($\sim x_8$)) x_8 x_8) x_2)
9	2→11	1→12	(+ (ITE x_1 x_5 x_4) 0)
10	2→2	2→9	(ITE x_5 x_5 (min (Odd x_3) ($\langle \rangle x_1 -2$)))
11	0→10	1→15	(+ (ITE (max x_5 x_1) x_5 x_3) 0)
12	2→5	1→15	(ITE x_3 x_7 (min (Odd x_3) ($\langle \rangle x_1$ x_4)))
13	3→14	1→2	(+ (> x_1 x_5) ($\sim x_5$))
14	3→2	2→18	(> (ITE ($\langle \rangle x_1$ ($\langle \rangle 0$ x_6)) x_8 x_8) x_2)
15	2→5	3→10	(> (ITE ($\langle \rangle x_1$ ($\sim x_8$)) x_8 (Odd x_3)) x_2)
16	3→0	3→18	(+ x_1 0)
17	1→0	2→12	(> $x_4 -1$)
18	0→4	1→5	x_3
19	0→6	2→10	(Odd x_1)

Fig. 4. An example of a GP-automata trained on the Tartarus task

The representation used for bulldozers in this study is described next. A *GP-automaton* is a finite state machine augmented by placing a parse tree representing a formula on each state. The formula is used to interpret inputs to the automaton and is called a *decider*. The parse trees use the

operations and terminals given in Table I. Formally, a GP-automaton consists of a collection of states and their associated deciders, including a distinguished initial state, together with a transition function and a response function. The initial state has an associated initial action. The transition function is used to determine what the next state of the GP-automaton will be, while the response function computes the output of the GP-automaton. Both functions are conditioned on both the current state and on inputs to the automata: in this case, the bulldozer’s sensors as processed by the decider. This permits the decider to function as a state-specific evolvable bandwidth compressor, reducing the 4496 possible sensor inputs to one bit of information (the integer parity of the output of the decider). An example of a GP-automaton trained on the Tartarus task appears in Figure 4. The responses and transitions are given in the form *action* → *next state*. Deciders are given in LISP-like notation. The GP-automata used in this study have 20 states. Preliminary data suggests that the number of states used is not a critical choice. GP-Automata with 12, 20, and 24 states were tried and achieved similar fitnesses on the Tartarus problem.

Alert readers will have noticed that there are three Tartarus actions (left, right, forward) while there are four actions visible in the GP-automaton shown in Figure 4. The action mapping is: 0=turn left, 1=turn right, 2=advance, and 3=think. The *think* action is not a Tartarus action, but, rather, is a form of λ -transition within the GP-automaton. A think action produces no external Tartarus action and does not count against the 80 moves the bulldozer is permitted. On executing a think, the GP-automaton makes a transition and then permits the next state to function normally. The advantage of think actions is that the GP-automaton may, if required, use multiple deciders to decide what action to take. A substantial disadvantage is that think actions permit infinite loops. These are prevented by placing a hard limit, called the *think limit*, on the number of consecutive think actions permitted. If a GP-automaton exceeds the think limit, then the bulldozer it is controlling does not move and prematurely ends its fitness evaluation. The think limit in this study is 8.

The variation (crossover and mutation) operators for GP-automata in this study are as follows. The list of states forms the basis for a crossover operator. The states are treated as atomic objects, making the list of states a linear gene. A two-point crossover of the list of states is used. The initial state and action are associated with the first state and follow it during crossover. This is different from the more traditional sub-tree crossover used in genetic programming. There is no potential for crossover-driven bloat and the offspring of two identical parents is identical to those parents. Such a crossover operator is termed *pure* or *conservative*.

Eight different mutation operators are used in this study. The type of each mutation is selected according to the following scheme to create a master mutation operator. Ten percent of mutations modify the initial state of the GP-automaton. Ten percent modify the initial action. Twenty percent modify a transition by replacing it with a new one selected uniformly at random. Twenty percent replace a uniformly selected action

Operation	Arity	Semantics
I	0	Ephemeral integer constant
$x_0 - x_7$	0	Input or sensor terminal
\sim	1	Integer negation
Com	1	Computes 1-x
Odd	1	Predicate for oddness*
+	2	Integer addition
-	2	Integer subtraction
=	2	Equality*
>	2	Greater than*
<	2	Less than*
>=	2	Greater than or equal to*
<=	2	Less than or equal to*
Max	2	Returns maximum of arguments
Min	2	Returns minimum of arguments
ITE	3	If-then-else: if first argument,** then return the second argument, otherwise return the third argument

*Returns 1 for true, zero for false.

