Hybridizing Novelty Search for Transfer Learning

Sabre Didi
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: sabredd0@gmail.com

Geoff Nitschke
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: gnitschke@cs.uct.ac.za

***Abstract*—This study investigates the impact of *genotypic* and *behavioral diversity maintenance* methods on controller evolution in multi-robot (*RoboCup keep-away soccer*) tasks. The focus is to examine the impact of these methods on the *transfer learning* of behaviors, first evolved in a *source task* before being transferred for further evolution in different but related *target tasks*. The goal is to ascertain an appropriate controller design (NE: *Neuro-Evolution*) method for facilitating improved *effectiveness* given policy transfer between source and target tasks. Effectiveness is defined as the average task performance of transferred behaviors. The study comparatively tests and evaluates the efficacy of coupling policy transfer with several NE variants. Results indicate a hybrid of *behavioral diversity maintenance* and *objective-based search* yields significantly improved effectiveness for evolved behaviors across increasingly complex target tasks. Results also highlight the efficacy of coupling *policy transfer* with the hybrid of behavioral diversity maintenance and objective based search in order to address *bootstrapping* and *deception* problems endemic to complex tasks.**

I. Introduction

In *Evolutionary Robotics* (ER) [1] there is increasing empirical evidence that maintaining diversity in *genotypes* (controller encodings) and *phenotypes* (controller behaviors) improves the quality (task performance) of evolved behaviors [2], [3], [4], [5]. Behavioral diversity maintenance methods to boost search *effectiveness* have received increased attention recently and significant benefits have been demonstrated in various simulated [6], [2], [4], and physical [5], [3] ER tasks.

Also, the benefits of genotypic diversity maintenance is well explored in evolutionary computation research [7], including niching techniques such as *fitness sharing* and *crowding* [8] and *multi-population models* [9]. Though genotypic diversity maintenance methods in ER tasks has received some research attention [2], there are relatively few studies using multi-robot [10], [4] and swarm-robotic tasks [11]. Also, ER studies comparing genotypic and behavioral diversity maintenance methods demonstrate that behavioral diversity maintenance consistently out-performs genotypic diversity maintenance [2], [12], [13].

Furthermore, there has been little research using NE coupled with genotypic and behavioral diversity maintenance to adapt robot behaviors and then facilitate behavior transfer between different but related tasks (*transfer learning* [14]). Previous work has demonstrated that such methods increase the effectiveness of behavior evolution in these related tasks [15], [16]. However, such work uses objective (fitness function) based search in controller evolution. Thus, the focus of this study is to investigate the impact of genotypic and behavioral diversity maintenance on transfer learning[1].

Transfer learning attempts to improve task learning by leveraging knowledge from learning a related but simpler task [14]. Transfer learning reuses learned information across tasks, where information is shared between a source and target task, and used as a starting point for learning new behaviors in target tasks. Transferring knowledge that is learned on a source task accelerates learning and increases solution quality in target tasks by exploiting relevant prior knowledge. Transfer learning has been widely studied in the context of *reinforcement learning* for single-agent tasks such as robot navigation and game-playing and some multi-agent tasks [17], where behavior transfer is between the same task with varying complexity.A popular multi-agent test-bed is *RoboCup keep-away soccer* [18], which has received significant attention in multi-agent transfer learning research [15] and is thus used in this study.

This study extends previous work [19] demonstrating that *Hypercube-Based Neuro-Evolution for Augmenting Topologies* (HyperNEAT) [20] for controller evolution and coupled with behavioral diversity maintenance (*novelty search* [6]) yielded significant benefits given behavior transfer between a range of keep-away tasks. This work compared three variants of NEAT and HyperNEAT (*novelty*, *objective* and *hybrid novelty-objective* based search). In all tasks and methods tested, the hybrid yielded the greatest benefits in terms of team task performance (*effectiveness*) when coupled with policy transfer. This study tests five variants of HyperNEAT, though comparatively evaluates these variants for longer evolution times in more *complex* tasks (defined by higher dimensional search spaces). We only used HyperNEAT as previous work demonstrated its efficacy over NEAT as an appropriate *Neuro-Evolution* (NE) method for facilitating behavior transfer between multi-agent tasks of increasing complexity [19].

Specifically, this study tests *objective-based search* (OS), *novelty search* (NS), *hybrid novelty-objective based search* (ONS), *genotypic novelty search* (GNS), and a *hybrid GNS-objective based search approach* (OGN). The NE search process is of particular interest, as several studies have supported the efficacy of objective-based search in transfer learning [15], [16], though the impact of *genotypic* and *behavioral diversity maintenance* on transfer learning has received little research attention. This study's key contribution is its demonstration of an effective NE search method to couple with policy transfer in (keep-away soccer) tasks of increasing complexity.

[1] *Transfer learning* and *policy (behavior) transfer* are used interchangeably.

A. Research Objectives (Hypotheses)

The study thus has two research objectives. First, to extend previous work [19] via coupling behavior transfer with genotypic and behavioral diversity maintenance variants of HyperNEAT. This coupling is tested for its capability to adequately address the problems of *bootstrapping* [9] and *deception* [21] in increasingly complex keep-away tasks.

Second, to further test (in keep-away soccer) a hypothesis that combining behavioral diversity maintenance and objective-based search is an effective means for balancing exploitation and exploration in controller synthesis for complex and deceptive evolutionary robotics tasks [2], [12], [13], [4].

II. Methods

To address this study's objectives (section I-A), experiments use five variants of HyperNEAT (section II-A) for controller adaptation in keeper teams in keep-away soccer (section III). HyperNEAT was selected given its successful application in related work [19]. Keeper teams are homogenous so a single controller is evolved to represent team behavior. The taker team uses a heuristic controller [19]. The following describes the HyperNEAT variants and their application.

