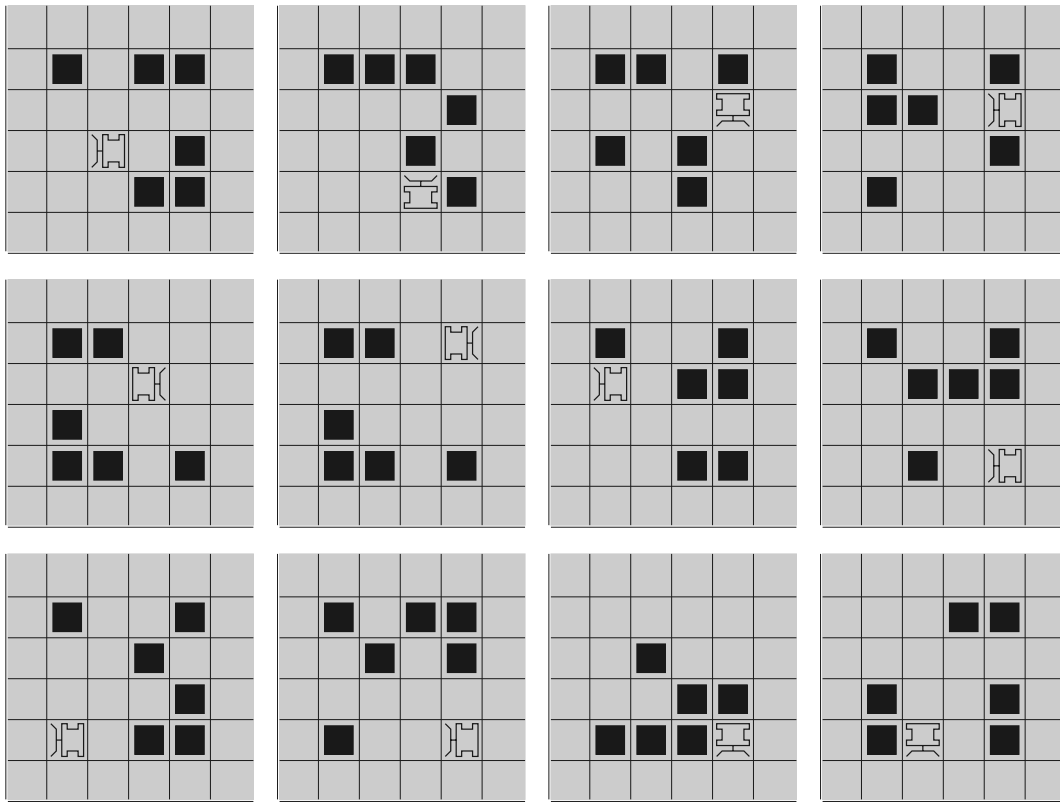# Coevolution and Tartarus

Dan Ashlock, Math, Complex Systems,

Joint work with:

Nicole Leahy, Bioinformatics, Complex Systems,
**Stephen Willson**, Math, Complex Systems.

Co-evolution.

In this study we use a test problem with thousands of problem cases. Using all the test cases in evaluation of each member of an evolving population is not practical. The typical approach is to sample the fitness cases. **Co-evolution** consists of selecting fitness cases via evolution. There are many way to co-evolve fitness cases. We will look at several.
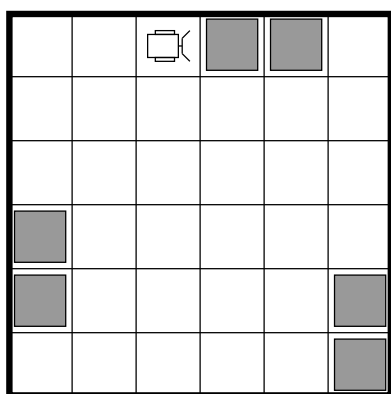
The Test Problem.

The **Tartarus** problem asks an agent, called the **bulldozer**, to move boxes to the corners or edges of a square room. There are six boxes starting away from the walls of a 6 × 6 room. The boxes start with no close groups of four boxes. The bulldozer is permitted eighty moves of the form **forward**, **turn left**, or **turn right**. A starting configuration for Tartarus might look like:
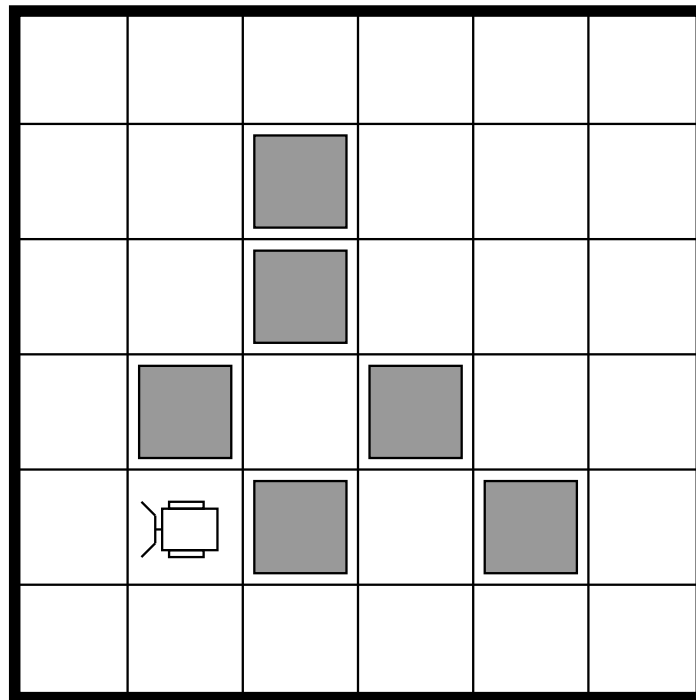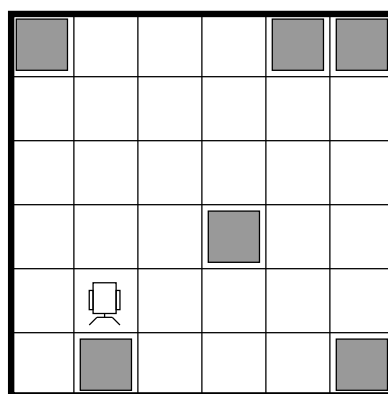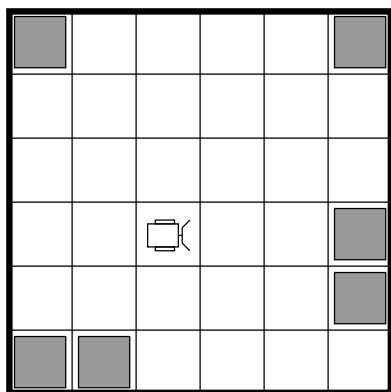
The Test Problem.

