Evolutionary Computation Variants for Cooperative Spatial Coordination

Georgios N. Yannakakis
Institute of Perception, Action and Behaviour
The University of Edinburgh
JCMB, EH9 3JZ
g.yannakakis@sms.ed.ac.uk

John Hallam
Mærsk Mc-Kinney Møller Institute for Production Technology
University of Southern Denmark
Campusvej 55, DK-5230, Odense M
john@mip.sdu.dk

John Levine
Department of Computer and Information Sciences
University of Strathclyde
26 Richmond Street, G1 1XH
john.levine@cis.strath.ac.uk

Abstract- This paper presents a comparative study between genetic and probabilistic search approaches of evolutionary computation. They are both applied for optimizing the behavior of multiple neural-controlled homogeneous agents whose spatial coordination tasks can only be successfully achieved through emergent cooperation. Both approaches demonstrate effective solutions of high performance; however, the genetic search approach appears to be both more robust and computationally preferred for this multi-agent case study.

1 Introduction

Cooperative behaviors within systems (environments, artificial worlds) of multiple agents is a prominent area of research. Designing agents for such systems could be a repetitive and tedious procedure. This task becomes even more difficult when the investigated multi-agent environment is fully dynamic, non-deterministic and the agents' motion is continuous. Additional complication is present when agents' communication is indirect (implicit) and partial (i.e. agents do not have complete information of the environment). Thus, when designing controllers for autonomous simulated agents for such environments, there is little guidance on how complex the controller must be for the agents to achieve good performance in particular tasks. Furthermore, when such a performance is to emerge via a learning mechanism, there is little knowledge about the mechanism's design and complexity.

To study this, we have developed a multi-agent simulated world called "*FlatLand*" to investigate the potential generation of cooperative complex behaviors amongst the agents given their type of communication and specific tasks they have to achieve. The two tasks that the agents are tested in are the antagonistic strategies of obstacle-avoidance and target-achievement. The work presented here is focused on the evolution of agents' controllers towards the emergence of the aforementioned spatial coordination in an adaptive fashion, using two forms of evolutionary learning: genetic algorithms (GAs) [1] and estimation of distribution algorithms (EDAs) [2]. In particular, for the first we use a generational mutation-based GA and for the latter we utilize a Univariate Marginal Distribution for Continuous Domains with tournament selection (UMDA$_c$). Among the few existing UMDA$_c$ applications in the literature we can distinguish its successful simple linear, and quadratic function approximations that appear in [3]. Moreover, Bengoetxea et al. [4] present a comparison between a UMDA$_c$ and a steady state GA for image recognition. In that comparative case study UMDA$_c$ appears much more efficient and faster than the GA approach. For our spatial coordination problem, the converse occurs.

FlatLand test-bed is used to assess and compare the performance, robustness and effort cost of the applied machine learning mechanisms over simulated multi-agent environments with increasing complexity. Overall, results in this presentation show that cooperative behavior amongst the agents constitutes an emerged necessity that is built on implicit and partial communication and that simple mutation-based genetic algorithms are more robust and computationally preferred than distribution estimation algorithms.

2 The *FlatLand* Simulated World

The name "*FlatLand*" is inspired by the title of E. Abbott's book [5] and its fundamental concept is based on previous research by Yannakakis [6]. Previous work on *FlatLand* is presented in [7] where the advantages of unsupervised evolutionary learning over supervised gradient-based learning mechanisms are demonstrated. The main purpose of this simulated world is to be used as a test-bed environment for investigating evolutionary [8] and gradient-based (to a lesser degree) learning techniques and furthermore, their ability to generate cooperative obstacle-avoidance and target-achievement.

FlatLand is a square two-dimensional multi-agent environment. The world's dimensions are predefined (e.g. 80 cm $\times$ 80 cm) so that actions take place in a closed frictionless plane. There are two simple figures visualized in *FlatLand* (see Figure 1): 1) white circles (radius of 5 mm) that represent the agents — artificial creatures; and 2) dashed straight lines connecting the agent's current position to its target point on the surface.

The population consists of a number of two-dimensional circular agents (20 in the original case — see [7]). One of these agents' properties is their permeability in case of a possible collision with each other. Therefore, their motion is not affected when they collide as they pass through each other. However, 'collisions' are penalized when assessing fitness.

As mentioned before, each agent is assigned a target point on the environment's surface. This point keeps changing during its life, hence as soon as an agent achieves its current target (i.e. manages to reach a circle of 5 mm around

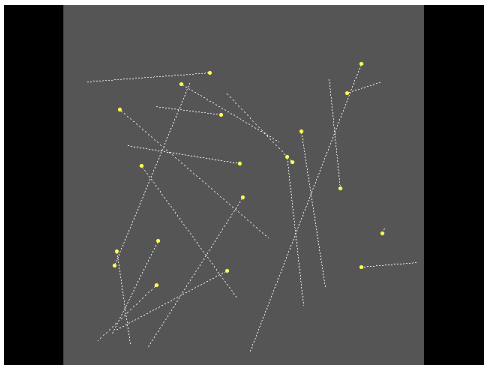

Figure 1: *FlatLand* world interface (the plane's dimensions are 80 cm $\times$ 80 cm for the experiments presented here).

the target point), then a new target point is selected. The new target point is picked from a uniform random distribution at a specified distance of 30 cm from the agent's center.