NS: The fitness function was replaced with a behavioral diversity maintenance metric [6].

OS: A fitness function [20] directs evolutionary search. Fitness (section II-B) is average keeper ball hold time (average simulation time that the keepers have possession of the ball).

ONS: The fitness function was replaced by a hybrid objective-novelty search function (section II-B).

GNS: The fitness function was replaced by a genotypic diversity maintenance (section II-C).

OGN: Objective-based GNS hybrid search (section II-B).

A. HyperNEAT: Hypercube-based NEAT

Hypercube-based NEAT (HyperNEAT) [20] is an indirect (generative) encoding method that extends NEAT and uses two networks, a *Composite Pattern Producing Network* (CPPN) [22] and a *substrate* (ANN: *Artificial Neural Network*). The CPPN uses generative encoding to indirectly map evolved genotypes to ANNs and encode pattern regularities, symmetries and smoothness of task geometry as the substrate. This mapping functions via having coordinates of each pair of nodes connected in the substrate fed to the CPPN as inputs. The CPPN outputs a value assigned as the synaptic weight of that connection and a value indicating whether that connection can be expressed or not. HyperNEAT uses the evolutionary process of NEAT [23] to evolve the CPPN and determine ANN fitness values. The main benefit of HyperNEAT is scalability as it exploits task geometry and thus effectively represents complex solutions with minimal genotype structure [20]. This makes HyperNEAT an appropriate choice for evolving complex multi-agent solutions [16], [24].

HyperNEAT's capability to evolve controllers that account for task geometry also makes it appropriate for deriving controllers that elicit behaviors robust to variations in state and action spaces and noisy, partially observable environments of multi-agent tasks [25]. Also, it has been demonstrated that transferring *connectivity patterns* [26] of evolved behaviors is effective for facilitating behavior transfer between multi-agent tasks [16]. That is, HyperNEAT evolved behaviors can be transferred to increasingly complex versions of keep-away without further adaptation, and transferred behaviors often yield comparable task performance to specially designed learning algorithms [16].

1) HyperNEAT Keeper-Team Controller: HyperNEAT is also used to evolve on controller for a keeper team. The CPPN has four coordinate inputs and a bias node with a constant value of 1.0, and two outputs (figure 1). The coordinates x_1, y_1, x_2, y_2 are of two sampled nodes. That is, x, y coordinates of *node 1* and *node 2* on the input of the substrate network. The CPPN outputs are synaptic weight values assigned to connections between *node 1* and *node 2*, and a connection expression value, *Link Expression Output* [27], determining if a connection can be created.

Methods from previous work are used to represent keep-away task states. Specifically, field size and relative positions of the ball, takers and keepers are represented using *Birds Eye View* (HyperNEAT-BEV) [16]. HyperNEAT-BEV uses indirect encoding so it can represent task complexity changes without genotype representation changes. A 20×20 square grid field was encoded on a two-dimensional substrate with 20×20 input layer and 20×20 output layer with coordinates in the x, y plane in the range $[-1.0, 1.0]$, where a 400×400 input-output vector yielded $160,000$ possible connections. Each field grid square was represented by a substrate network node. Keeper and taker positions were marked with values 1.0, -1.0, respectively.

In task simulation, straight line paths were calculated from the keeper with the ball to all other agents. If the path intersected another keeper then this node to node connection was assigned a 0.3 value. If the path intersected a taker a -0.3 value was assigned to this node to node connection. Otherwise, 0.0 was assigned if there was no agent in a grid square on the path. Thus, the number of keepers was indicated by the number of squares having a 1.0 value. Figure 1 presents an example of a HyperNEAT evolved controller, with the substrate (20×20 grid of inputs and outputs) encoding task environment state.

B. Behavioral Diversity and Objective-based Search

Many NE methods have incorporated behavioral diversity maintenance into their search processes as a means of discovering novel and higher quality solutions, compared to the same methods using objective based search [2].

1) Novelty Search (NS): NS [6] is a search process that rewards evolved behaviors based on their novelty. Thus, a genotype is more likely to be selected for reproduction if its encoded behavior is sufficiently different from all other behaviors produced thus far in artificial evolution. NS has been demonstrated as yielding solutions that out-perform objective based search in various multi-agent tasks [28], [4]. Given this, NS was selected as the behavioral diversity mechanism for controller evolution.

The function of NS is to consistently generate novel team (keep-away) behaviors. Hence, we define team behavior in terms that potentially influence team behavior but are not

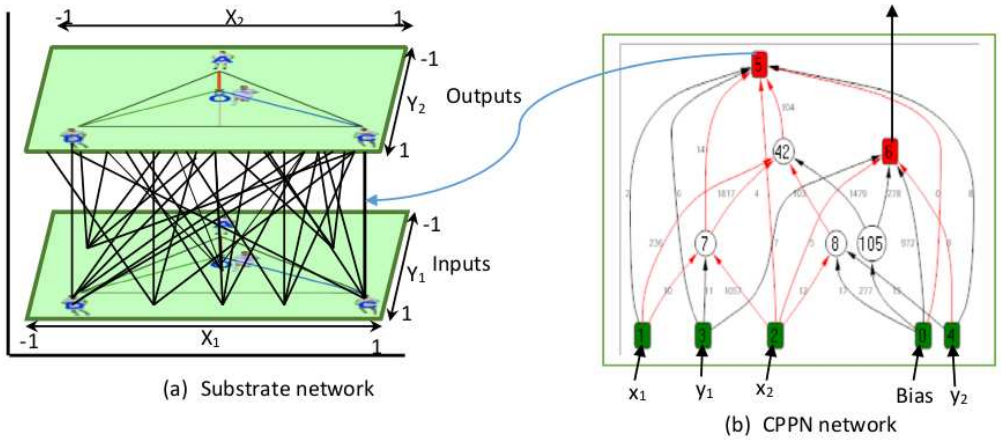

Fig. 1. **Left:** Substrate encoding the virtual field ($20x20$ grid of inputs and outputs). Connection values between input-output nodes represent agent positions relative to the center of the field. **Right:** Node-pair connections in the substrate are sampled and coordinates passed as CPPN inputs. The CPPN then outputs the synaptic weight of each sampled connection.

