odNEAT: An Algorithm for Distributed Online, Onboard Evolution of Robot Behaviours

Fernando Silva[1], Paulo Urbano[1], Sancho Oliveira[2] and Anders Lyhne Christensen[2]

[1]LabMAg, Faculty of Sciences, University of Lisbon (FC-UL)
[2]Institute of Telecommunications, University Institute of Lisbon (ISCTE-IUL)
Lisbon, Portugal
fsilva@di.fc.ul.pt

Abstract

We propose and evaluate a novel approach called Online Distributed NeuroEvolution of Augmenting Topologies (odNEAT). odNEAT is a completely distributed evolutionary algorithm for online learning in groups of embodied agents such as robots. While previous approaches to online distributed evolution of neural controllers have been limited to the optimisation of weights, odNEAT evolves both weights and network topology. We demonstrate odNEAT through a series of simulation-based experiments in which a group of e-puck-like robots must perform an aggregation task. Our results show that robots are capable of evolving effective aggregation strategies and that sustainable behaviours evolve quickly. We show that odNEAT approximates the performance of rtNEAT, a similar but centralised method. We also analyse the contribution of each algorithmic component on the performance through a series of ablation studies.

Introduction

The traditional evolutionary computation algorithm works in a discrete and centralised manner. An external component creates an initial population and is responsible for selecting, mutating and replacing individuals. Evolution is usually performed offline even if the subject of optimisation or design is an embodied agent such as a robot. Traditional evolutionary approaches have a number of shortcomings when evolving robotic controllers. Since evolution is typically conducted offline, controllers need to be transferred to robots post-evolution. Once deployed, the controllers are thus specialised to a particular task and environmental conditions. They are fixed solutions and exhibit limited capacity to adapt to environments and to tasks not seen during evolution.

In order for an evolutionary algorithm (EA) to give robots the capacity to continuously adapt, it has to be run on the robots themselves and execute as they perform their tasks, i.e., evolution must be conducted *online*. The first attempt at truly autonomous online evolution in multi-robot systems was proposed in (Watson et al., 1999) and denominated embodied evolution (EE). EE addresses long-term self-adaptation but relies on robots meeting and exchanging genetic material. Frequent encounters between robots is difficult to guarantee, especially in large and open environments. After EE, different approaches on online evolution have been proposed (discussed in the next section). Notwithstanding, in such contributions, neuroevolution is limited to evolving weights in fixed-topology artificial neural networks (ANN).

In this paper, we introduce odNEAT, a novel online and distributed version of NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002; Stanley, 2004). NEAT is a state-of-the-art neuroevolution (NE) method that evolves the weights and the topology of an ANN. odNEAT shares some features with rtNEAT, a real-time enhancement of NEAT designed for video games (Stanley, 2005). In rtNEAT, game characters are able to evolve online while they are playing against humans. Both NEAT and rtNEAT operate in a centralised manner. odNEAT, on the other hand, is completely decentralised. In odNEAT, robots adapt autonomously on the basis of local information. The EA is distributed across multiple robots which have to solve the same task, either individually or collectively. We demonstrate odNEAT in a simulated experiment where a group of e-puck-like robots (Mondada et al., 2009) running an EA independently, online and onboard, must perform an aggregation task. To the best of our knowledge, the contribution presented here is novel in two aspects: (1) an online and distributed version of NEAT has not been proposed and studied prior to this work; (2) this is the first demonstration of online and onboard evolution where both the weights and the topology of the ANN controllers are under evolutionary control.

Related Work

In this section, we review the background and related work in the online evolution of ANN robotic controllers, as well as the main characteristics of NEAT and rtNEAT.

Online Evolutionary Robotics

The first attempt at truly autonomous online evolution in multi-robot systems, *embodied evolution*, was presented in (Watson et al., 1999). In this approach, each robot carries

only a single genotype and is controlled by the corresponding phenotype — a fixed-topology neural network. Robots probabilistically broadcast a part of their (mutated) genes at a rate proportional to their fitness (Probabilistic Gene Transfer Algorithm, PGTA). Robots that receive gene transmissions incorporate this genetic material in their genome at a rate inversely proportional to their fitness. This way, selection and variation (reproduction) operators are implemented in a distributed manner through the interactions between robots. A variant of this scheme was implemented in (Wischmann et al., 2007) in a predator-prey scenario. The interplay of evolution and lifelong individual learning was investigated as a mean of providing adaptability to novel environmental conditions. Each robot had a maturation period during which no mating/replacement can take place. This mechanism allowed robots to adapt using individual learning before being subjected to any selective pressure. However, within the authors' experimental framework, the effects of learning were not significant. Considering both approaches mentioned above, the main disadvantage is the fact that the embodied evolution was dependent on the exchange of genetic information among the robots. In large environments, where such encounters may be rare, the evolutionary process is therefore prone to stagnation.

