Semantic Bias in Program Coevolution

Tom Seaton, Julian F. Miller, Tim Clarke

Department of Electronics
University of York
{tas507}{julian.miller}{tim.clarke}@york.ac.uk

Abstract. We investigate the occurrence of two pathological coevolutionary behaviours, specifically disengagement and cycling, in GP systems. An empirical analysis is carried out using constructed GP problems and a historical pursuit and evasion task, the Game of Tag. The effect of semantic bias on the likelihood of pathologies and performance in a coevolutionary context is examined. We present findings correlating semantic locality in the genotype to phenotype map to disengagement and cycling in a minimal competitive coevolutionary algorithm.

Keywords: Coevolution, Genetic Programming, Semantics, Benchmarks

1 Introduction

Pathological or unintended behaviours are a well-established issue in the design of coevolutionary algorithms (CoEA) [1]. Coevolutionary pathologies are processes, distinct to coevolutionary systems, that interfere with progress of the search towards a desired goal state. Analysis of these pathological behaviours in systems of minimal complexity has been a principal component of coevolutionary research in genetic algorithm (GA) representations, focusing on both their theoretical basis [2] and mitigation [3]. In this paper, we examine how coevolutionary pathologies can influence progress in coevolutionary forms of genetic programming (GP). Although pathological behaviours have been addressed in GP [4], to our knowledge no studies exist which explicitly recreate these concepts using a quantifiable, controlled approach under program representations. Our intention is to bridge this gap with an initial study demonstrating and analysing pathologies in a minimal form of GP.

Disengagement and cycling are patterns of search behaviour that occur in systems lacking an objective method of fitness evaluation. Informally, coevolutionary algorithms determine a search gradient through the interaction of sets of individuals rather a single individual. Disengagement therefore occurs when an element of the system has entered a state for which no search gradient can be induced by reference to the other coevolving elements. Cycling behaviour occurs when previously visited interactions recur so that the search is led to return repeatedly to a previous set of individuals. The former pathology results

in a period of unguided search. The latter pathology wastes computational effort by re-evaluating previously visited states. We postulate that disengagement and cycling are particularly significant in coevolutionary systems which use GP representations. Disengagement has been suggested to occur with greater frequency in situations where it is more difficult to make objective progress in some coevolved components than others [5]. Because most GP expressions are far from uniformly represented, such asymmetry may be common. Cycling behaviour is costly because (in general) re-evaluating GP structures is associated with more computation than their binary counterparts. However, it is not presently known which factors in the GP paradigm influence disengagement and cycling behaviours. This empirical work focuses on one possible factor, semantic bias. Research on semantics and the genotype-phenotype map in GP has been a prominent area since GP's inception [6–8], but it is not clear how previous findings in classical single population GP extend to coevolutionary systems. The experiments described here examine the effect of local and non-local semantic topology on disengagement, cycling and performance in a coevolutionary GP system.

The structure of this paper is as follows. In Section 2, we introduce a set of suitable, simple, benchmark coevolutionary problems. These have been selected to elicit measurable forms of pathological behaviour when using GP in a coevolutionary setting. Section 3 describes our experimental configuration and treatment of parameters. Section 4 summarises the results for each pathological case. The final section discusses how the relationships observed between pathological behaviour, semantic constraints and performance have informed our understanding of coevolution in GP.

2 Benchmark Selection

Few established benchmarks exist for coevolutionary forms of GP. Historically, one natural source of problems has been competitive pursuit and evasion games, where the objective is to develop strategies to intercept or escape an opponent. An example is the *Serengeti* problem, first analysed in GP by Haynes and Sen [9], which presents a classic predator-prey scenario. Strategies are developed for multiple predators ('lions') to capture a prey-agent (a 'gazelle') on a simulated grid-world. *Serengeti* is considered to be difficult to solve without a degree of cooperation between predators [10]. Pursuit and evasion is frequently used in the coevolutionary literature, but has been criticised as a method of benchmarking [11], primarily due to the complexity of interactions between different strategies and the ensuing difficulties when defining measures of progress. However given the breadth of practical applications, it remains an attractive area within which to analyse CoEA, provided a sufficiently simplified problem instance can be defined.