directly used for task performance evaluation. That is, we use the behavioral properties: *average number of passes*, *dispersion of team members*, and *distance of the ball to the center of the field*. To measure novelty we normalize each of these properties as task specific behavioral vectors, where the addition of these vectors is always in the range: $[0, 1]$. This team level behavioral characterization has been used previously [29] and out-performs individual behavioral characterizations. Behavioral distance is computed using equation 1:

$$\delta_i(x, y) = \|x_i - y_{ij}\| \tag{1}$$

Where, $x_i$and y_{ij} are normalized behavioral characterization vectors of two genotypes. Novelty is then quantified by equation 2, which replaces the fitness function of HyperNEAT.

$$nov_x = \frac{1}{3k} \sum_{i=1}^{k} \sum_{j=1}^{3} \delta(x_j, y_{ij}) \tag{2}$$

Where, x_j is the j^{th} behavioral property of genotype x, y_{ij} is the j^{th} behavioral property of the i^{th} nearest neighbor of genotype x and δ is the behavioral distance between two genotypes x and y computed in equation 1. The nov_x then is derived from the mean behavioral distance of an individual with k nearest neighbors. The parameter k (number of nearest neighbors) is user specified. Related work used $k = 20$ [30] and $k = [3, 10]$ [13] with varying results. Gomes et al. [13] found k values are highly dependent on the type of novelty archive, where $k = 15$ yielded relatively good performance across all tested archives. Hence this study uses $k = 15$.

As in related work [6], the novelty of newly generated genotypes is calculated with respect to previously novel behaviors stored in the novelty archive, where archived behaviors are ranked by diversity. The maximum archive size is 1000 (given results of related work [13]), where a maximum of 10 behaviors are added to the archive each generation (table I).

2) Objective-based Search (OS): Uses the following fitness function to compute mean episodic length using equation 3:

$$fit_x = \frac{1}{N} \sum_{j=1}^{N} T_j \tag{3}$$

Where, the length of an episode j is T_j, N is the number of task trials (simulation length), and T_j is the length of trial j. Objective-based search is attained by setting $\rho = 1.0$ in equation 4 giving a zero weighting to novelty search.

$$score_i = \rho \cdot \overline{fit_i} + (1 - \rho) \cdot \overline{nov_i} \tag{4}$$

Where, $\overline{fit_i}$ and $\overline{nov_i}$ are normalised fitness and novelty of i^{th} genotype respectively, $\rho \in [0, 1]$ is user selected to control the relative contribution of each metric to selection pressure.

3) Hybrid Objective-Novelty Search (ONS): In line with previous hybrid NS research [29], we use a metric that linearly combines NS with the objective-based search of HyperNEAT (equation 4). Previous work demonstrated that a medium to high novelty weight 50-80% yields the best results [11]. We found that a novelty weight of 40% yielded the best results in this case study (table I). All other novelty search parameters are the same as used for the NS variant.

C. Genotypic Diversity and Objective-based Search

1) Genotypic Novelty Search (GNS): Is similar to NS (section II-B) except that behavioral search is directed by a search for novel genotypes (controller encodings). The genotypic distance between two genotypes is measured using linear combination of *Excess* (E) and *Disjoint* (D) genes [23], and a mean weight difference of matching genes $\overline{W}$ [31] (equation 5). Genes that do not match are either *disjoint* or *excess* depending on whether they occur within or outside the range of parent innovation numbers [23].

$$\delta_g(a, b) = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{5}$$

Where, N is the number of genes in the longest genotype of the population, and coefficients c_1, c_2 and c_3 are parameters used to adjust the weighting of the three factors E, D and $\overline{W}$ respectively. The sparseness (S_g) of genotype x in population evolution is computed by equation 6.

$$S_g(x) = \frac{1}{k}\sum_{i=1}^{k}\delta_g(x, y_i) \tag{6}$$

Where, y_i is the i^{th} nearest neighbor of x, k is the number of nearest neighbors of x and δ_g is the compatibility distance measure (equation 5). The generation n exploration metric is then the population's mean sparseness (equation 7):

$$E_g(n) = \frac{1}{N}\sum_{x=1}^{N} S_g(x) \tag{7}$$

Where, N is the population size parameter and $S_g(x)$ is the sparseness of individual x in generation n computed in equation 6. If exploration measure $E_g(n)$ is high it means the population has genetically diverse genotypes. The *GNS* variant thus uses equation 6 in place of HyperNEAT's fitness function, so as genetically diverse genotypes are selected. The same nearest neighbor and archive parameters are used for GNS as used for the NS variant (section II-B).

2) Hybrid Objective-GNS (OGN): Uses equation 4, except that $\overline{nov_i}$ now represents the genotype diversity metric (equation 6). That is, equation 6 specifying the genotype sparseness in the population (normalised into the range $[0, 1]$) replaces the normalised novelty function value $\overline{nov_i}$ in equation 4. Similarly, $\rho \in [0, 1]$ controls the relative contribution of fitness versus genotypic diversity directed search. Parameter tuning experiments indicated that a genotypic diversity weight of 40% (table I) was appropriate for this study's experiments. All other parameters are the same as used for the GNS variant.