FlatLand concentrates on the creation of emergent efficient and robust obstacle-avoidance and target-achievement behavior. Consequently, the design of the simulated agents used in this environment is deliberately kept abstract.

2.1 "Humans"

Neural networks (NNs) are a suitable host for emergent adaptive behaviors in complex multi-agent environments, as stressed in [9] (among many). A feedforward neural controller is employed to manage the agents' motion. This "species" of agents are called 'Humans'.

2.1.1 Input

Using its sensors, each Human inspects the environment from its own point of view and decides about its next action. Sensors implemented are omni-directional with infinite range.

The neural controller's input data and format can be described as follows. Each Human receives information from its environment expressed in the neural network's input array of dimension D:

$$D = 2z + 1 \tag{1}$$

where z defines the number of the closest Humans that each Human perceives via its sensors. Thus, the input array consists of: (a) the polar coordinates $(\alpha_{\mathrm{i}}, \mathrm{r_i})$ — based on the axis determined by the current position of the Human and its target point (see Figure 2) — of the z $(\mathrm{i} = 1, \ldots, z)$ closest Humans and (b) an additional input that defines the distance between the Human's current position and its target point $(\mathrm{d_T})$. Figure 2 illustrates the Human's sensor information as described above.

All input values are linearly normalized into [0, 1] before they are entered into the neural controller. The input format in polar coordinates is based on Reynolds' work on artificial critters [10]. For the experiments presented in this paper $z = 2$, which was found to be the minimal amount

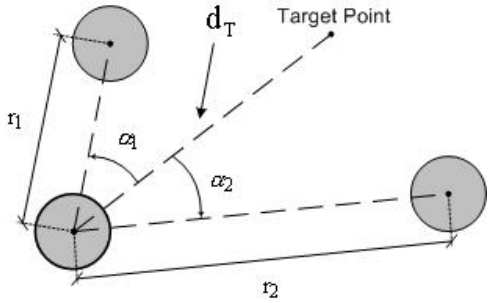

Figure 2: Human's input data in polar coordinates $(z = 2)$.

of information for a Human to successfully achieve the desired behavior (for $z = 1$ neural controllers are not able to generate satisfactory obstacle-avoidance strategies).

2.1.2 Architecture

The sigmoid function is employed at each neuron of the feedforward NN. The connection weights take values from -5 to 5 while the NN's output is a two-dimensional vector $[o_1, o_2]$ with respective values from 0 to 1. This vector represents the Human's step motion and is converted into polar coordinates according to (2) and (3).

$$\mathrm{r_{NN}} = o_1 M \tag{2}$$

$$\alpha_{\mathrm{NN}} = (2o_2 - 1)\pi \tag{3}$$

where $\mathrm{r_{NN}}$ is the Human's step motion (in cm/simulation step); α_{NN} is the Human's turn angle from the axis determined by the Human's current position and its target point (in degrees); M is the Human's maximum speed — in experiments presented in this paper, $M = 1$ cm/(simulation step).

2.2 "Animals"

Using the same environment, we explored an additional "species" of agents. These agents are called "Animals" and their only difference from Humans is in the control of their locomotion. Instead of a neural network, an Artificial Potential Field (APF), specially designed for this environment, controls the Animals' motion. The essence of the APF is that points along the Animal's path to its target point are considered to be attractive while obstacles (other Animals) in the environment are repulsive [11] (see also [12] for an application of APF in multiple robots). The overall APF causes a net force to act on the Animal, which guides it along a collision-free, target-achievement path. This force is calculated by each Animal at every simulation step and represents the function:

$$F(x,y) = \frac{M}{2}\sqrt{(x - x_{\mathrm{T}})^2 + (y - y_{\mathrm{T}})^2} + \delta \sum_{i=1}^{z} e^{-\left[\left(\frac{\Delta x_i}{4R}\right)^2 + \left(\frac{\Delta y_i}{4R}\right)^2\right]} \tag{4}$$

where

$$\Delta x_i = x - x_i$$

$$\Delta y_i = y - y_i$$

$F(x, y)$ is the potential field value for the Animal's cartesian coordinates x, y; $[x_{\mathrm{T}}, y_{\mathrm{T}}]$ are the coordinates of Animal's target point; $[x_i, y_i]$ are the coordinates of the Animal's i closest obstacle's (other Animal's) center; δ is a parameter that defines the height of the exponential "mountain-like" function presented in (4).

It is obvious that the APF of each Animal alters at every simulation step as a result of *FlatLand*'s dynamics (moving obstacles — other Animals — and changing neighbors). The Animals' motion, thence, consists of a fixed non-linear strategy that does not evolve and is determined by the two-dimensional discontinuously time-varying potential field represented by (4).

