To appear at Artificial Evolution, 2010
https://lsiit.u-strasbg.fr/ea09/index.php/Programme

Binary versus Real-valued Reward Functions under Coevolutionary Reinforcement Learning

Peter Lichodzijewski and Malcolm I. Heywood

Faculty of Computer Science
Dalhousie University,
6050 University Ave.,
Halifax, NS, Canada, B3H 1W5
{piotr,mheywood}@cs.dal.ca

Abstract. Models of coevolution supporting competitive and cooperative behaviors can be used to decompose the problem while scaling to large environmental state spaces. This work examines the significance of various design decisions that impact the deployment of a distinction-based formulation of competitive coevolution. Specifically, competitive coevolutionary formulations with and without point population speciation are compared to stochastic sampling of the environment under both binary and real-valued rewards. The additional structure implicit in the competitive coevolutionary models is shown to be of significant benefit under binary rewards, however, stochastic sampling results in more dependable performance under real-valued feedback. It is also observed that cooperation between multiple solutions is much more prevalent under real-valued rewards than under binary rewards.

Key words: Competitive Coevolution, Problem Decomposition, Genetic Programming, Teaming, Reinforcement Learning

1 Introduction

Model-based Evolutionary Computation such as Genetic Programming (GP) and Neural Evolution provide an effective basis for reinforcement learning, i.e., learning under environments with delayed reward [11]. Recent developments include comparisons of Sarsa TD-learning with Neuro-Evolution of Augmented Topologies (NEAT) [14], incorporating weight term refinement using Q-learning into NEAT [15], and augmenting GP with Q-learning to facilitate a strictly conditional tree-structured representation [5]. Application domains range from the behavioral modeling of non-player characters for video games [13] to robot rover control [1]. Most of the emphasis, however, has been on the model side of development with less attention placed on the interaction between model and environment. Such an interaction is important when scaling reinforcement learning to larger problem domains; that is, domains for which it is not sufficient to perform fitness evaluation over a fixed, *a priori*, set of training scenarios.

Methods for adapting training scenarios to the ability of the learners under an evolutionary framework are generally referred to as competitive coevolution.

In this work, a distinction-based [6] competitive coevolutionary framework is assumed. When combined with a model of GP that provides solutions in team form – multiple cooperating programs – the potential exists to support problem decomposition both internally to the team and across multiple teams. Naturally, such phenomena are emergent, potentially resulting in complex behaviors. Specifically, we are interested in identifying properties from competitive coevolution that are important for influencing the nature of the resulting cooperative behaviors. To do so we focus on the formulation for the interaction between point population (subset of configurations of the training environment) and the team population. Given that points are credited for distinguishing between the performance of teams, a design decision critical to this interaction is the definition of the reward function, i.e., at what sensitivity can a 'distinction' be registered. Previous works concentrated on binary rewards under immediate feedback (supervised learning) [4], [6]; where this can lead to a period of disengagement before points that distinguish between learners are identified. Conversely, a real-valued interaction between points and learners may result in too many distinctions (i.e., there are too many ways in which learner performance can vary) and also lead to disengagement as many meaningless distinctions are registered.

Section 2 develops the GP-based Symbiotic Bid-Based (SBB) model for the coevolution of *non-overlapping* learner behaviors under a competitive model of interaction. Assuming the SBB methodology provides the basis for problem decomposition between individuals within the same team as well as between different teams. Relative to the earlier SBB formulation (e.g., SBB as utilized under the context of supervised learning [8]) much more emphasis is placed on diversity maintenance, where this impacts variation operators, support for speciation, and the reward function. The problem domain adopted takes the form of the truck-backer-upper [2], made considerably more difficult than the original formulation through the introduction of an obstacle and a post-training generalization test based on 1,000 test configurations (Section 3). The definition of the reward function is shown to play a significant role in establishing the diversity of training scenarios and the corresponding support in the learner population for collective problem solving. Conclusions appear in Section 4.

2 Symbiotic Bid-Based Model

2.1 Model Overview

A bid-based metaphor [8] establishes the interaction between programs within a team. A team is composed of n team members where each member associates a bid behavior (evolved program) with a scalar action. The size of a particular team is not specified *a priori* but is determined through evolution. Likewise, evolution determines which actions need to be represented in a team, the number of times an action appears in the same team, and the contexts in which actions should be applied. To determine what action the team applies given an input state, the state features are presented to each team member and the action associated with the highest bidder is selected.