A different approach, *encapsulated evolution*, overcomes stagnation by using a time-sharing mechanism. To that end, alternative controllers are executed sequentially and their fitness is measured. Each robot maintains a population of genotypes stored internally and run self-sufficient (and possibly different) EAs locally. Within this paradigm, the robots individually adapt through evolution without the necessity of interacting with other robots. Such approach has been successfully applied to tasks of an individual nature such as phototaxis or obstacle avoidance (Haasdijk et al., 2010; Bredeche et al., 2009).

The two methodologies, embodied evolution and encapsulated evolution, can be combined, leading to a hybrid system similar to an island model (Tanese, 1989). In such a system, each robot acts like an island with genetic information being exchanged through intra-island variation and inter-island migration. An example of such a method is the one presented in (Elfwing et al., 2005). In that study, robots have to gather batteries while maintaining a virtual energy level that reflects their task performance. If a robot's energy level reaches 0, offspring is created by mating the current controller with one of the genomes collected during lifetime. In (Usui and Arita, 2003), six Khepera robots evolved an avoidance behaviour. Each physical robot ran an independent EA for a sub-population of virtual agents, evaluated by time sharing. Migrated genomes, broadcasted by other robots, were re-evaluated by the receiving robot.

One of the limitations of existing approaches to online evolution is that neuroevolution solely adjust the weights of the ANN. Previous experimentation to determine a suitable network topology is therefore necessary. Choosing an inappropriate topology affects the evolutionary process and, consequently, the potential for adaptation. In odNEAT, on the other hand, the network topology is a product of a continuous evolutionary process.

NeuroEvolution of Augmenting Topologies (NEAT)

The NEAT method, introduced by (Stanley and Miikkulainen, 2002) is one of the most prominent neuroevolution (NE) algorithms. The method is capable of optimising both the topology of the network and its connection weights. NEAT acts with global and centralised information like canonical GAs. It has been successfully applied to highly complex problems, such as the double pole balancing, outperforming several methods that use fixed topologies (Stanley, 2004). The high performance of the algorithm is due to three key features: tracking genes with *historical markers* to allow meaningful crossover between topologies, *a niching scheme*, and evolving topologies incrementally from simple initial structures (*complexification*).

The network connectivity is represented through a flexible genetic encoding. Each genome contains of a list of connection genes, each of these referring the two node genes connected. Furthermore, a connection gene encompasses the weight of the connection, a bit indicating if the connection gene is genetically expressed and a *global innovation number* (IN), unique for each gene in the population. INs represent a chronology of the genes introduced. With this feature, the difficulty of matching different network topologies (an NP-hard problem) is avoided and crossover can be performed without *a priori* topological analysis. During crossover, genes with the same historical markings are aligned, to produce meaningful offspring. In terms of mutations, NEAT allows for common connection weights perturbations and structural changes that may lead to the insertion of: (1) a connection gene between two previously unconnected nodes or, (2) a node gene, splitting an old connection into two new connections and disabling the former. Each new gene inserted receives an innovation number. This way, genomes representing networks of different topologies remain compatible throughout evolution because their origin is known.

The niching scheme is composed of two building block: speciation and fitness sharing. Speciation divides the population into non-overlapping sets of similar individuals based on a topological similarity measure. This mechanism protects new structural innovations by reducing competition between individuals representing differing structures and network complexities. In this way, newer structures have time to mature. If a species does not improve for a certain number of generations, it is removed from the population. Explicit fitness sharing dictates that individuals in the same species share the fitness of their niche. The fitness scores of existing members of a species are first *adjusted*, i.e., divided by the

number of individuals in the species. Species then grow or shrink depending on whether their average adjusted fitness is above or below the population average.

The third reason why NEAT often outperforms other NE approaches is the incremental exploration of the search space. The algorithm starts with a uniform population of simple networks with no hidden nodes as in SAGA (Harvey, 1993). Complexity is introduced incrementally as a result of structural mutations. Since only structural mutations that have proven to be fit survive, the exploration of the search space is conducted in an incremental manner.