Any motion strategy that guides an agent to quickly achieve its target, avoiding any possible collisions and keeping the straightest and fastest possible trajectory to its target, is definitely a "good" strategy in terms of *FlatLand* world. Hence, Animals present a "good" (near optimum) behavior in our simulated world and furthermore a reference case to compare to any Humans' behavior.

2.3 Target Achievers

The Target Achievers (TAs) are agents that move directly towards their target points with constant speed; $\alpha_{\mathrm{NN}} = 0^{\mathrm{o}}, \mathrm{r}_{\mathrm{NN}} = 0.5$ cm/simulation step.

3 Challenges

In this section we provide evidence of the problem's complexity and learning difficulty as well as its importance in the multi-agent systems research area. In fact, *FlatLand* is a hard environment for an agent to learn to perform in because of its following distinct features:

- **Fully dynamical multi-agent**. Agents move continuously. Each agent faces a number of moving obstacles in a specific squared environment and it has no *a priori* knowledge about their motion.
- **Partial information** The fact that each agent in the *FlatLand* environment is able to capture the position of only z — in all experiments presented here $z = 2$ — other agents adds the difficulty of partial information of the environment.
- **Implicit information**. An additional difficulty is that agents communicate just by "seeing" each other (see Figure 2). This kind of communication regarding the specific tasks (i.e. obstacle-avoidance and target-achievement) is very common in the animal world (e.g. predator-prey behaviors) as well as in human beings (e.g. crowded streets).
- **Discontinuous time-varying information** The agent's input information suffers from discontinuity because of frequent alterations of the z closest neighbors that it takes into account via its sensors. Hence, the values of the polar coordinates $\alpha_{\mathrm{i}}, \mathrm{r}_{\mathrm{i}}$ $(\mathrm{i} = 1, \ldots, z)$ alter in a discontinuous fashion.

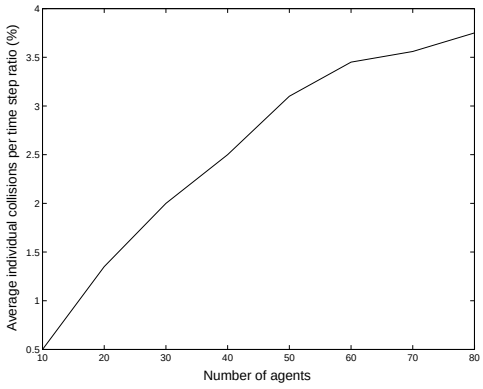

Figure 3: Worst collision-avoidance behaviors: average individual collisions per simulation step ratio over the number of simulated agents.

- **Very few collision examples**. One of the difficulties of the *FlatLand* world is the small number of collisions per simulation step in relation to the environments' complexity. In the worst obstacle-avoidance behaviors we observed, in the environment of 80 agents, Humans (in total) collide approximately about $3 \cdot 10^4$ times in 10^4 simulation steps. Thus, each Human collides less than 3.75% of its lifetime on average. For the simplest environment used (i.e. 10-agent) this percentage is approximately 0.5%. Therefore, it is both hard and computationally expensive for an obstacle-avoidance strategy to emerge from rewarding good examples of this strategy. Furthermore, when increasing the population of simulated agents, the average individual collisions per simulation step ratio increases logarithmically (see Figure 3).

FlatLand's basic concept and features make the proposed test-bed interesting for the multi-agent artificial life research area. The generality of this world extends into the area of computer games as successful applications have already shown [13], [14].

- **Emerging cooperation**. *FlatLand* is a simulated world in which we expect cooperative behaviors to emerge without any information exchange apart from spatial coordination (see above). Hence, emergent cooperation derives from 1) the way Humans move and 2) the way they interact with their environment (see Section 5).
- **Strong creature-environment interaction**. There is a strong interaction and relation between the simulated creatures and their environment. In other words, any creature in *FlatLand* faces an environment of a two-dimensional space that includes a number of other creatures. Creatures in *FlatLand* are part of their own environment. Furthermore, *FlatLand*'s main feature, as an environment, is its own creatures. This feature defines an important point in the research

of two-dimensional multi-agent dynamic simulated worlds. Computer games and artificial life offer a great arena of such worlds and a plethora of applications — see [15, 16, 13] among many.

4 Learning Mechanisms

In this section we present two different evolutionary computation (EC) learning mechanisms used for our experiments. Their common feature is the emergence of the desired behavior by rewarding agents when achieving overall good performance on the competing criteria of obstacle-avoidance and target-achievement.

4.1 Generational Genetic Algorithm

As previously stressed, our aim is to produce emergent complex cooperative behaviors by evaluating Humans from their own actions in *FlatLand*. A simple generational genetic algorithm (GGA) [1] is implemented, which uses an "endogenous" evaluation function that derives from the Humans' actions in the environment and promotes good collision-avoidance and target-achievement behaviors. Humans that learn to behave in this fashion are fit enough to be considered as good solutions of the problem.