The bulldozer can push one box ahead of it. The bulldozer cannot push two boxes ahead of it, nor can it crush a box or push a box through the wall. The objective function used to measure the performance of a bulldozer awards two points for each box in a corner and one for each box against a wall but not in a corner.

Score 7

Score 8

Score 9
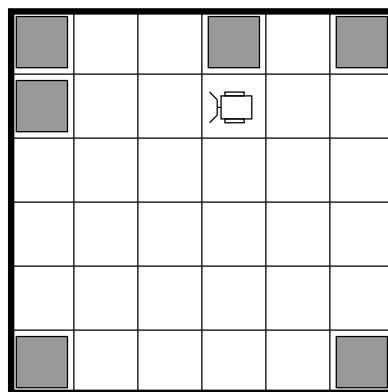
Score 10

The Test Problem.

There are two critical features of a representation for the Tartarus problem: sensory information and memory. Sensory information is conveyed by terminals X_1-X_8 corresponding to the squares adjacent to the bulldozer. The return 0 for space, 1 for a box, or 2 for a wall.

```
   X8        X1        X2



   X7                  X3



   X6        X5        X4
```
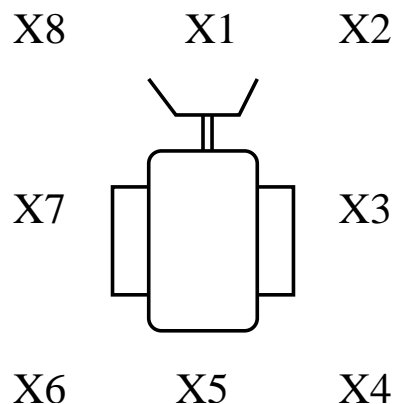
Memory in this study is supplied by a finite state device that is a portion of the bulldozer controller.

The Representation

The input terminals X_1-X_8 produce, in aggregate, thousands of combinations of input data. As a first step any collection of inputs will be processed by an integer formula into a single integer. These formulas use the following operations and terminals:

Op	Args	Semantics
e.g. 1	0	Ephemeral integer constants
x1-x8	0	Sensor terminals
~	1	Negation
Odd	1	Predicate: True if odd*
Com	1	1-*argument*
+	2	Addition
-	2	Subtraction
=	2	Predicate: equality*
>=	2	Predicate: greater than or equal to*
<=	2	Predicate: less than or equal to*
>	2	Predicate: greater than*
<	2	Predicate: less than*
<>	2	Predicate: not equal to*
max	2	Computes maximum
min	2	Computes minimum
ITE	3	If first argument return second argument else return third argument.

* False results are zero, true are nonzero.

The Bulldozer Controller

A *GP-Automaton* is a finite state device that has a formula associated with each state. The formulas reduce information from the environment (in this case $X_1 - X_8$) to a small set of values (in this case integer parity) used to drive the finite state device.

Start: 1→0			
State	If Even	If Odd	Formulas
0	0→10	3→4	(Com (ITE x_8 x_7 ($\sim x_7$)))
1	1→3	1→3	(ITE 0 (Com x_6) (Odd x_3))
2	0→4	0→8	(Odd (ITE x_4 x_5 ($\sim x_3$)))
3	3→5	2→0	(ITE x_1 ($\sim$ -2) (Com (min x_6 (Odd x_3))))
4	0→1	0→2	(> (= 1 0) (Com x_8))
5	1→4	0→3	(ITE ($\sim x_7$) (Com x_3) ($\sim x_6$))
6	0→2	0→5	(> ($\sim$ (Odd ($\sim$ (max x_8 x_7)))) (Com x_8))
7	2→7	2→0	(ITE x_6 (Com -2) (Odd x_4))
8	1→10	0→1	(- (ITE x_6 x_2 x_4) x_7)
9	1→4	3→4	(< (>= x_7 x_5) (Odd x_3))
10	3→1	2→1	(= (min x_6 (ITE x_5 0 x_8)) ($\sim$ (max x_1 x_7)))
11	3→3	2→2	(> ($\sim$ (Odd ($\sim$ (max x_8 x_7)))) (Com x_8))

An example of a GP-Automaton

Actions: 0-advance 1-left 2-right 3-think

Variation Operators on GP-Automata

Crossover of GP-Automata is achieved by treating the list of states as a linear chromosome and exchanging middle segments of the list of states between pairs of automata undergoing crossover.