Another commonly employed class of coevolutionary GP problem is a 'game vs. environment' where programs are coevolved for the purpose of controlling an

agent in conjunction with an increasingly challenging structure, such as a maze or series of obstacles. The *Tartarus* grid-world game proposed by Teller [12, 13] presents such a situation, in which an agent must manoeuvre a series of blocks into positions around the edges of the world. A more recent example can be found in Cartlidge [5], in which a maze navigation problem was defined where strategies to control robots to escape a maze are coevolved simultaneously with increasingly difficult maps. Both problems are examples of asymmetric problem difficulty in GP (challenging worlds are easier to obtain than good controllers.) However, the utility of established problems such as *Serengeti* and *Tartarus* is questionable when investigating coevolutionary pathologies. Firstly, both *Serengeti* and *Tartarus* lack a clearly defined notion of optimal behaviour or solution concept against which to measure progress. Secondly, their solutions require complex components whose contribution to the problem difficulty is not well understood. Notably, the *Tarturus* game requires that solutions incorporate memory, for example in the form of a finite state machine. It is unclear how these requirements interact with GP performance.

Given the paucity of suitable benchmarks, for this work we introduce two new minimal constructed problems after the style of existing analysis in GA systems, the *GP Greater Than Game* and *Simple Cycler*. These games are designed specifically to explore disengagement and cycling in a GP context. The problems are derived from the concept of GA 'number games' similar to those analysed in [3] and [14]. We will also consider a more complex historical pursuit and evasion task, the *Game of Tag*.

2.1 Problem Set

GP Greater-Than Game (GP-GTG) To investigate the pathology of disengagement in GP, the GA *Greater-Than* (GT) game described by [5] was generalised to the context of GP representations (GP-GTG). GP-GTG uses two symmetric populations of programs. Each program operates on real values formed from a constrained function set {+,-} accepting a single terminal input fixed at unity. Programs have a single output, derived by evaluating the expression at the root node of the program. The outcome of comparing a pair of programs (p, p') is given as a function of the program outputs g. We term this the interaction function:

$$g(p, p') = \begin{cases} 1.0 & o(p) > o(p') \\ 0.5 & o(p) = o(p') \\ 0.0 & o(p) < o(p') \end{cases} \quad (1)$$

where $o(p)$ and $o(p')$ are the real valued outputs of each program. The expressions are constrained to a maximum depth n, measured from root to terminal nodes. The game is solved after a program is found from the subset of programs which maximises the output.[1]

[1] GP-GTG is superficially similar to the GP 'MAX' problem [15]. The key distinction is that programs are evaluated using only their relative rather objective fitness.

Simple Cycler (SC) Informally, we define cycling behaviour in coevolutionary GP as exiting and revisiting the same phenotypic state in a program search space. *Simple Cycler* is an elementary game which is designed to simulate measurable, irregular cycles in a coevolutionary GP algorithm. Evolved programs operate on boolean values and are constructed from the function set {AND,OR,NOT,IF}, where the IF function accepts three arguments: a condition, response if the condition is true and response if the condition is false. The input is a fixed terminal with value TRUE. Programs are required to output a string of n boolean values (for example, from n selected output nodes). This output is mapped onto an unsigned integer $o \in \{1 : 2^n\}$, using a binary encoding. The interaction function is computed as:

$$g(p, p') = \begin{cases} 1.0 & o(p) > o(p') \text{ except } o(p) = 2^n \text{ and } o(p') = 1 \\ 1.0 & o(p) = 1,\ o(p') = 2^n \\ 0.5 & o(p) = o(p') \\ 0.0 & o(p) < o(p') \text{ except } o(p) = 1 \text{ and } o(p') = 2^n \\ 0.0 & o(p) = 2^n,\ o(p') = 1 \end{cases} \tag{2}$$

This expression states that the program corresponding to the greatest integer wins, except for the cases where the maximum integer (2^n) value is compared to the minimum integer (1). Therefore under this function all programs can be positioned on a single transitive chain of length 2^n. A cycle is said to occur when a program has changed from a structure corresponding to the smallest integer to the largest and back, traversing the intermediate states. Cycling behaviour is monitored by measuring the average period over a fixed number of generations.