**All nonzero values are considered true.

TABLE I
Operations and terminals of the integer valued parse tree language used in deciders

with a new one selected uniformly at random. Ten percent of mutations replace a decider with a new one generated at random. Ten percent perform sub-tree crossover on two deciders selected uniformly at random. Ten percent exchange two deciders. Finally, ten percent copy one decider over another.

The number of nodes (operations and terminals) in a new, random decider and the maximum size of a decider are set to 6 and 12 respectively in this study. This is less restrictive than it might seem because of the large collection of identities in the operation set chosen. The only way that decider size can change is as a result of sub-tree crossover during mutation of a GP-automaton. If the size of a decider exceeds the maximum decider size, then the decider is *chopped*. Chopping selects an argument (sub-tree) of the root node of a decider uniformly at random and uses it to replace the decider iteratively, until the decider is small enough. The effect of this method of controlling code size is to have a moderate implicit pressure toward economical solutions with solution size strongly controlled by the decider size parameters and the number of states.

V. EXPERIMENTAL DESIGN

Nine collections of 100 evolutionary simulations appear in this study. Initially, three experiments were performed, one baseline, one hard coevolution, and one soft coevolution. These experiments suggested a modification of soft coevolution worth studying and also suggested an additional baseline study might be worth performing.

All nine sets of runs shared much of their structure. A population of 200 GP-automata, with the variation operators described previously, is evolved for 1000 generations. When two automata breed they undergo crossover and are subjected to 1-3 mutations with the number of mutations selected uniformly at random. The model of evolution is single tournament selection with tournament size four. The population is shuffled

randomly into four member tournaments and the two most fit members of each tournament breed and replace the two least fit members of each tournament. Fitness is evaluated on 100 Tartarus boards selected either at random or according to the method of coevolution specified. Fitness of a GP-automaton is the sum of its scores on the 100 boards used in a given generation. In runs that used coevolution, a fitness was computed for each board and the least fit 50% of the boards were replaced with new boards selected uniformly at random. Details of particular experiments are as follows.

- **Baseline 1.** This experiment used 100 boards selected at random to test each generation with no close four boards, but with Willson boards permitted.
- **Hard Coevolution.** This experiment was identical to the baseline, save that boards were chosen by coevolution. The fitness function of a board was the points of fitness, out of ten per board, not earned by the bulldozers it was testing. Fitness was thus strictly divided between the GP-automata and the boards.
- **Soft Coevolution(SD).** This experiment was like the hard coevolution experiment, save for the fitness function used to coevolve the boards. This function was the standard deviation of ten minus the scores obtained by the population of GP-automata on that board. In this case, boards were rewarded for distinguishing GP-automata by achieving high standard deviation.
- **Soft Coevolution(SDE).** This experiment was identical to soft coevolution(SD), save that not all scores against GP-automata were used in computing the standard deviation used for fitness. Only scores from GP-automata that had already survived at least one round of selection (experts) were used. This was to test the hypothesis that new GP-automata are often quite incompetent and, so, serve as a source of fitness for boards that is of little worth in assessing the board's ability to generate a *useful* fitness gradient.
- **Soft Coevolution(SDE2)** This experiment was identical to soft coevolution(SDE), save that only scores against GP-automata that had survived at least two generations were used to compute the fitness of boards.
- **Soft Coevolution(SDE3)** This experiment was identical to soft coevolution (SDE2), save that only scores against GP-automata that had survived at least three generations were used to compute the fitness of boards.
- **Baseline II.** The outcome of the hard coevolution experiments suggested that the rejection of boards containing close fours might have been somewhat hasty. A second baseline study that permitted close four boards was performed.
- **Hard Coevolution 4.** Given the outcome of the second baseline experiment, it seemed worthwhile to see what would happen if close fours were permitted in the hard coevolution situation. This experiment was identical to hard coevolution, save that boards containing close fours were permitted.

- **Hard Coevolution, Expert (Hard E1).** Given the effect of using bulldozers that had survived one round of selection on soft coevolution, it was thought good to check its effect on hard coevolution. In this experiment, hard coevolution was rerun, computing board fitnesses only from bulldozers that had survived at least a single round of evaluation.

VI. RESULTS

Initially, three experiments were run: the first baseline experiment, hard coevolution, and soft coevolution with standard deviation based fitness for boards computed on all GP-automata. Looking at the fitnesses appearing *during the course of evolution*, it appeared that the fitnesses were best for the soft coevolution experiment. The fitnesses for these three experiments are not comparable, having been computed via quite different fitness-case-sampling schemes. For the best-of-run GP-automata from each of the 100 runs comprising each experiment, the average fitness on a fixed 5000 board cross validation set was computed. These cross-validation numbers are shown in Table II and Figure 5. The cross validation showed the initial soft coevolution experiment to be markedly inferior to the standard evolutionary algorithm.