Mutation does one of the following: Changes a state transition or an output, performs (sub-tree) crossover of two formulas, performs (sub-tree) mutation of a formula, replaces a formula with a new random formula, exchanges two formulas, or copies one formula over another.

Evaluating Bulldozer Controllers

A good bulldozer controller is one that scores well on any valid Tartarus board. Stephen Willson noticed that there are boards, not excluded by the original problem statement, that are as bad as those with a close four configuration of boxes. We call these *Willson Boards*.
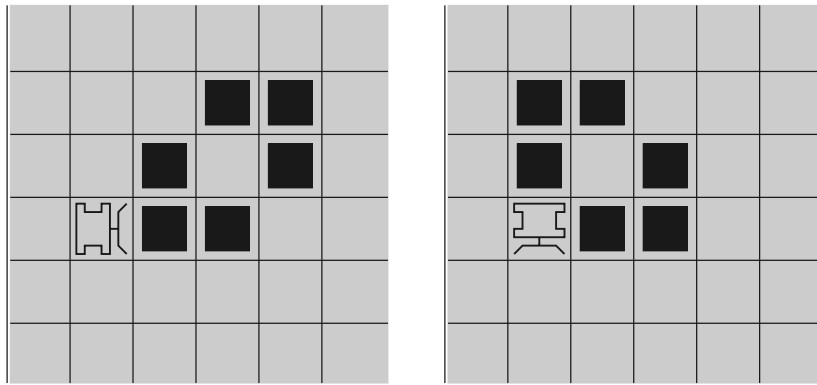
Theorem 1 (Enumeration of Configurations)
There are 320,320 ways of laying out a Tartarus board at all. A total of 23,280 of these boards contain close fours. The number of valid starting configurations for standard Tartarus is thus the difference, **297,040**. *Of the valid Tartarus configurations,* **288** *are Willson configurations.*

Evaluating Controllers: Co-evolution

With 297,040 valid configurations, evaluation of a controller on all available configurations is impractical, taking four to five minutes per controller. In most research on Tartarus the approach has been to test controllers on a random selection of Tartarus boards in each generation. Figuring out how to allocate fitness trials to members of an evolving population is an active area of research. The focus of this study is locating good boards by Co-evolution. Co-evolution attempts to locate a population of boards that improve the (final) performance of the bulldozers being evolved.

Types of Co-Evolution

Hard Co-evolution takes the available points of fitness and divides them between, for example, a bulldozer and a Tartarus board. The bulldozer's fitness is computed out of 10 in the usual manner and the remainder of the 10 points are awarded to the board. The fitness accumulated by boards is used as their fitness within their own evolving population.

Soft Co-evolution still evaluates the bulldozer in the usual fashion. The fitness for boards is computed so as to reward boards that create a fitness *gradient* on the current population of bulldozers. One might compute the standard deviation of the numbers that were summed to compute board fitnesses in hard co-evolution.

The Hypothesis

W. Daniel Hillis in a paper entitled *Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure*, in **Artificial Life II**, first proposed to use (hard) co-evolution to locate hardwired sorting networks for 16 element lists.

The hypothesis we test in this study is that: *soft coevolution is more effective than hard co-evolution to enhance performance on the Tartarus task*.

This hypothesis was tested and confirmed, for sorting networks, by **John Cartlidge** and **Seth Bullock** in a paper entitled *Lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization* that appeared in the Proceedings of the 2002 Congress on Evolutionary Computation.

Motivation

Hard co-evolution places the two evolving populations, in this case bulldozers and boards, into absolute conflict. This creates the potential for very hard fitness cases (boards) to appear and be retained, preventing or retarding the evolution of the bulldozers.

Soft co-evolution will not strongly encourage the appearance of "killer" fitness cases and so will permit evolution to proceed more efficiently than hard co-evolution.

Since the standard problem permits a known type of "hard" board, the Willson board, it follows we should look for this board's presence in our evolving population of boards as part of our analysis.

Experimental Design

In all the experiments performed in this study a population of 200 GP-Automata was used. The crossover and mutation operators given previously were used in breeding. Evolution is performed with *generational single tournament selection with tournament size four*. In this model of evolution, the population is shuffled randomly into four-member tournaments and the two most fit members of each tournament breed (copy, crossover, mutate) and replace the two least fit members of each tournament.

Experimental Design - continued

Fitness for a GP-Automata controlling a bull-dozer is evaluated on 100 Tartarus boards selected either at random or according to the method of coevolution specified. This fitness is the sum of its scores on the 100 boards used in a given generation. In experiments that use co-evolution, a fitness was computed for each board in a number of ways. The least fit half of the population of boards were replaced with new boards selected uniformly at random from the set of admissible boards.

Each evolutionary setup was run 100 times for 1000 generations. The fitness of the most fit member of the 100 population, estimated by testing on a cross-validation set of 5000 boards, was saved.

The First Three Experiments

Baseline. This experiment used 100 boards selected at random to test each generation with no close four boards, but with Willson boards permitted.

Hard Coevolution. In this experiment was identical to the baseline save that boards were chosen by coevolution. The fitness function of a board is 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 boards.

Soft Coevolution(SD). This experiment was like the hard coevolution experiment save for the fitness function used to co-evolve the boards. This function was the variance of ten minus the scores obtained by the GP-Automata on that board. In this case boards were rewarded for distinguishing GP-Automata by achieving high variance.

Soft and Hard : Willson Boards?