The SBB model evolves three populations [9]. The *team members*, each associating a bid behavior with an action, are evolved in the *learner* population. Since a single learner can suggest only one action, to form non-trivial solutions (involving different actions) it must be combined with other learners. A population of teams, representing useful combinations of learners, is therefore evolved in a symbiotic relationship with the learner population. Each team references a subset of the learner population restricting the bidding competition to this subset. The teams are of varying size, and each learner can appear in more than one team. A single team *may* represent a complete solution that can be applied to solving a non-trivial problem instance. The team population is thus evolved against a population of initial environmental states (points) through distinction-based competitive coevolution [6]. In this regard, the teams are evaluated on how well they perform given the initial states, while the points are evaluated on how well they distinguish between the teams.

Once a team is created the subset of learners it references does not change. Different combinations of learners are explored when offspring teams are created. Likewise, once an individual learner is defined its bid behavior and action are also fixed – alternate associations of bid behavior and action can only be explored in the offspring. While the set of possible actions is problem dependent, the space of possible bidding behaviors is a function of the GP implementation.

2.2 Training Algorithm

Algorithm 1 provides an overview of the SBB training process. The point, team, and learner populations are initialized in lines 3 and 4. The main loop, line 5, represents an iteration over a single generation: the creation of new points, teams, and learners, lines 6 and 7, the evaluation of teams on points, line 10, and the removal of points, teams, and learners, lines 13 and 14. Following the main training loop, a single best team is identified, line 17.

Initialization, lines 3, 4. No assumptions are made about the preference of certain points over others. As such, $P_{size} - P_{gap}$ points are sampled from the problem space with uniform probability. Here, P_{size} refers to the size of the point populations after point generation, line 6 of Algorithm 1, and P_{gap} is the number of points created and removed every generation.

Team initialization is performed in two steps and results in $M_{size} - M_{gap}$ teams (M_{size} and M_{gap} defined analogously as with respect to the points). In the first step, teams of size two are created by combining two new learners of different actions. Bid program generation and action selection is random and assumes a uniform probability. The learners created are included in the learner population. At the end of this first step, there are a total of $M_{size} - M_{gap}$ teams and twice as many learners, and each learner appears in exactly one team.

The goal of the second team initialization step is to explore larger combinations of learners and to place learners in multiple contexts. For each team generated in the first step, a team size is uniformly selected from $\{2, 3, \ldots, \omega\}$. Learners are then selected and added to the team until its chosen size is reached using a tournament heuristic that favours learners with fewer references.

Algorithm 1 Overview of the SBB training algorithm.

```
1: procedure TRAIN
2:     t = 0                                                    ▷ Initialization
3:     initialize point population P^t
4:     initialize team population M^t (and learner population L^t)
5:     while t ≤ t_max do                                       ▷ Main loop
6:         create new points and add to P^t
7:         create new teams and add to M^t (add new learners to L^t)
8:         for all m_i ∈ M^t do
9:             for all p_k ∈ P^t do
10:                evaluate m_i on p_k
11:            end for
12:        end for
13:        remove points from P^t                               ▷ Form P^{t+1}
14:        remove teams from M^t (remove learners from L^t)     ▷ Form M^{t+1}, L^{t+1}
15:        t = t + 1
16:    end while
17:    return best team in M^t
18: end procedure
```

Whenever a new bidding behavior (program) is generated, as in lines 4 and 7 of Algorithm 1, a check is performed against the learner population to determine if the new behavior is a duplicate of an existing behavior. This check is based on bidding profiles, where, for a given program, its bidding profile consists of its bid values on 100 uniformly selected points (this set is fixed over the entire run). If two programs result in the equal bids (values within 10^{-7}) across all points, they are considered duplicates. The set of implementation specific search operators is then applied to a new program as long as it duplicates the behavior of an existing program.

Creation of new individual, lines 6, 7. Point creation generates new points in one of two ways and adds them to P^t until it contains P_{size} points. With a probability of 0.9, a point is generated as an offspring of an existing point. First, a species is selected using roulette wheel selection with probability proportional to the species fitness. Once a species is selected, a parent point is selected from within that species with uniform probability. Problem dependent search operators are applied to the parent to produce an offspring that is then added to P^t. With a probability of 0.1, a point is generated by uniformly sampling the space as in the initialization step, line 3 of Algorithm 1.