The Game of Tag (GoT) The *Game of Tag* is a two dimensional GP pursuit and evasion game introduced by Reynolds [16]. Games consist of an idealised scenario in which control programs are developed which provide pursuit and evasion strategies. Analogous to the children's game, the objective for each control program is to minimise the length of time during which a program is designated as 'it' (the pursuer). Play occurs between pairs of competitors, which are point objects able to move at a fixed speed over a number of discrete timesteps. No account is made for momentum or limitations in change of heading. If the competitor designated as 'it' enters a certain capture radius of its opponent, the opponent is 'tagged' and the roles are exchanged. Successful programs must simultaneously evolve pursuit and evasion behaviours, depending on their role at that timestep. Effective algorithms should increase the capability of competitors in both roles progressively over time.[2] Our implementation closely follows Reynold's original version. At the start of a game, one competitor is placed at the centre of the two-dimensional play area. The other competitor is placed uniformly at random in a square region with width w centered on this position. Whilst a competitor is in pursuit, it is set to move at twice the speed of the

[2] In this work we do not consider external methods of assisting progress, such as archives. Understanding the components of GP that impact on pathological behaviours may provide systems which are less reliant on these approaches.

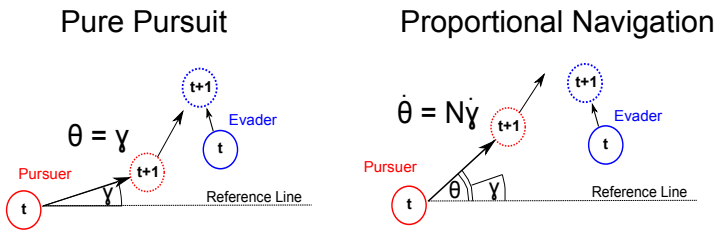

Fig. 1. *Pure and proportional navigation pursuit strategies. The pursuer is shown against a fixed trajectory evader is shown over a discrete timestep* $t \rightarrow t+1$.

evader. Inputs to each program are restricted to a real-valued vector (x, y) in the local coordinate system of the competitor and a boolean value, which specifies whether the competitor is in pursuit or not at that timestep. Programs provide a single real number output, which is interpreted as an updated heading. A score s is awarded to each competitor at the end of the game, equal to the number of timesteps spent as the evader. During training, the interaction function between two programs is evaluated as the average score obtained over a set of S games. In the first half of the set the first program p begins as the evader and in the second half the initial role is swapped.

$$g(p, p') = \frac{1}{|S|} \sum_{i \in |S|} s_i(p, p') \tag{3}$$

Reynolds assessed the objective quality of competitors by comparison against a robust artificial strategy, pure-pursuit. Competitors implementing pure-pursuit move directly towards their opponent whilst the pursuer and directly away whilst the evader. In the game of tag deviation by either adversary to any other strategy results in poorer performance, because a route other than the shortest path must be traversed (in game-theoretic terms this is a Nash equilibrium.) The present work includes an additional measure of solution quality using a further guidance strategy, proportional navigation. Proportional navigation is a widely applied guidance law, backed by a large body of analysis [17, 18]. The strategy employs the principle that an interception between the trajectories of two objects traveling with fixed speed will occur if the bearing between them is constant. In proportional navigation, the heading is updated at each timestep proportional to a constant N, which controls the magnitude of response (we assume N=3, see analysis in [18]). The angle γ gives the heading of the adversary. The angle θ gives the current heading. Angles are measured with respect to the local coordinate system. An example contrasting both strategies is sketched in Figure 1.