With the purpose of evolving increasingly complex ANNs online, rtNEAT was introduced (Stanley, 2005). Essentially, rtNEAT is a centralised real-time version of NEAT. rtNEAT contains some differentiating characteristics. While NEAT replaces the entire population at each generation, in rtNEAT one offspring is produced at regular intervals, every *n* time steps. The worst individual is removed and replaced with a child of a parent chosen from among the best. Unlike NEAT, rtNEAT attempts to keep the number of species constant by adjusting a threshold C_t, which determines the topological compatibility of an individual with a species. When there are too many species, C_t is increased to make species more inclusive; when there are too few, C_t is decreased to be stricter. rtNEAT has shown to preserve the dynamics of NEAT, namely protection of innovation through speciation and complexification (Stanley, 2004).

odNEAT: An Online and Distributed Evolutionary Algorithm

odNEAT runs on a group of agents whose objective is to evolve and adapt while operating in the environment. Each agent is controlled by an artificial neural network that represents a candidate solution to a given task. The evolutionary process takes place online and is an integral part of the agents' behaviour. The typical evolutionary operators (evaluation, selection and reproduction) are carried out autonomously by the agents in the environment without any need for external intervention.

In odNEAT, agents maintain a virtual energy level reflecting their individual performance in the task. The energy level increases and decreases as a result of the agent's behaviour, similarly to the work presented in (Elfwing et al., 2005). If an agent's energy reaches zero, its active chromosome (the genetic encoding of an ANN) is replaced. One general problem, especially for highly complex tasks, is that online evaluation is inherently noisy. Very dissimilar evaluation conditions may be presented to different chromosomes when they become active. Location of embodied agents and the proximity to other agents are factors that directly influence an agent's performance and behaviour. With the purpose of obtaining a reliable fitness estimate, odNEAT distinguishes between the fitness value of an agent and its current energy level. The fitness value is defined as the average of the energy level, sampled at regular time intervals.

In odNEAT, each agent maintains a local set of chromosomes in an internal repository. The repository is a genetic pool that stores a limited number of chromosomes and their respective fitnesses. The stored chromosomes are arranged into species based on the niching scheme of NEAT. The set of chromosomes include the agent's current and previous active chromosomes and those received from other agents. Each agent probabilistically broadcasts its active chromosome to agents in its immediate neighbourhood, an *inter-agent reproductive event*, with a probability computed as follows:

$$P(event) = \frac{\bar{F}_k}{\bar{F}_{total}} \tag{1}$$

where $\bar{F}_k$ is the average adjusted fitness of *local* species *k* to which the chromosome belongs and $\bar{F}_{total}$ is the sum of all *local* species' average adjusted fitnesses. Due to the broadcast of genetic information, the active chromosome of an agent may be present in another agent's repository. Such migrations approximate in a distributed manner and over time the reproduction dynamics of rtNEAT. This way, each repository is a local mirror of what happens in the population at large, but no agent has a complete global view of the system.

Besides the internal repository, each agent also maintains a local tabu list, a short-term memory which keeps track of recent *poor* solutions: chromosomes removed from the repository or that caused the robot to run out of energy. Newly received chromosomes must first be accepted by tabu list. The *acceptance condition* is only met if the received chromosomes are topologically dissimilar from all chromosomes in the tabu list.

After the pre-evaluation by the tabu list and if the acceptance condition was met, a received chromosome becomes part of the repository if it has a fitness score higher than the worst local chromosome thus enabling a progressive improvement. Due to the fixed size of the repository, whenever it is full, the insertion of a new chromosome is accompanied by the pre-requisite of removing the chromosome with the *worst adjusted fitness*. When a new chromosome is removed or added, the corresponding species has one less or one more element and therefore the adjusted fitness $\bar{F}$ is recalculated. Whenever an agent receives a copy *C'* of a chromosome *C* already contained in the repository (structurally the repository does not allow copies of the same chromosome), the energy level of *C'* is used to incrementally average the fitness of the *C* and provide a more reliable indicator of its value.

A particular characteristic of NEAT is the chronology of the genes due global to innovation numbers, which are assigned sequentially. In order to allow a decentralised implementation, odNEAT uses local high-resolution timestamps instead of innovation numbers. Each agent is responsible for

assigning a timestamp to each local innovation, be it a connection or a node. Using high-resolution timestamps for labels practically guarantees uniqueness and allows odNEAT to retain NEAT's concept of chronology.