Below are the first 12 of the 100 boards used in the final generation of the first hard coevolution population.
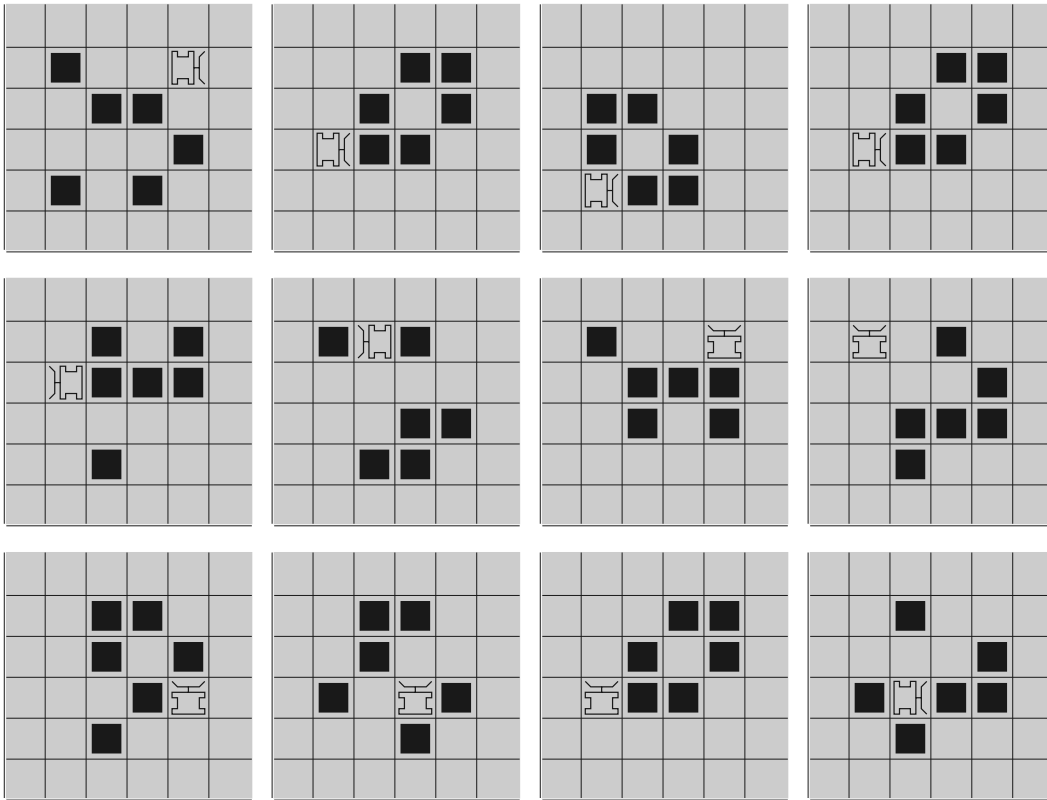
Note: 4 are Willson boards. The nominal density is 0.00097, not 1/3. The hypothesis that hard co-evolution would enrich difficult fitness cases among the boards is worth checking in more depth.

Number of Willson Boards
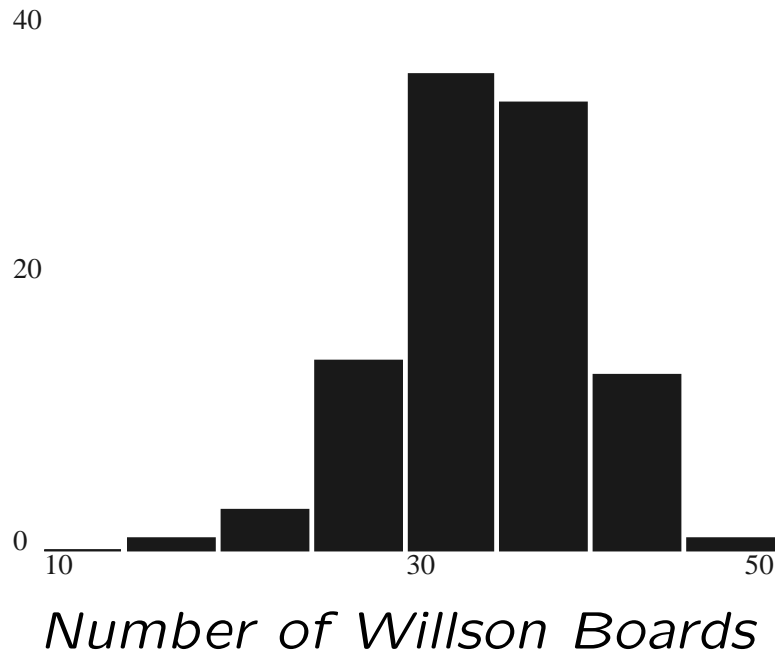
A histogram of the number of Willson boards in the final population of boards for all 100 simulations run with Hard co-evolution. The minimum number of Willson boards observed was 18, the maximum was 45. Enrichment is confirmed and consistent with *all* randomly generated Willson boards being retained.