3 Experiment

3.1 Algorithm and Representation

An intentionally minimal GP algorithm was used for ease of comparison with other techniques. An integer genotype representation, Cartesian Genetic Pro-

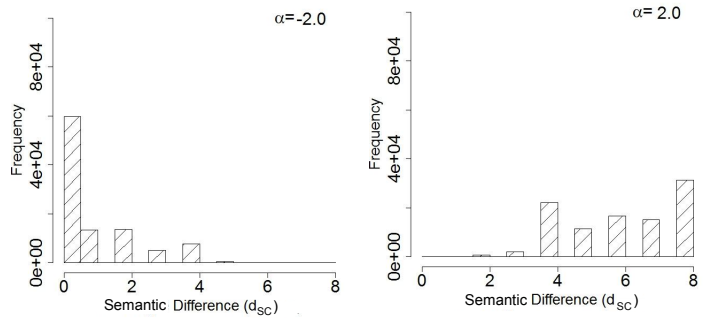

Fig. 2. *Example contrasting the effects of the semantic bias on distances of mutated offspring. Illustrated for* 10^5 *sampled mutation operations in Simple Cycler.*

gramming (CGP), was selected across all problems [19]. Standard CGP uses only a truncation selection strategy followed by uniform mutation without crossover. In addition, CGP has recently been applied to a coevolutionary setting [20]. Two populations of CGP programs were initialised uniformly and evaluated through the simplest coevolutionary interaction scheme (complete mixing) in which programs are tested versus all members of the other population P :

$$f(p) = \sum_{p' \in P} g(p, p') \tag{4}$$

Fitness values are given by the outcomes accumulated over all interactions.

3.2 Semantic Bias

A semantic bias was introduced to the mutation operator. A parameterised technique similar to the methods of Nguyen [7] was used, which has been previously applied to CGP to introduce a syntactic bias in [8]. A metric approximating the semantic difference between programs was defined for each problem, d_{GP-GTG}, d_{SC} and d_{GoT} respectively:

$$d_{GP-GTG} = |o(p) - o(p')| \tag{5}$$

$$d_{SC} = min \begin{cases} |o(p) - o(p')| \\ 2^n - |o(p) - o(p')|. \end{cases} \tag{6}$$

$$d_{GoT} = \sum_{K} |\theta(p) - \theta(p')| \tag{7}$$

Equation 5 is the absolute difference in the output of each program. Equation 6 is the shortest distance measured around the transitive cycle. Equation 7 is the absolute difference between output headings summed over a set of K input vectors to both programs. The set of vectors point to a grid of uniformly distributed fixed positions across the square starting region in the Game of Tag.

Table 1. Fixed Algorithm Parameters and Game Properties

Parameter	GP-GTG	SC	GoT
Nodes	20	20	50
Function Set	{+,-}	{AND, OR, NOT, TRUE}	Reynolds [16]
Terminal Set	{0,1}	{TRUE}	$\{x, y, isIt, [0.2{:}1.0]\}$
Mutation Rate	0.05	0.05	0.02
Selection Strategy	4+6 ES	1+1 ES	1+4 ES
Output Type	$o \in \{1 : 2^n\}$	$o \in \{1 : 2^n\}$	$o \in \mathbb{R}$, mapped onto $[0{:}2\pi]$
Populations	2×10	2×1	2×5
Runs	500	200	500
Max Generations	500	1000	500
β	1	1	0.05
Games/Opponent	1	1	Training: 5, Testing: 100
Game length	N/A	N/A	100 timesteps
Startbox Size w	N/A	N/A	7
Pursuer speed	N/A	N/A	2
Evader speed	N/A	N/A	1
Capture radius	N/A	N/A	1

A sigmoid function is used to bias the probability of mutating to individuals at particular semantic differences. The function gives the probability of accepting a prospective mutated individual, with respect to the semantic distance between parent and child. Control is provided by a pair of parameters $(\alpha, \beta) \in \mathbb{R}^2, \beta \geq 0$. The parameter α alters the slope of the sigmoid function, where $\alpha << 0$ and $\alpha >> 0$ correspond to a bias towards small and large semantic changes in each mutation. The parameter β offsets the function, giving the semantic distance at which there is a 50% chance of accepting a mutation, i.e. $sigmoid(\beta) = 0.5$. An example of the effects of the biased mutation operator on the distribution of mutation distances is illustrated in Figure 2, for SC with $\beta = 1$.