When an agent's energy reaches zero (because it is incapable of accomplishing the task), a new individual is created. In this process - *an intra-agent reproductive event* - a parent species is chosen with probability proportional to its average fitness, as defined in Equation 1. Then, two parents are selected from the species, each one via a tournament selection of size 2. Offspring is then created based on NEAT's genetic operators: crossover of the parents' genomes and mutation of the new chromosome.

One important aspect regarding newly created individuals is the importance of letting them act in the environment for a minimum amount of time α. This time, denominated as the *maturation period*, gives the new individuals a change to spread their genome by mating with other agents and provides a habituation period. An individual can continue to be active after it reaches α, if its energy is above 0. In Fig. 1, we summarise odNEAT as executed independently by each agent.

```
odNEAT()
  initialize_genes()
  energy = default_energy
  LOOP
    if(broadcast?)then
      send(all_genes, agents_in_range)
    endif
    if(has_received?) then
      for element in received do
        if(tabu_and_repository_accept(element))
          add_to_repository(element)
          adjust_repository_size()
          adjust_species_fitness()
        endif
      endfor
    endif
    act_in_environment()
    energy = update_energy_level()
    if(energy <= 0 AND not(IN_MATURATION_PERIOD?))
      add_to_tabu_list(old_controller)
      generate_offspring()
      assign_as_controller(offspring)
    endif
  ENDLOOP
```

Figure 1: Pseudo-code of odNEAT that runs independently on every agent (see text).

Experimental Methodology

To assess odNEAT, we applied the algorithm in a simulated collective robotics experiment. The simulated robots are modelled after the e-puck (Mondada et al., 2009), a small (75 mm in diameter) differential drive robot capable of moving at a maximum speed of 13 cm/s. Each robot is equipped with eight infrared sensors, capable of obstacle detection and communication at a range of up to 25 cm between emitter and receiver.[1] Each infrared sensor is subjected to noise, which is simulated by adding a random Gaussian component within $\pm$ 5% of the sensor saturation value. Besides these sensors, each robot has an internal energy level sensor and a counter, which allow it to respectively perceive its current virtual energy level and the number of *distinct* chromosomes received during the most recent *P* control cycles.

The environment consists of a square arena surrounded by walls. The size of the arena was chosen to be 3 x 3 meters. At any time, a robot can thus sense less than 2.90% of the environment. Each of the robots is controlled by an artificial neural network produced by odNEAT. The input layer consists of one neuron for each proximity sensor (detects walls and other robots), one neuron for the energy sensor, and one neuron for the counter. The output layer contains two neurons, one for each wheel of the robot.

Since we are working with a process of continuous evolution, experiments continue until all robots achieve sustainable energy levels or until a temporal upper bound of 100 hours of simulated time is reached, in which case the experiment is considered to have failed. We are primarily interested in: i) determining if odNEAT evolves controllers capable of solving the specified task, ii) the elapsed time, to measure the speed of the evolutionary process and, iii) the quality of the solution and the behaviours evolved, that is, how the robots search through the environment and locate each other.

The Aggregation Task

In an aggregation task, dispersed agents must move close to one another so that they form a cluster. Aggregation plays an important role in many biological systems since it is the basis for the emergence of various collective behaviours. For instance, several social animals use aggregation to increase their chances of survival, or as a pre-cursor of other behaviours. In robotics, self-assembly and collective transport of heavy objects require prior aggregation at the site of interest. Due to the collective nature of the task, the topological (and possible behavioural) heterogeneity of the evolved controllers is an intriguing aspect.