The First Three Experiments : Fitness

Below are given 95% confidence intevals for the mean fitness of a most fit member of a final population for each of the first three experiments.

```
Soft        ├─┤

Hard                              ├┤

Baseline                     ├─┤
         3       4       5       6       7       8
```
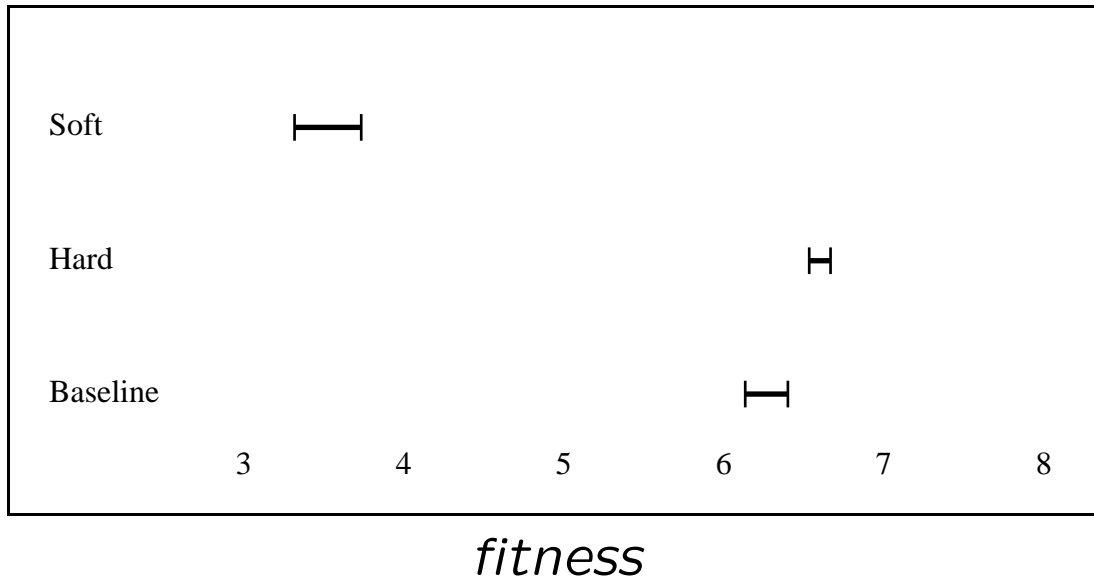
fitness

Experiment	mean	95% CI	Best
Baseline	6.33	(6.19,6.47)	7.459
Hard	6.66	(6.59,6.73)	7.489
Soft(SD)	3.58	(3.38,3.78)	7.187

Oh, dear!

Did we pick a bad soft fitness?

Soft Coevolution(ABS). This experiment was like the soft coevolution(SD) experiments save that fitness for boards was computed differently. The average of ten minus the score obtained by GP-Automata against the board was computed and then the positive difference of this number from five was used as the fitness for a board. In this case we reward a board for taking about half the available fitness.

Well:

fitness
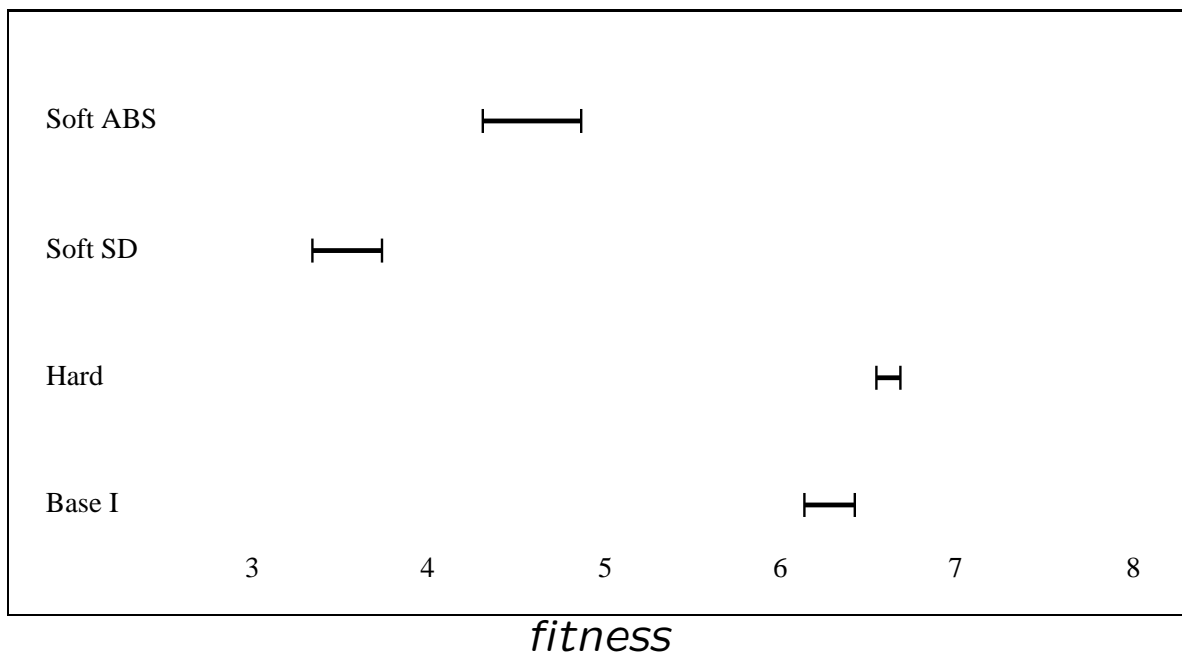
It's better but hard coevolution is still way ahead and soft is still below the baseline.

How bad ARE "hard" boards?

There is an assumption embedded in the Tartarus problem description: hard boards impair training. At this point we decided to check this hypothesis.