3.3 Summary of Fixed and Variable Parameters

In preliminary experiments, fixed algorithm parameters were tuned independently for each problem to give a locally optimal set of parameters for the CGP system, under no semantic bias ($\alpha = 0$). The range of values given in [19] was used as a basis. The Game of Tag parameters are based on those originally fixed by Reynolds [16]. Following the original work, the root node of all programs evolved in the Game of Tag is seeded with the 'IF-IT' function to provide a separate flow of execution for pursuit and evasion. Because there is no standardised approach to providing constants in CGP, the simplest technique is adopted here: the introduction of a small array of fixed constants as terminal values {0.2, 0.4, 0.6, 0.8, 1.0}. Although a full-factorial analysis of all parameters is outside the feasible scope of this work, in Section 4.2 we test the sensitivity of our experimental outcomes to mutation rate and length of CGP genotype. The offset β was fixed to the minimum semantic difference in the constructed problems and a representative small angular difference of 0.05 revolutions (18°) in the GoT,

to give semantically local differences in the limit $\alpha << 0$. A summary of all the parameters used in each problem case is given in Table 1.

4 Results

4.1 Disengagement in the GP Greater Than Game

The definition in [5] states that two populations can be considered to be disengaged when the variance of the accumulated fitness values across each population is zero. Figure 3 contrasts the probability of disengagement in GP-GTG and the magnitude of expected program output (performance), averaged over all runs as a function of α. A strong sensitivity to the semantic bias was observed. For $\alpha < 0$, the probability of disengagement is high and the evolved program output is low. Performance and disengagement were strongly correlated with the value of α (resp. Spearman 0.99 and -0.95, $p \leq 0.005$, exact). We infer that using a mutation operator with a high probability of making a small semantic change increases the likelihood of disengagement. Highest performance is observed for this case when the mutation operator is biased towards larger semantic changes.

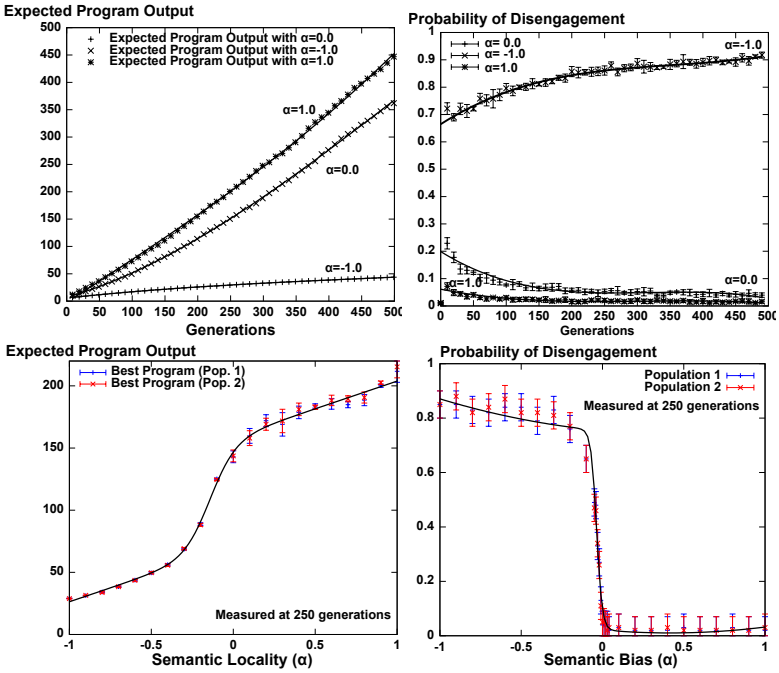

Fig. 3. Disengagement in the GP-GTG case. Top left: Expected output of the best evolved program. Top right: Probability of disengagement. Bottom left: Sensitivity of expected output to semantic locality. Bottom right: Sensitivity of probability of disengagement to semantic locality (reported at 250 generations).