Our experiments were conducted with a group of 5 robots placed in initial random positions at a minimum distance of 1.5 meters between neighbours. At each control cycle, a robot's virtual energy level *E* is updated according to the following equation:

$$\frac{\Delta E}{\Delta t} = \alpha(t) + \gamma(t) \tag{2}$$

where $\alpha(t)$ is a reward proportional to the number of controllers received in the last time period *P* (see Table 1). Since

[1]The original e-puck infrared range is 2-3 cm (Mondada et al., 2009). In real e-pucks, the *liblrcom* library, available at http://www.e-puck.org, allows to extend the range up to 25 cm and multiplex infrared communication with proximity sensing.

information is transmitted locally, this factor indicates the presence of robots nearby. $\gamma(t)$ is a factor related to the quality of movement and rewards robots that are capable of exploring the space in a relatively coordinated manner:

$$\gamma(t) = \begin{cases} \text{-1} & \text{if } v_l(t) \cdot v_r(t) < 0 \\ \Omega_s(t) \cdot \omega_s(t) & \text{otherwise} \end{cases} \quad (3)$$

where $v_l(t)$ and $v_r(t)$ are the left and right wheel speeds and $\Omega_s(t)$ is the ratio between the average and maximum speed achievable. $\omega_s(t) = 1 - \sqrt{|v_l(t) \cdot v_r(t)|}$ rewards robots for setting similar speeds on its two wheels, to avoid any turning-on-the-spot behaviour.

The experimental configuration is presented in Table 1. All parameters were fine-tuned through a trial-and-error process. Regarding NEAT, we have used the default parameters (as specified in Stanley (2005)) except for the crossover and mutation rates. Such parameters take values significantly lower than the default values (25% and 10%, respectively).

Parameter	Values
Repository size	40 chromosomes
Energy (initial/max)	1000/2000 e.u.
$\alpha(t)$	3 e.u. per chromosome
Time period *P*	10 cycles
Maturation period	500 cycles
Crossover rate	0.25
Mutation rate	0.1
Add node probability	0.03
Add connection probability	0.05
Weight mutation magnitude	0.5
Recurrent connection prob.	0.2
Maximum simulation time	100 hours

Table 1: Configuration for the aggregation experiments. Cycles represent robots' control cycles and e.u. denote energy units. The parameters were fine-tuned through a trial-and-error process.

Results and Discussion

In all 30 evolutionary runs performed, robots managed to evolve behaviours that could effectively explore the environment and keep the energy level above 0. We observed that aggregation into a single group was successfully achieved in 22 of the 30 runs. In the remaining 8 runs, the 5 robots formed two groups, one group of three robots and one group of two robots. In spite of such final configuration, robots still maintained self-sustainable energy levels. They evolve adequate behaviours for searching, locating and joining other robots in the environment. By analysing the details of each experiment, we observed the emergence of two types of strategies: *group clustering* (see Fig. 2) and *individual search* (see Fig. 3) behaviours.

Group Clustering: As a group, robots frequently evolve two distinct strategies (Fig. 2): a *static* and a *dynamic* clustering behaviour. In the static category, robots meet in some part of the environment and, by detecting one another, maintain their relative positions thus leading to a very stable behaviour. The other category, *flocking* or *dynamic* clustering creates loose and moving groups. In this case, robots meet and start moving together to explore the environment. The latter behaviour is less stable than the static clustering. When robots decide to flock and then collide with walls, it provokes a temporary de-synchronization of movement. As a consequence, and due to their short range of sensors, robots may lose sight of one another and will have to restart their search behaviour.

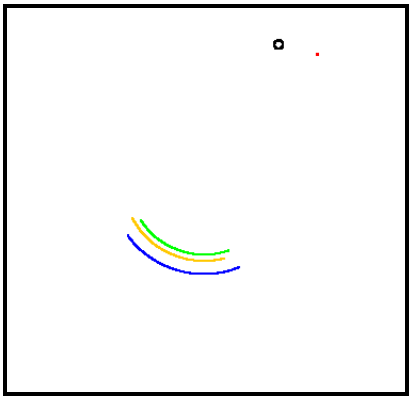

Figure 2: Traces of the robots' group clustering. Three robots exhibit a flocking behaviour while the remaining two form a static cluster, eventually leading to a single aggregate.

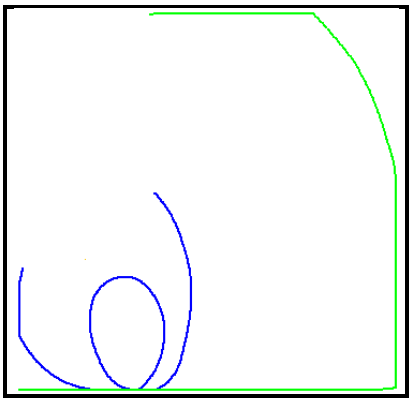

Figure 3: Traces of two of the most frequently evolved individual search strategies. One searches near the walls while the other presents a more circular trajectory thereby covering a larger area.