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.
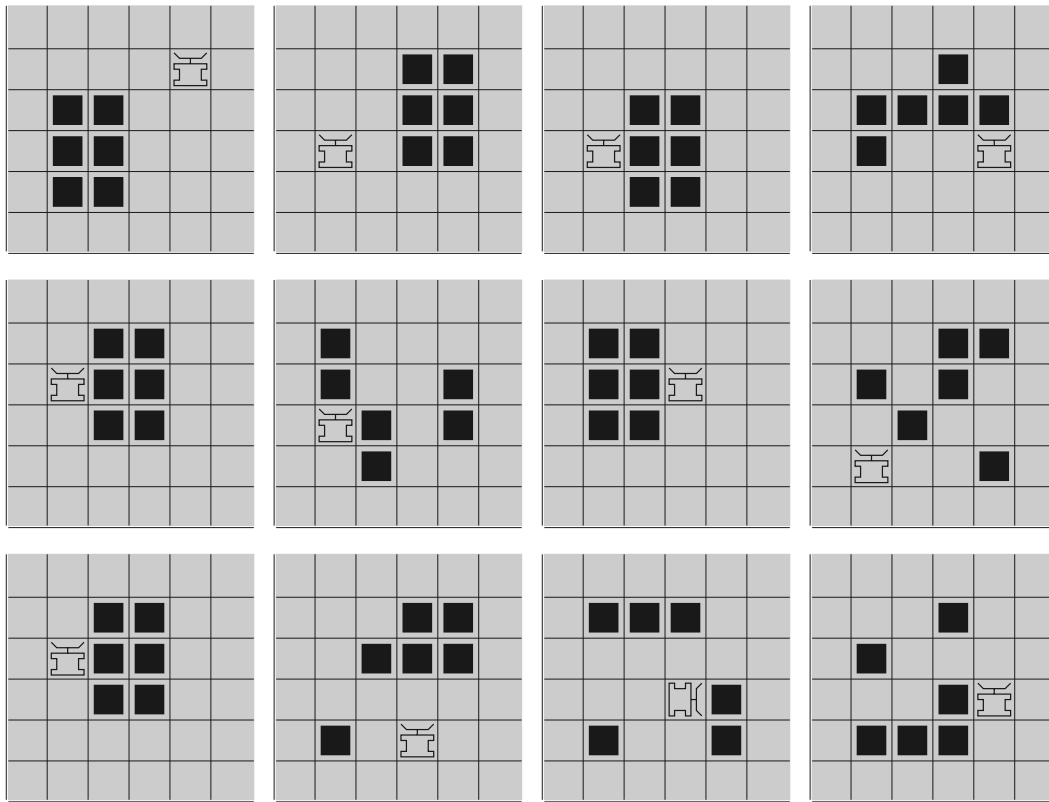

fitness
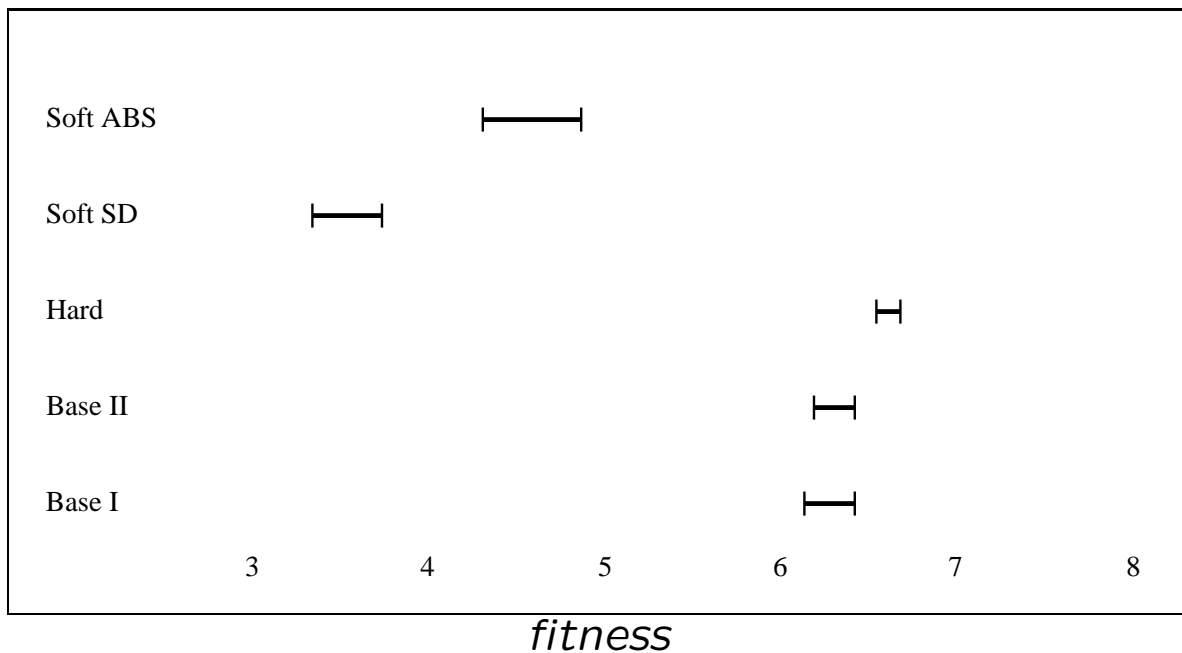
Including hard boards just makes the system slightly more reliable?!?

# Hard boards+hard coevolution?

What if we run the hard coevolution with close 4 boards *permitted*? There are far more of these than Willson boards and, perhaps, we can pile up enough "impossible" fitness cases to finally get bad behavior out of hard coevolution.

The first twelve boards in the first run are:

Look ma! Close sixes. (There are only 480 of them).

