Sample AAMAS Paper using the New ACM LaTeX Template

Paper #XXX

ABSTRACT

This file is based on ACM's sample-sigconf.tex. This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1] It has the appropriate boilerplate for AAMAS 2018.

KEYWORDS

AAMAS; ACM proceedings; LaTeX; text tagging

1 INTRODUCTION

The *proceedings* are the records of a conference.[2] ACM seeks to give these conference by-products a uniform, high-quality appearance. To do this, ACM has some rigid requirements for the format of the proceedings documents: there is a specified format (balanced double columns), a specified set of fonts (Arial or Helvetica and Times Roman) in certain specified sizes, a specified live area, centered on the page, specified size of margins, specified column width and gutter size.

2 METHODS

HyperNEAT [8] is an extension of NEAT (*Neuro-Evolution of Augmented Topologies*) [9], where ANNs are indirectly encoded using a CPPN (*Compositional Pattern Producing Network*) [7]. HyperNEAT was selected as it has a number of benefits demonstrated in previous work [3], [10]. This includes its capability to exploit geometric features such as symmetry, regularity and modularity in robot morphology and the task environment during controller evolution.

In this study, HyperNEAT evolves the connection weights between each robot's sensory input layer, hidden layer and motor output layer, where each robot used the same controller, making teams homogenous. Controller evolution experiments were initialized with a given morphology (table 1). However, in one experiment set, each robot's sensor configuration of team morphology could be *co-adapted* via HyperNEAT activating and deactivating sensory input node connections over the evolutionary process. THIS CAN BE REMOVED? For these experiments (section 3), *add connection* and *remove connection* mutation operators (table **??**) from previous work [4] were applied every generation to a sensory input node chosen with uniform random selection. The mutation operator applied depended on whether the chosen input node was connected or not. DOES THIS STILL REFER TO THE WHOLE APPROACH OF RANDOMLY ACTIVATING OR DEACTIVATING The construction zone sensor (table 1) was permanently activated for all morphologies and could not be disconnected, as this enabled robots to detect *construction zones* (section 3).

Table 1 presents a list of *morphology identification* (ID) numbers and the number and type of sensors that correspond to each morphology. For example, morphology 2 has four proximity sensors, one ultrasonic sensor, one colour-ranged sensor, and one low-resolution camera. FIND ANOTHER EXAMPLE TO USE HERE USING THE NEW MORPHOLOGIES. Note that all morphologies have a construction sensor as this is necessary to complete the collective construction task.

2.1 Robot Team Controller

Each robot in the team used an ANN controller with N sensory input nodes, determined by the given morphology being evaluated (table 1). Each robot's controller mapped sensory inputs, via a fully connected hidden layer, to two motor outputs, the robot's left and right wheels (figure 1).

Figure 1 illustrates the sensory configuration for $N = 11$ (morphology 1), and the associated substrate and CPPN used by HyperNEAT. For each robot morphology (table 1), the sensors corresponding to the input layer of the controller was a circle N nodes distributed about a robot's periphery, where the exact geometric configuration corresponded to the morphology being evaluated (figure 1 illustrates morphology 1)[3]. The intermediate ANN hidden layer reflects the configuration of the input layer, preserving the geometry of the sensory input layer, that is, the direction of each sensor's FOV (figure 1). THIS IS WHERE THE FOOTNOTE WITH THE LINK TO THE ILLUSTRATIONS ON GITHUB IS. REMEMBER TO CHANGE TO THE NEW REPO The ANN was initialized with random weights normalized to the range [-1.0, 1.0], with full connectivity between adjacent layers, however, partial connectivity was evolvable via the CPPN generating a zero weight. Collectively all sensors approximated up to a 360 degree *Field of View* (FOV).

The nodes comprising each robot's ANN controller, connected by the CPPN, were placed in the substrate illustrated in figure 1. Each node in the substrate was placed at specific (x, y) locations in the two-dimensional geometric space of the substrate (x, y axes were in the range: [-1, 1]). Connection weights in the controller were evolved via querying the CPPN for the weight of any connection between two points (x_1, y_1) and (x_2, y_2) by inputting (x_1, y_1, x_2, y_2) into the CPPN, which subsequently output the associated weight. During HyperNEAT's evolutionary process, the CPPN was evolved via having nodes and connections added and removed, as well as connection weight values mutated [8].