Individual Strategies: In terms of individual strategies for searching the environment, the evolved behaviours fall in

two categories (Fig. 3). The first one, *navigating near walls*, consists of exploring the environment by moving along the walls of the arena. In some instances of this behaviour, the searching robot moves away from the walls from time to time to explore. The second category consists of behaviours exhibiting a circular trajectory. The searching robot moves across the arena while rotating about itself. This way, the robot is capable of covering a wider area than the walls-based search strategy.

Figure 4 shows the time required to solve the task in each evolutionary run. The highest value is 24.43 hours of simulated time while the lowest is 1.10 hours. On average, each group of 5 robots takes 6.22 ± 5.55 hours of simulated time to aggregate.

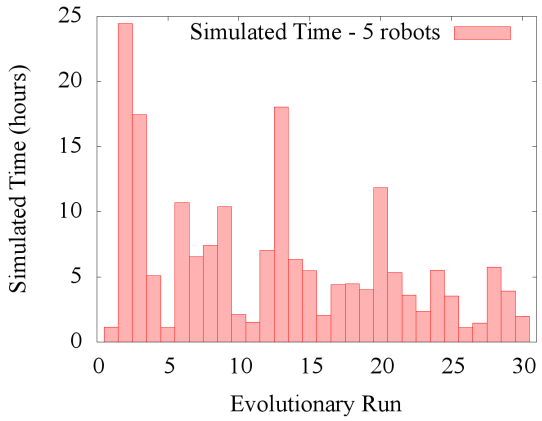

Figure 4: Experimental Results. Simulated time required to accomplish the task in each evolutionary run.

Measure	Average	Minimum	Maximum
Sim. Time	6.22 ± 5.55	1.12	24.45
Evaluations	104 ± 81	25	313

Table 2: Summary of the experimental results with a group of five robots. Time is listed in hours of simulated time.

Considering the average number of evaluations, i.e., controllers per robot, each robot was governed by 104 ± 81 controllers (see Table 2). The variance in the average time and number of evaluations can be explained by the non-linearity of the task. Robots have a short range of sensors and are placed in a large environment. Each robot may be able to search the environment very efficiently but, since it senses less than 2.90% of the total area, the process of finding other robots can be time consuming, especially if we consider that different robots are likely to execute different behaviours for exploring the environment, as happens with the individual search strategies.

rtNEAT and odNEAT

In spite of odNEAT being intentionally distributed, an interesting question is how the results of odNEAT compare to rtNEAT (Stanley, 2005), which relies on traditional centralised evolution. With the purpose of comparing the performance of odNEAT and rtNEAT and thus examine the costs of distributing NEAT, we setup a new series of experiments (with 30 independent runs) with a group of five robots. In order to provide a basis for comparison between the two EAs, two aspects of rtNEAT were altered. First, the dynamic compatibility threshold was fixed, as it is in odNEAT and NEAT. Second, offspring is not created based on a time condition but instead when a robot's energy level reaches zero. In the experiments, rtNEAT operated with a population size of 200 individuals, thus maintaining an average of 40 possible solutions per robot, as in odNEAT.

Method	Sim. Time	Evaluations
odNEAT	6.22 ± 5.55	104 ± 81
rtNEAT	4.44 ± 3.27	96 ± 60

Table 3: Performance comparison between odNEAT and rtNEAT, representing the costs of distributing NEAT. Time is listed in hours of simulated time.

Experimental results are listed in Table 3 and demonstrate the performance costs from distributing NEAT. In comparison with rtNEAT, odNEAT presents a slightly lower performance by requiring each robot to test approximately 8 controllers more (an equivalent to 8.33%). Notice that odNEAT, due to its distributed nature, does not assess the group level information from the global perspective. As a consequence, odNEAT requires more time to evolve solutions. The number of evaluations suggest that odNEAT provides results comparable to rtNEAT. The cost of operating solely based on local information is relatively low. odNEAT has another important advantage over the centralised EAs, namely the lack of dependency on an external mechanism which makes the approach resilient. If a robot fails, for instance, the group can adapt to accommodate for the faulty unit.

Ablation Studies

In order to verify the contribution of each algorithmic component in odNEAT, we performed a series of ablation studies considering the initial group of five robots. In particular, we tested the system's performance in three distinct experimental configurations: (1) without the maturation period, (2) with a minimal internal repository of size 2 and; (3) without the tabu list. Results, present in Table 4, are averaged over 30 independent evolutionary runs for each configuration. Averages in this table exclude runs that failed to find sustainable behaviours within 100 hours of simulated time.