Experiment	Mean	95% CI	Best
Baseline I	6.33	(6.19,6.47)	7.459
Baseline II	6.35	(6.24,6.46)	7.597
Hard	6.66	(6.59,6.73)	7.489
Hard E1	6.52	(6.37,6.67)	7.497
Hard 4	6.58	(6.53,6.70)	7.283
Soft(SD)	3.58	(3.38,3.78)	7.187
Soft(SDE)	6.23	(6.05,6.41)	7.531
Soft(SDE2)	6.65	(6.58,6.73)	7.874
Soft(SDE3)	6.68	(6.62,6.75)	7.262

TABLE II

Mean, 95% confidence intervals of mean fitness, and best fitnesses found for the most fit GP-automata in each of $n = 100$ simulations performed for each of the the nine experiments reported

In trying to think of reasons that soft coevolution performed so badly, the hypothesis that *fragility of representation* was a factor was advanced. GP-automata are, effectively, entire small computer programs. Mutation and especially crossover have the potential for massive disruption of their behavior. This suggests that many of the new GP-automata produced by the variation operators are of very low quality and, as such, may be of little worth in assessing the ability of boards to generate a fitness gradient for competent GP-automata. A simple solution to this problem is to use only information from GP-automata that have survived at least one round of evaluation and selection. These thoughts inspired the soft coevolution expert(SDE) experiments which achieved a fitness with no significant separation from the baseline studies (and still significantly worse than the hard coevolution experiment). The improvement obtained by paying attention only to fitness information derived from GP-automata that had survived at

least one round of selection (experts) is consistent with our hypothesis that fragility of representation is a factor.

In order to find the limit of the advantage of allowing only competent bulldozers to participate in the selection of coevolving boards, the SDE2 and SDE3 experiments were performed. These require additional competence, measured by additional survival time, in the bulldozers used to test the quality of boards. Requiring two generations of survival created a statistically significant impact. Requiring three generations of survival had a small additional impact that was not statistically significant. The SDE3 and hard coevolution runs produced almost identical results.

Excluding Hard Boards, Revisited

The second baseline study was performed as the result of analysis of the boards selected by coevolution. Recall that Willson configurations were permitted to maintain consistency with the initial Tartarus problem specification. In the hard coevolution experiments the boards selected were substantially enriched with Willson configurations. This suggested, given that hard coevolution provided significantly superior results, that the rejection of close four boards was inadvisable. The second baseline study, which did not reject boards with close four groups, was not significantly different from the first baseline study. Thus, we do not reject the null hypothesis that these “hard” boards have little impact on the evolution of GP-automata for the Tartarus problem. An additional set of runs to test the effect of using expert GP-automata in hard coevolution was performed. There was no significant impact.

This is the first time that the assumption that “hard” boards are bad for training Tartarus controllers has been experimentally examined. It makes intuitive sense that training software agents on impossible cases is a poor idea. The flaw seems to be in assuming that the “hard” boards in question are, themselves, impossible. With a maximum possible score of four, they still provide some fitness information. Perhaps more important is the fact that having to deal with such close four boards is good practice for close fours inadvertently created when a misstep is made on an otherwise solvable board.

VII. DISCUSSION AND CONCLUSIONS

This study failed to support the hypothesis that soft coevolution is superior to hard coevolution for the Tartarus task. In the end, hard and correctly designed soft coevolution were indistinguishable from one another and superior to all baseline studies and the initial soft coevolution. It suggests both that the design of an evolutionary algorithm that uses coevolution is a delicate task and that coevolution can yield superior results.”

The hypothesis that fragility of representation is contributing to the poor performance of soft coevolution was advanced. Suppose that we are trying to create a fitness gradient and hence reward parasites (Tartarus boards) that create the higher variation in fitness. In the initial soft coevolution experiment, it may be that these boards are mining very bad GP-automata for this fitness. Since GP-automata used as Tartarus controllers are subject to large disruptions of their performance during