D. Behavior (Policy) Transfer Method

For all HyperNEAT variants, the entire evolved population was transferred from the source task (at the final generation) and set as the initial population for evolution in the target task. Previous research [19] indicated that this method was most effective for HyperNEAT and various keep-away tasks. Using BEV [16] representation, an artificial neural network (substrate) encodes the keepaway task state variables directly to reflect the task geometry. This way the CPPN evolves the solution as a direct function of the task geometry, exploiting the regularities in the task geometry. In this representation a source task and a target task can be represented on the same two-dimensional substrate network. Policy transfer from source to target task can be performed with no change in task representation and in the input-output structure of the evolved CPPN that encodes the geometric relationships of input and output search space.

III. Experiments

Experiments test the impact of the five HyperNEAT variants (section II) as behavior search methods in a broader range of keep-away tasks and for increased evolution times in source and target tasks, compared to previous work [19]. The goal is for a given NE method to evolve keep-away behaviors that keep the ball in their possession and out of the possession of the taker agents for the duration of a task trial (table I). Keeper teams are homogenous so a single controller (evolved by HyperNEAT) represents keeper team behavior. A predefined rule-set controls taker agent behavior [19].

Five target tasks of increasing complexity were tested. NE adaptation time was increased by 50% and 100% in source and target tasks (respectively) [19]. A keeper team was first evolved in a source task of three keepers and two takers (*3vs2*) for 30 generations. Using behavior transfer (section II-D), team behavior evolved at generation 30 is transferred to a target tasks and further evolved for 100 generations (table I).

Keep-away task trials were played on a simulated bounded field[2] (20×20 square grid). A task trial ended when a taker gained control of the ball or the ball was out of bounds. In all experiments, each genotype (keeper team behavior) was evaluated over 30 task trials per generation, where each task trial tested random agent positions. Though, keepers always started near each of the field's corners, takers at random positions close to the center of the field, and the ball in the possession of a (randomly selected) keeper. Average task performance was computed over 30 task trials, where the highest task performance was selected from every generation and a team's average task performance was computed over 20 runs for a given method variant. Table I specifies the NE and simulation parameters used in all experiments.

IV. Results and Discussion

Policy (multi-agent behavior) transfer was applied between the source *3vs2* keep-away task and one of five more complex target tasks, after evolving keeper team behavior in the source task for 30 generations, transferring behaviors to a target task and further evolving for 100 generations (table I). The target keep-away tasks were *4vs3*, *5vs3*, *5vs4*, *6vs4* and *6vs5* (section III). Keep-away behavior was evolved with one of five NE variants (section II), and behavior evolved in the source task was transferred via transferring the *entire evolved population* as the initial population of the target task (section II-D).

Figure 2 presents the average normalized maximum task performance of each HyperNEAT variant in each keep-away target task. Task performance maximums are calculated each run (100 generations) and averaged over 20 runs. Task performance results of evolving from *scratch* (for 100 generations) in the most complex target task (6vs5) is included to indicate the benefits of using HyperNEAT to facilitate transfer learning.

The *MannWhitney U* test was applied in pair-wise comparisons between each HyperNEAT variant for a given target task. Statistical tests were also applied for each HyperNEAT variant in each target keep-away task, with and without policy transfer (for clarity, figure 2 presents only the *6vs5* task, with and without policy transfer). For all target tasks (*4vs3*, *5vs3*, *6vs4* and *6vs5*), statistical tests indicated a significantly higher task performance of each HyperNEAT variant *with* policy transfer, compared to evolution in the same tasks *without*

[2]All experiments were run in *RoboCup keep-away version 6* [15].

Simulation Parameters	Setting
Number of Runs	20
Task trials per generation	30
Agent positions	Random
Environment size	20 x 20 grid
Agent speed (per iteration)	1 grid cell
Ball speed (per iteration)	2 grid cells

HyperNEAT CPPN	Functions
Identity	x
Gaussian	$e^{-2.5x^2}$
Bipolar Sigmoid	$\frac{2}{1+e^{-4.9x}} - 1$
Absolute value	$\|x\|$
Sine	$\sin(x)$
NS / GNS Parameters	**Setting**
NS nearest neighbor k	15
Maximum archive size	1000
Compatibility threshold	3
Behavioral threshold	0.03

Neuro-Evolution (NE) Parameters		Setting
Population Size		150
Generations (Source task)		30
Generations (Target task) / No Policy Transfer		100
Maximum number of species		10
Maximum population per species		30
NEAT / HyperNEAT weight range		[-5.0, 5.0]
Survival threshold		0.2
Mutation rate	Add neuron	0.1
	Add connection	0.18
	Remove connection	0.15
	Weight	0.7
Mutation type		Gaussian

TABLE I. **LEFT:** KEEP-AWAY SIMULATION PARAMETERS. **CENTER:** HYPERNEAT (CPPN ACTIVATION FUNCTIONS), NOVELTY SEARCH (NS) AND GENOTYPIC NOVELTY SEARCH (GNS) PARAMETERS. **RIGHT:** NEURO-EVOLUTION (NE) PARAMETERS.