The evolutionary procedure used can be described as follows. Each agent has a genome that encodes the connection weights of its neural network. A population of $\mathrm{N_p}$ (for the experiments presented here $\mathrm{N_p} = 20$) neural networks is initialized randomly. Initial real values that lie within [-5, 5] for their connection weights are picked randomly from a uniform distribution. Then, at each generation:

Step 1 Every Human in the population is cloned N times (N being the number of agents in *FlatLand*). These N clones are placed in the *FlatLand* environment and tested for an evaluation period e_p (e.g. 300 simulation steps). The outcome of this test is to ascertain the total number of collisions C and target achievements T (see Figure 4).

Step 2 Each Human is evaluated via the following function:

$$f_i = \frac{max\left\{1 - \frac{C_i}{C_u}, 0\right\} + min\left\{\frac{T_i}{T_u}, 1\right\}}{2} \quad (5)$$

where f_i is the evaluation function of Human i; C_i is the total number of collisions of Human i's N clones; C_u is the total number of collisions' upper bound which is determined by the total number of collisions of N TAs in e_p simulation steps (see Table 2); T_i is the total number of target achievements of Human i's N clones; T_u is the total number of target achievements' upper bound which is determined by the total number of target achievements of N Animals in e_p simulation steps (see Table 2).

Step 3 A pure elitism selection method is used where only a small percentage $\mathrm{N_s}$ ($\mathrm{N_s} = 10\%$ in this paper) of the fittest solutions is able to breed and, therefore, determine the members of the intermediate population.

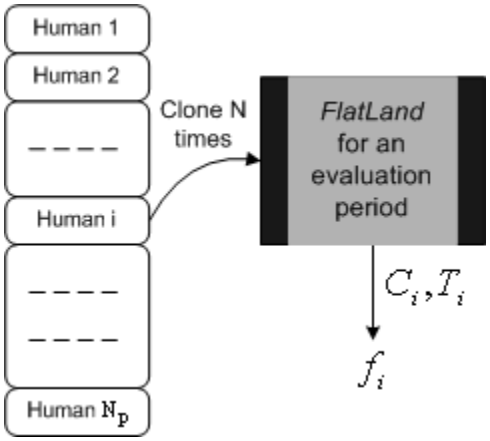

Figure 4: GGA: clonal evaluation of Humans.

Step 4 Each of the parents clones an equal number of offspring so that the total population reaches $\mathrm{N_p}$ members. Alternatively, uniform [17] and Montana and Davis [18] crossover operators have been used at this step but proved unsuccessful. The explanation is the disruptive feature of crossover operators when dealing with distributed knowledge representation (i.e. neural network). That is, crossover among parts of different successful neural networks is very likely to lead into unsuccessful offspring [19]. Results obtained from experiments with crossover operators are not presented in this paper.

Step 5 Mutation occurs in each gene (connection weight) of each offspring's genome with a small probability p_m ($p_m = 0.01$ in this paper). A uniform random distribution is used again to define the mutated value of the connection weight.

The algorithm is terminated when either a best fit Human (i.e. $f_i = 1.0$) is found or a large number of generations t_{max} is completed.

As mentioned before, a suitable evaluation function for the GGA approach promotes good obstacle-avoidance and target-achievement behaviors in an "endogenous" way. Furthermore, by using (5), we reward Humans (their N clones) that do not crash and achieve a determined number of targets (T_u) during an evaluation period. By this evaluation, we mainly promote clones capable of cooperating in order to successfully achieve the aforementioned desired behavior. Due to this, very interesting cooperative behaviors emerge within a homogeneous environment (see Section 5).

We mainly use small simulation periods e_p in order to evaluate Humans via (5) due to the implied computational effort. Thus, this evaluation function constitutes an approximation of the overall performance of the examined Humans in large simulation periods (see Section 5.1). The higher the number of e_p simulation steps, the better the Humans' performance estimation. Keeping an appropriate balance between computational effort and performance approximation

is one of the key features of the GGA approach.

4.2 Univariate Marginal Distribution for Continuous Domains

Estimation of Distribution Algorithms (EDAs) are a new and prominent area of evolutionary computation [2]. EDAs, as optimization tools, are based on the use of density estimators and search over probability distributions. An appropriate EDA for the *FlatLand* world is a modified Univariate Marginal Distribution [20] for Continuous Domains (UMDA$_c$) [3]. This algorithm is used as an alternative evolutionary learning mechanism to the GGA approach presented in Section 4.1.

The algorithm works as follows. At each generation t, an $n-$dimensional random variable $\mathbf{W}^t = (W_1^t, \ldots, W_n^t)$, that represents the connection weights of the neural controller, is maintained. We assume that the joint probability distribution of $\mathbf{W}^t$ follows an $n-$dimensional normal distribution which is factorized as a product of n unidimensional and independent normal densities. Thus, each component of $\mathbf{W}^t$ is unidimensional, normal distributed, that is $W_i^t \rightsquigarrow N(\mu_i^t, \sigma_i^t)$, where $f_{N(\mu_i^t,\sigma_i^t)}(w_i) = \frac{1}{\sqrt{2\pi\sigma_i^t}} e^{-(w_i-\mu_i^t)^2/2(\sigma_i^t)^2}$ with $i = 1, \ldots, n$ is the probability density function (PDF) of a normal distribution with mean μ_i^t and standard deviation σ_i^t in point w_i.