4.2 Periodicity in the Simple Cycler Game

Cycling is characterised by measuring the mean number of generations for each population to transition between the maximum program state 2^n and back. Use of a 1+1 EA ensures that each population is at a single position at any given instance.[3] Figure 4 shows the period obtained over 200 runs, for transitive chains of increasing length. As n is increased, the time to traverse the chain is larger. The effect on cycle rate of mutation rate and α is shown in the bottom two images. Predictably, in the limit of very low mutation rates the cycle frequency tends to zero (no change in state). Biasing the semantic mutation operator towards larger semantic changes ($\alpha \geq 1$) corresponded to longer periods. In the limit of high mutation rates and small α, the average frequency tended to $\sim \frac{1}{2^n}$, or one generation per program state in the transitive chain. The fastest cycling behaviour was apparent in both mutation operators which produced very small semantic changes in programs and also those approaching random search.

4.3 Performance in the Game of Tag

Performance in the Game of Tag was measured by relative evaluation with respect to four pursuit and evasion strategies. These included pure-pursuit (PP) and proportional navigation (PN). In addition, progress was also evaluated against a noisy control (R), which returned a randomised heading for all input vectors. Finally a mixed strategy was defined to give an intermediate quality opponent, which returned either a randomised response or the pure pursuit response with equal probability (PR). Figure 5 (left) shows the expected progress of unbiased CGP against these metrics. Best performance was achieved against the randomised and proportional navigation strategies (evolved strategies are expected to win $\approx 90\%$ of games versus random opponents). Weakest performance is evident against the PP and PR strategies. A modest expected performance ($\approx 20\%$) was observed against the pure pursuit case, though higher success rates were achieved in individual cases, similar to that of Reynolds [16]). A direct comparison with the performance of Reynold's implementation is not possible because only 5 individual runs were reported in the original work. The sensitivity of the CGP algorithm in the Game of Tag to semantic bias was examined. No significant change was observed when measuring against the PP strategy. A weak response was measurable against the PR, R and PN strategies. The measured Spearman correlation coefficients in each case are (PP = -0.17, PR = -0.82, PN = -0.79, R = -0.62) where the correlations in PR, PN and R are significant at $p \leq 0.005$ (exact). Figure 5 (right) shows the change in expected performance at 250 generations for each of the significant results. Best performance was observed at $\alpha = -50$, which corresponds to a strong bias towards small phenotypic changes. However, the net performance change is small ($\approx 5 - 10\%$ difference in win ratio) and the effect of varying semantic locality in this case is marginal.

[3] Devising an unambigious definition of cycling behaviour in larger populations of practical interest is an open issue for GP, which we leave for future debate.