New teams are always created from existing teams through mutation. First, a parent team is selected with uniform probability and an initial offspring formed as a copy of the parent. Mutation is applied in three steps and only to the offspring (so the parent team remains unaltered). (1) Respecting the minimum team size of two, uniformly selected references are removed from the offspring. The probability that exactly n references are removed is $(\mu_d)^{n-1}(1-\mu_d)$ where μ_d is the probability of learner deletion. (2) Respecting the maximum team size

ω, learners are uniformly selected from the learner population and added to the team. Whether a learner is to be added is determined as in learner deletion but considering the probability of learner addition, μ_a. (3) With probability μ_m that an individual learner is affected, learners in the offspring team are mutated. First, the learner to be mutated is copied into a new offspring learner and the reference in the offspring team is updated accordingly. With probability μ_n, the action associated with the learner is changed to a uniformly selected action, and the associated bid program is *always* altered (implementation specific). This step is repeated until at least one learner in the offspring team is mutated. By always operating on copies of parent teams/learners, the team generation procedure never disturbs existing teams. New teams are added to M^t until it contains M_{size} teams. New learners are added to L^t whose size may fluctuate but never surpasses $|M^t| \times \omega$.

Evaluation, line 10. The goal of the evaluation step is to determine the outcome of applying team m_i to point p_k, where this outcome is denoted as $G(m_i, p_k)$. It is assumed that outcomes fall in the unit interval and that higher outcomes are preferred.

Removal of individuals, lines 13, 14. Point fitness is determined using a form of genotypic speciation where in generation t the set of point species S^t is determined using the species in the previous generation (S^{-1} is assumed to be empty). First, the problem-specific genotypic distance between every pair of points is determined. Using the state feature vectors of points assigned to species in S^{t-1}, the centroid of each of these species is calculated as the arithmetic mean with respect to each feature. Next, the points not assigned to a species (those created during the current generation) are assigned to the closest species provided that it is within a radius of δ_s. Here, the distance to each species is determined by comparing the species' centroid to the unassigned point's state feature vector. If a point is not within δ_s of an existing centroid, it is used to form a new species whose centroid is (for the current generation) assumed to be that point. Once assigned a point's species remain fixed for the point's lifetime.

After the species are determined, the points' distinction vectors are calculated. Given the M_{size} teams, the distinction vector for the kth point p_k is defined as

$$\boldsymbol{d}_{p_k}[M_{size} \cdot i + j] = \begin{cases} 1 \text{ if } G(m_i, p_k) > G(m_j, p_k) \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

where i, j iterate over all teams and m_i, m_j are the ith and jth team (i.e., the size of the distinction vector is the square of M_{size}). When G returns real outcomes, a team epsilon ϵ_{team} defines when an outcome is considered to be strictly greater. The distinction vectors are then used to calculate a raw competitive fitness sharing score [12] as

$$f_{p_k} = \sum_q \left(\frac{\boldsymbol{d}_{p_k}[q]}{\sum_l \boldsymbol{d}_{p_l}[q]} \right)^3 \tag{2}$$

where q iterates over all dimensions of $\boldsymbol{d}_{p_k}$ and l iterates over all points. Thus, distinctions that are made by many points are assigned less worth. Finally, the

raw fitness sharing score for p_k is normalized by the cube of the size its species,

$$f'_{p_k} = \frac{f_{p_k}}{|S^t(p_k)|^3} \tag{3}$$

where $S^t(p_k)$ denotes p_k's species at time t. Based on this normalized fitness, the worst P_{gap} points are removed. Following point removal, a species fitness is calculated for each non-empty species as the mean fitness of its member points so that it can be used during point generation, line 6 of Algorithm 1.

The fitness calculation for team m_i follows Eq. 2 but is based on the outcomes $G(m_i, p_k)$ over all points p_k (i.e., there are P_{size} terms in the sum) instead of the distinction vectors. Genotypic speciation is not applied to the teams. The lowest-scoring M_{gap} teams are removed from the population. Any learners not referenced by at least one team are then removed from the learner population.

Two issues motivate the use of competitive fitness sharing instead of Pareto-dominance for selecting points and teams. First, the large number of objectives makes applying the Pareto-dominance relation impractical (e.g., all individuals tend to form a non-dominated set). Second, the competitive fitness sharing score is meant to discourage the overlapping behavior that tends to occur under Pareto-dominance [10].

Selection of best team, line 17. At the end of training, the team population contains multiple solutions. Given 1000 uniformly selected points, the best team is identified as the one with the highest mean outcome across this set[1].