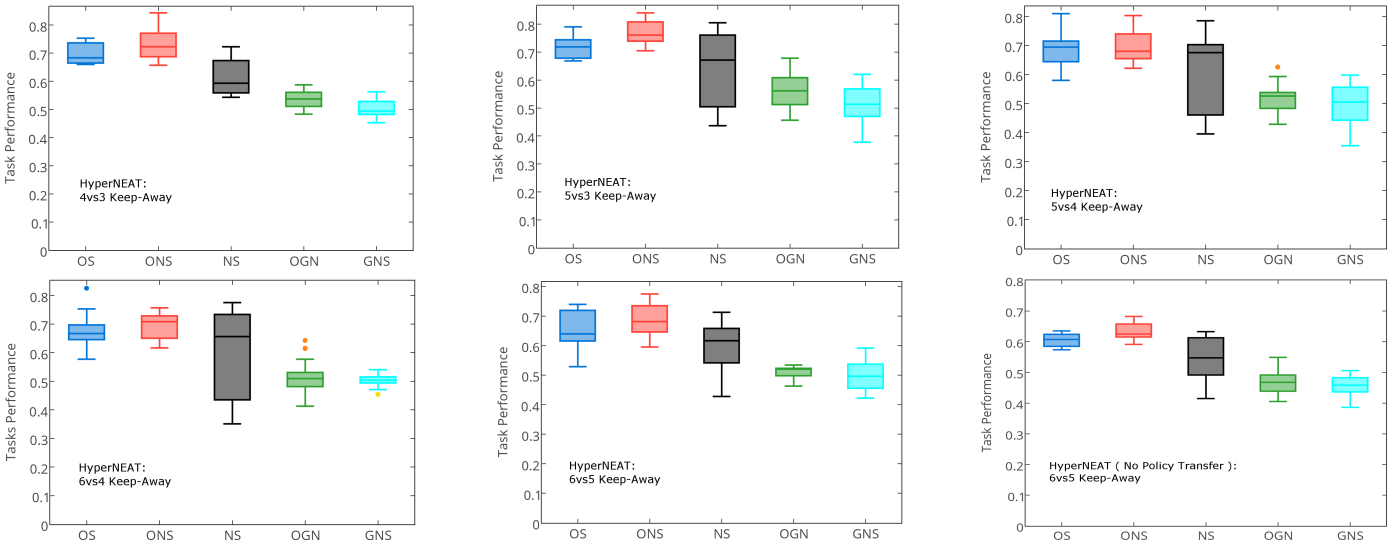

Fig. 2. Box plots of average normalized maximum task performance for the five HyperNEAT method variants: Objective-based search (OS), Novelty Search (NS), Objective-Novelty hybrid (ONS), Genotypic novelty search (GNS), Objective-GNS hybrid (OGN). Averages are calculated over 20 runs and for each target keep-away task. Top row: 4vs3, 5vs3, 5vs4 keep-away. Bottom row: 6vs4, 6vs5 keep-away, and 6vs5 keep-away with no policy transfer.

policy transfer. This supports previous work [19] indicating HyperNEAT as appropriate for policy (behavior) search where policy transfer enables the evolution of significantly higher performance behaviors (compared to evolving behaviors in the same tasks *from scratch*).

A. Complexity and Deception in Keep-Away

Previous research [19] indicated that keep-away task complexity increases not only due to the number of taker agents (making successful passes between team-mates more difficult), but also due to the number of keeper agents. That is, as the number of keeper agents increases, so too does the complexity of the decision making process for passing the ball and moving (as more team-mates must be accounted for), as well as the potential for interference between keeper agents (given that the size of the simulated field does not increase). Consider that at each task simulation iteration of *IvsJ* keep-away (where I and J are the number of keepers and takers, respectively), each keeper must process the 20×20 virtual field space, accounting for and processing $I-1$ team-mates, J taker agents, and the ball.

Task *complexity* is equated with the taker to keeper agent ratio plus the total number of agents (-1 as each keeper must account for its team-mates and $+1$ for the ball). Thus, keep-away tasks, in order of increasing complexity are, *4vs3*, *5vs3*, *5vs4*, *6vs4*, and *6vs5*. The *6vs5* keep-away task is most complex as the increased number of takers increases the likelihood of the ball being intercepted or taken from a keeper.

We consider keep-away *deceptive*, though not *perversely deceptive* as in deceptive maze navigation [6]. That is, in the OS variant, fitness is average keeper ball hold time (section II), and thus keeper behaviors that maintain ball control are rewarded and selected for. Hence, even though the action of moving directly towards the ball can result in ball control, the field position of this keeper is disadvantageous if the keeper is surrounded by takers, thus minimizing chances that the ball can be successfully passed to a team-mate, and maximizing the likelihood that the ball will be taken from the keeper. Thus, such behaviors (satisfying the fitness objective) are deceptive in that evolutionary stepping stones (such as minimizing distance to a ball before attaining an advantageous position for passing) are not rewarded and not selected for. The *6vs5* task is