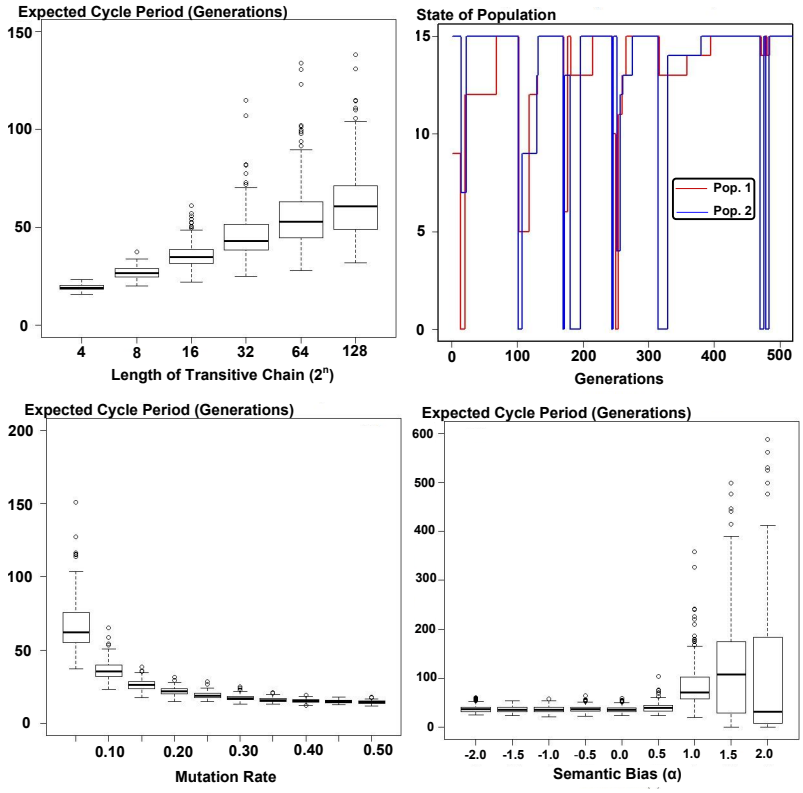

Fig. 4. Cycle rate in the Simple Cycler case. Top left: Scaling of unbiased cycle rate with problem size. Top right: Extract of a coevolutionary run, illustrating irregular cycling in both populations. Bottom left: Response to mutation rate. Bottom right: Response to semantic bias.

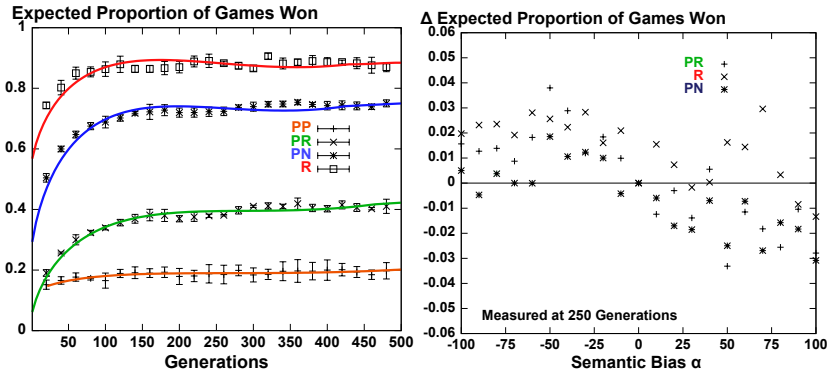

Fig. 5. Game of Tag Performance. Left: Unbiased CGP performance. Right: Sensitivity of performance to semantic bias.

5 Discussion

The strong correlation between performance and disengagement in GP-GTG supports our hypothesis that disengagement is a significant factor in program coevolution. The probability of disengagement increases for small changes because programs are less likely to be distinguished using the coevolving population. Although we must be wary of generalising too far from this simplified example, the result suggests there may be a tradeoff between the use of semantically local mappings and operators which permit sufficient diversity to ensure populations of programs remain engaged. An interesting result from the Simple Cycler example is the trend towards faster cycling in both random (high mutation rate) and highly local search (small α). Inspection of individual runs showed that this is a consequence of two distinct search behaviours. Under random search, programs do not traverse the chain of states and instead revisit each state with fixed probability. For small n this probability is high, therefore giving a short period. In contrast, when mutations are constrained to programs with a small difference in output, the search tends towards hillclimbing through the intermediate states, also giving fast cycles. The result implies that, because cycling in GP is a pathology which occurs when an algorithm follows a transitive chain of programs, mappings biased towards small semantic changes may worsen this behaviour. Varying semantic bias in the Game of Tag problem introduced only a minor change to performance. We theorise this is due to two issues. Firstly, relative to the constructed problems, the semantic mapping in the Game of Tag is significantly more complex. The relationship between the outcome of a game and changes to program output is not transparent. It is therefore not clear whether our metric for semantic difference is the most suitable for this case. Secondly, the randomised starting configuration introduces noise into each outcome. This reduces the measured likelihood of disengagement in this case because of the increased variation in fitness values.

6 Conclusions and Further Work

These experiments highlight that pathological behaviours are a factor in the performance of coevolutionary forms of GP and that semantic biases in the GP genotype-phenotype map can influence their occurence. We introduced two new constructed problems based on the concept of coevolutionary number games in GA systems. Disengagement in coevolving populations was strongly related to semantic locality. We showed that semantic locality changed the frequency of cycling in a simple GP system. A weak response to semantic bias was also observed in a more realistic coevolutionary system, the Game of Tag. The scope of this work could be extended by analysing further GP algorithms, problem sets and other pathological behaviours which have been characterised in binary representations (for example, overgeneralisation [21]). At present, archiving techniques are used to provide theoretical guarantees of progress in coevolutionary EAs. However, examining the genotype to phenotype map used in methods of GP

which have achieved good results without archives [22] may indicate how representations inherently robust to coevolutionary pathologies can be developed.