Thus, the CPPN evolved a connectivity pattern across the geometry of the ANN via querying all the potential connections for their weights. This connectivity pattern was effectively a function of the task and ANN geometry, which enabled HyperNEAT to exploit the structure (regularity, repetition and symmetry) of the task and robot morphology. For example, there was symmetry in the robot

[1] This is an abstract footnote

[2] This is a footnote

[3] Illustrations of all robot morphologies tested can be found at: https://github.com/not-my-name/SSCI_Paper_Appendix

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden

https://doi.org/doi

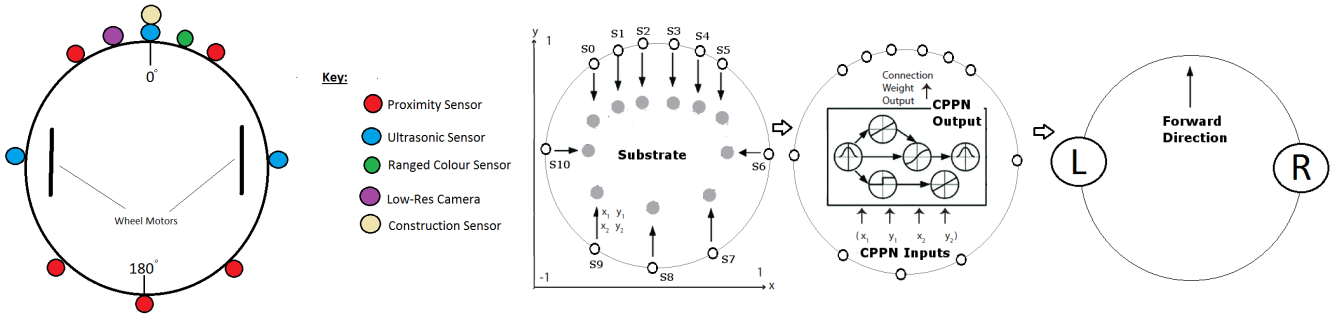

Figure 1: ***Left (color):*** **Robot morphology 1, with relative positions of various sensors on the robot.** ***Right (gray-scale):*** **ANN Topology as it relates to robot morphology 1: 11 Sensory inputs [S0, S10]. Sensory inputs connect to a hidden layer (left). Connection weight values between two nodes (x_1, y_1, x_2, y_2) are evolved by querying the CPPN (center) with x, y values in the range [-1, 1] (axis shown). The hidden layer is fully connected to all inputs and outputs (connectivity not depicted). Motor outputs (right) *L* and *R* determine the speed of the left and right wheels, respectively, and thus a robot's speed and direction.**

morphology in terms of the positioning of sensors about each robot's periphery (figure 1) and there was regularity and repetition in the collective construction task, in terms of repeating block types comprising modular and regular structures. In the collective construction task, *modularity* was defined as the composition of modular structures (buildings in construction zones) from a sequence of connected blocks and *regularity* was defined as the same sequence of blocks repeated in a building.

Previous work has demonstrated that the indirect encoding of an evolved CPPN facilitates the evolution of robot controllers with increased task performance enabled by a compact representation of task and robot geometry [2], [10]. Table **??** presents the HyperNEAT parameters used in this study, where *delta* was angle between the (x_1, y_1, x_2, y_2) positions of nodes in the substrate. These parameter values were determined experimentally. All other HyperNEAT parameters not listed in table **??**, were set as in previous work [2].

2.1.1 Detection Sensors. Each robot was equipped with various sensor types, where the exact sensor complement, including the relative position and direction on the robot depends upon the given experiment (section 3) and morphology being evaluated (table 1). Each robot had *N* sensors corresponding to the *N* inputs comprising the robot's ANN sensory input layer (figure 1), each with a range of *r* (portion of the environment's length). A robot's sensory FOV was split into *N* sensor quadrants, where all sensors were constantly active for the duration of the robot's lifetime. The *nth* sensor returned a value in the normalized range [0.0, 1.0], in the corresponding *nth* sensor quadrant. A value of 0.0 indicated that no blocks were detected and a value of 1.0 indicated that an object was detected at the closest possible distance to the given sensor.

Table **??** presents the different sensor types used in this study, where the functional properties of each sensor (range and FOV) were abstractions of corresponding physical sensors typically used on the Khepera III robots [5]. In table **??**, range values are units defined in relation to the environment size (20 x 20) and FOV values are in radians. Each morphology also included a special construction zone detection sensor that activated with a value in the range [0.0, 1.0] whenever a robot came into contact with a block that must be connected with other already connected blocks.