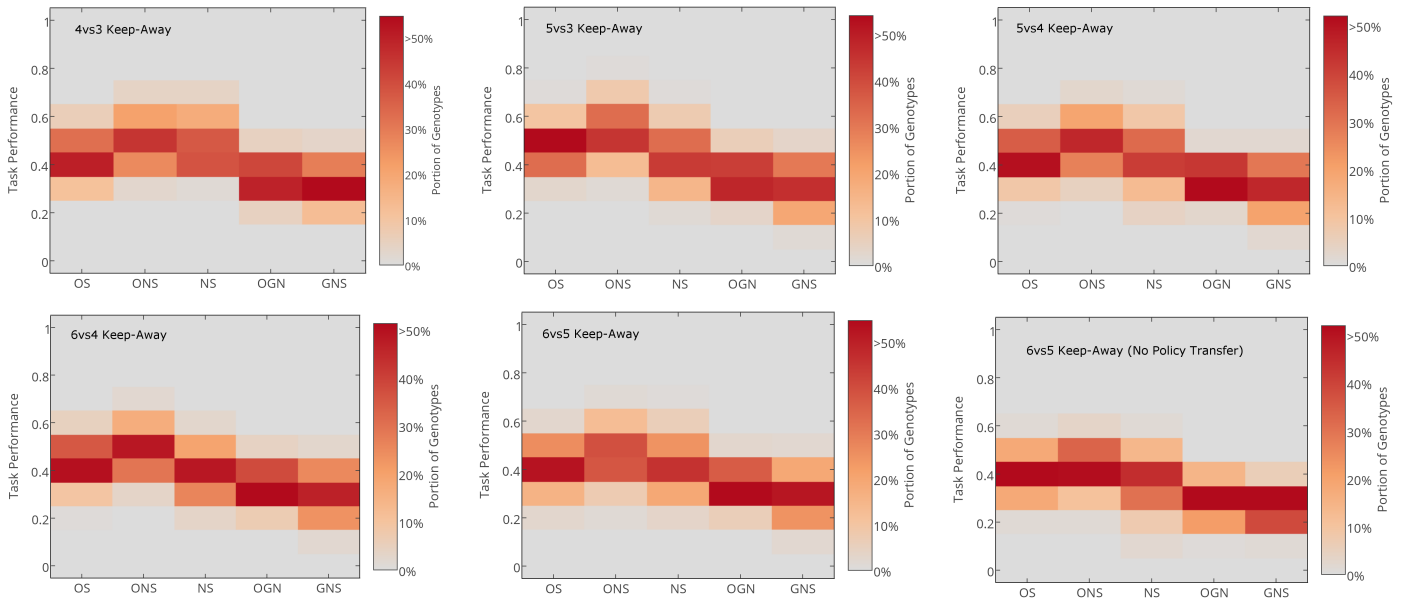

Fig. 3. Heat-maps presenting the portion of genotypes, in the final generation of evolution in a given target task (*4vs3*, *5vs3*, *5vs4*, *6vs4*, *6vs5*), that falls within each 20 percentile of normalized task performance [0.0, 1.0]. Bottom Right: Heat-map for *6vs5* keep-away with *no policy transfer* (to highlight efficacy of policy transfer, where *6vs5* is the most complex task tested). Darker shading indicates a higher portion of genotypes in the given range. OS: Objective-based search. NS: Novelty Search. ONS: Objective-Novelty hybrid search. GNS: Genotypic Novelty Search. OGN: Objective-GNS hybrid search.

construed as the most *deceptive* as with five takers there is an increased likelihood that the keepers (via satisfying the fitness objective) will execute detrimental behaviors.

B. Task Performance Comparisons

Statistical tests comparing all variants *with* policy transfer (in each task) indicated the following.

In *4vs3* and *5vs3* (figure 2, top left and center, respectively) keep-away, *Objective-Novelty* hybrid search (ONS) yielded a significantly higher average task performance, compared to all other variants. Also, for both *4vs3* and *5vs3* keep-away, Objective-based search (OS) yielded a significantly higher average task performance than the *Novelty Search* (NS) variant.

In *5vs4* keep-away (figure 2, top right), the ONS, OS and NS variants, all yielded the highest average performance, where there was no statistical difference between each, but a significantly higher performance compared to *Genotypic Novelty Search* (GNS) and *Objective GNS hybrid* (OGN).

In *6vs4* keep-away (figure 2, bottom left), the ONS variant yielded the highest average task performance, and the OS and NS variants yielded the second and third highest performance, respectively. There was a significant difference between each, though the greatest task performance differences were between the OS, NS, and ONS variants and the GNS and OGN variants.

In *6vs5* keep-away (figure 2, bottom center), the ONS and OS variants yielded the highest performance, with no significant difference between them. There was also a significantly higher performance between ONS and OS, and the NS, GNS and OGN variants. The greatest performance differences were between ONS, OS, NS and the GNS and OGN variants.

The following sections elucidate these results in terms of the *exploration* versus *exploitation* capacity of each variant.

C. Genotypic Diversity Maintenance (GNS, OGN variants)

In terms of the impact of the genotypic diversity maintenance variants (GNS and OGN), these results support related work [12], [13] indicating that in *complex* and *deceptive* [6] tasks, genotypic diversity maintenance approaches perform relatively poorly compared to behavioral diversity maintenance.

Consider search space *exploitation*, the average maximum task performance of GNS and OGN evolved populations. Figure 2 indicates, for all tasks, GNS and OGN yielded significantly lower average task performances compared to OS, ONS and NS (figure 2). In terms of search space *exploration* (the fitness and diversity of the evolved population), figure 3 indicates that, for all tasks, the largest portions of populations (at the final generation) evolved by GNS and OGN were in the lowest performing part of the behavior space. Each heat-map in figure 3 presents, for each task, the portion of genotypes within each 0.2 range of normalized task performance: $[0.0, 1.0]$.

For example, in *6vs5* keep-away (the most complex and deceptive task, section IV-A), 79% and 61% of GNS and OGN evolved behaviors, respectively, were in the (normalised) performance range: [0.0, 0.4], compared to 9% and 23% of behaviors evolved by the behavioral diversity maintenance variants (ONS and NS, respectively). Similarly, in *5vs4* keep-away (the next complex task, section IV-A), 68% and 55% of GNS and OGN evolved behaviors, were in the performance range: [0.0, 0.4], compared to 5% and 17% of ONS and NS evolved behaviors, respectively, in the same range. In the simplest task (section IV-A), *4vs3* keep-away, 67% and 54% of GNS and OGN evolved behaviors were in the performance range: [0.0, 0.4], compared to 3% and 2% of ONS and NS evolved behaviors, respectively, in the same range. This pattern of high portions of poorly performing behaviors in GNS and OGN evolved populations held for all tasks.

Thus, overall genetic-based diversity maintenance yielded no benefits either as the driving selection mechanism (GNS) or when combined with objective-based search (OGN) and applied to evolve behaviors in the keep-away tasks tested. Hence, for keep-away and its solution search space, we surmise that searching for genetically diverse genotypes does not facilitate effective behavior evolution and as in related work [13], behavioral diversity maintenance is a more effective approach.