We obtain a number of individuals N_T (for the experiments presented here $N_T = 8$) that defines the tournament size by drawing instances of the aforementioned $n-$dimensional random variable (i.e. connection weights). By using the GGA evaluation process (see Section 4.1), the fitness of these individuals is estimated and the best one is selected. By repeating this process $\mathrm{N_U}$ ($\mathrm{N_U}$ is 20 in this paper) times we obtain a population of best fit selected individuals. This population is used to estimate the means and standard deviations of the random variable $\mathbf{W}^{t+1}$. These parameters are estimated by using their corresponding maximum likelihood estimators. Table 1 presents the pseudocode for this algorithm.

The UMDA$_c$ algorithm is terminated as soon as either a best fit set of connection weights $\mathbf{W}^t$ is found ($f(\mathbf{W}^t) = 1.0$) or a large number of generations t_{max} is completed (e.g. $t_{max} = 2000$).

while no convergence ($f < 1.0$ or $t < t_{max}$) **do**
  begin
    for ($j = 1; j \leq \mathrm{N_U}; j++$)
      begin
        Draw $\mathbf{W}^t$ to obtain N_T individuals:
        $\mathbf{w}_{1,j}^t = (w_{1,j}^{1,t}, \ldots, w_{1,j}^{n,t})$
        $\vdots$
        $\mathbf{w}_{N_T,j}^t = (w_{N_T,j}^{1,t}, \ldots, w_{N_T,j}^{n,t})$
        Evaluate $\mathbf{w}_{1,j}^t, \ldots, \mathbf{w}_{N_T,j}^t$ by using (5)
        Select the best one:
        $\mathbf{w}_{(1:\ldots:N_T),j}^t = max\left\{f(\mathbf{w}_{1,j}^t), \ldots, f(\mathbf{w}_{N_T,j}^t)\right\}$
      end
    for ($j = 1; j \leq n; j++$)
      begin
        Estimate the parameters of the new PDFs
        $$\mu_i^{t+1} = \frac{\sum_{j=1}^{\mathrm{N_U}} w_{(1:\ldots:N_T),j}^{i,t}}{\mathrm{N_U}}$$
        $$\sigma_i^{t+1} = \sqrt{\frac{\sum_{j=1}^{\mathrm{N_U}} (w_{(1:\ldots:N_T),j}^{i,t} - \mu_i^{t+1})^2}{\mathrm{N_U}}}$$
      end
  end

Table 1: Pseudocode for UMDA$_c$ with tournament selection.

5 Results

In this section we conduct a comparative study between the two learning mechanisms applied in *FlatLand* as presented in Section 4. Hence, in Section 5.1 we introduce a way of evaluating the performance of a group of agents in *FlatLand* and in Section 5.2, Section 5.3 and Section 5.4 we respectively compare the performance, robustness and effort cost of the mechanisms in the 10, 20 and 40-agent *FlatLand* environments.

5.1 Performance Measurement

We present a methodology for measuring the performance of a team of agents whether these are emergent or hand-programmed. For the former, we pick up the best (in terms of the optimization function used) neural controller (Human). We, then, record the total numbers of both collisions C and target achievements T of a population of N copies of this agent in 10^4 simulation steps by placing these agents in *FlatLand* and running the simulation. In order to diminish the non-deterministic effect of the initialization phase (random choice of target points), we repeat the same procedure for ten simulation (i.e. evaluation) runs — we believe that this number of evaluation runs is adequate to illustrate a clear picture of the behavior — of different initial conditions and we compute the numbers of total collisions C_i and target achievements T_i for each run i. In addition, the agents' mean speed $E\{V\}$, and the agents' mean absolute turn angle $E\{a\}$ in degrees are calculated. Subsequently, the performance P_i of a team of agents in a single trial i is obtained as follows.

$$P_i = \frac{max\left\{1 - \frac{C_i}{C_{TA}}, 0\right\} + min\left\{\frac{T_i}{T_A}, 1\right\}}{2} \qquad (6)$$

where C_{TA} is the total number of collisions of N TAs in 10^4 simulation steps (see Table 2); T_A is the total number of target achievements of N Animals in 10^4 simulation steps (see Table 2). The average performance over the ten trials is denoted by P.

The maximum value of (6) is 1.0 and it is obtained only when the agents do not collide at all and achieve as many target points as the Animals do (T_A) or more. Additionally, the upper bound for the total number of collisions is the number that TAs produce (C_{TA}) because they just move directly towards their target points and therefore, present

N	P		GGA, UMDA$_c$			
	C_{TA}	T_A	e_p	C_u	T_u	t_{max}
10	440	1650	1000	44	165	
20	2000	3200	300	60	96	2000
40	8000	6350	200	160	127	

Table 2: P, GGA and UMDA$_c$ experiment parameter values for the 10, 20 and 40-agent environment.