The construction zone sensor calculated the squared Euclidean norm, bounded by a minimum observation distance, as an inversely proportional distance between *this* robot and the closest construction zone, where a value of 1.0 indicated the robot (pushing a block) was in contact with the construction zone and a value of 0.0 indicated that the robot (pushing a block) was the maximum possible distance from the closest construction zone. Robots were unable to detect each other, thus all cooperative interactions were *stigmergic* [1] where robots interacted via pushing blocks into the environment's construction zone. Furthermore, robots had no *a priori* knowledge of the construction schema, but rather must discover the construction schema rules by trial and error.NOT USING A CONSTRUCTION SCHEMA ANYMMORE. MENTION HERE THAT IT ONLY REQUIRES COOPERATION?? Also, once at least two blocks had been pushed and connected together this formed a construction zone (section 3), that was then visible to each robot's construction zone sensor.

2.1.2 Movement Actuators. Two wheel motors control a robot's heading at constant speed. Movement is calculated in terms of real valued vectors (*dx* and *dy*). Wheel motors (*L* and *R* in figure 1) need to be explicitly activated. A robot's heading is determined by normalizing and scaling its motor output values by the maximum distance a robot can traverse in one iteration (table **??**). That is:

$dx = d_{max}(o_1 - 0.5)$

$dy = d_{max}(o_2 - 0.5)$

Where, o_1 and o_2 are the motor output values, corresponding to the left and right wheels, respectively, producing an output in the range: [-1.0, 1.0]. These output values indicate how fast each respective wheel must turn. Equal output equates to straight forward motion and unequal output results in the robot rotating about its own axis. The d_{max} value indicates the maximum distance a robot can move in one simulation iteration (normalized to 1.0, table **??**).

Table 1: Sensory configuration (number of sensors) for each morphology.

Morphology ID	Proximity Sensors	Ultrasonic Sensors	Color Ranged Sensors	Low-Resolution Camera	Construction Zone Sensors
1	5	3	1	1	1
2	3	2	1	1	1
3	1	0	1	1	1

3 EXPERIMENTS

Experiments[4] tested 15 robots in a bounded two dimensional continuous environment (20 x 20 units) with randomly distributed type A, B and C blocks (table **??**). Robots and blocks were initialized with random orientations and positions throughout the environment.

A construction schema (table ■ dictated the sequence of block types that must be connected together in order that a specific structure be built

This task required the robot team to search the environment for building-blocks and cooperatively push them together in order that they connected to form a structure, where connected blocks then formed a construction zone. Task complexity was equated with the degree of cooperation required to collectively push blocks and connect them together in the construction zone and whether or not a construction schema was required. CAN JUST TAKE THIS LAST BIT OUT? In this construction task, there were three block types, A, B and C requiring one, two and three robots to push, respectively. MENTION THAT THE RESPECTIVE BLOCK TYPES ARE ONLY PRESENT IN EACH OF THE RESPECTIVE LEVELS, RESPECTIVELY Cooperation occurred when at least two robots simultaneously pushed a type B block, or at least three robots pushed a type C block. THEREFORE THE FIRST DIFFICULTY LEVEL REQUIRES NO COOPERATION BETWEEN THE ROBOTS

Table **??** presents the three levels of difficulty for the collective construction task. Level 1 was the least complex as it did not require any cooperation, given that it in this case there were only type A blocks in the environment. Level 2 was of medium complexity as there are equal numbers of type A, B, and C blocks in the environment, where block types B and C required at least two and three robots to push, respectively. LEVEL 2 IS OF MEDIUM COMPLEXITY BECAUSE THERE ARE ONLY TYPE B BLOCKS PRESENT IN THE ENVIRONMENT, REQUIRING ONLY 2 ROBOTS TO MOVE A SINGLE BLOCK. Level 3 was the most complex, as it required the same degree of cooperation as task level 2, though blocks had to be connected according to a construction schema. LEVEL 3 IS THE MOST COMPLEX/DIFFICULT AS IT CONSISTED OF ONLY TYPE C BLOCKS IN THE ENVIRONMENT WHICH REQUIRE 3 ROBOTS TO MOVE A SINGLE BLOCK THUS EQUATING TO THE HIGHEST DEGREE OF COOPERATION REQUIRED.