3 Experiments

3.1 The Truck Reversal Problem

Experiments were performed on the truck reversal environment [2] with an additional wall obstacle. Specifically, the point population specified starting configurations of the cab and semi, and the teams assumed steering responsibility given a starting state. Unlike maze-style problem domains the only inputs to the team were the cartesian coordinates of the semi, (x, y), and angle of the cab, Θ_c, and semi, Θ_s. Performance was evaluated over truck configurations established by the point population. The single action that each learner could assume was selected from $\{0°, \pm 35°, \pm 70°\}$, and an episode ended when: (1) the back of the semi crossed the y-axis, (2) the cab and semi jackknifed, (3) the cab and semi did not have sufficient time to return to the origin assuming a straight-line path, (4) the back of the semi entered a block with upper-right corner at $(45, 50)$ and lower-right corner at $(55, -50)$, or (5) 600 time steps elapsed. Relative to previous research employing the truck reversal environment, the formulation used here has an open-ended state space of training configurations, explicitly terminates a trial when jackknifing occurs, and introduces an obstacle into the world. Conversely, [7] applied GP to a problem formulation without the wall using a fixed symmetric set of 8 configurations for both training and testing.

[1] This set is different from the test points used in the evaluation.

The (x, y)-coordinates of starting states were always outside of the block and restricted to $(0, 100) \times (-100, 100)$, while Θ_s was always initially equal to Θ_c. The distance between two states (used in speciation) was defined as the square of the Euclidean distance with respect to the x, y, and Θ_s components, where the range of each was normalized to the unit interval. Offspring points were generated from parent points by independently mutating the state components by a normally distributed amount, assuming a mean of 0 and standard deviation of 5, 10 for x, y and 18 for the angles respectively.

The reward a team m_i received for a point p_k, $G(m_i, p_k)$, was based on the x, y, and Θ_s in the last time step of the episode. In the *binary* reward scheme, the team received a reward of 1 if the x and y components were both within 1 meter of the origin and $|\Theta_s|$ was less than 45°, and a reward of 0 otherwise (the absolute value of both angles was always within $[0, 180]$). In the *real-valued* reward scheme, the reward was set to $(x^2 + y^2 + \Theta_s^2 + 1)^{-1}$, so states closer to the origin and aligned with the x-axis received higher reward. Both fitness functions returned 0 for states where the cab and semi were in a jackknifed position.

3.2 Genetic Programming Implementation

Linear GP [3] was used to evolve the bid programs, assuming register-register and register-input modes, using eight registers (initialized to 0), and function set $\{+, -, \times, \div, cos, ln, exp, cond\}$. The *cond* operator compared the value of the destination register with the source value (input or another register) and if the destination was smaller flipped its sign. To obtain bids in the unit interval, the raw GP output y was transformed using $y' = (1 + e^{-y})^{-1}$. Four search operators were applied to parent programs to obtain offspring: *add* and *delete* removed and inserted arbitrary instructions, *mutate* flipped a single random bit, and *swap* exchanged the locations of two instructions (probabilities of 0.5, 0.9, 0.9, 0.9 respectively). Maximum program size was 48. A uniform distribution was assumed in setting bits and in selecting instructions to add, delete, and swap.

3.3 Configurations

The competitive interaction between training points and teams will be investigated using three variants of the algorithm: SBB with competitive coevolution (CC, the algorithm as described in Section 2.2), SBB with competitive coevolution but no speciation (CC-NS), and SBB with competitive coevolution replaced by random subset selection (RSS). CC-NS is like CC except that it sets the niching radius, δ_S, to infinity so that all points are assigned to the same species. Thus, CC-NS can be used to identify the contribution of the speciation mechanism. RSS replaces the competitive coevolution component with random subset selection so that at each generation P_{gap} points are uniformly sampled from the environment during point generation and P_{gap} uniformly selected points are purged during removal. Comparison with RSS is meant to establish the contribution of developing points through the competitive interaction versus a policy

based purely on diversity. This results in a total of six configurations given the two reward function formulations, binary and real-valued.

Unless noted otherwise, the parameters used were as follows: P_{size} 64, M_{size} 128, P_{gap} 8, M_{gap} 64, μ_d 0.3, μ_a 0.4, μ_m 0.1, μ_n 0.1, δ_S 0.2, ϵ_{team} 10^{-3}, ω 10. Thirty initializations, each training for 1000 generations, were performed using each configurations under both the binary and real-valued reward scheme.