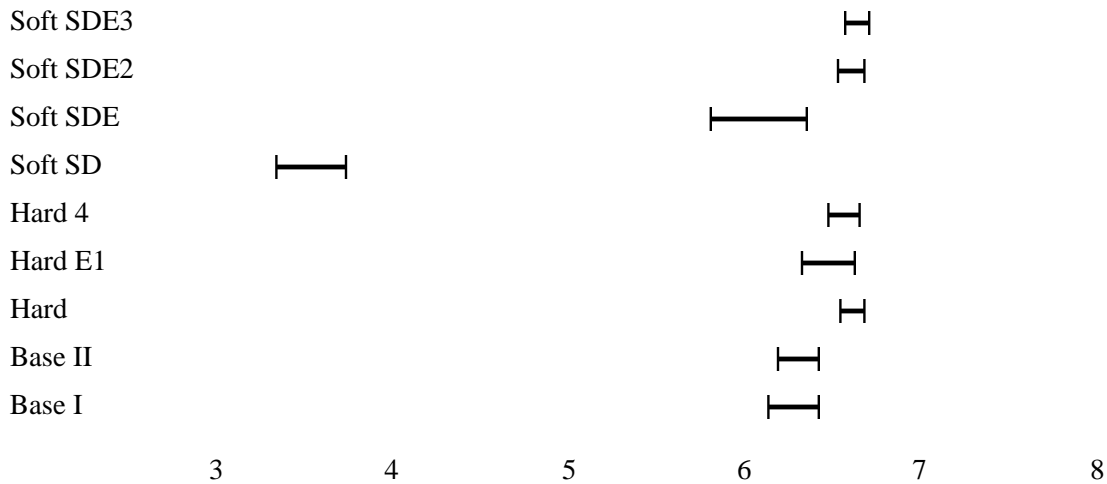


Fig. 5. 95% confidence intervals for the mean best score out of 10 over 100 simulations for each of the ten experiments reported (All fitnesses are computed on the cross validation set.)

breeding, there is quite likely to be a supply of very bad GP-automata to mine for fitness in this fashion. Boards that achieve high variation fitness for populations including idiot GP-automata may be quite easy for the competent GP-automata. In spite of being easy for a competent opponent, these boards obtain high variation-based fitness.

40

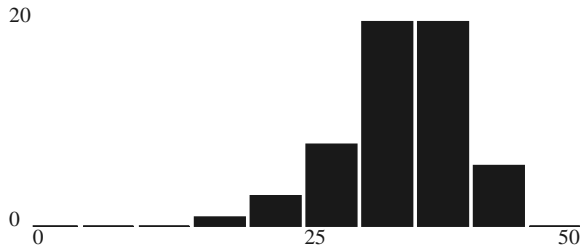


Fig. 6. Histogram of the number of Willson boards in the final population of boards for the 100 simulations run with hard coevolution (The minimum number of Willson boards observed was 18, the maximum was 45.)

When coevolution is used in the presence of fragile representations (disruptive variation operators) only somewhat tested population members should be used in assessing parasite fitness. The tracking of the number of selection steps a Tartarus agent has survived (its age) and use of age information to modify the selection criteria for the boards yielded improved results. This notion of tracking age and using it in coevolution has potential application well beyond the present study.

It was expected that the hard coevolution experiment would

concentrate, via selection, Willson configurations in its evolving population of parasites. These boards permit a maximum fitness of 4 out of 10 and so start with a substantial selective advantage over other admissible boards during hard coevolution. It was also expected that this concentration of Willson configurations would result in poor performance because such boards, which force the bulldozer to create a close four, would provide little useful fitness information. The former expectation was correct; the latter was not.

40

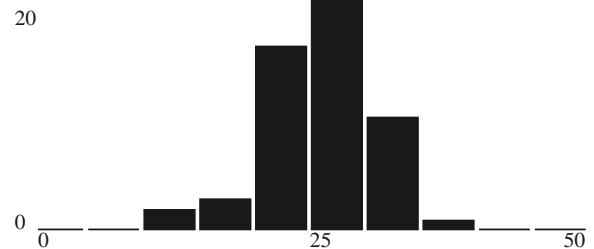


Fig. 7. Histogram of the number of Willson boards in the final population of boards for the 100 simulations run with hard coevolution with age 1 bulldozers. (The minimum number of Willson boards observed was 13, the maximum was 34.)

Figures 6 and 7 show the number of Willson configurations present in the final populations of 100 boards found during the hard coevolution experiments. Fifty of these boards were selected uniformly at random and fifty survive from previous generations. The expected number of Willson configurations in 100 randomly selected boards is 0.097, as we can see from

Theorem 1. The number found varies from 18 to 45. Clearly substantial enrichment is taking place. The observed numbers are what would be expected if all Willson configurations encountered were retained. In the other coevolution experiment, the majority (at least 90) of the final populations of boards had no Willson configurations and the maximum number appearing in any one final population was two. Enrichment was thus restricted to hard coevolution.