Figure ■ illustrates this construction schema, where the label on each of the four sides of each block type indicates what other block type can be connected to the given side. JUST REMOVE THIS PART REFERRING TO THE CONSTRUCTION SCHEMA? MIGHT NOT EVEN HAVE THE ILLUSTRATION IN THE FINAL PAPER? ASK GEOFF ABOUT THIS, BUT CREATE THE NEW IMAGE ANYWAYS The X label indicates that no block can be connected to a given side.

The construction zone was formed via at least two blocks pushed together and was thus any structure being built in the environment. Once a construction zone was created, all blocks attached to it were fixed in position and could not be disconnected. The task mandated a maximum of three construction zones and unconnected blocks had to be pushed and connected to one of these construction zones.

For each difficulty level of the task, a different type of resource block is implemented (as shown in table **??**) in order to control the level of cooperation required in the task.

Team task performance was calculated as the number of blocks connected in construction zones during a team's lifetime (equation **??**), where average task performance was calculated as the highest task performance selected at the end of each run (100 generations) and averaged over 20 runs (table **??**).

The fitness function to direct controller evolution was a weighted sum of the number of times *type A* blocks were pushed by *one robot* and connected (a in equation **??**), the number of times *type B* blocks were pushed by *two robots* and connected (b in equation **??**), and the number of times *type C* blocks were pushed by *three robots* and connected (c in equation **??**).

$$f = r_a a + r_b b + r_c c \tag{1}$$

Parameter tuning experiments found that setting the weights (reward values r_a, r_b and r_c) in equation **??** to 0.3, 0.6, and 1.0, respectively, resulted in functional controller evolution. Fitness was normalized to the range $[0.0, 1.0]$ using the maximum possible fitness yielded from all blocks being pushed and connected in construction zones.

3.2 Experiment Design

Experiments evaluated the *morphological robustness* of HyperNEAT evolved controllers for robot teams that must accomplish collective construction tasks of increasing of increasing degrees of cooperation/difficulty complexity (section 3). We measured the average comparative task performance of controllers evolved for a given team morphology and task complexity difficulty? where such controllers were then transferred to and re-evaluated in other team morphologies. Thus, teams that achieved an average task performance that was not significantly lower across all *re-evaluated* morphologies were considered to be *morphologically robust*.

This study comprised of three experiment sets where a single set consisted of evolving a controller for a given team morphology $1-3$

[4]Source code for all experiments is online at: https://github.com/not-my-name/AAMAS_Conference

Table 2: Experiment, Neuro-evolution and Sensor Parameters

Generations	100	
Sensors per robot	11, 8, 4	
Evaluations per genotype	3	
Experiment runs	20	
Environment length, width	20	
Max Distance (Robot movement per iteration)	1.0	
Team size	15	
Team Lifetime (Simulation iterations)	1000	
Lifetimes per generation	3	
Type A blocks (1 robot to push)	15	
Type B blocks (2 robots to push)	15	
Type C blocks (3 robots to push)	15	
Mutation rate	Add neuron	0.25
	Add connection	0.008
	Remove connection	0.002
	Weight	0.1
Population size	150	
Survival rate	0.3	
Crossover proportion	0.5	
Elitism proportion	0.1	
CPPN topology	Feed-forward	
CPPN inputs	Position, delta, angle	
Sensor	Range	FOV
Proximity Sensor	1.0	0.2
Ultrasonic Sensor	4.0	1.2
Ranged Colour Sensor	3.0	1.5
Low-Res Camera	3.0	1.5
Colour Proximity Sensor	3.0	3.0

Table 3: Task Complexity.

Construction Task Complexity	Level 1	Level 2	Level 3
Type A blocks (1 robot to push)	15	0	0
Type B blocks (2 robots to push)	0	15	0
Type C blocks (3 robots to push)	0	0	15

(table 1) for three levels of increasing task complexity, requiring increasing levels of cooperation (table **??**)

Each experiment set comprised a controller evolution stage and a re-evaluation stage (morphological robustness test). For controller evolution, each experiment applied HyperNEAT to evolve team behavior for 15 robots for 100 generations, where a generation comprised three team *lifetimes* (1000 simulation iterations). Each team lifetime tested different robot starting positions, orientations, and block locations in the simulation environment. The fittest controller evolved for each task level (yielding the highest absolute task performance) was then *re-evaluated* for morphological robustness in all other morphologies. For example, the fittest controller evolved for morphology 1 was re-evaluated in morphologies 2 and 3 and the average task performance calculated across all re-evaluation runs.