Agents	$E\{C\}$	$E\{T\}$	$E\{V\}$	$E\{a\}$	P
Random	198620	7	0.46	174^{o}	**0.0010**
TAs	2000	3348	0.5	0^{o}	**0.5000**
Animals	0	3200	0.5	3.2^{o}	**1.0000**
GGA	261	3376	0.9	44.5^{o}	**0.9347**
UMDA$_c$	335	3350	0.9	42.3^{o}	**0.9162**

Table 3: Best performance comparison table — average values are obtained from 10 evaluation runs (10^4 simulation steps each) of a 20-agent environment.

the worst collision-avoidance behavior from our viewpoint. Hence, (6) produces a clear picture of how far the performance of each learning mechanism is from the optimal performance of Animals ($P = 1.0$).

5.2 Emergent Performance

Table 3 illustrates the best obtained performances from the learning mechanisms applied into the 20-agent *FlatLand* environment (see Table 2 for the experiment parameter values). The neural controller employed is a 5-hidden neuron feedforward neural network. This controller experimentally generates the best performance, among all 1-hidden layer feedforward neural controllers with up to 15 hidden neurons, for both learning mechanisms applied.

In Table 3 we introduce the best obtained performance of a species of agents called "Random" ($P = 0.0010$). These agents are randomly initialized Humans and the variance of their performance over the 10 evaluation runs σ^2 equals to $12.03 \cdot 10^{-7}$. The Random agents along with the Target Achievers and the Animals are presented in Table 3 for comparison to any emergent Humans' behavior.

As seen from Table 3, the GGA approach ($P = 0.9347$, $\sigma^2 = 12.5 \cdot 10^{-5}$) gets closer to the desired behavior (i.e. Animals) than UMDA$_c$ or any other "species" of agents. However, both approaches achieve very high performances ($P > 0.9$) by generating Humans that keep a big distance from each other in order to avoid collisions. Furthermore, they move with an almost maximum speed (i.e. $E\{V\} = 0.9$) to achieve as many target points as possible. This behavior contradicts the Animal strategy where (given effective obstacle-avoidance) agents follow the shortest path between their current position and their target point. Small turn angles (i.e. $E\{a\} < 4^{o}$) and speeds that approximate 0.5 cm/(simulation step) portray the Animal strategy.

These results provide evidence that simple evolutionary learning mechanisms can generate cooperative coordination build on the partial, implicit and passive communication amongst Humans.

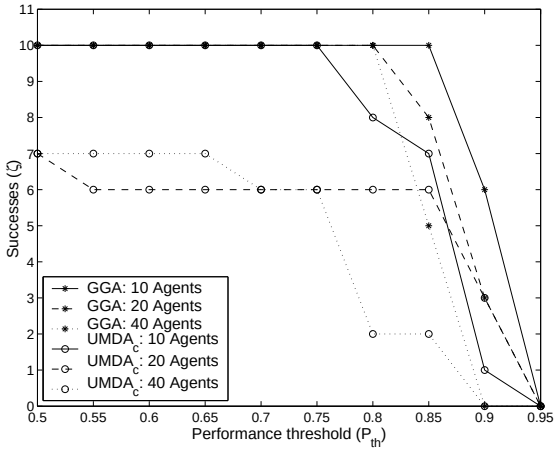

Figure 5: Increasing *FlatLand* Complexity. Number of successes out of 10 runs for specific performance values for both mechanisms.

5.3 Robustness Comparison

We are interested in obtaining a successful and robust learning mechanism with minimum efforts in our experiments. We can obviously experiment with parameter value adjustment of each method and therefore, be able to find more effective neural controllers (Humans) for the desired behavior. However, if a successful controller is determined with the lowest computing cost, the applied methodology can be recommended.

To ascertain the effort that each learning mechanism has tried to obtain a desirable robust neural controller, we assume that a single independent experiment is repeatedly run until a successful neural controller is found. A better mechanism will have a smaller number of runs to find a successful neural controller [21].

In addition to the computational effort analysis, we will experiment with teams of fewer (10) and more (40) than 20 agents in order to demonstrate the effectiveness of each approach over *FlatLand* environments of increasing complexity (see Section 3). The following procedure is applied for meeting the above-mentioned objectives. For each approach and each *FlatLand* environment (i.e. 10, 20 and 40-agent) a) we repeat the learning attempt ten times; b) we measure the performance of each run; c) we calculate the number of runs that present higher performance than specific performance values (i.e. thresholds). This number determines the successes of the approach for the respective performance threshold P_{th}. The higher the performance threshold value, the more demanding the procedure. Figure 5 illustrates the number of successes of both learning mechanisms applied for ten values of P_{th} for each *FlatLand* environment. The approaches' parameter values appear in Table 2.