Hard-hard : Fitness effects.

fitness
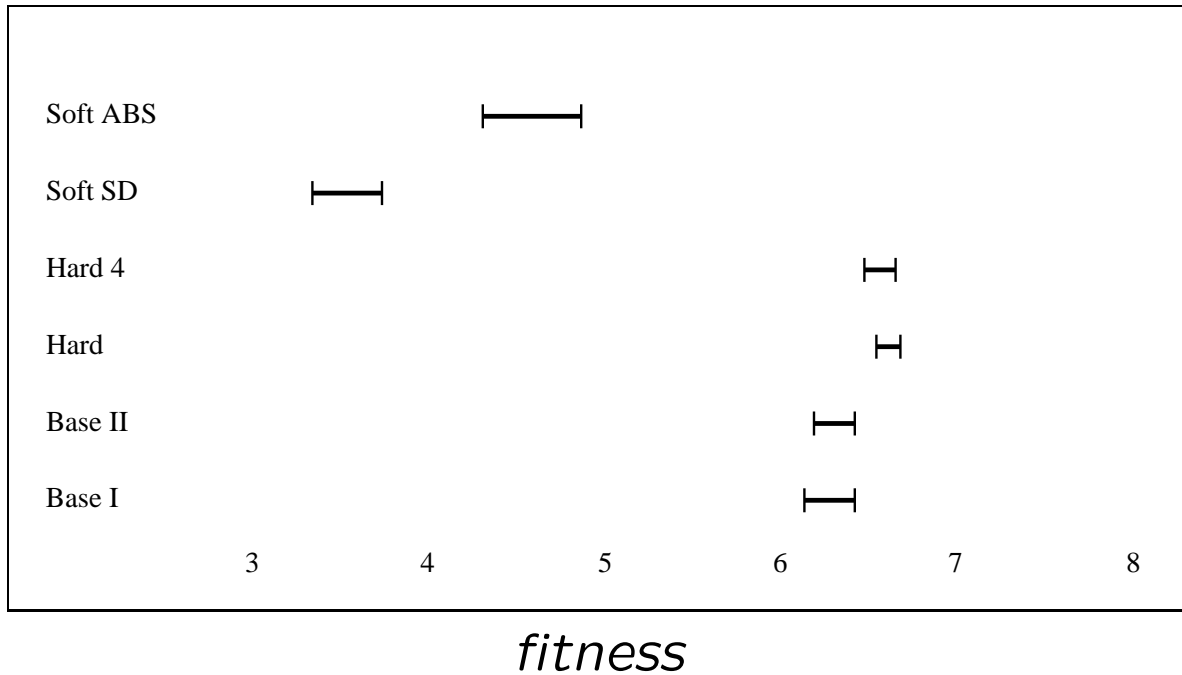
Apparently that hard boards did very little damage. There is not a significant separation between the Hard and Hard 4 experiments. The "too hard" boards either didn't do much damage or evolved slowly enough not to cause too much trouble.

Compensating for genetic operator disruption.

Suppose we took two computer programs, exchanged a middle block of statements, and then changed a couple statements at random. Even if the semantics of the programming system permit such programs to run there is an excellent chance of creating brain-damaged code.

Hypothesis: When computing soft fitness, GP-Automata that have not survived at least one round of fitness evaluation should be ignored. They are a source of vacuous fitness gradient. Experiment: Soft(SDE) (Standard Deviation Expert).

SDE Results
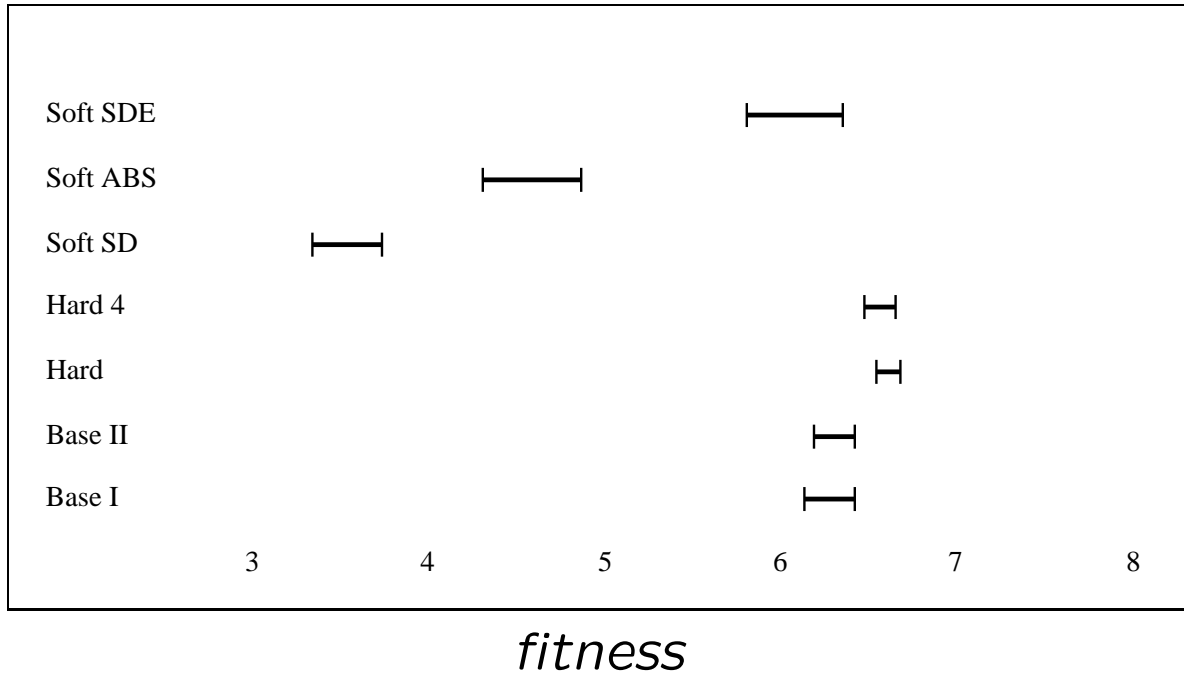
Ignoring the brain damaged helps. Not surprising, but good to know.