Each re-evaluation run was *non-evolutionary*, where controllers were not further evolved, and each re-evaluation run was equivalent to one team lifetime. Re-evaluation runs were repeated 20 times for a given morphology, in order to account for random variations in robot and block starting positions and orientations. For each fittest controller, re-evaluated in a given morphology, an average task performance was calculated over these 20 runs, and then an overall average task performance was computed for all re-evaluated morphologies.

As per this study's objectives, these morphological re-evaluation runs tested how robust the fittest evolved controllers (for a given morphology) were to variations in that morphology. Thus, re-evaluating the fittest controllers on other morphologies emulated sensor loss due to damage or new robot morphologies introduced due to changing task constraints.

A HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e., the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters.

A.2.2 Math Equations.

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations.

A.2.4 Tables.

A.2.5 Figures.

A.2.6 Theorem-like Constructs.

A Caveat for the TEX Expert.

A.3 Conclusions

A.4 References

Generated by bibtex from your `.bib` file. Run latex, then bibtex, then latex twice (to resolve references) to create the `.bbl` file. Insert that `.bbl` file into the `.tex` source file and comment out the command `\thebibliography`.

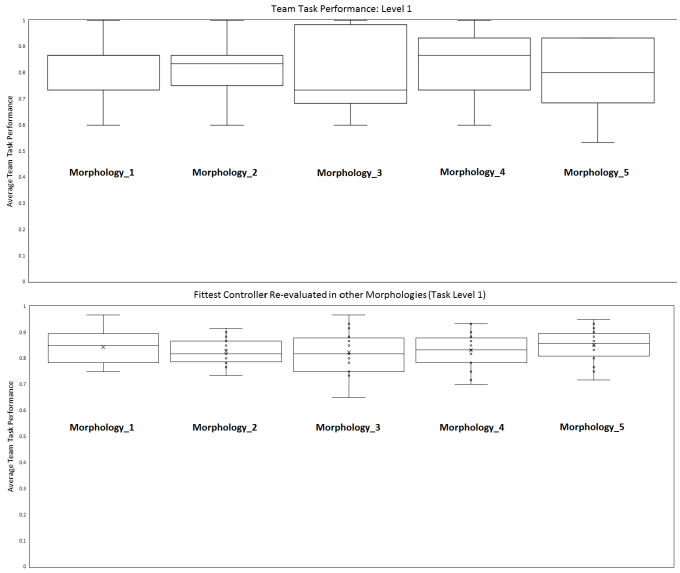

Figure 4: ***Left column:*** **Average team task performance for controller evolution (*task level 1*) given morphologies** $1-5$ **(depicted from left to right).** ***Right column:*** **Average task performance given the fittest controller evolved for each successive morphology** ($1-5$**, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology** 1**, re-evaluated in morphologies** $2-5$**. Right-most plot is the average task performance of fittest controller evolved for morphology** 5**, re-evaluated in morphologies** $1-4$**.**

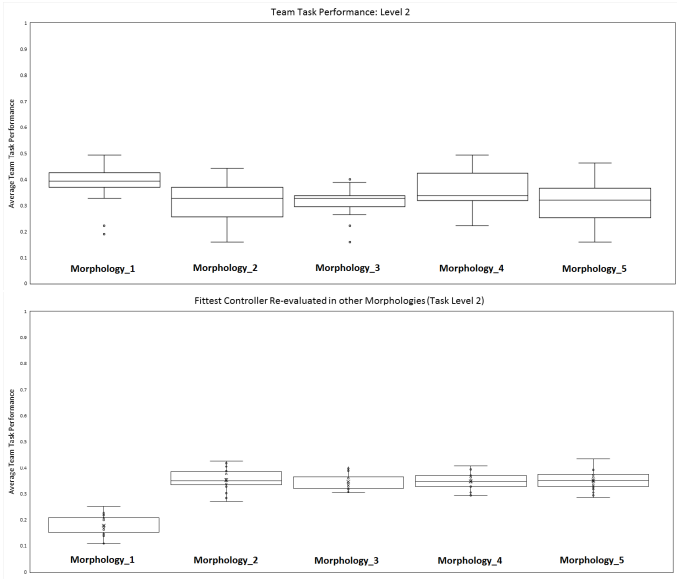

Figure 5: ***Left column:*** **Average team task performance for controller evolution (*task level 1*) given morphologies** $1-5$ **(depicted from left to right).** ***Right column:*** **Average task performance given the fittest controller evolved for each successive morphology** ($1-5$**, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology** 1**, re-evaluated in morphologies** $2-5$**. Right-most plot is the average task performance of fittest controller evolved for morphology** 5**, re-evaluated in morphologies** $1-4$**.**

B MORE HELP FOR THE HARDY

Of course, reading the source code is always useful. The file `acmart.pdf` contains both the user guide and the commented code.