The generational GA is more efficient and robust than UMDA$_c$ for all *FlatLand* environments tested. Both mech-

P_{th}	Approach	ζ	η	ECI[1]	MEC[1]
0.7–0.75	GGA	10	0	$[458.3, 639.4]$	503.1
	UMDA$_c$	10	0	$[449.4, 627.0]$	493.2
0.8	GGA	10	0	$[458.3, 639.4]$	503.1
	UMDA$_c$	8	2	$[477.1, 929.9]$	616.5
0.85	GGA	10	0	$[458.3, 639.4]$	503.1
	UMDA$_c$	7	3	$[503.6, 1149.1]$	704.6
0.9	GGA	6	4	$[596.8, 1955.8]$	1006.1
	UMDA$_c$	1	9	$[1086.2, 19666.6]$	4932.4

Table 4: 10-agent environment: effort cost comparison table ($\varepsilon = 0.05$) $Q_{\text{GGA}} = 457.28sec$, $Q_{\text{UMDA}_c} = 448.40sec$.

P_{th}	Approach	ζ	η	ECI[1]	MEC[1]
0.7–0.8	GGA	10	0	$[797.9, 1113.3]$	875.7
	UMDA$_c$	6	4	$[1353.9, 3660.9]$	2066.5
0.85	GGA	8	2	$[847.1, 1651.1]$	1094.6
	UMDA$_c$	6	4	$[1353.9, 3660.9]$	2066.5
0.9	GGA	3	7	$[1305.7, 7283.8]$	2919.1
	UMDA$_c$	3	7	$[1848.7, 10312.9]$	4133.1

Table 5: 20-agent environment: effort cost comparison table ($\varepsilon = 0.05$) $Q_{\text{GGA}} = 796.12sec$, $Q_{\text{UMDA}_c} = 1127.02sec$.

P_{th}	Approach	ζ	η	ECI[1]	MEC[1]
0.7–0.75	GGA	10	0	$[765.6, 1068.1]$	840.2
	UMDA$_c$	6	4	$[1268.8, 3430.7]$	1936.6
0.8	GGA	10	0	$[765.6, 1068.1]$	840.2
	UMDA$_c$	2	8	$[2040.1, 17546.8]$	5809.7
0.85	GGA	5	5	$[996.9, 3267.1]$	1680.4
	UMDA$_c$	2	8	$[2040.1, 17546.8]$	5809.7

Table 6: 40-agent environment: effort cost comparison table ($\varepsilon = 0.05$) $Q_{\text{GGA}} = 763.83sec$, $Q_{\text{UMDA}_c} = 1056.32sec$.

anisms, however, generate controllers with $P \geq 0.9$ when applied in the 10 and 20-agent environment. For $P_{th} < 0.7$, GGA is 100% successful (i.e. 10 out of 10 times) for all environments tested. This mechanism also produces 6 out of 10 successes for high performances ($P_{th} = 0.9$) in the 10-agents case study.

UMDA$_c$ is a population based evolutionary algorithm that emerges as a generalization of the GGA approach for the purpose of overcoming drawbacks such as efficient GA parameter and genetic operator selection. Instead of a genetic search by mutation, UMDA$_c$ searches through the solution's estimation of probability distribution. Despite its promise, it does not manage to get that robust solutions (see Figure 5). Hence, it appears that the most appropriate evolutionary process for the *FlatLand* case study is based on pure genetic search.

5.4 Effort Cost Comparison

Since the GGA approach is demonstrated to be more robust than the UMDA$_c$ approach, the next step is to compare these mechanisms via their effort cost interval and mean effort cost. Hence, we pick decent high values of P_{th} (i.e. $P_{th} \geq 0.7$) and proceed with a beta-distribution approximation of the effort cost interval and the mean effort cost [21] for both approaches. Learning mechanisms that experience zero successes out of ten runs are not considered worthy of further analysis.

Results from the effort cost comparison via the beta-distribution statistical method for the 10, 20 and 4-agent case studies are presented in Table 4, Table 5 and Table 6 respectively. More comprehensively, for each P_{th} value the number of successes (ζ) and failures (η) of each approach is presented; by use of (7) and (8) the lower and upper bound probability χ_l, χ_u for each method is found; then the 95% confidence interval $[1/\chi_u, 1/\chi_l]$ is calculated. This interval represents the 95% confidence bounds on the expected number of runs required to achieve the first successful outcome. The next column on the above-mentioned tables displays the effort cost interval (ECI) $[Q_A/\chi_u, Q_A/\chi_l]$ for each approach, where Q_A corresponds to the unit computing cost per run of the approach A. For the experiments presented here Q_A equals to the average CPU time of the ten runs (every experiment presented here ran in the same 1GHz processor). Finally, the mean effort cost (MEC) is calculated with $\frac{\zeta+\eta+1}{\zeta} Q_A$.

The conclusion that arises from Table 4, Table 5 and Table 6 is that the GGA approach is computationally preferred from the UMDA$_c$ approach for high performance values in any test-bed applied. The only case where the GGA consumes more computational effort than the UMDA$_c$ is when $0.7 \leq P_{th} \leq 0.75$ in the 10-agent environment. This implies that the more complex the problem and more demanding the solutions get, the more appropriate the GGA method seems to be for the *FlatLand* test-bed. On that basis, it appears that if there is need for a fast, relevantly low performance solution in low complexity case studies (10-agent), UMDA$_c$ is preferred.