The most critical algorithmic component of odNEAT is the internal repository, an evidence supported by statisti-

Method	Sim. Time	Evaluations	Failure Rate
Min. Repository	16.38 ± 21.15	236 ± 214	26.67%
No-tabu	8.88 ± 12.19	134 ± 119	6.67%
No-maturation	13.86 ± 22.53	211 ± 286	3.33%
Full odNEAT	6.22 ± 5.55	104 ± 81	0

Table 4: odNEAT ablations summary. The table lists the average simulation time (in hours), the average number of evaluations and the failure rate of each method, in the aggregation task. Each ablation leads to an inferior and less efficient algorithm.

cal significance ($\rho < 0.003$, Student's t-test). Since we are dealing with a process of continuous evolution, the repository maintains a local view of the system's history and provides the genetic basis for evolution. With a minimal repository, evolution is limited to a small set of chromosomes (in this case 2). In such scenario, the evolutionary process is much slower and may even be incapable of exploring enough of the solution space to find adequate solutions hence the high failure rate, simulation time and number of evaluations. Without the tabu list, odNEAT presents a failure rate of 6.67% but, when evolution is on the right track, it finds solutions relatively fast. Arguably, this means the tabu list keeps the evolutionary process from cycling around in one neighbourhood of the solution space, which sometimes happens due to the fact that robots act based only on local information. The tabu list promotes topological diversity in the repository by rejecting chromosomes similar to those that have already failed. The maturation period defines a lower bound of activity in the environment, giving the individuals a chance to spread their genome. If not for this component, good solutions could potentially be lost forever and evolution would be decelerated. Robots would still be capable of solving the task most of the times. However, they would not be able to and improve their behaviour iteratively through the exchange of genetic information unless they were situated close to each other. Arguably, the most important conclusion that can be drawn from the ablation studies is that all of the parts of odNEAT are necessary to guarantee its performance as an effective online distributed EA.

Scalability Experiments

The impact of the group size on performance was analysed by conducting 30 independent evolutionary runs for groups of 5, 10, 15, 20, 25 and 30 robots. The area of the arena was increased proportionally to the number of robots. Notice that if we maintained the same size of the environment, the experimental setup would not be fair: with the increasing density of robots in the environment, the task would be easier to solve simply because robots would encounter each other more frequently. Table 5 shows the area of the squared arena in each experimental configuration.

Group Size	Arena Area
5	9 m^2
10	18 m^2
15	27 m^2
20	36 m^2
25	45 m^2
30	54 m^2

Table 5: Environment size for each experimental configuration.

Experimental results are listed in Table 6. The time required to accomplish the task increases approximately 36% when the group size was increased from 5 to 10 robots. However, the average number of evaluations, a natural measure of performance, is almost similar except for a higher standard deviation. In fact, the increase in the time required is mainly due to 4 runs, displayed in Table 7. In these cases, robots managed to solve the task mainly by forming small aggregates, of two or three robots. Since small groups are difficult to detect by other robots, robots not belonging to any aggregate required more time to find a group of robots to join and to stabilise their behaviours.

Group Size	Sim. Time	Average Evaluations
5	6.22 ± 5.55	104 ± 81
10	8.49 ± 11.31	112 ± 117
15	3.71 ± 3.09	63 ± 44
20	3.49 ± 2.79	57 ± 37
25	3.33 ± 1.34	55 ± 22
30	3.78 ± 2.56	54 ± 28

Table 6: Summary of the scalability experiments. Time is listed in hours of simulated time.

Run	Sim. Time	Average Evaluations
5	33.87	419
8	27.47	364.6
14	36.68	135
16	43	500

Table 7: Outliers within the 10 robots' evolutionary runs. Time is listed in hours of simulated time.

Further increasing the group size, we observe that the performance improves substantially until it reaches a stable level around a group size of 15 robots. Results show that, for larger groups, the time required to accomplish the task and evolve sustainable behaviours is relatively constant. With the increase in the size of the environment, there is a larger area to search and explore. In relative terms and in spite of the group size increase, the robots sense a smaller portion of the environment. In this scenario, the stable performance

is, in fact, an argument in favour of odNEAT's scalability; the conditions for solving the task become more challenging and the robots are still able to evolve successful behaviours in the same amount of time.