3.4 Results

Performance was evaluated on an independent set of 1000 test points sampled uniformly but with the following restriction: Points which, along a straight line, backed the cab and semi into the block were not allowed. The number of these points solved by the best team, line 17 of Algorithm 1, and the entire population at the end of training were recorded for each initialization. The population-wide behavior was assessed to determine the degree to which the fitness sharing function, Eq. 2, promotes useful diversity in the teams. The thresholds on x, y, and Θ_s used in the binary reward definition were used to determine if the final state was sufficiently close to the origin for it to be considered solved.

Binary rewards, Figure 1 (left), appear to result in a strong preference for competitive coevolution; the RSS model effectively being unable to identify configurations of the environment that lead to non-zero outcomes. Moreover, the combination of the competitive interaction with point speciation (CC) results in improved team behaviour. Comparing individual to population-wide performance indicates that performance roughly doubles. Under real-valued rewards, Figure 1 (right), the level of diversity introduced by the RSS model of point sampling results in a significant benefit to the overall behavior (note the different y-axis scales between subplots). Indeed, the RSS population-wide performance is able to leverage team-wise fitness sharing, improving performance three-fold over that of the single best team. Finally, aside from the general quartile trends, strong outlier behaviors are observed, particularly with respect to the individual-wise behavior of CC. That said, CC under real-valued rewards is generally not able to capitalize on this at the population level. In a sense CC emphasizes exploitation more than exploration – a property that works to its advantage under binary rewards. RSS clearly emphasizes exploration, with different teams being associated with different subsets of test configurations. The penalty for this is that under environments limited to binary rewards RSS is unable to establish/maintain engagement between point and team populations.

The contribution of point speciation can be observed by reviewing the distribution of the points during training, Figure 2. The two plots on the left show that by generation 100 the CC run is able to form point species and maintain them through the rest of the run; this results in 853 of 1000 test cases solved. The best case CC-NS run, on the other hand, produces a point population that by generation 100 has converged, thus only generalizing to solving 271 test cases, most of which fell to the left of the obstacle (not shown). Points near and oriented towards the origin can be viewed as easier, and although they make many distinctions, Eq. 1, their aggregate usefulness is limited since they are in the

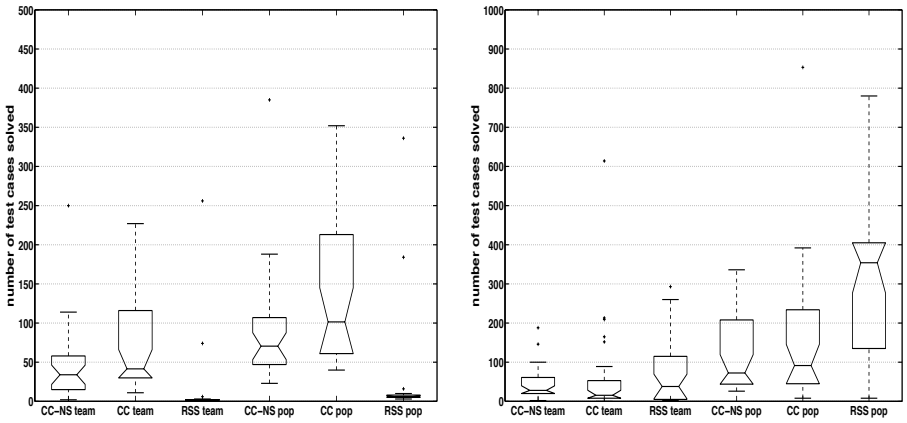

Fig. 1. Distributions under the binary (left) and real-valued (right) rewards across 30 initializations. Box endpoints represent first and third quartiles, the line inside the box the median. Whiskers extend to points within 1.5 times the interquartile range, and other outliers are noted with a '+'. A point in a distribution represents the number of cases solved by the best team in an initialization (distributions labeled 'team' on x-axis) or across all teams at the end of training in the initialization (distributions labeled 'pop').

same local region of the environment. This lack of point diversity precludes the development of a general set of solutions in the team population.