D. Behavioral Diversity Maintenance (ONS, NS variants)

The ONS variant yielded the most benefits overall in its capability to balance *exploitation* and *exploration* of the search space as task complexity increased. That is, for all tasks, ONS evolved populations explored diverse regions of the search space (a range of behaviors of varying quality) and discovered (exploited) high quality behaviors within these regions.

First, in terms of behavior *effectiveness* (search space exploitation) the ONS variant evolved a significantly (statistically) higher or comparable average task performance for all tasks (section IV-B). Second, in terms of search space exploration, the diversity of effective behaviors, the ONS variant yielded the greatest benefits for all tasks.

The efficacy of the ONS variant, resulting from its capability to balance exploration versus exploitation, is evidenced in figure 3. ONS evolved a greater portion of effective behaviors, compared to all other variants in all tasks except *6vs5* keep-away (the most complex and deceptive task, section IV-A). For example, in *5vs4* keep-away, 22% of ONS evolved genotypes were in the task performance range: [0.6, 0.8]. Though, only 6% and 10% of OS and NS evolved genotypes (respectively), and no OGN and GNS evolved genotypes, were in this same range. However, in *6vs5* keep-away, the NS variant was most beneficial in terms of the exploration versus exploitation trade-off in the evolved population. That is, 31% of NS evolved genotypes were in the performance range: [0.6, 0.8], where as, only 3% and 14% of OS and ONS evolved genotypes (respectively), and no OGN and GNS evolved genotypes, were in the same performance range.

This result supports the hypothesis that for difficult problems without too much deception, combining objective-based and novelty search is an effective approach [2], especially in related evolutionary robotics tasks [12], [13], [4]. Though, in keep-away, as task complexity and deception increased then pure NS yields the greatest benefits (compared to other variants) in terms of balancing exploration versus exploitation, via locating diverse regions of the behavior space containing high quality behaviors (figure 3). However, this only held for the most deceptive tasks (*6vs5* keep-away), and as such more deceptive and complex versions of keep-away are being investigated. Figure 3 also supports, for all tasks, the demonstrated exploratory capacity of NS, for discovering a diverse range of (not necessarily optimal) behaviors [12].

Furthermore, figure 3 supports the benefits of coupling behavioral diversity maintenance search *with policy transfer*, as it demonstrates each variant's boosted exploration versus exploitation capabilities *given* policy transfer. That is, method variants using policy transfer yield increased exploration of effective behaviors. Consider that ONS and NS evolved genotypes (100 generations, table I) in *6vs5* keep-away, given *no policy transfer*, 4% and 1% of behaviors (respectively) were in the [0.6, 0.8] task performance range. Comparatively, 14% and 31% of ONS and NS evolved genotypes *with policy transfer* (respectively) were in the same task performance range. Less of a performance jump was observed for the OS variant, and there was no change for the GNS and OGN variants. In *6vs5* keep-away, 2% of OS evolved genotypes *without policy transfer* were in the performance range [0.6, 0.8], where as, only 3% of OS evolved genotypes *with policy transfer* were in the same range.

Figure 3 only presents comparisons for *6vs5* keep-away as this was the most difficult and deceptive task tested (section IV-A) and similar results were observed for all other tasks tested *without policy transfer*. These results further support the hypothesis that combining behavioral diversity maintenance and objective-based search facilitates an effective exploration and exploitation trade-off (section I-A), where such benefits are increased *with policy transfer*.

E. Objective-based Search (OS variant)

Experimental results also indicate the appropriateness of objective-based search as the applied search approach for behavior evolution in keep-away, for all degrees of task complexity tested (section IV-A). Consider that, in all tasks (except *5vs4* keep-away, in which OS yielded a task performance comparable to ONS and NS), the OS variant was the second highest performing variant (figure 2). Hence, we surmise that keep-away is representative of a task domain that is definable in terms of sub-solution stepping stones, where such sub-solutions are readily discoverable by objective-based search. In keep-away, as with robotic biped gait evolution [12], these solution stepping stones are known and appropriate objective functions can be designed (section II-B).

However, these results also indicate that such objective functions are susceptible to deception. This is evident from the highest average task performance of the hybrid objective-novelty search variant (ONS), except in *5vs4* keep-away, where there was no significant difference between the best three performing variants, OS, ONS and NS. It is theorized that the deceptive aspects of keep-away prevent the OS variant from adequately exploring regions of the solution space containing the most effective (highest task performance) behaviors, especially as task complexity increases.

For example, consider *4vs3* keep-away, the simplest and least deceptive of the tasks tested (section IV-A), only 6% of OS evolved genotypes were in the task performance range: [0.6, 0.8], where as, 25% and 22% of ONS and NS evolved genotypes (respectively) were in this same performance range. For *6vs5* keep-away, the most complex and deceptive of the tasks tested (section IV-A), only 3% of OS evolved genotypes were in the task performance range: [0.6, 0.8], where as, 14% and 31% of ONS and NS evolved genotypes (respectively) were in this same performance range.

This further supports the hypothesis (section I-A) that controller evolution with behavioral diversity maintenance complements objective-based search via increasing solution space exploration, thus boosting the exploitative search capabilities (solution quality). These benefits were found to increase when

the hybrid novelty-objective search variant (ONS) was coupled *with policy transfer* (section IV-D).

V. Conclusions and Future Work

This study investigated coupling policy transfer with behavioral and genetic diversity maintenance as a means of alleviating two confounding problems with the evolutionary synthesis of controllers, the *bootstrap problem* [9] and *deception* [21]. To address this, the study tested multi-agent keep-away soccer task with policy transfer between tasks of increasing complexity and deception. Results supported the efficacy of a hybrid of behavioral diversity maintenance (novelty) and objective search for appropriately balancing *exploration* versus *exploitation* in the search process, in increasingly difficult and deceptive tasks. Supporting related work, the novelty-objective hybrid yielded a significantly higher task performance compared to all other search variants. Genotypic diversity maintenance based search approaches were found to be relatively ineffective in their coupling with policy transfer as well as without policy transfer. The key contribution was to highlight benefits of behavioral diversity maintenance for controller evolution when used in company with policy transfer.