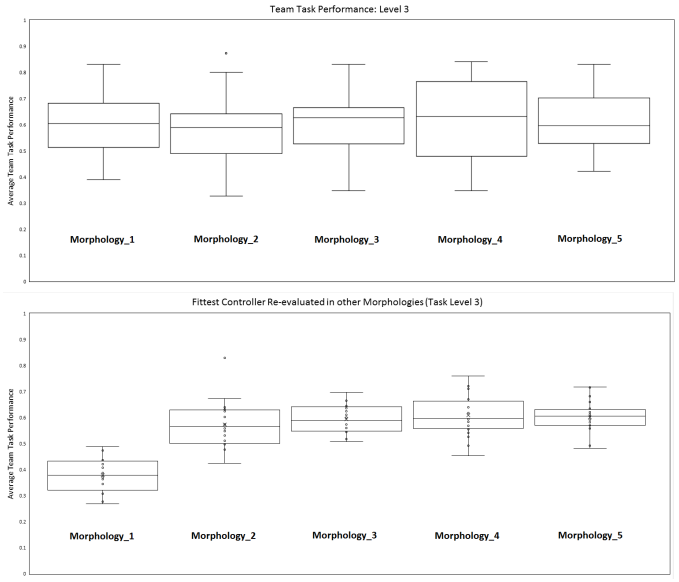

Figure 6: *Left column:* Average team task performance for controller evolution (*task level 1*) given morphologies $1-5$ (depicted from left to right). *Right column:* Average task performance given the fittest controller evolved for each successive morphology ($1-5$, shown left to right) re-evaluated in all other morphologies. For example: Left-most plot is average task performance of fittest controller evolved for morphology 1, re-evaluated in morphologies $2-5$. Right-most plot is the average task performance of fittest controller evolved for morphology 5, re-evaluated in morphologies $1-4$.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Yuhua Li for providing the matlab code of the *BEPS* method.

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. The work is supported by the National Natural Science Foundation of China under Grant No.: 61273304 and Young Scientsts' Support Program (http://www.nnsf.cn/youngscientsts).

REFERENCES

[1] R. Beckers, O. Holland, and J. Deneubourg. 1994. From local actions to global tasks: Stigmergy and collective robotics. In *Proceedings of the International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, Cambridge, USA, 181–189.

[2] D. D'Ambrosio and K. Stanley. 2008. Generative Encoding for Multiagent Learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, Atlanta, USA, 819–826.

[3] D. D'Ambrosio and K. Stanley. 2013. Scalable Multiagent Learning through Indirect Encoding of Policy Geometry. *Evolutionary Intelligence Journal* 6, 1 (2013), 1–26.

[4] J. Hewland and G. Nitschke. 2015. Evolving Robust Robot Team Morphologies for Collective Construction. In *The Benefits of Adaptive Behavior and Morphology for Cooperation in Robot Teams*. IEEE, Cape Town, South Africa, 1047–1054.

[5] F. Lambercy and J. Tharin. 2013. *Khepera III User Manual: Version 3.5*. K-Team Corporation, Lausanne, Switzerland.

[6] G. Nitschke, M. Schut, and A. Eiben. 2012. Evolving Behavioral Specialization in Robot Teams to Solve a Collective Construction Task. *Swarm and Evolutionary Computation* 2, 1 (2012), 25Ű–38.

[7] K. Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines: Special Issue on Developmental Systems* 8, 2 (2007), 131–162.

[8] K. Stanley, D'Ambrosio, and J. Gauci. 2009. Hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life* 15, 1 (2009), 185–212.

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

[10] J. Watson and G. Nitschke. 2015. Evolving Robust Robot Team Morphologies for Collective Construction. In *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, Cape Town, South Africa, 1039–1046.