Figure 3 summarizes several representative trajectories employed by the best CC team (same initialization as in Figure 2 (left)). In effect, the basic approach is to reverse the tractor-trailer to the top right (thus avoiding the wall obstacle) and then swing back to the origin. Needless to say, this results in a preference for solving test cases along this path, Figure 4 (left), where points not along this path are typically not solved, Figure 4 (middle). This particular team tends to fail on points because (1) they are too close to the origin leaving little room for maneuvering, (2) they result in jackknifing, or (3) they do not swing the tractor-trailer quite far enough so that it clips the top-right of the wall obstacle, Figure 4 (right). However, this reflects the single best team, with another 200 or so of the failed test cases correctly solved by other teams in the same population.

To gain insight into the nature of the CC versus RSS behavior across teams in the same population, the *absolute* contribution of each individual in the team population is compared to the combined *accumulative* improvement in test cases solved across the population as a whole, Figure 5. Here, the number of test cases solved by the best team is indicated at $x = 1$. The contribution made by considering additional teams, as measured by the number of *unique* test cases solved, is indicated by the solid line. The degree overlap in the test cases solved can be gauged by considering the absolute number of test cases solved by the team as indicated by the dashed line. Under CC (left) the initially very strong individual of Figure 3 is identified from which no more than five more individuals

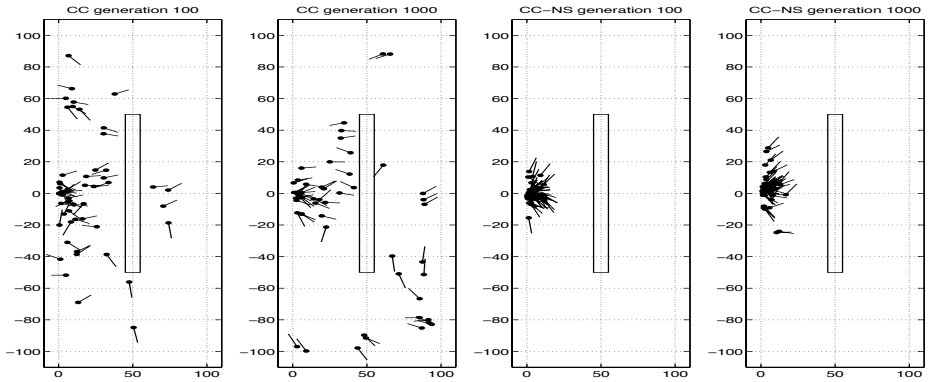

Fig. 2. Contents of point population in the best CC and CC-NS initializations (with respect to the test cases solved) after 100 and 1000 generations under real-valued rewards. Each plot shows the 56 surviving points, with the dot representing the coordinate of the back of the semi. The rectangle in the middle of the area represents the block.

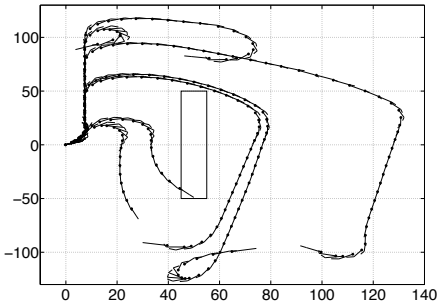

Fig. 3. Trajectories of the best case CC team under real rewards on a sample of solved cases. States are plotted every ten time steps.

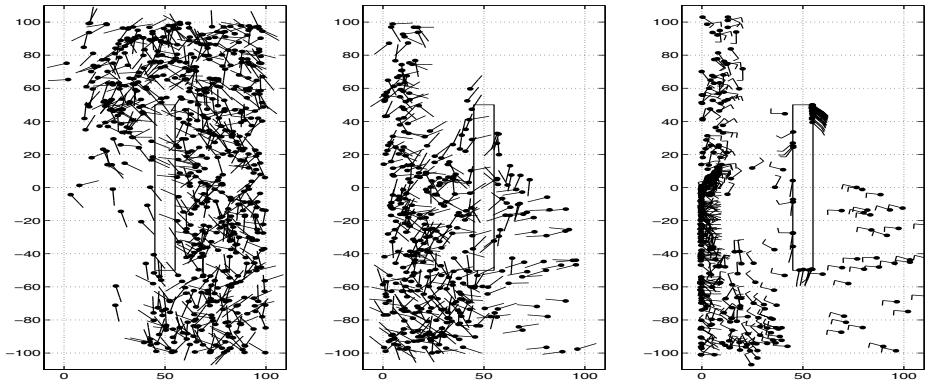

Fig. 4. Behaviour of the best (in terms of test cases solved) team across all CC runs: Solved cases (left), unsolved cases (middle) and final states for unsolved cased (right).

are able to provide a significant contribution to the base behavior. Conversely, under RSS the initial single best case team is relatively weak. However, about ten other teams are then able to significantly contribute resulting in a similar cumulative count of test cases solved.