Conclusions and Future Work

In this paper, we have introduced a novel approach called odNEAT, a completely distributed evolutionary algorithm for collective online learning in groups and swarms of embodied agents. We demonstrated odNEAT through a series of simulation-based experiments in which a group of e-puck-like robots evolved aggregation behaviours. Three points are worth mentioning about the experimental results. First, due to the asynchronous and distributed character of odNEAT, robots displayed different strategies for aggregating. Second, the behaviours evolved, static and dynamic clusters, as well as individual search strategies for exploring the environment, were observed simultaneously in the same group of robots. In spite of such behavioural diversity, robots manage to collaborate effectively towards the common goal. Finally, the comparison between rtNEAT and odNEAT suggest that, in spite of being a distributed EA, odNEAT provides results comparable to the standard centralised rtNEAT. The scalability experiments revealed that, for group sizes from 5 to 15 robots, odNEAT scales well considering the time required to achieve sustainable behaviours. For larger groups, odNEAT maintains the performance levels. Ablation studies show that the each of the algorithmic components provide a useful contribution to the performance of odNEAT, accelerating evolution and keeping the evolutionary process from cycling around in one neighbourhood of the solution space.

The immediate follow-up work will investigate a broader class of collective tasks in evolutionary robotics. One of the promising directions for odNEAT is to study to what extent agents are capable of continuously adapt in dynamically changing environmental conditions. In the future, we also intend to investigate the basic requirements for truly open-ended evolution, in which the evolutionary process should be capable of producing a large variety of different and novel solutions to a given task.

Acknowledgements

The authors gratefully acknowledge comments and discussions with Kenneth O. Stanley and Luis Correia during the preparation of this paper.

References

Bredeche, N., Haasdijk, E., and Eiben, A. E. (2009). On-line, on-board evolution of robot controllers. In Collet, P., Monmarche, N., Legrand, P., Schoenauer, M., and Lutton, E., editors, *9th International Conference on Artificial Evolution*, volume 5975 of *Lecture Notes in Computer Science*, pages 110–121. Springer-Verlag, New York, NY.

Elfwing, S., Uchibe, E., Doya, K., and Christensen, H. I. (2005). Biologically inspired embodied evolution of survival. In *IEEE Congress on Evolutionary Computation*, pages 2210–2216. IEEE Press, Edinburgh, UK.

Goldberg, D. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, volume 7. Kluwer Academic Publishers, Boston, MA.

Haasdijk, E., Eiben, A. E., and Karafotias, G. (2010). On-line evolution of robot controllers by an encapsulated evolution strategy. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE Press, Piscataway, NJ.

Harvey, I. (1993). Evolutionary robotics and SAGA: The case for hill crawling and tournament selection. In Langton, C. G., editor, *Proceedings of the Workshop on Artificial Life (ALIFE '92)*, volume 17 of *Sante Fe Institute Studies in the Sciences of Complexity*, pages 299–326. Addison-Wesley, Reading, MA.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65. IPCB, Castelo Branco, Portugal.

Stanley, K. O. (2004). *Efficient Evolution of Neural Networks through Complexification*. PhD thesis, The University of Texas at Austin, Austin, TX.

Stanley, K. O. (2005). Evolving neural network agents in the NERO video game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, pages 182–189. IEEE Press, Piscataway, NJ.

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

Tanese, R. (1989). *Distributed Genetic Algorithms for Function Optimization*. PhD thesis, University of Michigan, Ann Arbor, MI.

Usui, Y. and Arita, T. (2003). Situated and embodied evolution in collective evolutionary robotics. In Sugisaka, M. and Tanaka, H., editors, *Proceedings of the 8th International Symposium on Artificial Life and Robotics*, pages 212–215. Dept. of Electrical and Electronic Engineering, Oita University, Japan.

Watson, R. A., Ficici, S. G., and Pollack, J. B. (1999). Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 99)*, pages 335–342. IEEE Press, Piscataway, NJ.

Wischmann, S., Stamm, K., and Wörgötter, F. (2007). Embodied evolution and learning: The neglected timing of maturation. In e Costa, F. A., Rocha, L. M., Costa, E., Harvey, I., and Coutinho, A., editors, *European Conference on Artificial Life*, volume 4648 of *Lecture Notes in Computer Science*, pages 284–293. Springer-Verlag, Berlin, Germany.