The fact that superior performance was obtained in situations where the number of “hard” boards was enriched led to performance of the second baseline study which permitted both Willson configurations (quite rare) and boards with close fours. There was no significant difference between this baseline experiment and the first one performed. With 7.27 close four boards expected in each collection of 100 random boards used to evaluate fitness, it seems that the expectation that these boards are damaging as fitness cases is incorrect.

VIII. FUTURE WORK

The experiments reported in this study suggest several subtleties at work in the coevolution of fitness cases for Tartarus. First of all, the relationship of hard coevolution, soft coevolution, and the baseline algorithm are contrary to observed on sorting networks. Second, the details of how soft coevolution is implemented clearly have a substantial impact on performance. The issue of not permitting untested structures to influence the selection of the fitness cases is important. Fragility of representation is a mechanism that tentatively explains why it may be a poor idea to have entirely new structures influence the evolution of coevolving fitness cases. Given all of this, the following experiments are worth performing.

- Fragility of representation can be directly assessed by checking the average decrease in fitness of bulldozers when challenged with crossover and mutation. In addition to documenting such fragility such studies can be used to document which variation operators are the most disruptive.
- By repeating the experiments done here with an alternative representation, such as ISAc structures[4], we can look for a change in the response to excluding untested structures from those used to evolve the fitness cases.

The disparate results obtained for sorting networks and Tartarus controllers indicate that the soft/hard dichotomy in coevolutionary algorithms absolutely demand testing on additional problems. The problems should be chosen to have a broad variety of disruptiveness in their variation operators.

IX. ACKNOWLEDGMENTS

The authors would like to thank the Iowa State Bioinformatics and Computational Biology program for its support of this research. We would also like to thank the members of the Iowa State Complex Adaptive Systems Program for many helpful discussions.

REFERENCES

- [1] Peter J. Angeline and Jordan B. Pollack. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, 1993. Morgan Kaufmann.
- [2] Peter J. Angeline and Jordan B. Pollack. Coevolving high-level representations. In Christopher Langton, editor, *Artificial Life III*, volume 17 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 55–71, Reading, 1994. Addison-Wesley.
- [3] Daniel Ashlock and Jennifer Freeman. A pure finite state baseline for tartarus. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 1223–1230. IEEE Press, 2000.
- [4] Daniel Ashlock and Mark Joenks. ISAc lists, a different representation for program induction. In *Genetic Programming 98, proceedings of the third annual genetic programming conference.*, pages 3–10, San Francisco, 1998. Morgan Kaufmann.
- [5] Daniel Ashlock, Mark Smucker, E. Ann Stanley, and Leigh Tesfatsion. Preferential partner selection in an evolutionary study of prisoner’s dilemma. *Biosystems*, 37:99–125, 1996.
- [6] John Cartlidge and Seth Bullock. Lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1420–1425. IEEE Press, 2002.
- [7] S. Ficici and J. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In Adami et al, editor, *Proceedings of the Sixth International Conference on Artificial Life*, pages 238–247, Cambridge, MA, 1998. MIT Press.
- [8] S. Ficici and J. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In Cantu-Paz et.al., editor, *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, pages 467–476. Springer Verlag, 2003.
- [9] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.
- [10] W. Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In Christopher Langton, editor, *Artificial Life II*, volume 10 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 313–324, Reading, 1991. Addison-Wesley.
- [11] John H Holland. *Adaption in Natural and Artificial Systems*. The MIT Press, Cambridge, MA, 1992.
- [12] Vasant Honovar and Karthik Balakrishnan. On sensor evolution in robotics. In *GP 96*, pages 455–460. MIT Press, 1996.
- [13] J. J. Pollack, A. Blair, and M. Land. Coevolution of a backgammon player. In *Artificial Life V*. MIT Press, 1997.
- [14] Kenneth De Jong. Learning with genetic algorithms: An overview. *Machine Learning*, 3:121–138, 1988.
- [15] S. Luke, C. Hohn, J. Farris, G. Jackson, and J. Hendler. Co-evolving soccer softbot team coordination with genetic programming. In *RoboCup-97: Robot Soccer World Cup*, pages 398–411. Springer, 1997.
- [16] John H. Miller. The coevolution of automata in the repeated prisoner’s dilemma. *Journal of Economic Behaviour and Organisation*, 29:87–112, 1996.
- [17] Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998.
- [18] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.
- [19] Astro Teller. The evolution of mental models. In Kenneth Kinnear, editor, *Advances in Genetic Programming*, chapter 9. The MIT Press, 1994.
- [20] R. Watson and J. Pollack. Coevolutionary dynamics in a minimal substrate. In Lee Spector et.al., editor, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pages 702–709. Morgan Kaufmann, 2001.