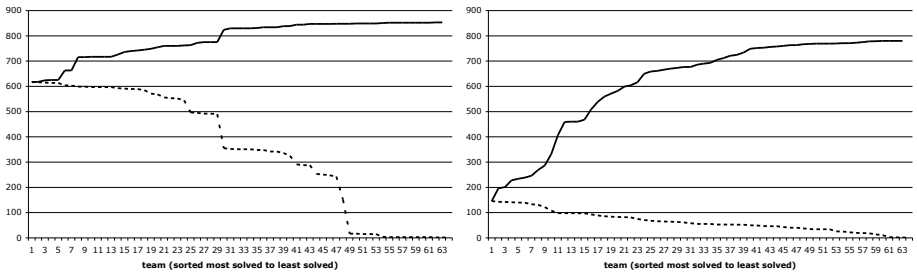

Fig. 5. Absolute and cumulative counts of test cases solved in the best CC (left) and RSS (right) initializations under real-valued rewards. The x-axis represents teams at the end of training sorted so that teams solving more cases are on the left. Going left to right, the number of test cases solved by the team (dashed line, absolute) and the number of test cases solved by the team and all the teams to its left (solid line, cumulative), are plotted.

4 Conclusions

A study was performed on problem decomposition under the SBB teaming model for building reinforcement learners under an environment with a large state space, thus necessitating a competitive coevolutionary interaction between candidate solutions and the environment. The fitness sharing component was found to be effective in producing non-overlapping behaviours across multiple teams during a single run, resulting in two to three times the performance achievable by the single best team. An evaluation was performed on the formulation of the competitive coevolutionary component of the model. Specifically, binary and real-valued rewards were considered, as was the significance of speciation in the point populations. Comparisons were also made against pure stochastic sampling of points (RSS). When the available rewards are limited to binary outcomes CC was found to outperform RSS, and speciation of the points provided an additional advantage. However, the structure of point replacement enforced by CC under real-valued rewards was generally not able to match RSS under real-valued rewards. The additional information provided by a real-valued reward function enabled the RSS heuristic to identify and support multiple teams making effective use of fitness sharing. Conversely, CC appeared to need additional diversity to support more teams concurrently. In the wider context, distinguishing classes of problem domains that explicitly conform to the binary/real-valued world view would naturally establish when to use each reward function. Opportunities for

future work include additional diversity mechanisms for the point population and further levels of symbiosis to provide a team integration mechanism by evolving 'team-of-teams'.

Acknowledgments

The authors gratefully acknowledge the support of MITACS, NSERC, CFI, and the Killam Scholarship program.

References

1. A. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.
2. C. W. Anderson and W. T. Miller. A challenging set of control problems. *Neural Networks for Control*, pages 475–508, 1990.
3. M. Brameier and W. Banzhaf. *Linear Genetic Programming.* New York, NY: Springer, 2007.
4. E. D. De Jong and J. B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12:159–192, 2004.
5. K. L. Downing. Reinforced Genetic Programming. *Genetic Programming and Evolvable Machines*, 2(3):259–288, 2001.
6. S. G. Ficici and J. B. Pollack. Pareto optimality in coevolutionary learning. In *Proceedings of the 6th European Conference on Advances in Artificial Life*, pages 316–325. Berlin: Springer-Verlag, 2001.
7. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: MIT Press, 1992.
8. P. Lichodzijewski and M. I. Heywood. Coevolutionary bid-based Genetic Programming for problem decomposition in classification. *Genetic Programming and Evolvable Machines*, 9(4):331–365, 2008.
9. P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with symbiotic bid-based Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.
10. A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in Pareto competitive classifier models of coevolution. In *Proceedings of the European Conference on Genetic Programming*, volume 4971 of *LNCS*, pages 289–300, 2008.
11. D. E. Moriarty, A. C. Schultz, and J. J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.
12. C. D. Rosin and R. K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5:1–29, 1997.
13. K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time Neuroevolution in the NERO Video Game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668, 2005.
14. M. E. Taylor, S. Whiteson, and P. Stone. Comparing evolutionary and temporal difference methods in reinforcement learning domain. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1321–1328, 2006.
15. S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7:877–917, 2006.