Discussion

This collection of experiments shows a potential trap in soft co-evolution. If the genetic operators routinely create inferior (brain damaged) population members then test cases could generate a steep fitness gradient without being particularly challenging. The simple expedient of paying attention only to information from population members that did not die during their first fitness evaluation really helped.

After seeing these results Steve Willson asked if it would help to pay attention only to members of the population that have survived two fitness evaluations. This question suggests that we search for the top of this hill: how much age should we require in the bulldozers used to evaluate the fitness of the boards? Two more experiments were performed with only age 2+ and age 3+ bulldozers used to evaluate boards.

The effects of age limits.
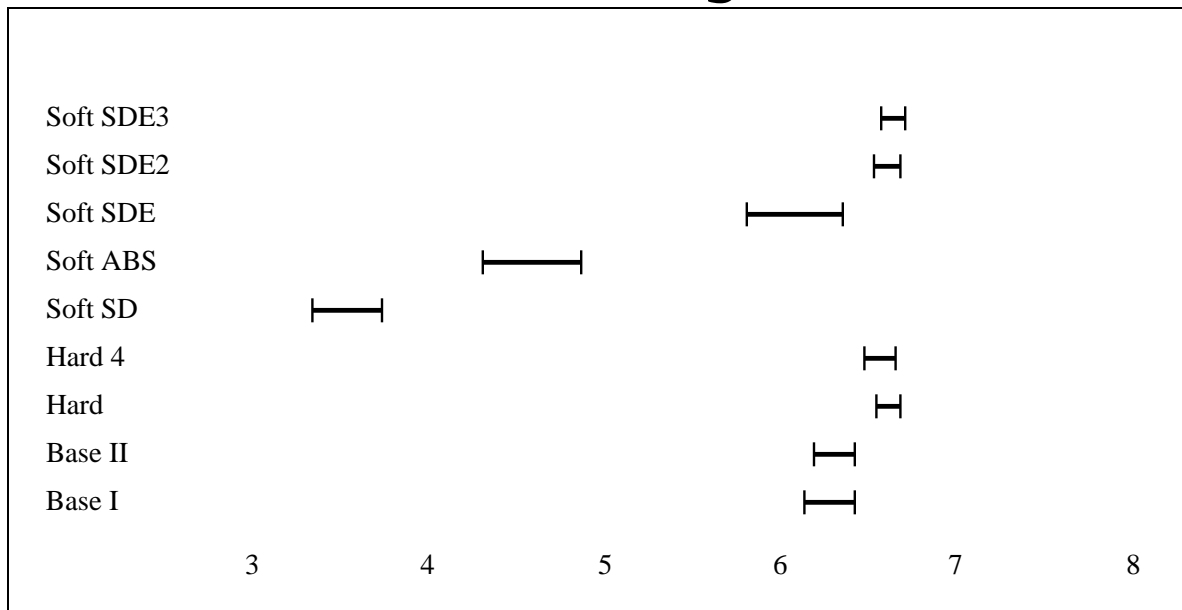
fitness

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 4	6.58	(6.53,6.70)	7.283
Soft(SD)	3.58	(3.38,3.78)	7.187
Soft(ABS)	4.63	(4.36,4.91)	7.419
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

Discussion

- Once the bugs are worked out, hard and soft coevolution are not significantly different on the Tartarus problem. The hard coevolution is slightly easier to code but they both run at the same speed and yield almost indistinguishable results. In addition, soft co-evolution still piles up some "hard" boards, albeit not as many as hard co-evolution.
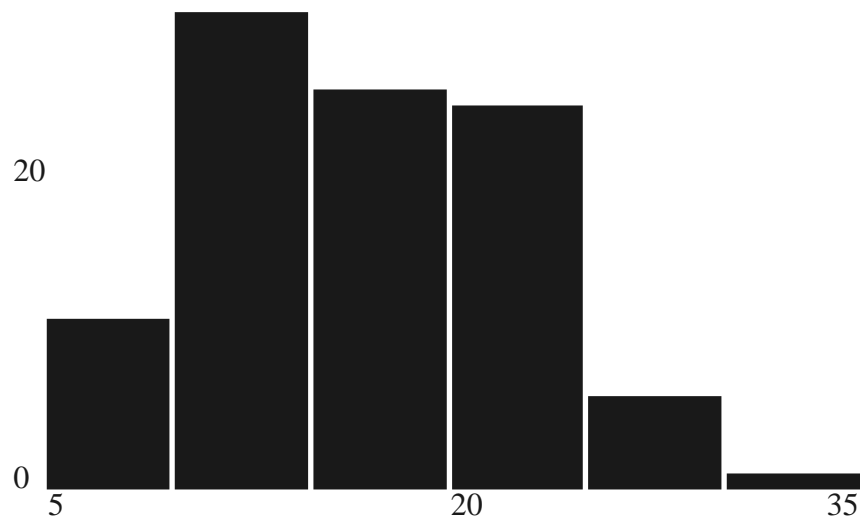
Number of Willson Boards in Soft(SDE3).

Discussion

- Co-evolution can be used to discover hard fitness cases. Willson boards, close 4, and close 6 boards are all *highly* enriched in co-evolved populations of boards where they are permitted at all.

- Given that co-evolution seems to behave very differently for sorting networks and Tartarus it would be nice to see experiments on other problems.

- A LARGE number of additional experiments are possible in the Tartarus environment, e.g. running the absolute value fitness experiments with age restrictions.

Thanks

The authors thanks the Iowa State University Complex Adaptive Systems program for its support and help.

Questions?