6 Conclusions

We introduced both a hard and interesting case study for the multi-agent dynamic simulated world research area. *FlatLand* shares common features of known artificial life and game worlds used for studying the emergence of cooperative global behaviors which are based on local interactions. In addition, agents are explicitly given individual spatial coordination tasks and their communication is limited to 'seeing' neighbor agents.

We saw that evolutionary computation techniques can generate robust and cooperative behaviors of high-performance as far as the complication of the *FlatLand* world is concerned. However, genetic search approaches (GGA) proved to be more robust and less computation-

[1] in seconds

ally expensive than EC probabilistic methods (UMDA$_c$) for nearly all case studies used. In particular, when high-performing solutions are demanded the GGA approach appears to be more appropriate as an optimization tool. On the other hand, UMDA$_c$ may be utilized for fast-obtained but relevantly low-performing behaviors. Sufficient epistasis may be the main reason for the failure of the univariate approach versus the genetic algorithm since UMDA$_c$ does not investigate the dependencies between the problem's variables. However, further epistasis testing experiments are required to confirm such a hypothesis.

Appendix

If a probability p is beta distributed the confidence limits χ_l, χ_u can be obtained by solving the equations:

$$\frac{\varepsilon}{2} = \int_0^{\chi_l} \frac{p^\zeta (1-p)^\eta}{B(\zeta+1, \eta+1)} dp \quad (7)$$

$$\frac{\varepsilon}{2} = \int_{\chi_u}^1 \frac{p^\zeta (1-p)^\eta}{B(\zeta+1, \eta+1)} dp \quad (8)$$

where $B(\zeta+1, \eta+1) = \int_0^1 p^\zeta (1-p)^\eta dp = \frac{\zeta!\eta!}{(\zeta+\eta+1)!}$.

Bibliography

[1] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[2] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A new tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[3] C. González, J. A. Lozano, and P. Larrañaga. Mathematical modelling of UMDA$_c$ algorithm with tournament selection. Behaviour on linear and quadratic functions. *Int. Journal of Approximate Reasoning*, 31(3):313–340, 2002.

[4] E. Bengoetxea, P. Larranaga, I. Bloch, and A. Perchant. Image recognition with graph matching using estimation of distribution algorithms. In *Proceedings of Medical Image Understanding and Analysis*, pages 89–92, 2001.

[5] E. A. Abbott. *Flatland*. Signet Classic, June 1984.

[6] G. N. Yannakakis. Evolutionary computation: Research on emerging strategies through a complex multi-agent environment. Master's thesis, Technical University of Crete, Chania, Greece, September 2001.

[7] G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou. Performance, robustness and effort cost comparison of machine learning mechanisms in *FlatLand*. In *Proceedings of the 11th Mediterranean Conference on Control and Automation MED'03*. IEEE, June 2003.

[8] A. D. Blair and E. Sklar. Exploring evolutionary learning in a simulated hockey environment. In *Congress on Evolutionary Computation*, pages 197–203. IEEE Service Center, 1999.

[9] D. H. Ackley and M. L. Littman. Interactions between learning and evolution. In *Artificial Life II*, pages 478–507, Reading, MA, 1992. Addison-Wesley.

[10] C. W. Reynolds. Evolution of corridor following behavior in a noisy world. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 402–410, 1994. MIT Press.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. for Robotics Research*, 5(1):90–99, 1986.

[12] R. C. Arkin. Cooperation without communication: Multiagent schema based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, April 1992.

[13] G. N. Yannakakis and J. Hallam. Evolving Opponents for Interesting Interactive Computer Games. In *From Animals to Animats 8: Proceedings of the 8^{th} International Conference on Simulation of Adaptive Behavior*, pages 499–508, 2004. The MIT Press.

[14] G. N. Yannakakis, J. Levine, and J. Hallam. An Evolutionary Approach for Interactive Computer Games. In *Proceedings of the Congress on Evolutionary Computation*, pages 986–993, June 2004.

[15] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[16] Icosystem. The Game, 2002. http://www.icosystem.com/game.htm.

[17] G. Syswerda. Uniform crossover in genetic algorithms. In Schaffer, editor, *Proc. of the Third International Conference on Genetic Algorithms*, pages 2–9, 1989.

[18] D. J. Montana and L. D. Davis. Training feedforward neural networks using genetic algorithms. In *Proc. of the Int. Joint Conference on Artificial Intelligence*, 1989.

[19] X. Yao. Evolving artificial neural networks. In *Proc. of the IEEE*, volume 87, pages 1423–1447, 1999.

[20] H. Muehlenbein and G. Paass. *From recombination of genes to the estimation of distributions*, volume 4, pages 178–187. Springer, 1996.

[21] DaeEun Kim. *A Quantitative Approach to the Analysis of Memory Requirements for Autonomous Agent Behaviours using Evolutionary Computation*. PhD thesis, University of Edinburgh, 2002.