References

[1] Popovici, E., Bucci, A., Wiegand, R.P., De Jong, K.: Coevolutionary Principles. In: Handbook of Natural Computing. Springer-Verlag, Berlin (2010)
[2] Ficici, S.G.: Solution Concepts in Coevolutionary Algorithms. PhD thesis, Brandeis University (2004)
[3] De Jong, E.: A Monotonic Archive for Pareto-Coevolution. Evolutionary Computation **15** (2007) 61–93
[4] Lipson, H., Bongard, J., Zykov, V.: Co-Evolutionary Methods in System Design and Analysis. In: 15th International CIRP Design Seminar, Shanghai (2005)
[5] Cartlidge, J., Ait-boudaoud, D.: Coevolutionary Optimization. IEEE Transactions on Evolutionary Computation **15** (2011) 215–229
[6] Whigham, P.: Search bias, language bias and genetic programming. In: First Annual Conference on Genetic Programming, MIT Press (1996) 230–237
[7] Nguyen, Q.U.: Examining Semantic Diversity and Semantic Locality of Operators in Genetic Programming. PhD thesis, University College Dublin (2011)
[8] Seaton, T., Miller, J.F., Clarke, T.: An Ecological Approach to Measuring Locality in Linear Genotype to Phenotype Maps. In: EuroGP 2012. (2012) 170–181
[9] Haynes, T., Sen, S., Schoenefeld, D., Wainwright, R.: Evolving multiagent coordination strategies with genetic programming. Technical Report UTULSA-MCS-95-04, The University of Tusla (1995)
[10] Luke, S., Spector, L.: Evolving Teamwork and Coordination with Genetic Programming. In: GECCO '96. Number July (1996) 150–156
[11] Cliff, D., Miller, G.F.: Co-evolution of Pursuit and Evasion II: Simulation Methods and Results. In: From Animals to Animats: Proc. of the Fourth International Conference on Simulation of Adaptive Behaviour, University of Sussex (1995)
[12] Teller, A.: Learning Mental Models. In: Proceedings of the Fifth Workshop on Neural Networks. (1993)
[13] Trenaman, A.: Concurrent Genetic Programming , Tartarus and Dancing Agents. In: EuroGP'99, Goteburg, Sweden, Springer-Verlag Berlin (1999) 270–282
[14] Watson, R., Pollack, J.: Coevolutionary Dynamics in a Minimal Substrate. In: GECCO 2001, Morgan Kaufmann (2001) 702–709
[15] Langdon, W., Poli, R.: Foundations of Genetic Programming. Springer-Verlag (2002)
[16] Reynolds, C.W.: Competition , Coevolution and the Game of Tag. In: Artificial Life IV. (1994) 59–69
[17] Yuan, L.C.L.: Homing and Navigational Courses of Automatic Target-Seeking Devices. Journal of Applied Physics **19** (1948) 1122–1129
[18] Shukla, U., Mahapatra, I.R.: The proportional navigation dilemma-pure or true? IEEE Transactions on Aerospace and Electronic Systems **26** (1990) 382–392
[19] Miller, J.F., ed.: Cartesian Genetic Programming. 1st edn. Springer (2011)
[20] Sikulova, M., Sekanina, L.: Coevolution in Cartesian Genetic Programming. In: EuroGP 2012, Springer-Verlag Berlin Heidelberg (2012) 182–193
[21] Wiegand, R.P.: An Analysis of Cooperative Coevolutionary Algorithms. PhD thesis, George Mason University (2003)
[22] Jaskowski, W., Krawiec, K., Wieloch, B.: Winning ant wars: Evolving a human-competitive game strategy using fitnessless selection. EuroGP (2008) 13–24