Future work aims to further support the efficacy of hybrid novelty-objective search via evaluating this study's variants with reinforcement learning for policy transfer [17] in keep-away and other tasks such as multi-agent predator-prey [24].

Acknowledgment

This research was funded by the Centre for Artificial Intelligence Research at the University of Cape Town and the Council for Scientific and Industrial Research Meraka Institute.

References

[1] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, USA: MIT Press, 2000.

[2] J. Mouret and S. Doncieux, "Encouraging behavioral diversity in evolutionary robotics: An empirical study," *Evolutionary Computation*, vol. 20(1), pp. 91–133, 2012.

[3] A. Cully and J. Mouret, "Evolving a behavioral repertoire for a walking robot," *Evolutionary Computation*, vol. 24(1), pp. 59–88, 2016.

[4] J. Gomes, P. Mariano, and A. Christensen, "Novelty-driven cooperative coevolution," *Evolutionary Computation*, vol. In Press, 2016.

[5] A. Cully, J. Clune, D. Tarapore, and J. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521(1), pp. 503–507, 2015.

[6] J. Lehman and K. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19(2), pp. 189–223, 2011.

[7] M. Crepinsek, S. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms," *ACM Computing Surveys*, vol. 45(3), pp. 1–33, 2013.

[8] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Transactions on Evolutionary Computation*, vol. 2(3), pp. 97–106, 2013.

[9] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5(1), pp. 317–342, 1997.

[10] G. Nitschke, A. Eiben, and M. Schut, "Evolving team behaviors with specialization," *Genetic Programming and Evolvable Machines*, vol. 13(4), pp. 493–536, 2012.

[11] J. Gomes and A. Christensen, "Generic behaviour similarity measures for evolutionary swarm robotics," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: ACM, 2013, pp. 199–206.

[12] J. Lehman, K. Stanley, and R. Miikkulainen, "Effective diversity maintenance in deceptive domains," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: ACM, 2013, pp. 215–222.

[13] J. Gomes, P. Mariano, and A. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proceedings of the Genetic Evolutionary Computation Conference*. Madrid, Spain: ACM, 2015, pp. 943–950.

[14] S. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22(10), pp. 1345–1359, 2010.

[15] M. Taylor, P. Stone, and Y. Liu, "Transfer learning via inter-task mappings for temporal difference learning," *Journal of Machine Learning*, vol. 8(1), pp. 2125–2167, 2010.

[16] P. Verbancsics and K. Stanley, "Evolving static representations for task transfer," *Journal of Machine Learning Research*, vol. 11(1), pp. 1737–1763, 2010.

[17] G. Boutsioukis, I. Partalas, and I. Vlahavas, "Transfer learning in multi-agent reinforcement learning domains," in *Recent Advances in Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 249–260.

[18] P. Stone, G. Kuhlmann, M. Taylor, and Y. Liu, "Keepaway soccer: From machine learning testbed to benchmark," in *Proceedings of RoboCup-2005: Robot Soccer World Cup IX*. Berlin, Germany: Springer, 2006, pp. 93–105.

[19] S. Didi and G. Nitschke, "Multi-agent behavior-based policy transfer," in *Proceedings of the European Conference on the Applications of Evolutionary Computation*. Porto, Portugal: Springer, 2016, pp. 181–197.

[20] K. Stanley, D. D'Ambrosio, and J. Gauci, "A hypercube-based indirect encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15(2), pp. 185–212, 2009.

[21] D. Whitley, "Fundamental principles of deception in genetic search," in *Proceedings of the 1st Workshop on Foundations of Genetic Algorithms*. San Mateo, USA: Morgan Kaufmann, 1991, pp. 221–241.

[22] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8(2), pp. 131–162, 2007.

[23] K. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10(2), pp. 99–127, 2002.

[24] D. D'Ambrosio and K. Stanley, "Scalable multiagent learning through indirect encoding of policy geometry," *Evolutionary Intelligence Journal*, vol. 6(1), pp. 1–26, 2013.

[25] S. Risi and K. Stanley, "Confronting the challenge of learning a flexible neural controller for a diversity of morphologies," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: ACM, 2013, pp. 255–261.

[26] J. Gauci and K. Stanley, "A case study on the critical role of geometric regularity in machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*. Chicago, USA: AAAI Press, 2008, pp. 628–633.

[27] P. Verbancsics and K. Stanley, "Constraining connectivity to encourage modularity in hyperneat," in *Proceedings of the Genetic and Evolutionary Computation Conference*. New York, USA: ACM, 2011, pp. 1483–1490.

[28] J. Gomes, P. Urbano, and A. Christensen, "Evolution of swarm robotics systems with novelty search," *Swarm Intelligence*, vol. 7(2), pp. 115–144, 2013.

[29] J. Gomes, P. Mariano, and A. Christensen, "Avoiding convergence in cooperative coevolution with novelty search," in *Proceedings of the International conference on Autonomous Agents and Multi-Agent Systems*. Paris, France: ACM, 2014, pp. 1149–1156.

[30] A. Liapis, G. Yannakakis, and J. Togelius, "Constrained novelty search: A study on game content generation," *Evolutionary Computation*, vol. 23(1), pp. 101–129, 2015.

[31] S. Risi, C. Hughes, and K. Stanley, "Evolving plastic neural networks with novelty search," *Adaptive Behavior*, vol. 18(6), pp. 470–491, 2010.